The University of Wisconsin Library
Manuscript Theses

Unpublished theses submitted for the Master's and Doctor's degrees and deposited in The University of Wisconsin Library are open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but passages may be copied only with the permission of the authors, and proper credit must be given in subsequent written or published work. Extensive copying or publication of the thesis in whole or in part requires also the consent of the Dean of the Graduate School of The University of Wisconsin.

This thesis by JORG UNGER
has been used by the following persons, whose signatures attest their acceptance of the above restrictions.

A Library which borrows this thesis for use by its patrons is expected to secure the signature of each user.

NAME AND ADDRESS	DATE

Structural Analysis of Differential–Algebraic Equation Systems

by

Jörg Unger

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
CHEMICAL ENGINEERING

at the
University of Wisconsin — Madison

December 1990

Acknowledgements

At the beginning of my thesis I would like to thank several people and institutions for their support during may stay at the University of Wisconsin – Madison.
During the first nine months the *"Deutsche Akademische Austauschdienst"* supported me financially. I am grateful to Professor Michael Zeitz, Universität Stuttgart, for organizing this program which provided me with the lasting experience of studying abroad.
During the last seven months Professor W. Harmon Ray supported me by a Research Assistantship. During my entire stay he and his group also provided the academical and social environment that made my work here very enjoyable.
A key role during this project was played by Dr.Wolfgang Marquardt, Universität Stuttgart, whose great insight, methodology, and determination were very motivating and inspiring. I am deeply grateful for his great contribution and his pleasant way of advising.
Furthermore, I would like to thank the faculty, staff, and graduate students of the Department of Chemical Engineering for having me as guest.
The support, especially the *encouraging trips to the "Essen Haus"*, of my fellow german colleagues, Peter (The Piper) Häring, Eberhard Kleinbach, Christine Maul, and Andreas Tietze, helped me to "survive" the first semester and made the following ones much more enjoyable.

Outside the university I also had the chance to meet many nice people. In particular, I am grateful to the congregation and staff of Middleton Baptiste Church for their hospitality and their inspiring congregational live.

Several of my relatives and friends back home stayed in touch with me by writing and even calling. I appreciated their dedication and patience, even when I was not

able to answer immediately.
My parents supported me financially and emotionally in everything I did. I am very grateful for this unconditioned support.
A very important role through my entire stay played my girlfriend Ulrike. Her honesty and dedication were very motivating and comforting. The summer vacation we had will remain unforgettable for me.

Concluding, I would like to emphasize that although studying abroad was not only plain pleasure, every single experience was well worth it. I am very grateful for this opportunity.

Madison, December 1990

Jörg Unger

Contents

Chapter 1

Introduction

1.1 Significance of DAE's

Realistic mathematical models of most physical processes are usually represented by systems of equations that contain coupled algebraic equations, as well as, coupled partial and ordinary differential equations. In different areas of application these systems are referred to as singular, implicit, differential–algebraic, descriptor, generalized state space, noncanonic, noncausal, degenerate, semi–state, constrained, reduced order model, or nonstandard systems [5], more recently the term "differential equations on manifolds " has been used [42].
In the remainder they are just called *Differential Algebraic Equations, DAE's. Partial Differential Algebraic Equations (PDAE's)* are not considered separately here since by use of the *method of lines* they can be transferred into DAE's in principle.

Differential equations usually arise from energy, momentum, or material balances. *Algebraic equations* which are sometimes referred to as *constraints* represent important properties such as:

– closure conditions

– geometrical constraints

– thermodynamic equations of state (e.g. boiling point curves).

Another way of obtaining algebraic equations is the introduction of *quasi-steady state assumptions.*

1.2 Properties of DAE's

The numerical solution of DAE's can be by far more complicated than the integration of standard ordinary differential equations (ODE's)[39]. Important properties of DAE's which cause this difference are illustrated by two short examples below.

Example 1.1: Fixed Length Pendulum

The following equations arise from the Euler-Lagrange formalism if the pendulum is modelled as a massless bar with an infinitesimally small ball of mass m at its end and if (x, y) are the cartesian coordinates of the ball with respect to the bearing:

$$\ddot{x} = \frac{2\lambda}{m} x \,, \tag{1.1}$$

$$\ddot{y} = \frac{2\lambda}{m} y - g \,, \tag{1.2}$$

$$0 = x^2 + y^2 - L^2 \,, \tag{1.3}$$

where g is the gravitational constant and λ the Lagrange multiplier which represents the tension in the bar.
In state space representation with

$$z_1 = x \,, \quad z_2 = \dot{x} \,, \quad z_3 = y \,, \quad z_4 = \dot{y} \,, \quad z_5 = \lambda$$

the system above can be rewritten as

$$\dot{z}_1 = z_2 \,, \tag{1.4}$$

$$\dot{z}_2 = \frac{2}{m} z_5 z_1 , \tag{1.5}$$

$$\dot{z}_3 = z_4 , \tag{1.6}$$

$$\dot{z}_4 = \frac{2}{m} z_5 z_3 - g , \tag{1.7}$$

$$0 = z_1^2 + z_3^2 - L^2 . \tag{1.8}$$

By differentiating (1.8) with respect to time and substituting (1.4) and (1.5) one obtains

$$0 = z_1 z_2 + z_3 z_4. \tag{1.9}$$

If this equation is differentiated once more with respect to time and if (1.4)–(1.7) are substituted then

$$0 = m (z_2^2 + z_4^2) - 2mg z_3 + 2L^2 z_5 \tag{1.10}$$

is obtained.

Equations (1.9) and (1.10) represent relations which do not occur explicitly in the original system (1.4)–(1.8), but are satisfied implicitly; they are called *"hidden"* relations. Further differentiations would not yield additional relations among the variables $z_i, i = 1(1)5$, since they would introduce higher derivatives of those variables and could not place further restrictions on the ones occurring in (1.4)–(1.8).

An important characteristics of a DAE is its *index* , which is *the number of differentiations that have to be applied to it in order to be able to transform it into an ODE by algebraic manipulations.* This means that the index is an integer and that it is zero for ODE's.

Obviously, after one more differentiation of (1.10) it can be solved for

$$\dot{z}_5 = \dot{z}_5(z_1, z_2, z_3, z_4, z_5) \tag{1.11}$$

and along with (1.4)–(1.7) an ODE is obtained after three differentiations which means that the DAE (1.4)–(1.8) is of index three. Equation (1.11) can also be called

a "hidden" equation since it does not appear in the set of original equations (1.4)–(1.8).

If (1.4)–(1.8) ought to be solved numerically then initial conditions have to be provided in order to start the integration. A set of initial values z_{0i},$\dot{z}_{0i}$, $i = 1(1)5$ is called *consistent* if it satisfies

$$z_{0i} = z_i^*(t_0) \ , \quad \dot{z}_{0i} = \dot{z}_i^*(t_0) \ , \quad i = 1(1)5 \tag{1.12}$$

where t_0 is the initial time and $z_i^*(t)$, $\dot{z}_i^*(t), i = 1(1)5$ represent a solution of the DAE. Quite a few currently available numerical methods and codes require consistent initial values in order to carry out a successful and efficient numerical treatment of DAE's. In order to provide *consistent initialization* $z_i, \dot{z}_i, i = 1(1)5$, must satisfy both, the original (1.4)–(1.8) and the hidden (1.9)–(1.11) equations since a solution of the DAE satisfies all of them. Thus, one is left with eight equations (1.4)–(1.11) in ten variables $z_{0i}, \dot{z}_{0i}, i = 1(1)5$ for $t = t_0$. Even though there are four differential equations only two arbitrary initial conditions may be chosen. All others are determined by the system.

Summarizing it can be said that:

1. The number of free initial conditions of a DAE is in general less or equal the number of equations of the DAE containing time differentials.

2. In order to provide consistent initial values for all variables $z_i, \dot{z}_i, i = 1(1)5$ one has to pick the subset of variables which are assigned arbitrary initial conditions such that all other variables are determined uniquely by the joint system of original and hidden equations.

Example 1.2: CSTR-Cascade with Dynamic Design

Another possibility of introducing algebraic equations is the assignment of desired

time courses to some state variables for reasons of design [38]. An example for this is a series of N_r continuously stirred tank reactors (CSTR's) where the output concentration $c_{N_r}(t)$ of the desired product should have some time course $w(t)$. If reactions are modelled by rates R_i and if the mean residence time of the ith reactor is τ_i then the equations become:

$$\dot{c}_i = \frac{1}{\tau_i}(c_{i-1} - c_i) + R_i\,, \quad i = 1(1)N_r\,, \tag{1.13}$$

$$0 = c_{N_r} - w(t)\,. \tag{1.14}$$

c_0 is the feed concentration of the first reactor of the cascade and would therefore be considered an *input variable* which can be specified by the modeler.

If reactions are omitted and if all reactors are assumed to have the same volume such that

$$\tau_i = \tau\,, \quad i = 1(1)N_r\,, \tag{1.15}$$

then the solution for the ith concentration with respect to the normalized time $t^* = \frac{t}{\tau}$ becomes [1]:

$$c_i(t^*) = \sum_{k=0}^{N_r-i} \binom{N_r - i}{k} \frac{d^k}{dt^k} w(t^*) \qquad i = 0(1)N_r \tag{1.16}$$

These solutions show important features of DAE's:

1. $c_0(t^*)$ is determined uniquely by the system. Thus, it cannot be specified by the modeler although it seemed to be an input variable initially.

2. The solutions for $c_i, i = 0(1)N_r - 1$, involve *higher order derivatives of* $w(t)$ although only $w(t)$ occurs in the original equations. This causes numerical problems if $w(t)$ is smooth only up to order $k < N_r - 1$, i.e., $w(t) \in C^k$, $k < N_r - 1$.

3. There is *no* arbitrary initial condition for any of the $N_r + 1$ variables c_i, $i = 0(1)N_r$, although the system contains N_r ordinary differential equations. For

DAE's it is generally required to carry out *further examinations of the equations in order to provide consistent initialization*, this has already been observed in the previous example.

4. In order to transform (1.13),(1.14) into an ODE, $N_r + 1$ differentiations have to be applied to it, i.e., this DAE is of index $N_r + 1$. For a large number of CSTRs the index will be large as well. DAE's with large indices arise from dynamic design tasks quite frequently.

It has been known for several years that integration of index-1-systems is no more complicated than integration of ODE's (i.e., index-0-systems)[19]. However, it is still very difficult to solve higher-index-systems (i.e., systems of index > 1) properly. There are index two and three DAE's for which all multistep and Runge-Kutta methods are unstable [24], for some index three systems they are even not convergent. In recent years this led to the development of numerical methods and codes which can cope with higher-index-systems of *particular structure* [5]
Only most recently a method which is supposed to be able to cope with *any* higher-index-DAE has been published [11]. To our knowledge, so far there have not been any publications on the performance of its implementation for realistic problems.

1.3 Overview

The problems and comments above clearly indicate the need for a characterization of a DAE *prior* to any numerical solution. This thesis addresses the problem of determining all properties of a DAE required prior to any solution in order to allow its successful and efficient numerical treatment. Properties of interest are the *index*, which affects the choice of a proper numerical method, and the number of free initial

conditions, which in the remainder is called *the number of degrees of freedom.* Since many methods require consistent initial values it is also important to know *which* variables can be assigned free initial conditions and which equations have to be satisfied by the remaining variables in order to provide consistent initialization.

Based on a proposition by Gear [22] we formulate an algorithm which determines the index and the number of degrees of freedom of DAE's of the most general form:

$$F(z, \dot{z}, u) = 0 \tag{1.17}$$

where $F, z, \dot{z} \in \mathcal{R}^n$, $u(t) \in \mathcal{R}^p$, and $u(t)$ represents input variables and other explicit time dependencies.

For large and highly nonlinear systems it is computationally not tractable or even not possible to carry the algorithm out *symbolically.* Therefore, a relaxed *structural* version of the algorithm is suggested here. It is shown that its results are consistent with the definitions of structural properties of DAE's. By keeping track of the sequence of computations within the algorithm it is possible to tell how the equations required for consistent initialization can be obtained and which steps could be taken to reduce the index as far as required. This is an advantage of this algorithm over a structural algorithm proposed earlier by Pantelides [37]. His approach mainly addresses the problem of consistent initialization.

After a concise review of the *Theory of DAE's* and the introduction of an *Algorithm for Index Reduction* in Chapter 2 there are some comments on different *Concepts of the Numerical Treatment of DAE's* in Chapter 3. In Chapter 4 the idea of *Structural Considerations* is introduced. The meaning of propositions and definitions given in Chapter 2 is examined from the structural perspective. *Limitations* of the structural approach are pointed out. Based on these *relaxed definitions* the *Structural Algorithm* is formulated. The consistency of the algorithm and its results with previously defined structural ideas is proven. At the end of the fourth chapter Pan-

telides' algorithm is introduced briefly and compared with our approach. Chapter 5 gives results obtained from implementations of both algorithms for various examples. Comments on implementation, computational time, efficiency, performance, and limitations of both algorithms are given. Finally, along with recommendations for further theoretical work in the area of DAE's, concluding remarks on possible applications of the algorithm developed are given in Chapter 6.

Chapter 2

Theory of DAE's

This chapter is meant to be a "dictionary" which *defines* the terminology associated with DAE's. *Relations and consequences* of those definitions are summarized in several *propositions.*

Before the formal introduction of various definitions and propositions, important questions associated with the numerical solution of DAE's are highlighted below. These questions should be answered *prior* to attempting any numerical solution of a differential–algebraic model:

1. Does the DAE have a solution, i.e., is it solvable ?

2. If it is solvable, what is its index ?

3. If the index exceeds one, can the numerical method intended for the numerical solution cope with higher-index-systems ?

4. How can the index be reduced if it is required for a successful numerical treatment ?

5. Does the numerical method intended to solve the DAE require consistent initialization ?

6. If consistent initialization is required, which variables can be assigned arbitrary initial values and which equations have to be satisfied by the initial values of the remaining variables ?

In order to be able to answer those questions important properties of DAE's such as *solvability, index, degrees of freedom, and consistent initialization* have to be defined precisely. Based on reviews of the theory of both, DAE's and matrices [5,8,13,18,32,35,48], this is done in the first section of this chapter. Relations among those characteristics are pointed out and an algorithm for index reduction is introduced. Later in this chapter solvability conditions and particular characteristics of DAE's are indicated. The last section of this chapter deals with the problem of consistent initialization.

2.1 Definitions and Relations

2.1.1 Particular Types of DAE's

The most general representation of a DAE is

$$F(z, \dot{z}, u) = 0 \,, \tag{2.1}$$

where $z, \dot{z}, F \in \mathcal{R}^n$, $u(t) \in \mathcal{R}^p$. This can be referred to as the *nonlinear implicit form.* Input variables and other explicit time dependencies are represented by $u(t)$.

There are some special cases of this most general form; even a *system of purely algebraic equations* $h(z, u) = 0$ is one of them. If $detF_{\dot{z}} \neq 0$ then it is an *implicit ODE* . Most of the systems in Chemical Engineering are of the *linear implicit form*:

$$M(z, u)\dot{z} + h(z, u) = 0 \,. \tag{2.2}$$

If $detM \neq 0$ then it is just a *linear implicit ODE* . A special case of (2.1) which is

used in the dynamic simulation environment DIVA [30] is:

$$A(x,y)\,\dot{x} + h_1(x,y,u) = 0\,, \tag{2.3}$$

$$C(x,y)\,\dot{x} + h_2(x,y,u) = 0\,, \tag{2.4}$$

where $A(x,y) \in \mathcal{R}^{r\times r}$, $detA(x,y) \neq 0$, $C(x,y) \in \mathcal{R}^{(n-r)\times r}$, $z = (x,y)^T$, $z \in \mathcal{R}^n$, $x, h_1 \in \mathcal{R}^r$, $y, h_2 \in \mathcal{R}^{n-r}$. This is equivalent to:

$$\dot{x} - f(x,y,u) = 0\,, \tag{2.5}$$

$$g(x,y,u) = 0\,, \tag{2.6}$$

where $y \in \mathcal{R}^{n-r}$, $x, f \in \mathcal{R}^r$, $u \in \mathcal{R}^p$, and $r = rank(A\ C)^T$. To show this equivalency we prove

Proposition 2.1.1: *Each DAE* $F(z,\dot{z},u) = \begin{pmatrix} F^1(z,\dot{z},u) \\ F^2(z,\dot{z},u) \end{pmatrix} = 0$ *can at least locally be brought into the form*

$$F^1(x,\dot{x},y,\dot{y},u) = 0\,, \tag{2.7}$$

$$G(x,y,u) = 0\,, \tag{2.8}$$

where $F^1_{\dot{x}}$ *is a regular* $(\kappa \times \kappa)$*-matrix with* $\kappa = \max(\mathrm{rank}F_{\dot{z}})$, x, y *subsets of* $z = (x,y)^T \in \mathcal{R}^n$, *and* $F^2, G \in \mathcal{R}^{n-\kappa}$.

Proof of Proposition 2.1.1: The basic idea of this proof can also be found in [22]. By the implicit function theorem $detF^1_{\dot{x}}|_* \neq 0$ is a necessary condition for being able to solve for $\dot{x}^* = \dot{x}^*(x^*, y^*, \dot{y}^*, u(t^*))$ locally at some point $(\dot{x}^*, x^*, y^*, \dot{y}^*, t^*)$. For highly nonlinear equations it might not be possible to obtain a closed symbolical expression $\dot{x} = \dot{x}(x,y,\dot{y},u)$ by inverting $F^1(x,\dot{x},y,\dot{y},u) = 0$ [1]. For those systems the form

[1]Since for linear implicit systems

$$M(z,u)\,\dot{z} + h(z,u) = 0$$

(2.7),(2.8) can only be obtained by carrying out the procedure below locally at some point $(\dot{x}^*, x^*, y^*, \dot{y}^*, t^*)$.

By the argument above we can assume without loss of generality that for $detF_{\dot{x}}^1 \neq 0$ we can at least locally determine $\dot{x} = \dot{x}(x, y, \dot{y}, u)$ and substitute it into

$$F^2(x, \dot{x}(x, y, \dot{y}, u), y, \dot{y}, u) = 0 \tag{2.9}$$

to obtain

$$G(x, y, \dot{y}, u) = 0\,. \tag{2.10}$$

Assume G really depends on the ith component of $\dot{y}$ explicitly then $G_{\dot{y}_i}$ is not identically zero. Thus,

$$\dot{y}_i \;=\; \dot{y}(x, y, \dot{\tilde{y}}, u)\,, \tag{2.11}$$

could be determined by the implicit function theorem at least locally, where $\dot{\tilde{y}}$ represents the subset of $\dot{y}$ which is obtained by taking out $\dot{y}_i$. But, if F^1 was chosen such that $F_{\dot{x}}^1$ is a regular $(\kappa \times \kappa)$–matrix with $\kappa = \max(\text{rank}F_{\dot{z}})$, and x, y subsets of $z = (x, y)^T \in R^n$ then there is a contradiction since in addition to $\dot{x}$ also $\dot{y}_i$ can be determined by the implicit function theorem. This on the other hand implies that rank $F_{\dot{z}}(z, \dot{z}, u) = \kappa + 1$ which violates the assumption that

$$\kappa \;=\; \max(\text{rank } F_{\dot{z}}). \tag{2.12}$$

Thus, if the partition $F(z, \dot{z}, u) = \begin{pmatrix} F^1(z, \dot{z}, u) \\ F^2(z, \dot{z}, u) \end{pmatrix} = 0$ is done such that assumption (2.12) holds then $G_{\dot{y}} \equiv 0$ is satisfied generally. □

Further special cases of the linear implicit DAE are the *linear time varying and time invariant systems* where h is of the form

$$h(z, u) = L\,z + E\,u(t)\,, \tag{2.13}$$

it is in principle possible to obtain $\dot{x} = \dot{x}(x, y, \dot{y}, u)$ symbolically, these systems can actually be transformed into the form (2.7),(2.8) symbolically.

where $L \in \mathcal{R}^{n\times n}$, $u(t) \in \mathcal{R}^p$, $E \in \mathcal{R}^{n\times p}$, and where M, L, E, u are just time dependent or constant respectively. Linear DAE's are treated by quite a few authors in control theory. Lewis [32] and Dai [13] gave surveys of linear singular systems which both contain an extensive bibliography for this particular class of DAE's.

2.1.2 Solvability

Although solvability obviously is a very important property of a DAE there are several definitions of it in the literature [5,8,18,35,48]. For *linear time invariant systems*

$$M\,\dot{z} + L\,z + u(t) = 0 \tag{2.14}$$

which are frequently dealt with in control problems Yip and Sincovec [48] showed that various definitions in the literature are equivalent to:

Definition 2.1.1: *The system* $M\dot{z} + Lz + Eu(t) = 0$ *is solvable if the matrix pencil* $(\lambda M + L)$ *is regular, i.e.,* $\det(\lambda M + L) \not\equiv 0$ *for all except a finite number of* $\lambda \in \mathcal{C}$ *where* $\mathcal{C}$ *is the field of complex numbers.*

The probably most general and abstract definition is given in [5,p.16]

Definition 2.1.2 : *Let* $\mathcal{I}$ *be an open subinterval of* $\mathcal{R}$, Ω *a connected open subset of* $\mathcal{R}^{2n+1}$, *and* F *a differentiable function from* Ω *to* R^n. *Then the DAE* $F(z, \dot{z}, u(t)) = 0$ *is solvable on* $\mathcal{I}$ *in* Ω *if there is an* r*-dimensional family of solutions* $\phi(t, c)$ *defined on a connected open set* $\mathcal{I} \times \tilde{\Omega}$, $\tilde{\Omega} \subset \mathcal{R}^r$ *such that:*

1. $\phi(t, c)$ *is defined on all of* $\mathcal{I}$ *for each* $c \in \Omega$.
2. $(t, \phi(t, c), \dot{\phi}(t, c)) \in \Omega$ *for* $(t, c) \in I\times\tilde{\Omega}$.
3. *If* $\psi(t)$ *is any other solution with* $(t, \psi(t), \dot{\psi}(t)) \in \Omega$, *then* $\psi(t) = \phi(t, c)$ *for some* $c \in \tilde{\Omega}$.

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

Simplified, the precise definition above means that a DAE is solvable if at least locally a solution $z^*(t)$ (i) *exists* and (ii) *is uniquely defined* on an interval $\mathcal{I} =]t_0, T]$ if the initial state of the system at $t = t_0$ is determined by specifying a set c of r arbitrary initial values. T denotes the upper limit of the time range of interest. Since *Definition 2.1.2* considers the solution only on a connected open *subset* Ω of $\mathcal{R}^{2n+1}$ it is sufficient for the DAE to satisfy (i) and (ii) above *locally* in order to be solvable according to this definition.
A geometrical illustration of the meaning of this definition is given at the end of this chapter (see 2.1.8). At this point we only indicate that it *excludes bifurcating solution trajectories* .

In order to motivate necessary solvability conditions for general DAE's $F(z, \dot{z}, u) = 0$ we have to define and illustrate several more characteristics of DAE's.

2.1.3 Well-Posedness

Sometimes in literature it is not clearly distinguished between solvability and and well-posedness. The following definition is according to Hadamard [34]:

Definition 2.2 : *A DAE $F(z, \dot{z}, u) = 0$ is called well-posed if*

1. *it has a solution (i.e., is solvable)*
2. *its solution is unique*
3. *its solution is continuous in changes of the data (e.g. small changes of initial conditions)*

The first and the second requirement of this are met by systems that are solvable according to *Definition 2.1.2*. However, systems being solvable according to this definition do not necessarily have to satisfy the third condition above. Thus, well-posedness requires more than solvability. Since we are aiming for *structural* properties of DAE's we are merely interested in the existence of a solution rather than in well-posedness. Note that solvability according to *Definition 2.1.2* requires existence *and* uniqueness of a solution and therefore, also requires more than needed for the structural considerations in Chapter 4.

2.1.4 Index

The origin of the term "index" is in matrix theory where *the index of nilpotency* of a matrix $N \neq 0$ is ν if $N^{\nu} = 0$ and $N^{\nu-1} \neq 0$ [18]. Sometimes the index of nilpotency is also called the *degree of nilpotency.* After the following formal definition of the *(differential) index of a solvable DAE* it is pointed out below how this relates to nilpotent matrices and the *index of a matrix pencil.*

Definition 2.1.3 : *The minimum number of times that all or subsets of the equations of the DAE* $F(z, \dot{z}, u) = 0$ *itself or of equations derived from it after previous differentiations have to be differentiated with respect to time* t *in order to determine* $\dot{z}$ *as a continuous function of* z, t, *is called the (differential) index of the DAE.*

Basically, this definition initially given by Petzold [39] is adopted by various authors [11,22,38,45]. The relation to matrix theory can be seen by considering the linear constant coefficient DAE of index k

$$M\dot{z} + Lz + Eu(t) = 0\,, \tag{2.15}$$

where $M, L \in \mathcal{R}^{n\times n}$, $E \in \mathcal{R}^{n\times p}$, $z \in \mathcal{R}^n$, $u \in \mathcal{R}^p$. If $det(\lambda M + L) = 0$ only for some singular λ then the system can be transformed into the *Kronecker Canonical Form*

$$\dot{\zeta}_1 + C\zeta_1 - g_1(t) = 0\,, \tag{2.16}$$

$$N\dot{\zeta}_2 + \zeta_2 - g_2(t) = 0\,, \tag{2.17}$$

where $\zeta_1, g_1 \in \mathcal{R}^{r_0}$, $\zeta_2, g_2 \in \mathcal{R}^{n-r_0}$, $C \in \mathcal{R}^{r_0\times n}$, and $N \in \mathcal{R}^{(n-r_0)\times(n-r_0)}$ by means of the transformation

$$z = Q\zeta \quad g = -Ph\,, \tag{2.18}$$

where P, Q are regular $(n \times n)$-matrices. N is a nilpotent matrix of degree k since the DAE is assumed to be of index k. We illustrate that this is true by solving for $\dot{\zeta}_2 = \dot{\zeta}_2(\zeta, t)$ explicitly by applying successive differentiations and substitutions to (2.17). Since det $N = 0$ (2.17) cannot be solved for $\dot{\zeta}_2$ right away. If (2.17) is rewritten as

$$\zeta_2 = -N\dot{\zeta}_2 + g_2 \tag{2.19}$$

and this is differentiated $(k-1)$ times then one finally is left with k equations

$$\zeta_2 = -N\dot{\zeta}_2 + g_2(t)\,, \tag{2.20}$$

$$\dot{\zeta}_2 = -N\ddot{\zeta}_2 + \dot{g}_2(t)\,, \tag{2.21}$$

$$\vdots$$

$$\zeta_2^{(k-1)} = -N\zeta_2^k + g_2^{(k-1)}(t)\,, \tag{2.22}$$

where $\zeta_2^{(j)}$ and $g_2^{(j)}$ represent the jth time derivative of ζ_2 and $g_2(t)$. By backward successive substitution of $\zeta_2^{(j)}$ into equation $(j-1)$ one obtains

$$\zeta_2 = (-1)^k N^k \zeta_2^{(k)} + \sum_{j=0}^{k-1}(-N)^j \frac{d^j g_2}{dt^j}. \tag{2.23}$$

If the index of nilpotency of N is k, i.e., if $N^k = 0$ then

$$\zeta_2 = \sum_{j=0}^{k-1}(-N)^j \frac{d^j g_2}{dt^j}. \tag{2.24}$$

Thus, after one more differentiation of these equations with respect to time t

$$\dot{\zeta} = (\dot{\zeta}_1 \; \dot{\zeta}_2)^T = \dot{\zeta}(\zeta, t) \tag{2.25}$$

is obtained since $\dot{\zeta}_1(\zeta_1, t)$ has already been determined by (2.16). In matrix theory the degree of nilpotency k of N is defined to be the *index of the matrix pencil* $(\lambda M + L)$. Looking at *Definition 2.1.3* now shows that both, the index of the matrix pencil $(\lambda M + L)$ and the (differential) index of the DAE, are k.

A more detailed observation of nonlinear systems leads to the distinction between a *local* and a *global (differential) index.* Since *Definition 2.1.3* does not state anything about exceptions at some points along a trajectory of a solution it defines the *global* (differential) index ν_G, i.e., after ν_G differentiations of the entire DAE or subsets of equations of it with respect to time t, it has to be possible to determine $\dot{z}$ as a continuous function of z, t *at any point along the trajectory of any possible solution.* Considering the implicit function theorem one can see that there can be differences among different points in state space concerning the number of differentiations required to determine $\dot{z}$ as a continuous function of z, t at these points. These differences are due to the fact that the numerical Jacobians of nonlinear systems can be regular for some points and singular at others. This motivates the notation of the *local* (differential) index. Its definition according to [5,p.33] is given below:

Definition 2.1.4: *The local index ν_l of the DAE $F(z, \dot{z}, u) = 0$ at $(z^*, \dot{z}^*, t^*)$ is the index of the pencil $(\lambda F_{\dot{z}}(z^*, \dot{z}^*, t^*) + F_z(z^*, \dot{z}^*, t^*))$.*

Comparing *Definition 2.1.3* and *Definition 2.1.4* one observes that the following relation always holds:

Proposition 2.1.2: *The local (differential) index ν_l of a DAE is always less or equal*

to its global (differential) index ν_G, i.e., the equivalent relations

$$\nu_G = \max \nu_l \quad \textit{and} \quad \nu_l \leq \nu_G$$

hold for every DAE at every point of a trajectory of a every possible solution.

A different approach introduces the *perturbation index* [27]:

Definition 2.1.5: *The DAE $F(z, \dot{z}, u) = 0$ has perturbation index m along a solution $z(t)$ for t in a bounded interval $\mathcal{I}$, if m is the smallest integer such that, for all functions $\hat{z}(t)$ having a defect such that*

$$F(\dot{\hat{z}}, \hat{z}) \;=\; \delta(t) \tag{2.26}$$

there exists an estimate on I

$$\|\hat{z}(t) - z(t)\| \;\leq\; W\|\hat{z}(0) - z(0)\| + \max_{0\leq\tau\leq t}\|\delta(\tau)\| + \ldots + \max_{0\leq\tau\leq t}\|\delta^{(m-1)}(\tau)\| \tag{2.27}$$

whenever the expression on the right-hand side is sufficiently small. W denotes a constant which depends only on F and the length of the interval $\mathcal{I}$.

The relationship between the (differential) index and the perturbation index is indicated in [27]: The perturbation index is either the same as the (differential) index or bigger by one. The introduction of the perturbation index and its relation with the commonly considered (differential) index illustrates why systems whose (differential) index exceeds one cause numerical problems. The perturbation index of those systems is at least one. If $\hat{z}$ in *Definition 2.3.2* above is assumed to be a numerical solution which due to roundoff and discretization errors is bound to have a defect with respect to the "correct" solution z then the estimate for the error $\|\hat{z}(t) - z(t)\|$ involves at least first order derivatives of $\delta(t)$. The higher the (differential) index the higher will

be the maximal order of the derivatives of $\delta(t)$ involved in this estimate. Due to truncation and roundoff errors in a numerical solutions $\delta(t) \neq 0$ will generally be the case. Additionally, it can be assumed that $\delta(t)$ is a noisy signal, whose generally high frequency is related to the stepsize of integration. If the differential index exceeds one then the error estimate $\|\hat{z}(t) - z(t)\|$ involves derivatives of $\delta(t)$ of order ≥ 1. In general the derivatives of a noisy signal is even noisier than the signal itself. This illustrates why the numerical solution of higher-index-problems tends to be unstable if ODE methods are used without proper adjustments.
In the remainder the term *"index"* stands for *global (differential) index.*

2.1.5 An Algorithm for Index Reduction

In order to motivate further important notations it is convenient to introduce an algorithm for index reduction at this point. For the first time the idea of this algorithm appeared in [43] for linear constant coefficient DAE's in control problems. For general nonlinear DAE's it was introduced but not formulated rigorously by [22]. The key idea of this algorithm is to find all algebraic relations among components of z which are satisfied implicitly without occurring in the original formulation of the DAE. This is accomplished by differentiating algebraic equations and substituting known relations for components of $\dot{z}$ in order to obtain further algebraic equations. This is done until all components of $\dot{z}$ can be determined as functions of z, t. A formal description of this algorithm is given below:

Algorithm 2.1.1:

Step 0: Initialization:

$$
\begin{array}{lcl}
i & = & 0\,, \\
n_0 & = & n\,, \\
r & = & n\,, \\
\tilde{z}_0 & = & z\,, \\
F_0 & = & F\,, \\
X_{-1} & \ldots & \text{is a "vector" with no elements}\,.
\end{array}
$$

Step 1: Determine: $r_i = \max \operatorname{rank} \frac{\partial F_i}{\partial \tilde{z}_i}$, where $\tilde{z}_i \in \mathcal{R}^{n_i}$,
and set $r \rightarrow r - (n_i - r_i)$.

Step 2: Partition $F_i(z, \dot{z}_i) = \begin{pmatrix} F_i^1(z, \dot{x}_i, \dot{y}_i) \\ F_i^2(z, \dot{x}_i, \dot{y}_i) \end{pmatrix}$ such that $det \frac{\partial F_i^1}{\partial \dot{x}_i} \neq 0$,
where $F_i^1, x_i \in \mathcal{R}^{r_i}$, $\tilde{z}_i = (x_i, y_i)^T \in \mathcal{R}^{n_i}$, $F_i^2, y_i \in \mathcal{R}^{(n_i - r_i)}$.

Step 3: Solve $F_i^1(z, \dot{x}_i, \dot{y}_i) = 0$ for $\dot{x}_i = \dot{x}_i(z, \dot{y}_i)$ and add it to
$\dot{X}_{i-1} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{i-1})^T = \Upsilon_{i-1}(z, \dot{\tilde{z}}_i)$
to obtain
$\dot{X}_i = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{i-1}, \dot{x}_i)^T = \Upsilon_i(z, \dot{x}_i, \dot{y}_i)$.

Step 4: Substitute $\dot{x}_i = \dot{x}_i(z, \dot{y}_i)$ into $\dot{X}_i = \Upsilon_i(z, \dot{x}_i, \dot{y}_i)$
to obtain $\dot{X}_i = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{i-1}, \dot{x}_i)^T = \Upsilon_i(z, \dot{y}_i)$.

Step 5: If $r_i = n_i$ then STOP.

Step 6: Substitute $\dot{x}_i = \dot{x}_i(z, \dot{y}_i)$ into $F_i^2(z, \dot{x}_i, \dot{y}_i) = 0$ to obtain $G_i(z) = 0$
according to proposition 2.1.1.

Step 7: Differentiate $G_i(z) = 0$ with respect to time:
$(G_i)_{X_i}\, \Upsilon_i + (G_i)_{y_i}\, \dot{y}_i = 0$.

Step 8: Substitute $\dot{X}_i = \Upsilon_i(z, \dot{y}_i)$ into the result of **Step 7** to obtain:

$$F_{i+1}(z, \dot{y}_i) = 0 .$$

Step 9:

$$\begin{aligned} \tilde{z}_{i+1} &= y_i , \\ n_{i+1} &= n_i - r_i , \\ i &\rightarrow i+1 . \end{aligned}$$

Step 10: Restart from **Step 1.**

On termination of the algorithm $i = \nu$, where ν denotes the index of the DAE and r contains *the number of degrees of freedom of the DAE* which will be defined in 2.1.7. For both, *linear time invariant and linear time variant,* systems it has been shown [25,43] that solvability actually is a *necessary and sufficient* condition for the termination of the algorithm. Since in Chapter 4 we are going to carry out the algorithm *structurally* . It is important to note that this does not restrict the applicability of this algorithm on linear systems. This is true since after the introduction of a *structural algebra* in Chapter 4 we will show that the nonlinear system and its corresponding linearized representation have the *same structural index.* Thus, if *Algorithm 2.1.1* is carried out by using structural computations then it yields identical results for the nonlinear system and its corresponding linearized representation. This means that for structural considerations it is completely sufficient to know that *Algorithm 2.1.1* terminates with consistent results for (structurally) solvable linear systems. "Consistent results" in this context means that the results obtained from a structural representation of *Algorithm 2.1.1* are consistent with the relaxed structural definitions of the index and further properties that can be determined by the algorithm.

Before the meaning of steps and notations in *Algorithm 2.1.1* is illustrated by a small example it should be emphasized that this algorithm can in principle be used for the most general class of DAE's $F(z, \dot{z}, u) = 0$. However, due to highly nonlinear equations it might be possible that **Step 3** cannot be carried out by symbolical manipulations but can only be accomplished *locally* using the implicit function theorem, i.e., for highly nonlinear DAE's the algorithm can only be used to determine the *local index* at points along a trajectory of a solution. However, it will be shown in Chapter 4 this does not affect the computation of the *structural index* by this algorithm. For *linear implicit* DAE's

$$M(z,u)\,\dot{z} \;+\; h(z,u) \;=\; 0 \tag{2.28}$$

Step 3 just involves a matrix inversion which always can be done symbolically. Thus for this class of DAE's it is theoretically possible to carry out *Algorithm 2.1.1* symbolically. The following simple example belongs to this class, it is even more special since M is a constant matrix in this case.

Example 2.1:
Consider the DAE:

$$\dot{z}_1 - f_1(z_1, z_2, z_3) \;=\; 0\,, \tag{2.29}$$

$$\dot{z}_2 - f_2(z_1, z_2, z_3) \;=\; 0\,, \tag{2.30}$$

$$z_1 + z_2 \;=\; 0\,. \tag{2.31}$$

The initialization of *Algorithm 2.1.1* (**Step 0**) yields:

$$i = 0,\ n_0 = 3,\ r = 3,\ \tilde{z}_0 = (z_1, z_2, z_3)^T\,,$$

$$F_0 \;=\; \begin{pmatrix} \dot{z}_1 - f_1(z_1, z_2, z_3) \\ \dot{z}_2 - f_2(z_1, z_2, z_3) \\ z_1 + z_2 \end{pmatrix}, X_{-1} \text{ being a "vector " with no elements .}$$

Carrying out **Steps 1–6** one obtains the algebraic relation to be differentiated which in this case is:

$$G_0(z) = z_1 + z_2 = 0. \tag{2.32}$$

Differentiation of this with respect to time t and substitution of

$$\dot{X}_0 = \dot{x}_0 = (\dot{z}_1, \dot{z}_2)^T = (f_1(z_1, z_2, z_3), f_2(z_1, z_2, z_3))^T \tag{2.33}$$

(**Steps 7–8**) yields the new (algebraic) equation

$$F_1 = f_1(z_1, z_2, z_3) + f_2(z_1, z_2, z_3) = 0 \tag{2.34}$$

Since $\dot{z}_3$ could not have been determined yet

$$\tilde{z}_1 = z_3\ ,\ \ n_1 = 3 - 2 = 1\ ,\ \ i = 0 \rightarrow i = 1\ . \tag{2.35}$$

are set in **Step 9** before going back to **Step 1**.

Since $\frac{\partial F_1}{\partial \dot{z}_1} \equiv 0$ after the restart **Steps 1–6** will trivially yield:

$$\begin{gathered} F_1^1 \equiv 0\ ,\ \ x_1 \equiv 0\ , \\ F_1^2 = f_1(z_1, z_2, z_3) + f_2(z_1, z_2, z_3) = 0\ ,\ \ y_1 = z_3\ , \\ \dot{X}_1 \equiv \dot{X}_0 = (\dot{z}_1, \dot{z}_2)^T\ , \end{gathered} \tag{2.36}$$

and the further algebraic equation

$$G_1(z) \equiv F_1^2 = f_1(z_1, z_2, z_3) + f_2(z_1, z_2, z_3) = 0\ . \tag{2.37}$$

Differentiation of $G_1(z) = 0$ with respect to time leads to:

$$\sum_{i=1}^{2} \left(\frac{\partial f_1}{\partial z_i} + \frac{\partial f_2}{\partial z_i}\right) \dot{z}_i + \left(\frac{\partial f_1}{\partial z_3} + \frac{\partial f_2}{\partial z_3}\right) \dot{z}_3 = 0\ . \tag{2.38}$$

which after substitution of $\dot{z}_1 = f_1(z_1, z_2, z_3)$, $\dot{z}_2 = f_2(z_1, z_2, z_3)$ can be rewritten (**Step 8**):

$$F_2(z, \dot{z}_3) = 0\ . \tag{2.39}$$

Before restarting with **Step 1** the following assignments have to be made in order to match the notation in *Algorithm 2.1.1*:

$$\tilde{z}_1 = z_3\ ,\ \ n_2 = 1 - 0 = 1\ ,\ \ i = 1\ \rightarrow\ i = 2\ .$$

Under the assumption that the DAE is solvable

$$\frac{\partial F_2}{\partial \dot{\tilde{z}}_1} = (\frac{\partial f_1}{\partial z_3} + \frac{\partial f_2}{\partial z_3}) \neq 0 \tag{2.40}$$

holds[2] and implies $r_2 = 1$ (**Step 1**).

Carrying out **Steps 2–4** one obtains:

$$\dot{z}_3 \ = \ \bar{f}_3(z_1, z_2, z_3) \tag{2.41}$$

$$\dot{X}_2 \ = \ (\dot{z}_1, \dot{z}_2, \dot{z}_3)^T \ = \ \Upsilon_2(z_1, z_2, z_3) \tag{2.42}$$

The algorithm terminates since all components of $\dot{z}$ are determined such that $r_2 = n_2 = 1$. Since $i = 2$ on termination the index ν of this DAE is two. In Section 2.1.7 it will be shown that r is the number of arbitrary initial conditions in order to allow consistent initialization. In this example there is one arbitrary initial condition since $r = 1$ on termination of the algorithm.

Example 2.1 illustrates that X_i denotes the set of variables z_i whose time derivative $\dot{z}_i$ could be determined as a function of $z, \dot{y}_i, t$, where $\dot{y}_i \in \dot{z}$ but $y_i \ni X_i$, after i differentiations. Thus, for $i = \nu$, where ν denotes the index of the DAE, all components of $\dot{z}_i$ could be determined as functions of z, t according to *Definition 2.1.3*, i.e.,

$$X_\nu \ \equiv \ z\ ,\quad \dot{X}_\nu \ \equiv \ \dot{z}\ ,\quad y_\nu \ \equiv \ 0\ . \tag{2.43}$$

In the upcoming Sections 2.1.6 and 2.1.7 further properties of DAE's are defined based on the notations of *Algorithm 2.1.1*.

[2] A necessary solvability condition which implies $(\frac{\partial f_1}{\partial z_3} + \frac{\partial f_2}{\partial z_3}) \neq 0$ for this DAE to be solvable is given in Section 2.2.2.

2.1.6 Hidden Equations and Extended System

By carrying out *Algorithm 2.1.1* equations are derived which do occur explicitly in the original formulation of the DAE $F(z, \dot{z}, u) = 0$ but are satisfied implicitly. This motivates

Definition 2.1.6: *The equations*

$$\dot{x}_i = \dot{x}_i(z, \dot{y}_i), \quad i = 1(1)\nu \tag{2.44}$$

$$0 = G_i(z), \quad i = 1(1)\nu - 1 \tag{2.45}$$

which are derived by applying Algorithm 2.1.1 on a general DAE and cannot be obtained from its original formulation $F(z, \dot{z}, u) = 0$ of the DAE by means of algebraic manipulations are called hidden equations.

This means that equations $G_0 = 0$ which can be obtained without differentiations are not considered hidden equations although they might not occur in the original representation of the DAE. This becomes clearer by looking at

Example 2.2:

$$\dot{z}_1 = f_1(z_1, z_2, z_3), \tag{2.46}$$

$$\dot{z}_1\, \dot{z}_2 = f_2(z_1, z_2, z_3), \tag{2.47}$$

$$\dot{z}_1 = f_3(z_1, z_2, z_3). \tag{2.48}$$

Using *Algorithm 2.1.1* **Steps 1–6** yield:

$$\dot{z}_1 = f_1(z_1, z_2, z_3), \tag{2.49}$$

$$\dot{z}_2 = \frac{f_2(z_1, z_2, z_3)}{f_1(z_1, z_2, z_3)}, \tag{2.50}$$

$$0 = f_3(z_1, z_2, z_3) - f_1(z_1, z_2, z_3) =: G_0(z). \tag{2.51}$$

This shows that this system is of index one since after one differentiation of $G_0(z) = 0$ with respect to time $\dot{z}_3 = \dot{z}_3(z_1, z_2, z_3)$ can be computed and represents the only hidden equation of this DAE.

Another definition motivated by *Algorithm 2.1.1* is

Definition 2.1.7: *The joint system of the original equations of the DAE* $F(z, \dot{z}, u) = 0$ *and the hidden equations obtained by applying Algorithm 2.1.1 on those is called the corresponding extended system of the DAE.*

There is no general relation between the index ν and the dimension n of the DAE which would allow to compute the number q of hidden equations without actually deriving the corresponding extended system by use of *Algorithm 2.1.1.* . The following proposition and the geometrical illustration in Section 2.1.8 indicate why this number q is of interest.

Proposition 2.1.3: *The corresponding extended system of the DAE* $F(z, \dot{z}, u) = 0$ *can in general at least locally be represented in the form*

$$\dot{z} = \phi(z, u) \tag{2.52}$$

$$0 = K(z, u) \tag{2.53}$$

where $\phi \in \mathcal{R}^n$, $K \in \mathcal{R}^q$.

Proof: The limitation on local considerations for some highly nonlinear DAE's is due to **Step 3** of *Algorithm 2.1.1* . For linear implicit systems however, this step can always be carried such that in this case this form can actually be obtained by symbolical manipulations.

It is not obvious that the number of algebraic equations in the extended system is equal to the number q of hidden equations. This can be verified by observing that each hidden equation of the type

$$\dot{x}_i = \dot{x}_i(z, \dot{y}_i) , \qquad i = 1(1)\nu \tag{2.54}$$

corresponds to an algebraic equation of the set

$$0 = G_{i-1}(z) , \qquad i = 1(1)\nu \tag{2.55}$$

before the i^{th} differentiation. For a DAE $F(z, \dot{z}, u) = 0$, $F, z \in \mathcal{R}^n$ with $rank\frac{\partial F}{\partial \dot{z}} = r_0$ there have to be $q_d = n - r_0$ hidden equations of type $\dot{x}_i = \dot{x}_i(z, \dot{y}_i)$, $i = 1(1)\nu$ in order to obtain each component of $\dot{z}$ as a continuous function of z, t. The formulation $\dot{z} = \phi(z, u(t))$ can be chosen since on termination of the algorithm all dependencies on $\dot{y}_i$ have been replaced by dependencies on z, t due to **Step 4**. $rank\frac{\partial F}{\partial \dot{z}} = r_0$ also implies $G_0 \in \mathcal{R}^{n-r_0}$ which can be seen from the argument preceeding *Example 2.2* and by examining **Steps 1–6** of the algorithm for $i = 0$. Thus, the number of algebraic equations which can be obtained from the original formulation of the DAE without differentiations equals q_d. If now it is assumed that q_a *algebraic* equations can be obtained by differentiations and substitutions, i.e., if

$$(G_1, \ldots, G_{\nu-1})^T \in \mathcal{R}^{q_a} \tag{2.56}$$

is assumed, then the final number q of hidden equations is

$$q = q_a + q_d . \tag{2.57}$$

Since there are q_d algebraic equations $G_0(z) = 0$ which are not considered hidden equations because they can derived by solely algebraic manipulations of the original equations this is also the number of algebraic equations on termination of *Algorithm 2.1.1.* □

This proposition has some practical use for one way of achieving consistent initialization, which will be dealt with in Section 2.1.7.

A further rather basic property of DAE's can be found by observing that no solvable DAE of dimension n can hide more than n independent equations, i.e., $q \leq n$. This is true since otherwise, the corresponding extended system would be overdetermined with respect to $z, \dot{z}$ and there would not be a solution. This on the other hand implies

Proposition 2.1.4: *For a solvable DAE* $F(z, \dot{z}, u) = 0$, $z, F \in \mathcal{R}^n$, $u \in \mathcal{R}^p$ *the index* ν *always satisfies* $\nu \leq n$.

Proof: If after each differentiation of the original and previously found hidden equation one new hidden equation is obtained then the system would not be solvable if one would have to differentiate more than n times. This is true since by assumption each differentiation yields one new hidden equation. Thus, after n differentiations n hidden equations are obtained and the corresponding the extended system consists of $2n$ equations in $2n$ unknowns. If a further differentiation would yield a new equation then the extended system would be overdetermined and thus, not solvable.
If one differentiation yields $k > 1$ hidden equations then there cannot be more than $(n-k)$ further differentiations each of which could supply just one more hidden equation in order to avoid the explicit system being overdetermined. This means that in this case $\nu \leq (n-k)$. □

2.1.7 Consistent Initialization and Degrees of Freedom

As indicated in the preceeding section the extended system has some practical use if the state of the system at initial time t_0 has to be determined.

Definition 2.1.8a: *The vectors z_0, $\dot{z}_0$ are called consistent initial conditions of the DAE $F(z, \dot{z}, u) = 0$ at t_0 if they satisfy the corresponding extended system at t_0.*

An alternate definition of consistent initial conditions is given in [8]:

Definition 2.1.8b: *A complete initial vector $(z_0, \dot{z}_0)$ at initial time t_0 which admits a smooth solution of $F(z, \dot{z}, u) = 0$ is called a set of consistent initial conditions.*

Applying *Algorithm 2.1.1* on systems of index>0 it is generally required to carry out differentiations. Due to these differentiations it might happen that derivatives of $u(t)$ occur in the equations of the extended system. Since any solution of the DAE always satisfies the extended system the extended system cannot contain non–smooth higher order derivatives of $u(t)$ if the solution ought to be smooth. This indicates that the two above are equivalent if $u(t)$ is sufficiently differentiable in order to allow a smooth solution.

At initial time t_0 the extended system

$$\dot{z}_0 = \phi(z_0, u(t_0)) \tag{2.58}$$

$$0 = K(z_0, u(t_0)) \tag{2.59}$$

where $\phi \in \mathcal{R}^n$, $K \in \mathcal{R}^q$, can be considered an *algebraic system* of $n + q$ equations in $2n$ unknowns which are the initial values $z_0, \dot{z}_0$. If $r = n - q$, $0 \leq r \leq n$ then in order to solve this underdetermined algebraic system r of those variables must be considered "parameters" and have to be assigned certain values. This motivates

Definition 2.1.9: *Variables which can be assigned arbitrary initial conditions and still allow consistent initialization are called degrees of freedom.*

A geometrical motivation of the term "degree of freedom" is given in Section 2.1.8

2.1.8 Geometrical Illustration

After the introduction of various in the preceeding sections they are related by a geometrical illustration.

Formally, the corresponding extended system of a DAE can be considered a system of n ordinary differential equations (ODE) $\dot{z} = \phi(z, u)$ whose solution has to satisfy $q = n - r$ additional (algebraic) constraints $K(z, u) = 0$. These constraints can be viewed as a $(n-r)$-dimensional hyper plane in n-dimensional state space. Any feasible solution of the DAE can only lie inside this hyper plane. Looking at the extended system one observes that at some time t in general all derivatives $\dot{z}_i^*(t), i = 1, .., n$ of a solution $z^*(t)$ can be determined if the solution is known. This is also true for the initial state at initial time $t = t_0$. A feasible solution at initial time has to lie inside the hyper plane as well. Since the n equations (2.58) are going to be used to determine the initial slopes of the system, i.e., the starting directions of the solution $z^*(t)$, only $(n - r)$ equations remain for the computation of the n variables $z_i^*(t_0), i = 1, .., n$. Since this is an underdetermined system of nonlinear equations r of those variables can be assigned free arbitrary initial values. Going back to the illustration with the hyper plane this means that the starting point of the solution on the hyper plane can be chosen by picking r initial values arbitrarily, which motivates to call this a "DAE with r degrees of freedom". If r_s initial slopes $\dot{z}_{0_i}$ are specified arbitrarily then only $r_c = r - r_s$ coordinates z_{0_i} of the starting point can also be assigned arbitrary values. In geometrical terms this mean that the hyper plane has to be shaped such that a solution through a point of the specified coordinates and initial slopes lies inside the plane if consistent initialization is desired. This brief qualitative argument indicates that in general consistent initialization is more difficult if instead of picking r arbitrary initial states z_0 a combination of components of z_0 and $\dot{z}_0$ is assigned

arbitrary initial values.

For simple systems it is sometimes possible to relate the free initial conditions to storage of mass or energy. The arbitrary initial conditions can then be interpreted as initial loads of these storages. However, in chemical engineering it is frequently very hard to make such a physical interpretation since there are many complicated relations and complex crosslinks among variables. This makes it hard to assign concrete pairs of variables and associated storages.

The argument above is rather qualitative. However, it shows that the *number of degrees of freedom* r of a DAE is unique and an important characteristic.

The significance of consistent initialization and how it can be accomplished will be pointed out in section 2.3.

The geometrical illustration can also be used to examine the vivid meaning of "Solvability" according to *Definition 2.1.2.* For this purpose the four conditions given in this definition are restated form the geometrical point of view:

1. $\phi(t,c)$ *is defined on all of* $\mathcal{I}$ *for each* $c \in \tilde{\Omega}$:

   For a (feasible) set of free initial conditions $c \in \tilde{\Omega} \subset \mathcal{R}^r$ there exists a solution $z(t) = \phi(t) \in \mathcal{R}^n$ for all $t_0 \leq t \leq T$.

2. $(t, \phi(t,c), \dot{\phi}(t,c)) \in \Omega$ *for* $(t,c) \in \mathcal{I} \times \tilde{\Omega}$:

   The solution lies inside the hyper plane which is defined implicitly by the subset Ω.

3. *If* $\psi(t)$ *is any other solution with* $(t, \psi(t), \dot{\psi}(t)) \in \Omega$, *the* $\psi(t) = \phi(t,c)$ *for some* $c \in \tilde{\Omega}$:

   Each arbitrarily specified starting point on the hyper plane is the starting point of exactly one solution. Considering trajectories of solutions this means that according to this definition systems which have solutions whose trajectories branch

at some points due to multiple solutions of nonlinear equations at those points are *not* solvable. This kind of bifurcation which is different from bifurcations known from ODE's is discussed in more detail in Appendix B.

4. *The graph of ϕ as a function of (t, c) is an $(r+1)$-manifold.*
   Since the initial time t_0 can be picked additionally to the values of r arbitrary initial conditions $(r+1)$ values have to be specified for an initial point. Frequently $t_0 = 0$ is chosen.

2.2 Solvability Conditions

2.2.1 Linear Systems

For *linear constant coefficient* DAE's

$$M\,\dot{z} + L\,z + h(t) = 0 \tag{2.60}$$

with $h(t) \equiv E\,u(t)$ to shorten notation, the definition of solvability and the actually necessary and sufficient solvability condition

$$\det(\lambda M + L) \neq 0 \tag{2.61}$$

for all but a finite set of values λ_i, are practically identical. The solvability condition above has been indicated by several authors [13,48]. In the remainder of this section it is shown how this condition determines solvability and what it means practically if this condition is violated. Since (2.60) is a linear constant coefficient DAE the Laplace transform may be applied to it:

$$\mathcal{L}: \qquad (sM + L)\,Z(s) = -H(s) \tag{2.62}$$

This inhomogeneous linear system is known to have a unique solution if and only if condition (2.61) holds. If it is violated just for some singular values s_i, $i = 1..k$,

of s then there is still a solution for $z(t)$ since the inverse Laplace transform can be performed by using the residual theorem :

$$\mathcal{L}^{-1}: \quad z(t) = \oint_C Z(s)\, ds = \sum_{i=1}^{k} \mathrm{Res}\{Z(s), s_i\} \tag{2.63}$$

where C is a closed curve surrounding an area in the complex plain which contains all singular points s_i, $i = 1, .., k$ and

$$Z(s) = -(sM + L)^{-1} H(s)\,, \tag{2.64}$$

$$\mathrm{Res}\{Z(s), s_i)\} = \frac{1}{(m_i - 1)!} \lim_{s \to s_i} \{\frac{d^{m_i - 1}}{ds^{m_i - 1}} (s - s_i)\, Z(s)\}\,, \tag{2.65}$$

where m_i denotes the multiplicity of pole s_i in $det(sM + L) = 0$. The computation above is only possible if $(sM+L)^{-1}$ exists for all s but the singular points s_i, $i = 1, .., k$, i.e. $det(sM + L) = 0$ may only occur for the finite set of poles s_i, $i = 1, .., k$. For $s = -\lambda$ this is obviously the same condition as in *Definition 2.1.1.*

In order to obtain a practical interpretation of the solvability condition (2.61)*Definitions 2.2.1 and 2.2.2* as well as *Proposition 2.2.1* are introduced below.

Definition 2.2.1: *The matrices $L, M \in \mathcal{R}^{n \times n}$ have the same linear dependencies if one or both the statements below hold:*

1. *There is at least one column index l such that*

$$\mathrm{col}_l L = \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i\, \mathrm{col}_i L \tag{2.66}$$

$$\mathrm{col}_l M = \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i\, \mathrm{col}_i M \tag{2.67}$$

$$\alpha_i = \text{constant}\,, \quad i = 1, .., n\,, \quad i \neq l. \tag{2.68}$$

2. *There is at least one row index m such that*

$$\mathrm{row}_l L = \sum_{\substack{i=1 \\ i \neq m}}^{n} \beta_i\, \mathrm{row}_i L \tag{2.69}$$

$$\text{row}_l M = \sum_{\substack{i=1 \\ i \neq m}}^{n} \beta_i \, \text{row}_i M \tag{2.70}$$

$$\beta_i = \text{constant}\ , \quad i = 1,..,n\ , \quad i \neq m. \tag{2.71}$$

Definition 2.2.2: *The linear time invariant DAE* $M\dot{z} + Lz + h(t) = 0$ $h, z \in \mathcal{R}^n$, $L, M \in \mathcal{R}^{n \times n}$ *consists of*

a. n linearly independent equations if M, L *do not have the same linear dependencies,*

b. linearly dependent equations if M, L *do have the same linear dependencies.*

Proposition 2.2.1: *A linear time invariant DAE* $M\dot{z} + Lz + h(t) = 0$, $h, z \in \mathcal{R}^n$, $L, M \in \mathcal{R}^{n \times n}$ *which contains linearly dependent equations is not solvable.*

Proof: Assume:

$$\text{col}_l L = \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i \, \text{col}_i L \tag{2.72}$$

$$\text{col}_l M = \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i \, \text{col}_i M \tag{2.73}$$

$$\alpha_i = \text{constant}\ , \quad i = 1,..,n\ , \quad i \neq l. \tag{2.74}$$

With this assumption the l^{th} column of $(\lambda M + L)$ is a linear combination of the remaining columns in this matrix:

$$\text{col}_l(\lambda M + L) = \lambda \, \text{col}_l M + \text{col}_l L \tag{2.75}$$

$$= \lambda \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i \, \mathrm{col}_i M + \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i \, \mathrm{col}_i L \tag{2.76}$$

$$= \sum_{\substack{i=1 \\ i \neq l}}^{n} \alpha_i \, \mathrm{col}_i (\lambda M + L) \tag{2.77}$$

Thus, $det(\lambda M + L) \equiv 0$ for all λ and the DAE is not solvable according to *Definition 2.1.1.*

If same linear dependencies of rows in M, L occur then an analogous argumentation can be used for showing $det(\lambda M + L) \equiv 0$ for all λ. □

An illustration of nonsolvability due to row defects in $(\lambda M + L)$ is the following simple example:

Example 2.3:

$$\dot{x} - y + h_1(t) = 0 \tag{2.78}$$

$$\dot{x} - y + h_2(t) = 0 \tag{2.79}$$

With the notation above:

$$n = 2 \,, \quad z = (x, y)^T \,, \quad \dot{z} = (\dot{x}, \dot{y})^T \tag{2.80}$$

$$M = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \,, \quad L = \begin{pmatrix} 0 & -1 \\ 0 & -1 \end{pmatrix} \,, \quad h(t) = \begin{pmatrix} h_1(t) \\ h_2(t) \end{pmatrix} \tag{2.81}$$

In this case

$$\lambda M + L = \begin{pmatrix} \lambda & -1 \\ \lambda & -1 \end{pmatrix} \tag{2.82}$$

where both rows are identical, which could be expressed by $\beta_1 = 1$ using the notation of *Definition 2.2.1.* Obviously, $det(\lambda M + L) \equiv 0$. The fact that this system is not solvable can also be seen from a different perspective:

In order to avoid contradictions which would make the system unsolvable without even looking at $det(\lambda M + L)$, $h_1(t) \equiv h_2(t)$ has to be satisfied. However, this means that (2.78), (2.79) really contains only one equation for two variables x, y. Thus, there are not enough (independent) equations .

Another example illustrates unsolvability due to a column deficiency of the pencil $(\lambda M + L)$:

Example 2.4:

$$\dot{v} + x + y + h_1(t) = 0 \tag{2.83}$$

$$x + y + h_2(t) = 0 \tag{2.84}$$

$$v + h_3(t) = 0 \tag{2.85}$$

With the notation above:

$$n = 3\,, \quad z = (v, x, y)^T\,, \quad \dot{z} = (\dot{v}, \dot{x}, \dot{y})^T \tag{2.86}$$

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\,, \quad L = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}\,, \quad h(t) = \begin{pmatrix} h_1(t) \\ h_2(t) \\ h_3(t) \end{pmatrix} \tag{2.87}$$

In this case

$$\lambda M + L = \begin{pmatrix} \lambda & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{2.88}$$

where the second and third column are identical, which could be expressed by $\alpha_1 = 0, \alpha_2 = 1$ using the notation of *Definition 2.8.* Again, $det(\lambda M + L) \equiv 0$. Approaching this system in a different way one observes that it actually represents a system for two variables $v, (x + y)$. Thus, even if requirements for $h_1(t), h_2(t), h_3(t)$, which would avoid contradictions, are met, it is impossible to compute x, y separately, since the system only provides equations to determine the sum of those, $(x + y)$.

2.2.2 General DAE's

In [8] a linear implicit DAE

$$M(z,t)\dot{z} = h(z,t) \tag{2.89}$$

is *defined* to be solvable *at some point* z_0 if

$$\det(\lambda\bar{A}(t_0) + B_1(t_0) \neq 0 \tag{2.90}$$

at this point for almost every λ, where

$$\bar{A}(t) := M(z_0,t) \tag{2.91}$$

$$B_1(t) := -h_z(z_0,t) + A_z^{\mathcal{T}}(z_0)\dot{z}_0 \tag{2.92}$$

$A_z^{\mathcal{T}}(z_0)$ denotes a 3-dimensional tensor which consists of the partial derivatives of the elements of matrix M with respect to the components of z. This is motivated by expanding the system about some point $z_0, \dot{z}_0, t$ and then lumping explicit time dependencies into a function G(t) to obtain:

$$\bar{A}(t)\dot{z} + B_1(t)z = G(t) \tag{2.93}$$

Thus, he considers the DAE *locally* and defines solvability *locally*. Considering this along with the solvability condition for linear constant coefficient DAE's in the previous section one would like to relate solvability of a general DAE $F(z,\dot{z},u) = 0$ to the matrix pencil $(\lambda F_{\dot{z}} + F_z)$.

However, for nonlinear DAE's this pencil can change significantly along the trajectory of a solution $z^*(t)$ since it generally depends on $(z,\dot{z},t)$. Thus, in order to evaluate this pencil the solution has to be known. It is in general not appropriate to consider the corresponding linearized system

$$F_{\dot{z}}|_*\,(\dot{z} - \dot{z}^*) \;+\; F_z|_*\,(z - z^*) \;+\; F_t|_*\,(t - t^*) \;=\; 0 \tag{2.94}$$

in the vicinity of a solution $z^*(t)$ at time $t = t^*$ to check the nonlinear system on solvability[3] [5, p.31].

A local approach also does not seem to be practical for an a priori check on solvability since it would require to check a solvability condition along a solution which obviously is not known at this stage. Thus, we confine ourselves to a rather crude condition which can be shown to be sufficient to rule a DAE *globally unsolvable*, which means that systems that do not have a solution at any point in state space can be detected and excluded from further considerations by checking this condition.

Proposition 2.2.2a: *A sufficient condition for the DAE* $F(z, \dot{z}, u) = 0$ *to be not solvable is*

$$det(\lambda F_{\dot{z}} + F_z) \equiv 0 \tag{2.95}$$

for all $z, \dot{z} \in \mathcal{R}^n$, *all* , $\lambda \in \mathcal{R}$, *and all* $t \geq 0$.

A formal proof of this using further propositions introduced in this section is given in Appendix A. *Proposition 2.2.2a* can be reformulated to obtain

Proposition 2.2.2b: *A necessary condition for the DAE* $F(z, \dot{z}, u) = 0$ *to be solvable is that* $det(\lambda F_{\dot{z}} + F_z)$ *does not vanish identically for all* $z, \dot{z} \in \mathcal{R}^n$, *all* , $\lambda \in \mathcal{R}$, *and all* $t \geq 0$.

In order to illustrate, why the necessary solvability condition formulated in the equivalent propositions above holds, we have to examine what its implications are. As a first step towards that, the concept of linearly dependent equations is generalized for nonlinear equations $F_i(z, \dot{z}, u) = 0$, $i = 1(1)n$ forming a general DAE.

[3]Campbell [8] could do this since he defined solvability locally rather than deriving a condition for solvability according to *Definition 2.1.2.*

Definition 2.2.3: *An equation* $F_i(z, \dot{z}, u) = 0$ *in the DAE* $F(z, \dot{z}, u) = 0$ *is called redundant if it is automatically satisfied if the remaining equations of the DAE are satisfied.*

Thus, the most general form of a redundant equation is

$$F_i(z, \dot{z}, u) \equiv \Psi(F_1, F_2, \ldots, F_{i-1}, F_{i+1}, \ldots, F_n) = 0 \tag{2.96}$$

with Ψ being some function of $n-1$ arguments and and the additional property:

$$\Psi(0, 0, \ldots, 0) = 0 \tag{2.97}$$

If $F(z, \dot{z}, u) = 0$ happens to be a linear system then

$$\Psi(F_1, F_2, \ldots, F_{i-1}, F_{i+1}, \ldots, F_n) \equiv \sum_{\substack{j=1 \\ j \neq i}} \alpha_j F_j(z, \dot{z}, u) = 0 \tag{2.98}$$

is the most general representation of a redundant equations, with $\alpha_j, j = 1(1)n$, $j \neq i$, being constants. This explains why redundant equations can also be called *dependent equations. Definition 2.2.3* is consistent with the meaning of the term *linearly dependent equations* as it is used in matrix theory [28].

The next proposition relates the occurrence of redundant equations and and the priorly introduced extended system

Proposition 2.2.3: *The corresponding extended system of a DAE* $F(z, \dot{z}, u) = 0$ *with* $1 \leq \mu \leq n-1$ *redundant equations is of the form*

$$\dot{z}_1 = \phi(z, \dot{z}_{n-\mu+1}, \ldots, \dot{z}_n)\,, \tag{2.99}$$

$$\vdots \tag{2.100}$$

$$\dot{z}_{n-\mu} = \phi(z, \dot{z}_{n-\mu+1}, \ldots, \dot{z}_n)\,, \tag{2.101}$$

$$0 = \bar{K}(z)\,, \tag{2.102}$$

where $\bar{K} \in \mathcal{R}^{\bar{q}}$, and $\bar{q}$ denotes the number of algebraic equations after k differentiations in Algorithm 2.1.1.

Essentially, this means that not even locally all components of $\dot{z}$ can be determined as continuous functions of z, t. This is derived in Appendix A by applying *Algorithm 2.1.1* on a DAE with μ redundant equations.

A consequence of *Proposition 2.2.3* is that being at some point of the trajectory of a solution $z^*(t)$ of the DAE at time $t = t_1$, i.e., $z = z^*(t_1)$ it is not possible to determine how this trajectory is going to continue for $t > t_1$. This is due to the fact that the components $\dot{z}_{n-\mu+1}, \ldots, \dot{z}_n$ of $\dot{z}$ are not determined at this point even though the point itself is known. As a consequence it can be seen that $\dot{z}_1, \ldots, \dot{z}_{n-\mu}$ are not determined either, what follows from *Proposition 2.2.3.* This means that the entire derivative $\dot{z}(t_1)$ which determines the propagation of the trajectory cannot be computed.

For highly nonlinear systems it might be possible that *Algorithm 2.1.1* can only be carried out locally. If there are points where the equations become locally redundant due to numerical peculiarities then by the same argument as above it is not possible to state anything about the propagation of a trajectory from those points on. Thus, those points can be called *singular points* . It is generally very difficult to state anything about the behavior of solutions at or after those points. One possibility is that such a point gives rise to a *bifurcation of a trajectory,* e.g., the trajectory splits into several more branches after running into such a singular point[4]

These considerations can be summarized by

Proposition 2.2.4: *A DAE which contains redundant equations is not solvable ac-*

[4]As indicated in Section 2.1.8 systems whose solutions can show this type of bifurcation are not solvable according to *Definition 2.1.2.*

cording to Definition 2.1.2.

What remains to be shown is how all this relates to the matrix pencil $(\lambda F_{\dot{z}} + F_z)$, or in other words how this leads to *Propositions 2.2.2a/b*. As shown in Appendix A

$$det(\lambda F_{\dot{z}} + F_z) \equiv 0 \tag{2.103}$$

for all $z, \dot{z} \in \mathcal{R}^n$, all , $\lambda \in \mathcal{R}$, and all $t \geq 0$ implies that $F(z, \dot{z}, u) = 0$ either contains redundant equations and therefore is not solvable by *Proposition 2.2.4* , or it contains a set of variables $z_1, \ldots, z_\rho$, $1 \leq \rho \leq n$, which occurs in identical terms in all equations such that those terms can be replaced by a lumped variable

$$\sigma = \sigma(z_1, \ldots, z_\rho, \dot{z}_1, \ldots, \dot{z}_\rho) \tag{2.104}$$

in all equations. In such a case generally those components $z_i(t)$, $i = 1(1)\rho$ of $z(t)$ cannot be determined from such a system, and therefore there is no complete solution which specifies *all* components of $z(t)$. The reason for this can be seen from the linear case in *Example 2.4* and from a more detailed discussion in Appendix A.

Summarizing it can be said that solvability according to *Definition 2.1.2* excludes systems that allow bifurcations of trajectories by requiring that the third condition in this definition has to be satisfied by solutions of systems which are solvable in this sense. The solvability condition formulated in the equivalent *Propositions 2.2.2a/b* allow singular points and bifurcations of trajectories. However, they rule DAE's for which all points $z, \dot{z} \in \mathcal{R}^n$, at all $t \geq 0$ are singular points unsolvable. This is important, since for those system the corresponding explicit system cannot be derived completely at any point of state space and thus, by the arguments above there is not a single point from which the propagation of a trajectory is defined.

In conclusion of the considerations above it can be stated that the solvability condition in *Propositions 2.2.2a/b* is less restrictive than required by *Definition 2.1.2.*

However, in Chapter 4 it will be indicated that the solvability condition according to *Propositions 2.2.2a/b* is only slightly more restrictive than a condition for *structural* solvability. Especially, its formulation in *Propositions 2.2.2a* turns out to be a rather practical way to check a priori if a DAE is structurally solvable. This is important to know since the structural algorithm based on *Algorithm 2.1.1* only terminates for structurally solvable DAE's.

2.3 Consistent Initialization

2.3.1 Significance of Consistent Initial Conditions

A great many of presently available numerical software for the solution of DAE's requires consistent initial values. In particular this is true for codes like SPRINT [2,3] and DASSL [40] which use *backward differentiation formulas (BDF)*. If DASSL for instance is supplied with initial values z_0 and with an initial guess for the values of $\dot{z}_0$ then it uses a small backward Euler step and a damped Newton iteration in order to determine consistent initial values of $\dot{z}_0$ from the generally nonlinear DAE $F(z, \dot{z}, u) = 0$ [10,30]. For poor initial guesses of $\dot{z}_0$ the Newton iteration may converge very slowly, may converge to an undesired solution, or may even fail to converge at all. Convergence to an undesired solution means that this solution implies crude or even non-physical assumptions.

As indicated more detailed in the next section it is only a necessary condition for consistent initial $z_0, \dot{z}_0$ conditions that they satisfy the original equations at $t = t_0$, i.e., $F(z_0, \dot{z}_0, u(t_0)) = 0$. In DASSL the calculation of $\dot{z}_0$ as described above is only optional. It is sometimes possible to start DASSL without consistent initial values of $\dot{z}_0$. However, the error estimate of the first step is not correct and may lead to successive error test failures which cause the code to stop. For the index-two-representation of the reactive distillation column considered in *Example 5.2* in Chapter 5 DASSL

converged to a non-physical solution. For a more detailed discussion of this particular example we refer to Chapter 5 and [4].

In general it can be stated that present numerical software without consistent initial conditions often fails or becomes extremely inefficient. Failure of numerical software means either convergence to a wrong solution or no convergence at all. In many instances consistent initial conditions seem to be the only reasonable choice for real physical systems since they admit smooth solutions and avoid impulse functions at initial time $t = t_0$. However, Cobb [12] pointed out that this argument for consistent initial conditions is not necessarily valid for all physical systems. To illustrate that he cites examples of electric circuits which are formed at a certain instant of time with the opening and closing of switches. The separation of different storages by open switches allows arbitrary values of the initial loads of these storages at time t_0, "immediately" after all switches were closed.

Thus, the most striking argument for the computation of consistent initial conditions is the fact that many presently available numerical codes require them for a successful and efficient solution of DAE's.

2.3.2 Computation of Consistent Initial Conditions

The actual computations of consistent initial conditions can be a difficult and lengthy computational task. Even for some index-one-systems it might not be possible to provide complete consistent initial conditions $\dot{z}_0, z_0$ without considering differentials of the original equations. This can be observed in the following small index-1-system:

Example 2.5:

$$\dot{z}_1 + \dot{z}_2 = -z_1 + h_1(t) \tag{2.105}$$

$$z_2 = h_2(t) \tag{2.106}$$

After differentiation of (2.106) and substitution in (2.105) the corresponding extended system becomes:

$$\dot{z}_1 = -z_1 - \dot{h}_2(t) + h_1(t) \tag{2.107}$$

$$\dot{z}_2 = \dot{h}_2(t) \tag{2.108}$$

$$0 = z_2 - h_2(t) \tag{2.109}$$

Clearly, at $t = t_0$ this is a system of three equations in four unknowns $z_{0_1}, z_{0_2}, \dot{z}_{0_1}, \dot{z}_{0_2}$ thus, according to *Definition 2.1.9* the number of degrees of freedom is one, i.e., one of the four initial values can be chosen arbitrarily, the remaining three are determined by (2.107)–(2.109). Thus, an only necessary condition for consistent initial conditions is that they satisfy the original equations $F(z_0, \dot{z}_0, u(t_0)) = 0$ at $t = t_0$.

Basically, there are two different approaches to provide consistent initialization:

1. numerical approach
2. analytical/symbolical approach.

In the remainder of this section different methods using these two basic ways are introduced briefly.

Numerical Approach

For linear implicit systems of the form:

$$M\dot{z} + l(z) + h(t) = 0\,, \tag{2.110}$$

where $M \in R^{n\times n}$ is a constant matrix and $z, h, l \in R^n$, Campbell [9] derived a *necessary* condition which has to be satisfied if z_0 allows consistent initialization, i.e., a set of initial conditions for z that admits a smooth solution of (2.111.). However,

since this is only a necessary condition it can only be used to check if a chosen vector z_0 represents a set of values that allows consistent initialization of (2.110). It cannot be used to compute consistent initial conditions since it is *not sufficient.*

Considering general DAE's $F(z, \dot{z}, t) = 0$[5] *Example 2.3.1* indicates that in order to provide consistent initialization it might be required to differentiate the original equations. A numerical method which basically uses the definition of the index ν and differentiates the *entire* DAE $F(z, \dot{z}, u) = 0$ ν times with respect time to is introduced in [31]:

Given a DAE $F(z, \dot{z}, t) = 0$ of index ν and some information $B(z_0, \dot{z}_0, t_0) = 0$ about the initial values of a subset of $(\dot{z}_0, z_0)^T$ they define the *consistency equations* of the DAE to be the system:

$$F(z_0, \dot{z}_0, t) = 0 \tag{2.111}$$

$$\frac{dF}{dt}(z_0, \dot{z}_0, \ddot{z}_0, t) = 0 \tag{2.112}$$

$$\vdots$$

$$\frac{d^\nu F}{dt^\nu}(z_0, \dot{z}_0, \ddot{z}_0, \ldots, z_0^{\nu+1}, t) = 0 \tag{2.113}$$

$$B(z_0, \dot{z}_0, t_0) = 0 \tag{2.114}$$

In order to allow consistent initialization $B(z_0, \dot{z}_0, t_0) = 0$ can only restrict r of the complete initial vector $(\dot{z}_0, z_0)^T$, where r denotes the number of degrees of freedom. These relations can be chosen according to physical information about the initial state of the system considered. They can also be chosen arbitrarily as long as they represent r relations which are both, independent of each other and the other consistency equations. Usually these conditions are such that they directly assign certain values to r components of $(\dot{z}_0, z_0)^T$. (2.112)–(2.114) contain higher order partial derivatives

[5] In order to shorten notation in the definition of the consistency equations the time dependent vector $u(t)$ in the previously used notation $F(z, \dot{z}, u) = 0$ is replaced by t. This replacement does not restrict generality.

of $F(z, \dot{z}, t)$ which are not known or extremely difficult to compute symbolically. In [31] these higher order partial derivatives are approximated numerically by linear combinations of $F(z, \dot{z}, t)$ evaluated at various points (of time). This allows to compute the components of $(\dot{z}_0, z_0)^T$ which are not determined by $B(z_0, \dot{z}_0, t_0) = 0$ with only computing the first partials of F.

For linear constant coefficient DAE's there is a rigorous proof that this method yields consistent initial values. For nonlinear systems there has not been an equivalent proof yet. However, it is expected that this method works for most nonlinear systems as well. However, so far no numerical experience with this approach has been reported. In order to set up the consistency equations properly the index ν and the number of degrees of freedom r has to be known. The consistency system contains
$(\nu + 1)n + r$ equations which for large which for large systems, i.e., for large n, is a very large system which has to be solved by least square techniques in order to provide the $2n$ initial values $z_0, \dot{z}_0$. As Pantelides [37] indicated it is generally only required to differentiate subsets of the original equations in order to provide consistent initialization. The structural algorithm he introduced which will be compared with our approach in Chapter 4 determines how often which sets of equations have do be differentiated. Thus, it appears to be more efficient to derive this smaller system by differentiating only subsets of $F(z, \dot{z}, t) = 0$ and then using Leimkuhler's numerical approximations for the higher order partial derivatives of F. So far this combination has not yet been examined or tested practically. However, it seems to be worthwhile doing that.

For index-one-systems there are practical methods which take less computational effort than the one introduced above. By using a small Euler step and a Newton iteration they determine consistent values of $z, \dot{z}$ at $t = t_0 + h$,where t_0 denotes the initial time and h is a small time step (for a discussion see [30]). Although from a theoretical standpoint this approach does not solve the problem of determining

consistent initial values at initial time $t = t_0$ it allows successful numerical solution (of an approximate problem) in practical applications.

Symbolical/Analytical Approach

Based on the considerations in Section 2.1 the problem of consistent initialization can be tackled in a rather rigorous way. As indicated in Section 2.1.5 it is possible to derive the corresponding extended system

$$\dot{z} = \phi(z,u) \qquad \phi \in R^n \,, \tag{2.115}$$

$$0 = K(z,u) \qquad K \in R^{n-r} \,, \tag{2.116}$$

where r denotes the number of degrees of freedom, for any system, which is solvable[6] in the sense of *Definition 2.1.2* , at least locally. Thus, for those systems it can also be derived at initial time $t = t_0$. The Jacobian J_0 of this system with respect to the initial conditions $(\dot{z}_0, z_0)^T$ which ought to be determined at $t = t_0$ can be represented by

$$J_0 = \begin{pmatrix} -I_n & \phi_z|_0 \\ 0 & K_z|_0 \end{pmatrix} \,, \tag{2.117}$$

where $J_0 \in R^{(2n-r)\times 2n}$, I_n represents the n-dimensional identity matrix, and $|_0$ denotes that these matrices are evaluated at $\dot{z} = \dot{z}_0, z = z_0, t = t_0$. If at $t = t_0$ r components $(\bar{\dot{z}}_0, \bar{z}_0)^T$ of $(\dot{z}_0, z_0)^T$ are arbitrary assigned initial values then columns in J_0 corresponding to those components can be removed to obtain the $((2n-r)\times(2n-r))$-matrix J_0^*. If $detJ_0^* \neq 0$ at $t = t_0$ then the subset $(\bar{\dot{z}}_0, \bar{z}_0)^T$ is a *feasible* subset to accomplish consistent initialization. Since after assigning free initial values to those variables (2.105),(2.106) can be solved for the remaining components of $(\dot{z}_0, z_0)^T$ due to the implicit function theorem, if $detJ_0^* \neq 0$.

Since J_0 for nonlinear systems generally depends on $\dot{z}_0, z_0, t_0$ a *symbolical* derivation

[6] For being able to derive the corresponding extended system at initial time t_0 it is sufficient that for $t = t_0$ $det(\lambda F_{\dot{z}} + F_z)$ does not vanish.

of the extended system would be required to use the method described above. As indicated before there might be problems doing that for highly nonlinear systems. However, it has been shown in Section 2.1.5 that for linear implicit DAE's

$$M(z,u)\dot{z} + h(z,u) = 0 \tag{2.118}$$

which arise quite frequently from applications in chemical engineering the symbolical derivation of the corresponding extended system is always possible. In principle the method above is possible for solvable DAE's. However, it takes a lot of computational effort to derive the corresponding system symbolically. Although this problem definitely limits the practical applicability of this approach it is still worthwhile considering. Since the extended system only depends on the DAE, i.e., the modeling equations, it could be computed as soon as the modeling equations are derived, if a symbolical derivation is possible. In fact, one could consider it a part of the complete model along with the modeling equations themselves. With this approach the extended system would not have to be derived for each simulation. Consistent initialization would only require:

1. a check if $\det J_0^* \neq 0$ for the chosen set of arbitrary initial conditions $(\bar{\dot{z}}_0, \bar{z}_0)^T$
2. a Newton iteration to determine the remaining values of $(\dot{z}_0, z_0)^T$ consistently if $\det J_0^* \neq 0$.

If only components of z_0 are assigned free initial conditions, $\bar{z}_0$ i.e., $\dim \bar{z}_0 \equiv r$, $\dim \bar{\dot{z}}_0 \equiv 0$, and if $\bar{z}_0$ is a feasible set to be assigned free initial conditions it is sufficient to determine the remaining components of z_0 from

$$K_0(z_0, u(t_0)) \;=\; 0 \tag{2.119}$$

by a Newton iteration. This is always possible since $K_z|_0$ must have a regular submatrix corresponding to those remaining components of z_0. If this is not true then

$det J_0^* = 0$ and the presupposition that $\bar{z}_0$ is a feasible choice for a set of variables that can be assigned free initial conditions is violated. After consistent initial values of z_0 are determined by solving (2.116) the computation of consistent values for $\dot{z}_0$ is straight forward using (2.115).

Concluding this section on consistent initialization it can be said that presently there is no software which solves the problem completely for general DAE's of any index. There are some implementations of practical methods for index-one-systems [30]. Furthermore, there are numerical and analytical ideas how this problem can be assessed for general DAE's of any index. All of those ideas require in some way that the index and the number of arbitrary initial conditions are known. The determination of each of those quantities is one of the goals to be achieved efficiently by the structural algorithms of Chapter 4.

Chapter 3

Concepts of Numerical Treatment for DAE's

The integration of index-one-systems is generally no more complicated than the integration of ODE's [19]. However, the numerical treatment of higher-index-systems, i.e., DAE's of index exceeding one, is significantly more difficult. For those systems there are basically three strategies of successful numerical treatment:

1. Direct solution by specifically designed higher-index-solvers.
2. Analytical transformation into an equivalent system of lower index, preferably one.
3. Modification of the mathematical model based on physical insight in order to reduce the index.

Each of the strategies above is introduced briefly in this chapter.

3.1 Direct Solution by Higher–Index–Solvers

Presently available numerical methods and codes for higher-index-systems mostly represent modified ODE-solvers. These methods such as *backward differentiation formulas* (BDF), *Runge-Kutta* (RK), and *Extrapolation* methods are generally only

capable of dealing with higher-index-systems of restricted structure and of an index frequently limited to values of three or less.

For constant stepsize h and $k < 7$ k-step BDF methods can cope with nonlinear semi-explicit systems

$$f(x, \dot{x}, y, t) = 0\,, \tag{3.1}$$

$$g(x, y, t) = 0\,, \tag{3.2}$$

where $det\frac{\partial f}{\partial \dot{x}} \neq 0$, $f, x \in \mathcal{R}^{r_0}$, $g, y \in \mathcal{R}^{n-r_0}$ of index two, if they are started with initial values which have an error of at most $O(h^k)$ with respect to consistent initial conditions. For linear time invariant systems $M\dot{z}+Lz+f(t) = 0$ k-step BDF methods with constant stepsize h and $k < 7$ converge for any index (after an initial boundary layer)[5]. Index-three-systems in *Hessenberg Form*

$$\dot{z}_1 = F_1(z_1, z_2, z_3, t)\,, \tag{3.3}$$

$$\dot{z}_2 = F_2(z_1, z_2, t)\,, \tag{3.4}$$

$$0 = F_2(z_2, t)\,, \tag{3.5}$$

where $F_1, z_1 \in \mathcal{R}^{n_1}$, $F_2, z_2 \in \mathcal{R}^{n_2}$, $F_3, z_3 \in \mathcal{R}^{n-n_1-n_2}$, can be solved by k-step BDF methods with $k < 7$ and constant stepsize h. However, the solution is highly sensitive to the accuracy of initial conditions, whose computation therefore requires much care and effort in order to obtain reliable results. The performance of BDF methods in solving DAE's is investigated more detailed in [5]. Concluding the remarks on BDF methods it should be emphasized that they require initial conditions which differ at most by $O(h^k)$ from consistent initial values. This has to be satisfied for index-one-systems as well. For index-three-systems in Hessenberg form initial values have to be consistent of even one order higher, $O(h^{k+1})$.

Stability and convergence of multi-step and RK methods is also limited to special types of DAE's. Modified RK methods, especially *singly implicit RK* (SIRK)

methods, are expected to perform very well on systems of index two. [27] give an extended overview of recent advances in applications of RK methods on higher-index-systems. They obtained a preliminary convergence result for index-three-systems in Hessenberg Form.

The investigation of extrapolation methods which can be considered as implicit RK (IRK) methods is just beginning [5]. The code LIMEX [14] implements an extrapolation of the semi-explicit Euler scheme for the solution of semi-explicit index-one-systems. A proof for the semi-explicit index-two-systems has been given recently by Lubich.

Only most recently a method which is supposed to be capable of coping with *all* higher-index-systems has been published [11]. This method involves numerical rank determinations at each time step during the integration. Such an operation tends to be ill-conditioned and is very time consuming for large systems. Despite those drawbacks this new approach appears to be worthwhile being tested on practical examples.

3.2 Analytical Transformation into Lower–Index–Systems

As indicated in the previous section currently there are only a few settled numerical methods for the direct solution of higher-index-systems. This implies that presently and at least for the near future it is required to reduce the index if it exceeds one, two, or three depending on the structure of the DAE. Therefore, the techniques introduced in this and the next section remain useful even though there are higher-index-solvers.

In a more general way the problem addressed in this section could be stated: How can equivalent representations which are easier to solve numerically be obtained from the original equations of the higher-index-system ?

Earlier we introduced the corresponding extended system, which is equivalent to the DAE it is derived from. A proposition by Gear [22] uses this approach to formulate the semi-explicit index-two-system

$$\dot{z} = \phi(z,u) + K_z^T \xi \,, \tag{3.6}$$

$$0 = K(z,u) \,, \tag{3.7}$$

by introducing the new variables $\xi \in \mathcal{R}^{n-r}$. He shows that for a solution of the original equations $\xi \equiv 0$ is always satisfied. Index-two-systems of this type can be solved by BDF and appropriate RK methods. The system (3.6),(3.7) is of the index-two-Hessenberg form.

For large systems this approach might not be desirable since the system to be solved numerically consists of $2n - r$ equations which is even more than there are in the original system $F(z, \dot{z}, u) = 0$ with $F, z \in \mathcal{R}^n$.

An approach which would yield an index-one-representation with only n equations by using only a subsystem of the extended system is motivated by the geometrical illustration in Section 2.1.8:

According to this point of view one could choose r differential equations of the extended system which would govern the movement of a solution on the $(n-r)$-dimensional hyper plane defined by $K(z,u) = 0$. This would yield an index-one-system of the form

$$\dot{z}_1 = \phi_1(z,u) \,, \tag{3.8}$$

$$\vdots$$

$$\dot{z}_r = \phi_r(z,u) \,, \tag{3.9}$$

$$0 = K(z,u) \,, \tag{3.10}$$

which could be solved by several available standard methods and codes. Both, the algebraic equations in the original formulation and the hidden algebraic constraints,

would be satisfied by a solution of this system. Obviously, there are $\binom{n}{r}$ such subsystems of the corresponding extended system. So far there are no experiments or theoretical results which could suggest an "optimal" subset or which would show that this choice does not affect the solution significantly.

For a more detailed discussion of Gear's method sketched in the beginning of this section and of other approaches such as regularizations and transformations by introduction of new variables we refer to [5,15,22].

In general it can be said that by applying *Algorithm 2.1.1* it is principally possible to reduce the index of a DAE as far as desired. However, there is need for further theoretical and experimental work which clarifies the relation between the true solution of the original higher-index-formulation and the numerical solution of its corresponding lower-index-representation.

3.3 Modification of the Mathematical Model

Quite frequently there are several possible modeling equations for certain physical phenomena. Often times higher-index-systems arise from approximations such as quasi-steady state assumptions or as a more concrete example from neglecting pressure drop over a distillation column . However, this cannot be generalized meaning that those assumption always cause higher-index-DAE's. Depending on the particular situation similar assumptions can in fact cause the index to decrease (see *Example 5.2*).

From these brief considerations it can be seen that it is of interest to know which variables and equations cause the index to exceed a certain desired limit. Once this is detected one can think about different modeling approaches for those equations which would render a system of lower index. This possibility will be illustrated more detailed by various examples in Chapter 5.

As a general guideline it should be emphasized that frequently higher-index-DAE's and the numerical problems associated with them can be avoided by using systematic and careful modeling approaches. This will also be illustrated in further detail in Chapter 5.

Chapter 4

Structural Considerations and DAE's

As indicated in previous chapters it is important to know certain properties of DAE's in order to achieve a reliable numerical solution or even to obtain a numerical solution at all. Important properties such as the index and the number of degrees of freedom can be determined by applying *Algorithm 2.1.1* on the DAE. However, as indicated in Chapter 2 it takes a lot of computational effort to carry out this algorithm symbolically. Also, for highly nonlinear systems its applicability might be limited to local considerations.

Thus, there is need for an approach that takes less effort and still provides a useful characterization of the DAE that is not limited to local validity.

As we will show in this chapter this is possible by examining the *structure* of the DAE. It will be pointed out that the "structure" of the DAE $F(z, \dot{z}, u) = 0$ can described in a useful way by examining which components of $z, \dot{z}$ occur explicitly in which equations $F_i(z, \dot{z}, u) = 0$, $i = 1(1)n$.

After introducing the idea of *structural computations* and *structural properties* of DAE's in Section 4.1 we will suggest *structural representations of Algorithm 2.1.1* in Section 4.2. This structural algorithm provides a *lower bound on the index* and an

upper bound on the number of degrees of freedom of the DAE at *any* point in state space. This means that if, for instance, the (structural) index obtained from this algorithm is two, then the local index at any point in state space is *at least* two. For practical computations this is useful to know, since as indicated in Chapter 3, the numerical treatment of higher-index-systems requires more effort than the numerical integration of ODE's or index-one-systems. It will also be shown in Section 4.2 how further results of this algorithm can additionally be used to cope with the problems of consistent initialization and index reduction.

Finally, in Section 4.3 another structural algorithm proposed earlier by Pantelides [37] will be introduced briefly. This method was mainly developed to address the problem of consistent initialization of higher-index-DAE's. A comparison of this method with our approach will show that both algorithms yield identical results concerning the (structural) index and the (structural) number of degrees of freedom.

4.1 The Idea of Structural Considerations

In this section we first introduce a *structural algebra* and then use those rules to define *structural properties of DAE's* .

In *structural considerations* it is only distinguished between (hard) zeros (0) and non-zero ($*$). Therefore, the transformation from numerical (and symbolical) representations to the corresponding structural representation can be accomplished by a map S from the real numbers $\mathcal{R}$ to the two elements of the set $\{0, *\}$:

$$S: \quad \mathcal{R} \rightarrow \{0, *\} \quad . \tag{4.1}$$

Structural *computations* , i.e., operations on the elements of $\{0, *\}$, can be considered the "crudest way of computation". This becomes clear from the most general rule for structural operations:

Proposition 4.1.1: *Whenever there is a numerical (or symbolical) representation of operands in $\mathcal{R}$ such that a certain operation in $\mathcal{R}$ yields a nonzero result then the result of the corresponding structural representation of this operation is $*$. Conversely, if there is no such numerical (or symbolical) representation then the structural result is 0.*

This rather formal statement becomes more vivid by looking at its consequences for arithmetic operations. The following rules also illustrate why structural zeros can also be called *hard zeros.*

Rule 1: Addition and Subtraction

$$\begin{aligned} 1.1: \quad * + * &= * \\ 1.2: \quad * + 0 &= * \\ 1.3: \quad 0 + 0 &= 0 \end{aligned}$$

Subtraction can be viewed as addition of a negative number (nonzero) and is therefore no different from addition itself. Note that the addition/subtraction of two nonzero numbers a, b always yields a nonzero result according to 1.1 above and *Proposition 4.1.1* . Evidently, for $a = -b$ this sum $a + b$ is zero *numerically.* Numerical zeros are considered *weak zeros* since they occur only if there is some special numerical relations among the operands of a certain operation. In the structural sense of *Proposition 4.1.1* these are "non-zeros", or more precisely "no hard zeros".

Rule 2: Multiplication and Division

$$\begin{aligned} 2.1: \quad * \cdot * &= * \\ 2.2: \quad * \cdot 0 &= 0 \end{aligned}$$

Division is equivalent to a multiplication with the reciprocal of the denominator,

consequently:

$$2.3: \quad \frac{*}{*} = *$$

$$2.4: \quad \frac{0}{*} = 0$$

$$2.5: \quad \frac{*}{0} \ \ldots \ \text{not defined.}$$

The two rules above are just the *structural equivalent* of the arithmetic operations defined on the real numbers $\mathcal{R}$.

Before we generalize these two rules to obtain a *structural matrix algebra* we introduce:

Definition 4.1.1: *The representation of a matrix $M \in \mathcal{R}^{n\times m}$ where nonzero elements are represented by $*$ and zero elements by 0 is called the pattern of M and is abbreviated by $patM$.*

With this the definition of a structural matrix algebra based on *Proposition 4.1.1* and **Rules 1,2** is straightforward:

$$(pat\,A \,+\, pat\,B)_{ij} \;=\; (a_{ij})_s \,+\, (b_{ij})_s\,, \tag{4.2}$$

$$(pat\,A \,\cdot\, pat\,B)_{ij} \;=\; \sum_{k=1}^{n} (a_{ij})_s \,\cdot\, (b_{ij})_s\,, \tag{4.3}$$

where $A \in \mathcal{R}^{m\times n}$, $B \in \mathcal{R}^{n\times l}$ and $(a_{ij})_s, (b_{ij})_s$ denotes the structural representation of element ij in matrix A, B respectively, i.e., $(a_{ij})_s, (b_{ij})_s \in \{0, *\}$. The scalar operations on the right-hand sides above are subject to **Rules 1,2.**

It should be noted that in general

$$pat\,A \,+\, pat\,B \;\neq\; pat\,(A+B)\,, \tag{4.4}$$

$$pat\,A \,\cdot\, pat\,B \;\neq\; pat\,(A\cdot B)\,. \tag{4.5}$$

This can be seen from the small examples below:

Example 4.1:

1. Addition

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad B = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} ,$$

$$pat\,A = pat\,B = \begin{pmatrix} * & 0 & * \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} \rightsquigarrow pat\,A + pat\,B = \begin{pmatrix} * & 0 & * \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} ,$$

$$A + B = \begin{pmatrix} 1-1 & 0 & 2 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \rightsquigarrow pat\,(A+B) = \begin{pmatrix} 0 & 0 & * \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} ,$$

$$pat\,A + pat\,B \neq pat\,(A+B)\,.$$

2. Multiplication

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad B = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} ,$$

$$pat\,A = \begin{pmatrix} * & 0 & * \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} \quad , \quad pat\,B = \begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ * & 0 & * \end{pmatrix}$$

$$\rightsquigarrow pat\,A \cdot pat\,B = \begin{pmatrix} * & 0 & * \\ 0 & * & 0 \\ * & 0 & * \end{pmatrix} ,$$

$$A \cdot B = \begin{pmatrix} 1-1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \rightsquigarrow pat\,(A \cdot B) = \begin{pmatrix} 0 & 0 & * \\ 0 & * & 0 \\ * & 0 & * \end{pmatrix} ,$$

$$pat\,A \cdot pat\,B \neq pat\,(A \cdot B)\,.$$

Due to 1.1 in **Rule 1** it can be stated:

Proposition 4.1.2: *The ij-element* $(pat\,A + pat\,B)_{ij}$ *of* $(pat\,A + pat\,B)$ *is zero if and only if* $(pat\,A)_{ij} = (pat\,B)_{ij} = 0$.

Practically this means that $(pat\,A + pat\,B)$ can be constructed by adding non-zeros of $pat\,A$ to $pat\,B$ or vice versa. Thus, adding two matrices $A, B \in \mathcal{R}^{n\times}$ structurally can be viewed as *superposing* $pat\,A$ and $pat\,B$ in the way described above. This motivates the following

Definition 4.1.2: *The pattern obtained by superposing pat A and pat B such that all nonzero entries of pat A are added to pat B or vice versa is called the merged pattern of A, B .*

With this it is convenient to formulate

Rule 3: Matrix–Addition
Two matrices $A, B \in \mathcal{R}^{n\times m}$ *are added structurally by building their merged pattern.*

Finally, for completeness we also give a formal rule for structural matrix multiplication which is an immediate consequence of **Rules 1,2**:

Rule 4: Matrix–Multiplication
Two matrices $A, B \in \mathcal{R}^{n\times m}$ *are multiplied structurally by formally multiplying their patterns as defined for numerical matrices while obeying* **Rules 1,2**.

Before we define structural properties of DAE's based on the considerations above and the definitions in Chapter 2 we have to introduce the *structural rank* of a matrix.

Definition 4.1.3: *Let $N_M \in \mathcal{R}^{n\times m}$ be the the set of all possible numerical representations M_i of pat M. Let rank M_i be the numerical rank of a numerical representation $M_i \in N_M$ of pat M. Then the maximum of all values rank M_i is called the structural rank s_r of pat M, i.e., $s_r = \max_{M_i \in N_M} (rank\, M_i)$.*

An immediate consequence of this definition is that there are matrices whose numerical rank is smaller than their structural rank, whereas the opposite is not possible. This can be seen from the (2×2)-matrix

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{4.6}$$

whose numerical rank is one but whose structural rank is two since

$$pat\, M = \begin{pmatrix} * & * \\ * & * \end{pmatrix} \tag{4.7}$$

and there are a great many (2×2)-matrices with such a pattern that have numerical rank two. One example of those is:

$$M' = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}. \tag{4.8}$$

For square matrices $M \in \mathcal{R}^{n\times n}$ the statements

$$rank M < n \text{ and } det\, M \equiv 0$$

are equivalent [28]. Using this well known equivalency along with *Definition 4.1.3* we can conclude:

Proposition 4.1.3: *If the structural rank s_r of a square pattern pat M, $M \in \mathcal{R}^{n\times n}$, satisfies $s_r < n$ then*

$$det\, M_i \equiv 0$$

for all numerical representations $M_i \in N_M$ of pat M. All numerical representations $M_i \in N_M$ of pat M are then called structurally singular.

Proposition 4.1.2 along with the last proposition can be used to reformulate the solvability conditions in *Propositions 2.2.2a/b* from the structural point of view:

Proposition 4.1.4a: *A sufficient condition for the DAE* $F(z, \dot{z}, u) = 0$, $F, z \in \mathcal{R}^n$ *to be not solvable is that the structural rank of the merged pattern* $(pat\, F_z + pat\, F_{\dot{z}})$ *of* $F_z, F_{\dot{z}} \in \mathcal{R}^{n \times n}$ *is less than* n.

This can be reformulated to obtain:

Proposition 4.1.4b: *A necessary condition for the DAE* $F(z, \dot{z}, u) = 0$, $F, z \in \mathcal{R}^n$ *to be solvable is that the structural rank of the merged pattern* $(pat\, F_z + pat\, F_{\dot{z}})$ *of* $F_z, F_{\dot{z}} \in \mathcal{R}^{n \times n}$ *is* n.

The relationship between *Proposition 4.1.4a* and *Proposition 2.2.2a* is illustrated by the following *Example 4.2* and the brief discussion preceding it. In *Example 4.3* we will point out that the structural condition formulated in *Proposition 4.1.4a* is *less restrictive* than the condition given in *Proposition 2.2.2a*.

As indicated in Section 2.2 the condition formulated in *Proposition 2.2.2a* rules a DAE unsolvable if it consists of redundant equations or if it consists of equations that contain terms involving a subset of variables which can be substituted by one variable σ in all equations. For the upcoming argument it is convenient to introduce

Definition 4.1.5: *An equation* $F_i(z_1, \ldots, z_\rho, \dot{z}_1, \ldots, \dot{z}_\rho, u) = 0$, $1 \leq \rho \leq n-1$, *of the DAE* $F(z, \dot{z}, u) = 0$, $F, z \in \mathcal{R}^n$ *is called structurally redundant if the DAE contains a subset* $\tilde{F} = 0$ *of* ρ *equations* $F_j(z_1, \ldots, z_\rho, \dot{z}_1, \ldots, \dot{z}_\rho, u) = 0$, $1 \leq \rho \leq n-1$, $j \neq i$, *for which the structural rank of the merged pattern* $(pat\, \tilde{F}_{\tilde{z}} + pat\, \tilde{F}_{\dot{\tilde{z}}})$, $\tilde{z} = (z_1, \ldots, z_\rho)^T$, *is* ρ *and* $\tilde{F}_{\tilde{\tilde{z}}} \equiv \tilde{F}_{\dot{\tilde{z}}} \equiv 0$, *where* $z = \begin{pmatrix} \tilde{z} \\ \tilde{\tilde{z}} \end{pmatrix}$.

Less formal this means that an equation $F_i = 0$ of the DAE $F(z, \dot{z}, u) = 0$ involving only a subset $\tilde{z}, \dot{\tilde{z}}$ of ρ components of $z, \dot{z}$ respectively is called redundant if there are (at least) ρ other equations forming $\tilde{F} = 0$ which also involve only variables of the subset $\tilde{z}, \dot{\tilde{z}}$.

This definition is consistent with *Definition 2.2.3* since $F_i = 0$ has to be satisfied automatically if the ρ equations $F_j = 0$, $j \neq i$ are satisfied. If this would not be the case then the DAE would consist of contradictory equations and would not have a solution due that reason. Motivated by *Propositions 2.2.2a/b* one might expect that for a structural solvability check $pat(\lambda F_{\dot{z}} + F_z)$ should be examined. However, before this pattern could be set up one would have to carry out *symbolical* computations since $(\lambda F_{\dot{z}} + F_z)$ would have to be known *symbolically.* Since we are interested in *purely structural* considerations we can only use $pat\, F_z$ and $pat\, F_{\dot{z}}$. Considering (4.5) one might suspect that this yields inconsistencies. However, one can verify easily that generally the number of nonzeros in $pat(\lambda F_{\dot{z}} + F_z)$ is less or equal to the number of nonzeros in $pat(\lambda F_{\dot{z}}) + pat\, F_z$, which is due to possible symbolical cancellation. Therefore, if $pat(\lambda F_{\dot{z}}) + pat\, F_z$ is structurally singular, then $pat(\lambda F_{\dot{z}} + F_z)$ is structurally singular. If additionally **Rule 2** is observed then $\lambda \neq 0$ can be removed without changing the impact of $pat\, F_{\dot{z}}$ on the merged pattern considered. This shows that *Proposition 4.1.4a* represents a sufficient condition for structural non-solvability and that *Proposition 4.1.4b* represents a necessary condition for structural solvability.

In order to make the rather formal argument above more vivid we consider

Example 4.2:

$$F_1(z_1, z_2, \dot{z}_1, \dot{z}_2, u) = 0 \tag{4.9}$$

$$F_2(z_1, z_2, \dot{z}_1, \dot{z}_2, u) = 0 \tag{4.10}$$

$$F_3(z_1, z_2, \dot{z}_1, \dot{z}_2, u) = 0 \tag{4.11}$$

$$F_4(z_1, z_2, z_3, z_4, \dot{z}_1, \dot{z}_2, \dot{z}_3, \dot{z}_4, u) \;=\; 0 \tag{4.12}$$

$$\tag{4.13}$$

If we assume that these equations actually contain all variables listed in the brackets then the patterns of the Jacobians $F_z, F_{\dot{z}}$ become

$$pat\, F_z \;=\; pat\, F_{\dot{z}} \;=\; \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & * \end{pmatrix} . \tag{4.14}$$

With the notation of *Definition 4.1.2* this implies

$$\rho = 2\,,\;\; \tilde{z} = (z_1, z_2)^T\,,\;\; \tilde{\tilde{z}} = (z_3, z_4)^T\,,\;\; \tilde{F} = (F_1, F_2)^T\,,\;\; F_i = F_3$$

and

$$pat\, \tilde{F}_{\tilde{z}} \;=\; pat\, \tilde{F}_{\dot{\tilde{z}}} \;=\; \begin{pmatrix} * & * \\ * & * \end{pmatrix} . \tag{4.15}$$

such that the structural rank of this merged pattern is two.

Thus, $\tilde{F} = (F_1, F_2)^T = 0$ can be considered a system which can be solved to obtain $z_1^*(t)$ and $z_2^*(t)$ without considering F_3, F_4 at all. Since $F_3 = 0$ only involves $z_1, z_2, \dot{z}_1, \dot{z}_2$ as well, it has to be satisfied by $z_1^*(t)$ and $z_2^*(t)$ obtained from the subsystem $\tilde{F} = (F_1, F_2)^T = 0$. If this is not true then there are contradictions and the DAE $F = 0$ is not solvable for this reason.

By using vector algebra one can see that F_3 is redundant at all points in state space since

$$pat\, F_z \;=\; pat\, F_{\dot{z}} \;=\; \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & * \end{pmatrix} . \tag{4.16}$$

is structurally singular, or in other words its structural rank is less than four. This is true since the first three rows in this pattern can be considered three two-vectors $v_i \in \mathcal{R}^2$, $i = 1, 2, 3$. Only pairs of two of these vectors can be linearly independent.

No matter which numerical representation is chosen the third vector can always be represented as a linear combination of the other two

$$v_k \;=\; \alpha_l\, v_l \;+\; \alpha_m\, v_m \;, \tag{4.17}$$

where α_l, α_m are constants and $k, l, m = 1, 2, 3$ with $k \neq l \neq m$.
Therefore, F_3 (or F_2 or F_1) can be removed from the DAE without loosing any information. The merged pattern of the Jacobians of the resulting DAE $\bar{F}(z, \dot{z}, u) = 0$, $\bar{F} \in \mathcal{R}^3$, $z \in \mathcal{R}^4$ becomes

$$pat\, \bar{F}_z \;=\; pat\, \bar{F}_{\dot{z}} \;=\; \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & * \end{pmatrix} . \tag{4.18}$$

This obviously represent a system of three (structurally) non-redundant equations in four unknowns. Using the notation of Section 2.2 one can only solve this system for $z_1(t), z_2(t)$ and some lumped variable $\sigma = \sigma(z_3, z_4, \dot{z}_3, \dot{z}_4)$. Therefore, the system cannot be used to determine $z(t)$ completely, i.e., it is not solvable with respect to the complete state vector $z = (z_1, z_2, z_3, z_4)^T$.

The example above has illustrated that the structural condition formulated in *Proposition 4.1.4a* is sufficient to rule a DAE unsolvable. The next example points out that this condition is less restrictive than the condition formulated in *Proposition 2.2.2a.* Less restrictive means that there are DAE's which are not solvable by the condition in *Proposition 2.2.2a* but cannot be ruled unsolvable by the structural condition given in this section.

Example 4.3:

$$F_1(z_1, z_2, z_3, \dot{z}_1, \dot{z}_2, \dot{z}_3, u) \;=\; 0 \tag{4.19}$$

$$F_2(z_1, z_2, z_3, \dot{z}_1, \dot{z}_2, \dot{z}_3, u) \;=\; 0 \tag{4.20}$$

$$F_3 \;\equiv\; F_1(z_1, z_2, z_3, \dot{z}_1, \dot{z}_2, \dot{z}_3, u) + F_2(z_1, z_2, z_3, \dot{z}_1, \dot{z}_2, \dot{z}_3, u) \;=\; 0 \tag{4.21}$$

According to *Definition 2.2.3* $F_3 = 0$ is a redundant equation implying that this DAE can be ruled unsolvable by *Proposition 2.2.4*. However, this DAE cannot be ruled *structurally* unsolvable according to *Proposition 4.1.4a* since

$$pat\, F_z \;=\; pat\, F_{\dot{z}} \;=\; \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \tag{4.22}$$

and consequently

$$pat\, F_z \;+\; pat\, F_{\dot{z}} \;=\; \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}, \tag{4.23}$$

whose structural rank according to *Definition 4.1.2* is three.

This last example indicates *limitations* of the structural approach, which will also be responsible for the structural index being only a lower bound on the index and the structural number of degrees of freedom being an upper bound on the number of degrees of freedom. In practical applications these bounds are very frequently the actual index and the actual number of degrees of freedom of the DAE according to the definitions given in Section 2.1. However, one has to keep in mind that structural considerations cannot detect singularities that are due to numerical or symbolical cancellation.

As indicated in previous chapters a very important characteristics of a DAE is its index. Since the computation of the index can require repeated differentiation we have to define this operation in the structural sense in order to define the *structural index.*

To motivate a rule for structural differentiations consider the scalar function

$$f:\;\; x \;\rightarrow\; f(x)\,, \tag{4.24}$$

where $f, x \in \mathcal{R}$ and its derivative $\frac{df}{dx} \in \mathcal{R}$. Note that there are representations of f

such that its derivative

$$\frac{df}{dx} = \text{function}(x) \tag{4.25}$$

still depends on x explicitly. In fact, the only two cases for which this is not true are f being a constant or a linear function. Based on *Proposition 4.1.1* we can generalize this for $z \in \mathcal{R}^n$ and $f(z)$ being a function from $\mathcal{R}^n$ to $\mathcal{R}$ in

Rule 5: *If differentiations are carried out in the structural sense then* $f = f(z)$, $f \in \mathcal{R}$ $z \in \mathcal{R}^n$ *implies*

$$\begin{aligned} 5.1: &\quad \tfrac{\partial f}{\partial z_j} = \textit{function}(z_j) \quad \textit{if } f(z) \textit{ is nonlinear in } z_j \; j = 1(1)n \textit{ of } z. \\ 5.2: &\quad \tfrac{\partial f}{\partial z_j} \neq \textit{function}(z_j) \quad \textit{if } f(z) \textit{ is linear in } \underline{\textit{all}} \; z_j \; j = 1(1)n \textit{ of } z, \end{aligned}$$

where $\frac{\partial f}{\partial z_j} \in \mathcal{R}$.

In other words this means for repeated differentiations that 5.1 assumes that $f(z) = 0$ is differentiable with respect to all components z_j $j = 1(1)n$ of z *as often as required*, whereas 5.2 assumes that f is of the form $f = \sum_{j=1}^{n} a_j z_j$ with a_j being constant, $j = 1(1)n$. Later in this section it will be shown that these two different approaches can be used to estimate the impact of nonlinearities on the choice of feasible sets of arbitrary initial conditions.

For differentiations with respect to time t which are required for determining the index by application of *Algorithm 2.1.1* this means that in the structural sense

$$G_i(z) = 0 \tag{4.26}$$

implies that

$$F_{i+1}(z, \dot{z}) \equiv (G_i)_z \, \dot{z} = 0 \tag{4.27}$$

depends on *all* components of z that occurred explicitly in $G_i(z) = 0$ if according G_i is treated according to 5.1. This can be viewed as the assumption that the DAE

considered represents "one end of the nonlinear scale"(it is "infinitely nonlinear"), whereas differentiations according to 5.2 would assume the DAE to be on the "other end" of this scale being linear ("not nonlinear") This is implied by assuming that $\frac{dG}{dt} = \sum_{j=1}^{n} a_j \dot{z}_j$ with a_j, $j = 1(1)n$, being constant which also means $G(z) = \sum_{j=1}^{n} a_j z_j$.
One might suspect that the two different ways of carrying out differentiations would yield different results for the index and the number of degrees of freedom. In general this is true, however we will show that for structural considerations the only two factors determining the *structural* index ν_s and the *structural* number of degrees of freedom r_s are *pat* F_z and *pat* $F_{\dot{z}}$. This means that higher order partial derivatives of F such as $F_{zz}, F_{\dot{z}z}, F_{\dot{z}\dot{z}}, \ldots$ do not play any role in the computation of r_s and ν_s. This indicates that the way differentiations are carried out, according to 5.1 or according to 5.2, has no impact on the results obtained for r_s and ν_s.
Before these important observations are summarized in a proposition we have to introduce

Definition 4.1.6 *The structural index ν_s of a DAE $F(z, \dot{z}, u(t)) = 0$ is the index obtained if all computations are carried out according to the structural* **Rules 1–5.**

In the same fashion we can formulate

Definition 4.1.7: *The structural number of degrees of freedom r_s of a DAE $F(z, \dot{z}, u(t)) = 0$ is the number of degrees of freedom obtained if all computations are carried out according to the structural* **Rules 1–5.**

With the two definitions above we can state

Proposition 4.1.5: *The structural index ν_s and the structural number of degrees of*

freedom r_s of a DAE $F(z, \dot{z}, u(t)) = 0$ is solely determined by pat F_z and pat $F_{\dot{z}}$.

and as an immediate consequence of this

Proposition 4.1.6: *Let ν_l be the index and r_l be the number of degrees of freedom obtained by applying only structural computations on the linear DAE*

$$F_{\dot{z}}|_p \Delta\dot{z} + F_z|_p \Delta z + F_u|_p \Delta u = 0 \tag{4.28}$$

which represents the linearization of the DAE $F(z, \dot{z}, u(t)) = 0$ at a point $(z_p, \dot{z}_p, t_p)$ of a solution $z^*(t)$ where $|_p$ denotes that these matrices are evaluated at $(z_p, \dot{z}_p, t_p)$ and $\Delta z = z - z_p, \Delta\dot{z} = \dot{z} - \dot{z}_p, \Delta t = t - t_p$.
Let ν_n be the number of differentiations with respect to time t that have to be applied to the DAE $F(z, \dot{z}, u(t)) = 0$ until the system

$$F(z, \dot{z}, u(t)) = 0 \tag{4.29}$$

$$\frac{dF}{dt}(z, \dot{z}, \ddot{z}, u(t), \dot{u}(t)) = 0 \tag{4.30}$$

$$\vdots$$

$$\frac{d^{\nu_n} F}{dt^{\nu_n}}(z, \dot{z}, \ddot{z}, \ldots, z^{(\nu_n+1)}, u(t), \ldots, u^{(\nu_n)}(t)) = 0 \tag{4.31}$$

can be solved for $\dot{z} = \dot{z}(z, t)$ by structural computations and let r_n be the number of degrees of freedom obtained from the same procedure,.
Then $\nu_l = \nu_n$ and $r_l = r_n$ is always satisfied.

It is not obvious that the two statements above hold. A proof for the validity of both propositions is given in Appendix C. From both, a practical as well a theoretical point of view, the observation formulated in *Propositions 4.1.5 and 4.1.6* is very interesting. From the *theoretical* point of view one can say that the nonlinear DAE $F(z, \dot{z}, u(t)) =$

0 and its corresponding linearized representation

$$F_{\dot{z}}|_p \Delta\dot{z} + F_z|_p \Delta z + F_u|_p \Delta u = 0 \tag{4.32}$$

have the *same structural index* ν_s and the same *structural number of degrees of freedom* r_s. This is true since the patterns of the Jacobians of both systems with respect to $z, \dot{z}$ are identical. These patterns indicate which variables occur in which equations. They do not contain information *how* variables appear, i.e., in what kind of terms they occur.

An important *practical* implication of this result is that the structural representation of *Algorithm 2.1.1* which will be discussed in greater detail in the upcoming Section 4.2 can be set up as if $F(z, \dot{z}, u(t)) = 0$ would be a *linear constant coefficient* DAE

$$M\dot{z} + Lz + \tilde{u}(t) = 0\,, \tag{4.33}$$

where $pat\, M = pat\, F_{\dot{z}}$ and $pat\, L = pat\, F_z$ [1]. This is very convenient since it allows the the application of a structural representation of *Algorithm 2.1.1* without requiring information on patterns of higher order partial derivatives of F with respect to $z, \dot{z}$ and still leads to consistent results for the structural index and the structural number of degrees of freedom of the nonlinear DAE $F(z, \dot{z}, u(t)) = 0$.

4.2 Structural Representations of Algorithm 2.1.1

In this section structural representation of *Algorithm 2.1.1* are suggested. In a first step we will formulate a structural algorithm, *Algorithm 4.2.1,* as if the DAE considered would be a *linear constant coefficient DAE*

$$M\dot{z} + Lz = 0\,, \tag{4.34}$$

[1] $\tilde{u}(t)$ represents some time dependent function which does not play any role in the computation of the structural index.

where $z \in \mathcal{R}$ and $M, L \in \mathcal{R}^{n \times n}$ are considered to be *constant* matrices [2]. Due to *Proposition 4.1.5* this will yield consistent results for the structural index ν_s and the structural number of degrees of freedom r_s of the general nonlinear DAE $F(z, \dot{z}) = 0$ if

$$pat\, M \equiv pat\, F_{\dot{z}} \text{ and } pat\, L \equiv pat\, F_z\,, \tag{4.35}$$

since with these relations the pattern of the Jacobians with respect to $z, \dot{z}$ of both systems are identical.

This indicates the practical significance of *Proposition 4.1.5* and *Proposition 4.1.6*. By these two propositions it is possible to carry out differentiations according to **Rule 5.2** without having to worry about higher order partial derivatives of F and still obtains consistent results for ν_s and r_s. As indicated in the preceeding section doing this implies that the DAE is assumed to be a linear constant coefficient DAE. Differentiating according to **Rule 5.2** on the other hand would imply that the DAE is considered "as nonlinear as possible". After the formulation of *Algorithm 4.2.1* which applies **Rule 5.2** it will be indicated that a very similar formulation, *Algorithm 4.2.n*, applying **Rule 5.1** will yield additional information for the subject of consistent initialization. However, it is important to note that the structural index ν_s and the structural number of degrees of freedom r_s obtained from both representations are *identical* by *Propositions 4.1.5* and *4.1.6*.

Before the introduction of *Algorithm 4.2.1* below we want to clarify the notation used in it. The algorithm is formulated as if the DAE $F(z, \dot{z}) = 0$ considered would be a linear constant coefficient DAE $M\,\dot{z} + L\,z = 0$. In order to shorten notation we omitted "*pat*" in front of all matrices, but it should be emphasized that all matrices occurring in it are represented by their pattern and that all computations are carried out according to **Rules 1,2,3,** and **5.1**.

[2] Again, without loss of generality input variables and explicit time dependencies $u(t) \in \mathcal{R}^p$ can be omitted in the structural considerations carried out here.

Algorithm 4.2.1:

Step 0: Initialization:

$$\begin{array}{lcl} i & = & 0\,, \\ n_0 & = & n\,, \\ r & = & n\,, \\ \tilde{z}_0 & = & z\,, \\ M_0 & = & M \equiv F_{\dot{z}}\,, \\ L_0 & = & L \equiv F_z\,, \\ X_{-1} & \ldots & \text{is a "vector" with no elements}\,. \end{array}$$

Step 1: Determine: $r_i = \text{struc.rank}\, M_i$, where $M_i \in \mathcal{R}^{n_i \times n_i}$,
and set $r \rightarrow r - (n_i - r_i)$.

Step 2: Partition $M_i = \begin{pmatrix} A_i & B_i \\ C_i & D_i \end{pmatrix}$, $L_i = \begin{pmatrix} L_i^1 \\ L_i^2 \end{pmatrix}$ such that
$A_i \in \mathcal{R}^{r_i \times r_i}$ with $det\, A_i \neq 0$, $B_i \in \mathcal{R}^{r_i \times n_i}$, $C_i \in \mathcal{R}^{(n_i - r_i) \times r_i}$, $D_i \in \mathcal{R}^{(n_i - r_i) \times (n_i - r_i)}$,
$L_i^1 \in \mathcal{R}^{r_i \times n}$, $L_i^2 \in \mathcal{R}^{(n_i - r_i) \times n}$, $x_i \in \mathcal{R}^{r_i}$, $y_i \in \mathcal{R}^{(n_i - r_i)}$, $\tilde{z}_i = (x_i, y_i)^T \in \mathcal{R}^{n_i}$.

Step 3: Solve $A_i\, \dot{x}_i + B_i\, \dot{y}_i = L_i^1\, z$ for $\dot{x}_i = A_i^{-1} L_i^1\, z - A_i^{-1} B_i\, \dot{y}_i$ and add it to
$\dot{X}_{i-1} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{i-1})^T = \Upsilon_{i-1}^z z + \Upsilon_{i-1}^{\dot{y}} \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix}$.
to obtain
$\dot{X}_i = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{i-1}, \dot{x}_i)^T = \tilde{\Upsilon}_{i-1}^z\, z + \Upsilon_{i-1}^{\dot{x}_i} \dot{x}_i + \Upsilon^{\dot{y}_i}\, \dot{y}_i$,
where $\tilde{\Upsilon}_{i-1}^z \in \mathcal{R}^{d_i \times n}$, $\Upsilon_{i-1}^{\dot{x}_i} \in \mathcal{R}^{d_i \times r_i}$, $\Upsilon_i^{\dot{y}} \in \mathcal{R}^{d_i \times (n_i - r_i)}$, $d_i = \sum_{j=1}^{i} r_i$

Step 4: Substitute $\dot{x}_i = A_i^{-1} L_i^1\, z - A_i^{-1} B_i\, \dot{y}_i$ into $\dot{X}_i = \tilde{\Upsilon}_{i-1}^z\, z + \Upsilon_{i-1}^{\dot{x}_i}\, \dot{x}_i + \Upsilon_i^{\dot{y}}\, \dot{y}_i$,
to obtain $\dot{X}_i = \Upsilon_i^z\, z + \Upsilon_i^{\dot{y}}\, \dot{y}_i$, where $\Upsilon_i^z \in \mathcal{R}^{d_i \times n}$, $d_i = \sum_{j=1}^{i} r_i$.

Step 5: If $r_i = n_i$ then STOP.

Step 6: Substitute $\dot{x}_i = A_i^{-1} L_i^1\, z - A_i^{-1} B_i\, \dot{y}_i$ into $C_i\, \dot{x}_i + D_i\, \dot{y}_i = L_i^2\, z$ to obtain

$G_i(z) = (L_i^2 - C_i A_i^{-1} L_i^1)\, z = 0$ according to *Proposition 2.1.1.*

Step 7: Differentiate $G_i(z) = 0$ with respect to time:

$$(L_i^2 - C_i A_i^{-1} L_i^1)\, \dot{z} \;=\; (L_i^2 - C_i A_i^{-1} L_i^1) \begin{pmatrix} \dot{X}_i \\ \dot{y}_i \end{pmatrix} \;=\; 0$$

Step 8: Substitute $\dot{X}_i = \Upsilon_i^z z + \Upsilon^{\dot{y}_i} \dot{y}_i$, into the result of **Step 7** to obtain:

$$M_{i+1}\, \dot{y}_i \;+\; L_{i+1}\, z \;=$$
$$(L_i^2 - C_i A_i^{-1} L_i^1) \begin{pmatrix} \Upsilon_i^z \\ 0 \end{pmatrix} z \;+\; (L_i^2 - C_i A_i^{-1} L_i^1) \begin{pmatrix} \Upsilon_i^{\dot{y}_i} \\ I_{(n_i - r_i)} \end{pmatrix} \dot{y}_i \;=\; 0,$$

where $I_{(n_i - r_i)}$ denotes the $(n_i - r_i) \times (n_i - r_i)$ unity matrix.

Step 9:

$$\begin{aligned} \tilde{z}_{i+1} &= y_i\,, \\ n_{i+1} &= n_i - r_i\,, \\ i &\rightarrow i+1\,. \end{aligned}$$

Step 10: Restart from **Step 1.**

Before we move on to possible further analyzation of the DAE based on computations within this structural algorithm we briefly comment on the inversion of

$$A_i\, \dot{x}_i \;+\; B_i\, \dot{y}_i \;=\; L_i^1\, z \tag{4.36}$$

which in **Step 3** yields

$$\dot{x}_i \;=\; A_i^{-1} L_i^1\, z \;-\; A_i^{-1} B_i\, \dot{y}_i\,. \tag{4.37}$$

The structural procedure that accomplishes this inversion can be called *Structural Gauss Elimination* since it is just the regular Gauss Elimination

$$M\, x \;=\; y \;\rightarrow\; I_n\, x \;=\; M^{-1}\, y\,, \tag{4.38}$$

with $x, y \in \mathcal{R}^n$, $M, I_n \in \mathcal{R}^{n \times n}$, $det\, M \neq 0$, $I_n = diag(1, \ldots, 1)$ carried out according to the structural **Rules 1–4**.

Interpretation of *Algorithm 4.2.1*

Basically *Algorithm 4.1.1* determines the structural index ν_s and the structural number of degrees of freedom r_s of a DAE $F(z, \dot{z}) = 0$ by reducing the structural index of the DAE to zero by means of structural computations. Therefore, it also provides information on how the (structural) index of $F(z, \dot{z}) = 0$ can be reduced to values smaller than ν_s by symbolical manipulations. This information can be provided by keeping track of what the algorithm does during the computation of the structural index, i.e., which equations are differentiated and which substitutions are carried out. The algorithm also detects variables (and equations) which cause higher-index-problems. *Variables* z_j that cause the structural index to exceed one are simply the ones for whose time derivative $\dot{z}_j$ it cannot be solved after at most one differentiation with respect to time. *Equations* that are definitely involved as causes of this problem are those which have not been used to determine any component $\dot{z}_i$ of $\dot{z}$ after one differentiation. In Chapter 5 we will show how these observations can be implemented to obtain as many information on the DAE as possible. Applications of an implementation of *Algorithm 4.1.1* with these additional examinations on several examples will show advantages and limitations of this approach. subsubsectionDifferentiations According to Rule 5.2 – *Algorithm 4.2.n* So far we have only considered the structural representation of *Algorithm 2.1.1* as given by *Algorithm 4.2.1*, where the general nonlinear DAE $F(z, \dot{z}) = 0$ is treated as a linear constant coefficient DAE. As indicated in the beginning of this section for the problem of consistent initialization it might be of interest to consider a structural representation of *Algorithm 2.1.1* where differentiations are carried out according to **Rule 5.1**. This representation is called *Algorithm 4.2.n* and can be obtained by just changing **Step 7** of *Algorithm 4.2.1*, all the other

steps remain the same. **Step 7** has to be changed such that all dependencies on variables z_i that occurred in an equation $G_i(z) = 0$ prior to its differentiation with respect to time still occur in the time derivative $(G_i)_z \dot{z} = 0$ of this equation. This can be assured by structurally replacing **Step 7** in *Algorithm 4.2.l* by

$$\textbf{Step 7n: } (L_i^2 - C_i A_i^{-1} L_i^1)\dot{z} + \underbrace{(L_i^2 - C_i A_i^{-1} L_i^1) z}_{\text{additionally}} = 0 .$$

Implications of Differences Between *Algorithms 4.2.l/n*

It should be emphasized that due to *Propositions 4.1.5 and 4.1.6* the two possible ways of structural differentiation do not affect the results for the structural index ν_s and the structural number of degrees of freedom r_s. This means that both, *Algorithm 4.2.l* and *Algorithm 4.2.n* obtain the same values for ν_s and r_s.However it affects the decision whether a subset $(\bar{z}_0, \bar{z}_0)^T \in \mathcal{R}^{r_s}$ of the the complete initial vector $(\dot{z}_0, z_0)^T \in \mathcal{R}^{2n}$ allows consistent initialization if it is assigned r_s arbitrary values. To show this we refer to Chapter 2 where we indicated that such a subset allows consistent initialization if $det\, J_0^* \neq 0$ where $J_0^* \in \mathcal{R}^{(2n-r)\times(2n-r)}$ is the submatrix arising if the columns corresponding to the variables $(\bar{z}_0, \bar{z}_0)^T$ are removed from

$$J_0 = \begin{pmatrix} -I_n & \phi_z|_0 \\ 0 & K_z|_0 \end{pmatrix} , \tag{4.39}$$

which is the Jacobian with respect to $\dot{z}, z$ of the corresponding extended system at initial time $t = t_0$. Using the same notation as in the algorithms introduced so far the corresponding extended system can be represented by

$$\dot{z} = \phi(z) \tag{4.40}$$

$$0 = G_0(z) \tag{4.41}$$

$$\vdots$$

$$0 = G_0(\nu - 1)(z) . \tag{4.42}$$

Structurally information on *pat* J_0 can be obtained by saving the patterns of the Jacobians $(G_i)_z$, $i = 1(1)\nu_s - 1$, of **Step 6** of *Algorithms 4.2.1 and 4.2,n*. On termination of these algorithms *pat* ϕ_z is known as well such that *pat* J_0 can be set up completely. If then a set of r_s variables $(\bar{\dot{z}}_0, \bar{z}_0)^T$ is specified then *pat* J_0^* can be set up and its structural rank r_f can be determined. There are two cases of interest:

1. $r_f = 2n - r_s$, i.e, *pat* J_0^* has full structural rank.

2. $r_f < 2n - r_s$, i.e, *pat* J_0^* is structurally singular.

The interpretation if these two possible results depends on the whether differentiations were carried out according to **Rule 5.1** or according to **Rule 5.2**, i.e., whether *pat* J_0 was computed by *Algorithm 4.2.n* or by *Algorithm 4.2.1*. The reason for this can be seen by the following brief argument which will be generalized in *Proposition 4.2.1* and further illustrated in *Example 5.1* in Chapter 5:
Assume $G_i(z) = 0$ contains an equation

$$g(z_1, z_2) \;=\; f_1(z_1) \;+\; c_2\, z_2 \tag{4.43}$$

with c_2 being a constant. Therefore, after one more differentiation with respect to time $G_{i+1}(z) = 0$ contains an equation

$$g^* \;=\; f_{z_1}\, \tilde{\phi}_1(z) \;+\; c_2\, \tilde{\phi}_2(z) \tag{4.44}$$

if $\dot{z}_1 = \tilde{\phi}_1(z)$ and $\dot{z}_2 = \tilde{\phi}_1(z)$ could be eliminated in prior steps. If we furthermore assume that $(\tilde{\phi}_1)_{z_2} = (\tilde{\phi}_2)_{z_1} = 0$ then the dependencies of g^* with respect to z_1, z_2 are

$$(g^*)_{z_1} \;=\; f_{z_1 z_1}\, \tilde{\phi}_1(z) \tag{4.45}$$

$$(g^*)_{z_2} \;=\; 0 \tag{4.46}$$

Therefore, if computed *symbolically*, g^* depends on z_1 but not on z_2.
If the same steps are carried out with the linearized representation

$$g_l(z_1, z_2) \;=\; c_1\, z_1 \;+\; c_2\, z_2 \tag{4.47}$$

of g then

$$(g_l^*)_{z_1} = (g_l^*)_{z_2} = 0 \tag{4.48}$$

if the same assumptions for $\dot{z}_1, \dot{z}_2$ as above hold.

If structural differentiations are carried out according to **Rule 5.2** the same equation $g = 0$ is treated like $g_n = g_n(z_1, z_2)$ which implies that it is infinitely differentiable with respect to z_1, z_2. Therefore, with the same assumptions for $\dot{z}_1, \dot{z}_2$ as above one obtains

$$g_n^* = (g_n)_{z_1} \tilde{\phi}_1(z) + (g_n)_{z_2} \tilde{\phi}_2(z) \tag{4.49}$$

and

$$(g^*)_{z_1} = g_{z_1 z_1} \tilde{\phi}_1(z) \tag{4.50}$$

$$(g^*)_{z_2} = g_{z_2 z_2} \tilde{\phi}_2(z) \tag{4.51}$$

which generally implies that g^* depends on both, z_1 and z_2.

Therefore,depending on the the assumptions for (structural) partial differentiation *pat* g^* becomes

$$\begin{array}{cc} g^*_{z_1} & g^*_{z_2} \\ \downarrow & \downarrow \\ (* & 0) \\ (0 & 0) \\ (* & *) \end{array} \qquad \begin{array}{l} \text{analytically} \\ \text{linearized (\textbf{Rule 5.2})} \\ \text{most nonlinear (\textbf{Rule 5.1})} \end{array}$$

The small example above motivates motivates the following generalization:

1. *pat* J_0^l which is *pat* J_0 as determined by *Algorithm 4.2.l* has the *minimal* number of nonzero entries possible for the Jacobian of the corresponding extended system of a DAE $F(z, \dot{z}) = 0$ of a given structure *pat* $F_{\dot{z}} =$ *pat* M, *pat* $F_z =$ *pat* L. This is due to differentiations according to **Rule 5.2** where a dependency on a variable disappears after one partial differentiation with respect to it.

2. *pat* J_0^n which is *pat* J_0 as determined by *Algorithm 4.2.n* has the *maximal* number of nonzero entries possible for the Jacobian of the corresponding extended

system of a DAE $F(z, \dot{z}) = 0$ of a given structure $pat\, F_{\dot{z}} = pat\, M$, $pat\, F_z = pat\, L$. This is due to differentiations according to **Rule 5.1** where a dependency on a variable in an equation never vanishes no matter how many partial derivatives with respect to this variable are applied to the equation.

As the small example above indicated both approaches are generally *approximations* of what would happen if differentiations would be carried out *symbolically* rather than structurally.

Based on these observations we can formulate:

Proposition 4.2.2: *Let $F(z, \dot{z}) = 0$, $F, z \in \mathcal{R}^n$, be a DAE with r_s structural degrees of freedom. Let pat J_0^l be the pattern of the Jacobian, $J_0^l \in \mathcal{R}^{(2n-r_s)\times 2n}$, with respect to $\dot{z}, z$ of the corresponding extended system derived by* Algorithm 4.2.1 *and let pat J_0^n be the analogous Jacobian, $J_0^n \in \mathcal{R}^{(2n-r_s)\times 2n}$, obtained from* Algorithm 4.2.n. *Let the subset $(\bar{\dot{z}}_0, \bar{z}_0)^T \in \mathcal{R}^{r_s}$ of the complete initial vector $(\dot{z}_0, z_0) \in \mathcal{R}^{2n}$ be the set to be assigned arbitrary initial conditions. Let pat J_0^{l*}, pat J_0^{n*}, $J_0^{l*}, J_0^{n*} \in \mathcal{R}^{(2n-r_s)\times(2n-r_s)}$, be the pattern arising from pat J_0^l, pat J_0^n if r_s columns corresponding to $(\bar{\dot{z}}_0, \bar{z}_0)^T$ are removed, then the set $(\bar{\dot{z}}_0, \bar{z}_0)^T$ of arbitrary initial conditions*

1. *allows consistent initialization if the structural rank of pat J_0^{l*} equals $2n - r_s$, i.e., if pat J_0^{l*} is structurally regular.*

2. *does not allow consistent initialization if the structural rank of pat J_0^{n*} is less than $2n - r_s$, i.e., if pat J_0^{n*} is structurally singular.*

Note that the inversion of both conditions above does *not* hold necessarily. This will also be illustrated in Chapter 5 along with further comments on the implementation of this method for the examination whether a certain set $(\bar{\dot{z}}_0, \bar{z}_0)^T$ is feasible for consistent initialization or not.

Structural Solvability and Termination of *Algorithms 4.2.1/n*

At the end of this section we briefly want to indicate the significance of the *structural solvability condition* of Section 4.1 for the structural algorithms suggested here. As indicated in Section 4.1 it is not possible to derive the corresponding extended system of a DAE $F(z,\dot{z}) = 0$ if $det(\lambda F_{\dot{z}} + F_z)$ vanishes structurally. Since *Algorithm 2.1.1* and its structural representations determine the structural index ν_s and the structural number of degrees of freedom r_s by deriving the corresponding extended system structurally, they cannot terminate if this system cannot be derived. Therefore, it is a *necessary* condition for termination of these algorithms that the merged pattern $(pat\, F_{\dot{z}} + pat\, F_z)$ is not structurally singular. In [5,p.20] it is indicated that *Algorithm 2.1.1* terminates after ν differentiations if it is applied to a *solvable* linear constant coefficient DAE $M\dot{z} + Lz + \tilde{u}(t) = 0$. As pointed out in Section 2.2 the necessary and sufficient condition for these systems to be solvable is that $det(\lambda M + L)$ does not vanish for all $\lambda \in \mathcal{R}$. Since there is a close relation between linear time invariant DAE's and structural considerations (see *Propositions 4.1.5 and 4.1.6*) it is plausible that the necessary condition for structural solvability given in Section 4.1 and above is also *sufficient* for the termination of *Algorithm 4.2.1* and *Algorithm 4.2.n*.

4.3 Structural Algorithm Proposed by Pantelides

After being mentioned at various points in earlier sections another structural algorithm proposed earlier by Pantelides [37] is introduced briefly in this section. Only the basic ideas behind it are indicated. For a more detailed discussion we refer to Pantelides [37]. The basic goal of this approach is to detect all subsets of equations of the DAE $F(z,\dot{z},u) = 0$, $F, z \in \mathcal{R}^n$ which need do be differentiated one or more times with respect to time in order to provide consistent initialization. These sets are identified by the *graph oriented* procedure motivated and described briefly below.

For a basic discussion of graph algorithms we refer to [47].

In order to have a similar notation as [37] we represent the DAE $F(z, \dot{z}, u) = 0$ by $F(x, \dot{x}, y) = 0$, where $z = (x, y)^T$ such that x represents the components z_i of z whose $\dot{z}_i$ occurs explicitly in an equation $F_i(z, \dot{z}) = 0$ whereas y is the subset of variables whose derivative does appear in any equation of $F(z, \dot{z}) = 0$. Again, explicit time dependencies and input variables $u(t)$ are omitted since they do not affect the considerations here. The *bipartite graph* set up and examined by this algorithm consists of two types of nodes: *equation (E-nodes) and variable nodes (V-nodes).* These nodes are connected by *edges* where in this case an $edge(i, j)$ exists if the variable represented by V-node j occurs in the equation represented by E-node i. The algorithm terminates after it has been able to assign each E-node to one V-node without using any E-node or V-node in more than one assignment pair. E-note i and V-node j can only form such a pair if $edge(i, j)$ exists.

In order to understand which variables and equations related to $F(z, \dot{z}) = 0$ form the nodes introduced above one can define $\eta = (\dot{x}, y)^T$ such that $F(z, \dot{z}) = F(x, \eta) = 0$. After differentiating this with respect to time t one obtains

$$F_\eta \dot{\eta} + F_x \dot{x} = 0 \,. \tag{4.52}$$

If F_η is invertible then this implies

$$\dot{\eta} = \phi(x) \tag{4.53}$$

by the implicit function theorem. If F_η is not invertible then equation (4.49) can be rewritten

$$F^1(\dot{\eta}, \eta, x) = 0 \tag{4.54}$$

$$G(\eta, x) = 0 \tag{4.55}$$

according to *Proposition 2.1.1*, where $det \frac{\partial F^1}{\partial \dot{\eta}} \neq 0$. This shows that in this case there are further constraints on the variables $(x, \eta)^T = (x, \dot{x}, y)^T$ beyond the ones

formulated by the original equations $F(z,\dot{z}) = F(x,\eta) = 0$. As a conclusion this means that if F is regular with respect to the "highest derivatives" $\eta = (\dot{x},y)^T$ of all variables $z = (x,y)^T$ then a differentiation of $F(z,\dot{z}) = F(x,\eta) = 0$ with respect to time yields n new equations in $\dot{\eta} = (\ddot{x},\dot{y})^T = \phi(x)$ in n new variables $(\ddot{x},\dot{y})^T$ without placing further constraints on the original set of variables $(x,\dot{x},y)^T$. Therefore, this case is of no further interest whereas in the other case there are further constraints (hidden equations) $G(x,\dot{x},y) = 0$ on those variables such that this differentiation is of interest. Basically, this is the same argument as the one used in Appendix C to prove *Proposition 4.1.5.*

If the observations above are extended on successive differentiations one can see that further differentiations will not yield further constraints on the set $(x,\dot{x},y)^T$ if they just yield as many new equations as new variables. This stage is reached if after differentiations applied to various subsets of equations there is a system of n equations originating from the n distinct equations $F(z,\dot{z}) = F(x,\eta) = 0$ whose Jacobian with respect to the highest derivatives of $(x,y)^T$ is regular. To see this assume that there is such a system

$$S_{\nu-1}(\eta_{\nu-1},\psi) \;=\; 0\,, \tag{4.56}$$

where $S_{\nu-1},\eta_{\nu-1} \in \mathcal{R}^n$ denotes the vector of the highest derivatives of $(x,y)^T$ in $S_{\nu-1}$,i.e., $\eta_{\nu-1}(x_1^{(hd(x_1))},\ldots,x_{r_0}^{(hd(x_{r_0}))},y_1^{(hd(y_1))},\ldots,y_{n-r_0}^{(hd(y_{n-r_0}))})^T$ and ψ denotes the vector whose components are $x_i^{(hd(k))}$, $1 \le k < hd(x_i)$, $i = 1(1)r_0$ and $y_i^{(hd(k))}$, $1 \le k < hd(y_i)$, $i = 1(1)n - r_0$ with $hd(x_i), hd(y_i)$ denoting the highest derivative of x_i, y_i occurring in $S_{\nu-1} = 0$. With this (4.53) after one more differentiation with respect to time becomes

$$(S_{\nu-1})_\eta\,\dot{\eta} \;+\; (S_{\nu-1})_\psi\,\dot{\psi} \;=\; 0 \tag{4.57}$$

$$\dot{\eta} \;=\; -\,(S_{\nu-1})_\eta^{-1}\,(S_{\nu-1})_\psi\,\dot{\psi} \tag{4.58}$$

meaning that then there are n new equations in n new variables $\dot{\eta}$ such that there are no further constraints on the "old" variables η and ψ.

Because of this relation the V-nodes of interest are the ones representing the highest derivatives of $(x, y)^T$. The E-nodes represent either original equations $F_i(x, \dot{x}, y) = 0$ or equations arising by differentiating those an appropriate number of times with respect to time.

The algorithm proposed by Pantelides uses this idea and basically, determines which equations have to be differentiated in order to obtain the system $S_{\nu-1}$ described above. This is done by examining the DAE equation by equation and attempting to assign pairs of highest derivatives (V-nodes) and equations (E-nodes). If it fails to accomplish an assignment after one more equation is added to the graph, it colors all V-nodes whose edges to the E-node that could not be assigned a V-node exist. It also colors all E-nodes that were assigned to now colored V-nodes. By this procedure one obtains a set of k colored E-nodes and $k-1$ colored V-nodes which represents a system of k equations in $k-1$ variables (highest derivatives) and is defined to be *minimally structurally singular.*

The formal outline of this algorithm below describes crudely the sequence of steps required to determine which sets of equations have to be differentiated in order to provide consistent initialization.

Algorithm 4.3.1:

Step 0: Initialization:

- number of equations examined already: $next = 0$
- no E-nodes in graph
- n V-nodes in graph representing $(\dot{x}, y)$

Step 1: $next \rightarrow next + 1$

pathfound = 'false'
Add equation $F_{next}(x, \dot{x}, y) = 0$ as an E-node to the graph
and call this node F_{ex}.

Step 2: Establish $edge(F_{ex}, j)$ if the variable represented by V-node j occurs in $F_{ex} \hat{=} F_{next} = 0$.

Step 3: Try to assign the E-node representing F_{ex} to an unassigned V-node j if $edge(F_{ex}, j)$ exists.
If possible set pathfound = 'true'

Step 4: If pathfound = 'true' and $next < n$ continue with **Step 1.**
If pathfound = 'true' and $next = n$ terminate.
If pathfound = 'false' continue with **Step 5.**

Step 5: Try to assign the E-node representing F_{ex} to a V-node by changing prior assignments
If possible such that no E-node stays unassigned
set pathfound = 'true' .

Step 6: If pathfound = 'true' and $next < n$ continue with **Step 1.**
If pathfound = 'true' and $next = n$ terminate.
If pathfound = 'false' continue with **Step 7.**

Step 7: Color all V-nodes for which $edge(F_{ex}, j)$ exists.

Step 8: Color E-node F_{ex} and all E-nodes which have assigned

a colored V-node

Step 9: Replace all colored E- and V-nodes by E- and V-nodes representing the time derivatives of what the same nodes represented before they were colored.

Step 10: Remove edges of replaced nodes and establish feasible edges of the new ("differentiated") nodes.

Step 11: Try to assign all E-nodes to V-nodes.
If possible set pathfound = 'true'.

Step 12: If pathfound = 'true' and $next < n$ continue with **Step 1.**
If pathfound = 'true' and $next = n$ terminate.
If pathfound = 'false' call the E-node without an assignment F_{ex} and continue with **Step 7.**

Clearly, in order to provide information which equation has been differentiated how often the algorithm requires additional bookkeeping which is not given explicitly in [37]. This can be done by having a counter $diff(i) \in \mathcal{R}^n$ which is incremented by one whenever the E-node representing equation $F_i(x, \dot{x}, y) = 0$ $i = 1(1)n$, of the DAE $F(x, \dot{x}, y) = 0$ is colored implying that this equation has to be differentiated (**Step 9**). A counter $hd(j) \in \mathcal{R}^n$ which is incremented by one whenever a V-node is colored yields the order of the highest derivative of (x, y) occurring in $S_{\nu-1}$ on termination if

$$\text{initially} \begin{cases} hd(j) = 0 & , \text{ for } z_j \in y \\ hd(j) = 0 & , \text{ for } z_j \in x \end{cases} . \tag{4.59}$$

With this the *structural index* ν_s can be determined to be

$$\nu_s = 1 + \max_{i=1(1)n} diff(i) \tag{4.60}$$

if one or several of the $hd(j)$ are zero on termination[3], if $hd(j) = 1$ for all $j = 1(1)n$ then the DAE is an ODE implying $\nu_s = 0$. The *structural number of degrees of freedom* r_s is

$$r_s = \sum_{j=1}^{n} hd(j) - \sum_{i=1}^{n} diff(i) . \tag{4.61}$$

This relation is not obvious but can be obtained by an argument that is practically identical to the one at the end of Appendix C.

The rather qualitative discussion above is meant to introduce the basic ideas of this approach in order to provide the basis for a comparison with the structural representations of *Algorithm 2.1.1* suggested in Section 4.2. For a more detailed discussion we refer to [37].

4.4 Comparison of the Structural Algorithms

The algorithm proposed by Pantelides was mainly designed to address the problem of consistent initialization. The structural representations of *Algorithm 2.1.1* given in Section 4.2 were motivated by the desire to have a tool that provides as many information on a DAE as possible with reasonable computational effort and time. Before differences due to these two different motivations are pointed out it should be emphasized that these two approaches yield the *same structural index* ν_s *and the same structural number of degrees of freedom* r_s, if they are applied to the same DAE, i.e., if the input *pat* F_z and *pat* $F_{\dot{z}}$ is identical for both.

[3]This does not hold if $hd(j) \leq 1$ for all $j = 1(1)n$ and if all equations that have been differentiated once (diff(i)=1) are used to determine variables $\dot{z}_j$. It is easy to verify that for such a situation the DAE considered has to be of index *one*. In an implementation this special case must be handled separately.

Algorithm 4.3.1 just carries out differentiations of original equations or their time derivatives. It does not carry out any substitutions such as *Algorithms 4.2.n/l* do. This basic difference could be characterized by calling Pantelides' approach *equation oriented.* In an analogous manner the approach in Section 4.2 could be called *variable oriented* since its goal is to solve for the time derivatives $\dot{z}$ of all variables z in the DAE $F(z, \dot{z}) = 0$ by means of differentiation *and* substitution. As indicated in Section 4.2 it is therefore rather simple to identify variables that cause a higher-index-problem if *Algorithms 4.2.n/l* are used. Due the inherent "inaccuracy" of structural differentiations (**Rule 5**) and the substitutions following each of the successive differentiations however, it is impossible to keep track of the number of differentiations applied to each *original* equation. This on the other hand is straightforward in Pantelides' approach. As pointed out in Section 2.3 this encourages us to think that a combination of this algorithm and the numerical initialization method proposed by Leimkuhler [31] would be an efficient solution to the problem of consistent initialization if symbolical manipulations want to be avoided.

For index reduction and for consistent initialization after symbolical manipulations the algorithms suggested in Section 4.2 seem to be better suited since they provide the sequence of steps that have to be taken in order to reduce the index by differentiations and substitutions.

Another advantage of those representations is that they can be stopped after having encountered that the structural index exceeds a certain limit. If this limit is one then it is even possible to use the results of the computations up to this point to formulate various suggestions how the index could be reduced by using a different modelling approach. Although it is in principle also possible to stop *Algorithm 4.3.1* after having encountered that the structural index exceeds one (or any other limit) it is not as desirable to do so since at such a point the algorithm generally just has examined a subset of equations rather than the entire DAE. Therefore, the result obtained at

such a point can in principle not be used to provide suggestions similar to the ones based on the structure of the *entire* DAE above.

With respect to computational effort the two approaches are not significantly different. In Chapter 5 further details on that issue are indicated based on implementations of both methods in FORTRAN 77. The performance of both implementations is demonstrated on various practical examples.

Chapter 5

Implementations and Examples

This chapter summarizes practical implications of the rather theoretical and formal considerations in preceeding chapters.

In Section 5.1 we indicate how the structural representations of *Algorithm 2.1.1*, i.e., *Algorithms 4.2.1 and 4.2.n*, and the structural algorithm proposed by Pantelides [37], i.e., *Algorithm 4.3.1*, can be implemented. Section 5.1 also describes the basic functionality of implementations of both approaches. A more detailed documentation of the actual codes developed is given in Appendix D. Their application on several examples in Section 5.2 demonstrates how they perform in practical problems and how results obtained by these implementations can be used to cope with practical problems that are frequently associated with DAE's. Finally, Section 5.3 provides a brief comparison of the performance of the two approaches with respect to both, the quality of the results and the computational effort and time required.

5.1 Implementations

Although both approaches are significantly different their implementations contain components that are identical for both approaches or that are closely related to each other.

As indicated before both approaches require just $pat\, F_z$ and $pat\, F_{\dot{z}}$ as input. Therefore, the input components allowing manual input of these patterns or specification of an appropriate input file are practically identical. Both algorithms terminate only if the DAE considered is structurally solvable,i.e., if the merged pattern $(pat\, F_z + pat\, F_{\dot{z}})$ is not structurally singular. To ensure termination even if a structurally unsolvable DAE is entered, both implementations contain a routine that checks if the DAE entered is structurally solvable and stops the terminates the program without further computations if the condition above is violated. It also provides some hints why the DAE could be structurally unsolvable. This routine has to carry out a *structural rank determination* for $(pat\, F_z + pat\, F_{\dot{z}})$. This is accomplished by an algorithm proposed by Duff [16]. The same rank determination plays an important role in both, the implementation of *Algorithms 4.2.l and 4.2.n* which is called ALGO as well as in the implementation of Pantelides' algorithm as represented in *Algorithm 4.3.1* which is called PALG.

5.1.1 ALGO – Implementation of Algorithms 4.2.l and 4.2.n

Based on the structural representations of *Algorithm 2.1.1* in Chapter 4 an implementation of this approach is pretty much straightforward. The central component in such a program is the *structural rank determination* required in **Step 1** of both, *Algorithm 4.2.l* and *Algorithm 4.2.n.* As indicated above technically this can be accomplished by Duff's *" Algorithm for Obtaining a Maximal Transversal"*. There are other methods available [26,29] but we chose this one since it is based on a settled approach and quite simple to implement.

ALGO was implemented such that prior to any computation the user can decide if differentiations are carried out according to **Rule 4.1** or according to **Rule 4.2**, i.e., if *Algorithm 4.2.l* or *Algorithm 4.2.n* is carried out. As emphasized earlier either one yields the same structural index ν_s and the same structural number of degrees

of freedom r_s. However, this distinction is important if after the computation of ν_s and r_s the user would like to check if a candidate set of r_s variables is feasible to be assigned arbitrary initial conditions. The examination of such a user specified set is optional.

Another option provided is that the user can indicate prior to any computation if he/she is only interested in whether the index exceeds one and how it could be reduced if it does so. If this option is chosen then ALGO terminates after encountering "$\nu_s > 1$" providing suggestions for index reduction based on structural heuristics. The physical feasibility of these suggestions has to be examined by the user.

If the user is interested in the actual values of ν_s and r_s then the option described above should be rejected. With this on termination ALGO yields ν_s and r_s and provides a sequence of steps to reduce the index by successive differentiations and eliminations. Futhermore, ALGO then provides the option to give more information on consistent initialization as described above.

5.1.2 PALG – Implementation of Algorithm 4.3.1

The formal representation of the algorithm proposed by Pantelides in *Algorithm 4.3.1* is different from its pseudo-code formulation in [37]. The formulation in [37] solved the assignment problem in **Steps 3,5,** and **11** of *Algorithm 4.3.1* very elegantly by means of a *recursive* procedure. However, this cannot be accomplished using FORTRAN 77. Examining Duff's algorithm used for structural rank determinations in the solvability check of both programs and in **Step 1** of ALGO one can see that this algorithm actually solves an assignment problem. Therefore, after some modifications it can be used to accomplish the assignment tasks in **Steps 3,5,** and **11** of *Algorithm 4.3.1.*

The way of keeping track of the number of differentiations applied to certain equations and variables of the DAE in PALG is also different from the one suggested in [37]. However, despite those technical differences the basic idea of the approach in PALG

is the one suggested in [37].

On termination PALG provides the value of the structural index ν_s and the structural number of degrees of freedom r_s. Additionally, the number of differentiations $diff(i)$ that have to be applied to each equation $F_i(z, \dot{z}) = 0$, $i = 1(1)n$ in order to obtain an index-one-system system is given.

A further component which could examine if a certain set of r_s variables to be assigned arbitrary initial conditions allows consistent initialization is principally possible. Such a component would be very similar to the corresponding one in ALGO. So far, we have not included such a component in PALG.

5.2 Examples

In this section we demonstrate capabilities and limitations of the two structural algorithms implemented in ALGO and PALG. The examples used also reveal that higher-index-models in practical applications can arise for various reasons. The first example is the fixed length pendulum mentioned at several points in preceeding chapters. It demonstrates that the choice of an appropriate coordinate system can avoid higher-index-problems. This example is also used to demonstrate the two different options in ALGO concerning consistent initialization. The model of a distillation column is considered as the second example. It illustrates the impact of certain modelling assumptions on the index. As a further example we indicate that dynamic design can yield models of fairly large index. Three practical applications of that type are given. Finally, in the fourth example a PDAE is considered. This example is meant to demonstrate that both, ALGO and PALG can to a fair extend also be used to examine PDAE's. It is also meant to motivate further theoretical work in the field of PDAE's.

Example 5.1: Fixed Length Pendulum

This system (Fig. 5.1) can be modeled in two different coordinate systems:

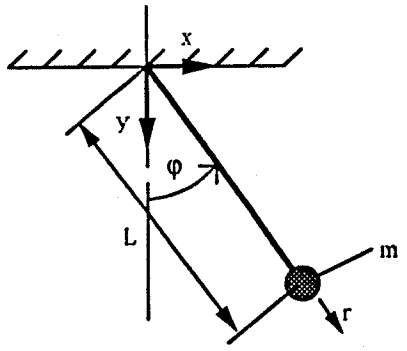


Figure 5.1: Possible Coordinate Systems and Parameters

1. *Polar Coordinates* φ, r:

   With $\dot{\varphi} = \psi$ and λ being the force in the bar the balance on the angular momentum about the bearing

$$\ddot{\varphi} = \frac{g}{L} \sin\varphi \tag{5.1}$$

   and the balance on the forces on the bar

$$\lambda = m L \dot{\varphi}^2 + m g \cos\varphi \tag{5.2}$$

   along with the constraint $r = L$ can be rewritten:

$$\dot{\varphi} = \psi \tag{5.3}$$

$$\dot{\psi} = \frac{g}{L} \sin\varphi \tag{5.4}$$

$$0 = r - L \tag{5.5}$$

$$0 = \lambda - m L \psi^2 - m g \cos\varphi. \tag{5.6}$$

   Evidently, this is an *index-one-system* since after one differentiation with respect to time (5.5) and (5.6) yield

$$\dot{r} = 0 \tag{5.7}$$

$$\dot{\lambda} = 2mL\psi\dot{\psi} - mg\dot{\varphi}\sin\varphi \tag{5.8}$$

and after substitution of $\dot{\varphi}$ and $\dot{\psi}$ one obtains

$$\dot{\lambda} \;=\; mL\psi \sin\varphi\,, \tag{5.9}$$

which along with (5.3)–(5.7) forms the corresponding extended system. Since the corresponding extended system contains two algebraic equation and since the dimension of the model (5.3)–(5.6) is four the number of degrees of freedom of this system is $4 - 2 = 2$.

2. *Cartesian Coordinates* x, y:
Using the Lagrange Formalism with

$$z_1 \;=\; x\,,\quad z_2 \;=\; y\,,\quad z_3 \;=\; \dot{x}\,,\quad z_4 \;=\; \dot{y}\,,\quad z_5 \;=\; \lambda\,,$$

the model of the pendulum becomes

$$\dot{z}_1 \;=\; z_3 \tag{5.10}$$

$$\dot{z}_2 \;=\; z_4 \tag{5.11}$$

$$\dot{z}_3 \;=\; \frac{2}{m}\, z_1\, z_5 \tag{5.12}$$

$$\dot{z}_4 \;=\; \frac{2}{m}\, z_2\, z_5 \;-\; g \tag{5.13}$$

$$0 \;=\; z_1^2 \;+\; z_2^2 \;-\; L^2\,. \tag{5.14}$$

By carrying out *Algorithm 2.1.1 symbolically* one obtains the hidden equations: after the first differentiation:

$$0 \;=\; z_1\, z_3 \;+\; z_2\, z_4\,, \tag{5.15}$$

after the second differentiation:

$$0 \;=\; m(z_3^2 \;+\; z_4^2) \;+2(z_1^2 \;+\; z_2^2)\, z_5 \;-\; mg\, z_2\,, \tag{5.16}$$

and

$$\dot{z}_5 = -\frac{2}{z_1^2 + z_2^2}(z_1 z_3 z_5 + z_1 z_3 z_5 - g z_4) + \frac{mg}{2}\frac{z_4}{z_1^2 + z_2^2} \tag{5.17}$$

after the third differentiation. Therefore, the index of this representation is three, whereas the number of degrees of freedom remains two since the corresponding extended system (5.10)–(5.17) contains three algebraic equations the original model (5.10)–(5.14) consists of five equations.

The brief discussion above demonstrates that the index can depend on the coordinate system used to describe the geometry of the system, whereas the number of degrees of freedom does not. This ties in with the interpretation of the number of degrees of freedom as the number of independent storages in the system in Section 2.3. One can verify easily that the pendulum has two independent storages: kinetic and potential energy. Therefore, one can specify an initial velocity and an initial position. For the polar coordinate representation a natural choice would be $(\dot{\varphi}_0, \varphi_0)$ for the cartesian coordinates an analogous choice could be either $(\dot{x}_0, x_0)$,$(\dot{x}_0, y_0)$, $(\dot{y}_0, y_0)$, $(\dot{y}_0, x_0)$ but neither one of $(\dot{x}_0, \dot{y}_0)$ (x_0, y_0) since each of these pairs is related to only one of the two storages.

Using the representation in cartesian coordinates we now demonstrate what difference it makes if ALGO is run according to *Algorithm 4.2.1* or according to *Algorithm 4.2.1.* As indicated earlier *Algorithm 4.2.1* considers the DAE to be *linear*. In this case the linearized representation of (5.10)–(5.14) becomes

$$\dot{\xi}_1 = \xi_3 \tag{5.18}$$

$$\dot{\xi}_2 = \xi_4 \tag{5.19}$$

$$\dot{\xi}_3 = a^0_{31}\xi_1 + a^0_{35}\xi_5 \tag{5.20}$$

$$\dot{\xi}_4 = a^0_{42}\xi_2 + a^0_{45}\xi_5 \tag{5.21}$$

$$0 = a^0_{51}\xi_1 + a^0_{52}\xi_2\,, \tag{5.22}$$

such that the corresponding extended system after three differentiations and substitutions consists of

$$0 = a^1_{53}\xi_3 + a^1_{54}\xi_4 \tag{5.23}$$

$$0 = a^2_{51}\xi_1 + a^2_{52}\xi_2 + a^2_{55}\xi_5 \tag{5.24}$$

$$\dot{\xi}_4 = a^3_{53}\xi_3 + a^0_{54}\xi_4 \tag{5.25}$$

and equations (5.18)–(5.22), where a^k_{ij} denotes the (constant) coefficient of variable j in the equation obtained by differentiating equation i k times with respect to time. If the analogous procedure is carried out according to *Algorithm 4.2.n* then (5.10)–(5.14) is considered as

$$F_1(\dot{z}_1, z_3) = 0 \tag{5.26}$$

$$F_2(\dot{z}_2, z_4) = 0 \tag{5.27}$$

$$F_3(\dot{z}_3, z_1, z_5) = 0 \tag{5.28}$$

$$F_4(\dot{z}_4, z_2, z_5) = 0 \tag{5.29}$$

$$F_5(z_1, z_2) = 0\,. \tag{5.30}$$

The structural representation of this is

$$pat\,F_{\dot{z}} = \begin{pmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad pat\,F_z = \begin{pmatrix} 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & * \\ 0 & * & 0 & 0 & * \\ * & * & 0 & 0 & 0 \end{pmatrix}. \tag{5.31}$$

After one (trivial) structural Gauss Elimination one obtains

$$\dot{z}_1 = \phi_1(z_3) \tag{5.32}$$

$$\dot{z}_2 = \phi_2(z_4) \tag{5.33}$$

$$\dot{z}_3 = \phi_3(z_1, z_5) \tag{5.34}$$

$$\dot{z}_4 = \phi_4(z_2, z_5) \tag{5.35}$$

and

$$G_0(z_1, z_2) \;=\; 0\,. \tag{5.36}$$

The first differentiation and substitution yields

$$(G_0)_{z_1}\,\phi_1(z_3) \;+\; (G_0)_{z_2}\,\phi_2(z_4) \;=\; 0\,. \tag{5.37}$$

Since *Algorithm 4.2.n* differentiates according to **Rule 4.1** this can be rewritten

$$G_1(z_1, z_2, z_3, z_4) \;=\; 0\,. \tag{5.38}$$

With this the second differentiation and substitution yield

$$(G_1)_{z_1}\,\phi_1(z_3) \;+\; (G_1)_{z_2}\,\phi_2(z_4) \;+\; (G_1)_{z_3}\,\phi_3(z_1, z_5) \;+\; (G_1)_{z_4}\,\phi_4(z_1, z_5) \;=\; 0\,, \tag{5.39}$$

which due to differentiation according to **Rule 4.1** yields

$$G_2(z_1, z_2, z_3, z_4, z_5) \;=\; 0\,. \tag{5.40}$$

After one more differentiation, substitution,and elimination this becomes

$$\dot{z}_5 \;=\; \phi_5(z_1, z_2, z_3, z_4, z_5)\,. \tag{5.41}$$

This indicates that the index and the number of degrees of freedom is the same in all three computations, the symbolical, according to *Algorithm 4.2.l*, and according to *Algorithm 4.2.n*. By *Proposition 2.1.3* each corresponding extended system can be represented by

$$\dot{z} \;=\; \phi(z) \tag{5.42}$$

$$0 \;=\; K(z)\,, \tag{5.43}$$

where $z, \phi \in \mathcal{R}^n$ and $K \in \mathcal{R}^q$, $q \leq n$. If the Jacobians with respect to $\dot{z}, z$ of the corresponding extended systems obtained from symbolical computations, *Algorithm*

4.2.l, and *Algorithm 4.2.n* are denoted J^s, J^l, and J^n then the patterns of these Jacobians are

$$pat\, J^s \equiv pat\, J^n = \begin{pmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ 0 & 0 & * & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 & * \\ 0 & 0 & 0 & 0 & * & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \end{pmatrix}$$

and

$$pat\, J^l = \begin{pmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ 0 & 0 & * & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 & * \\ 0 & 0 & 0 & 0 & * & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & * \end{pmatrix}.$$

Evidently, $pat\, J^l$ has *less* nonzero entries than $pat\, J^s \equiv pat\, J^n$. In Sections 2.3 and 4.1 we introduced a procedure that allows to check if a certain set of variables is feasible for to be assigned arbitrary initial conditions. Certainly, one expects that the results of this procedure depend on which of the pattern above is used. All three patterns become structurally singular patterns if columns 1,2 or columns 3,4 are removed at the same time. This can be expected from physical insight since the variable pairs corresponding to these pairs of columns either consist of only positions (z_1, z_2) or only velocities (z_3, z_4). Therefore, by the argument with the independent storages and by looking at the symbolical representation one can see that (x_0, y_0) and $(\dot{x}_0, \dot{y}_0)$ do not represent sets that are feasible to be assigned arbitrary initial conditions. Removing columns 1,5 or columns 2,5 from $pat\, J^l$ renders a structurally singular pattern. However, if the same is done in $pat\, J^s \equiv pat\, J^n$ this renders a structurally non-singular pattern. From physical insight one would also expect that

(x_0, λ_0) or (y_0, λ_0) would be feasible since the force in the bar involves both, position and velocity. Therefore, specifying λ_0 in this case indirectly specifies an initial velocity. This demonstrates one part of the practical implications of *Proposition 4.2.2*: If based on computations according to *Algorithm 4.1.1* a set of variables is ruled feasible to be assigned arbitrary initial conditions then it *is* feasible. If as set cannot be ruled feasible by this argument then it does *not necessarily* have to be infeasible.

On the other hand one can also imagine that there are cases where the pattern $pat\, J^n$ obtained from *Algorithm 4.2.n* has by far more nonzero entries than $pat\, J^s$ such that sets variable for which the corresponding submatrix in $pat\, J^s$ actually becomes singular do not cause a structural singularity in in the corresponding submatrix of $pat\, J^n$.

This illustrates that $pat\, J^l$ and $pat\, J^n$ provide *bounds* on the actual pattern $pat\, J^s$ that would arise in a symbolical computation. $pat\, J^l$ contains the *minimum* number of nonzeros and $pat\, J^n$ contains the *maximum* number of nonzeros possible for $pat\, J^s$ if numerical and symbolical cancellation is neglected. Therefore, $pat\, J^l$ alone cannot be used to decide whether a set is *infeasible* since it might contain *less* nonzeros than $pat\, J^s$, whereas $pat\, J^n$ alone cannot be used to decide whether a set is *feasible* since it might contain *more* nonzeros than $pat\, J^s$.

In his publication Pantelides [37] also showed how his approach can be used to detect if a set is feasible. However, he only mentioned the necessary condition for feasibility of a set, even though his algorithm is also capable of providing information that can be used to rule a set infeasible.

Summarizing, this example showed that the index but not the number of degrees of freedom can depend on the coordinate system chosen. For both representations the results for the structural index ν_s and the structural number of degrees of freedom r_s obtained from ALGO and PALG are identical.

The effects of running ALGO according to *Algorithm 4.2.1* or according to *Algorithm*

4.2.n on the characterization of candidate sets to be assigned arbitrary initial conditions were illustrated.

Example 5.2: Reactive Distillation Column

The problems pointed out in this example were encountered during the development

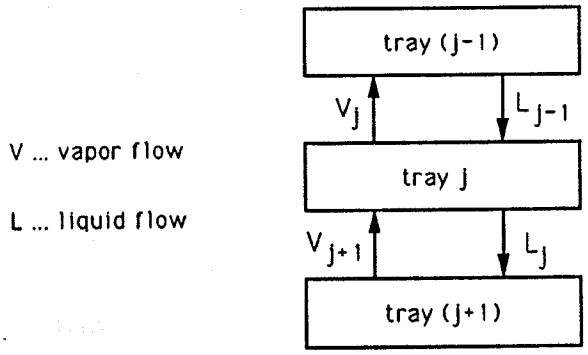


Figure 5.2: Notation for trays of the Column

of a module for the CAD-package POLYRED [4]. The goal of this work was to model a reactive distillation column for homopolycondensation. The numerical problems during this development were solely caused by the distillation. The complex kinetics could be represented by first order ordinary differential equations involving reaction rates that generally depend on the concentrations of the species and solvent involved and on almost all other variables of the system such as flow rates, density, temperature and pressure. The kinetics could be represented by a subsystem of ordinary differential equations which did not affect the considerations below. Therefore, in this example we only consider a simple non-reactive distillation model.

Energy and mass balances along with thermodynamical and flow hydraulic relations

on tray j of the column yield

$$\dot{m}_j = V_{j+1} + L_{j-1} - V_j - L_j \tag{5.44}$$

$$m_j \dot{H}_j^l + \dot{m}_j H_j^l = V_{j+1} H_{j+1}^v + L_{j-1} H_{j-1}^l - V_j H_j^v - L_j H_j^l + Q_j \tag{5.45}$$

$$0 = m_j - m(L_j, \rho_j^l) \tag{5.46}$$

$$0 = y_{i,j} - y_{i,j+1} - E_{MV_{i,j}} (y_{i,j}^* - y_{i,j+1}) \ , \quad i = 1(1)N^c - 1 \tag{5.47}$$

$$0 = 1 - \sum_{i=1}^{N^c} y_{i,j} \tag{5.48}$$

$$0 = 1 - \sum_{i=1}^{N^c} y_{i,j}^* \tag{5.49}$$

$$0 = H_j^l - H^l(p_j, T_j) \tag{5.50}$$

$$0 = p_j - p_{j-1} - P_{dr}(V_j, y_{i,j}, h_{Lj-1}, \rho_{j-1}^l) \tag{5.51}$$

if no feed is assumed on tray j. For completeness it should be indicated that in addition to these equations there should be a thermodynamic relation for the vapor enthalpy H_j^v and mass balances for the components of the liquid. Another simplification is that we omitted the dependency on the composition of the liquid in (5.50). These additional relations are required for physical reasons. However, they do not have any impact on the considerations below. Therefore, we omitted them in order to shorten notation. Using further thermodynamical equations of state to obtain the equilibrium mole fractions $y_{i,j}^* = y_{i,j}^*(T_j, p_j)$ (5.49) basically determines the (equilibrium) temperature T_j. This model can be shown to be of *index one*. In practice the pressure drop between two stages represented by the last equation above is frequently negligible. Additionally, it is hard to get realistic data for the pressure drop correlation P_{dr}. Therefore, one would like to neglect this pressure drop by omitting the last equation above and assuming $p_j = p_0$ for all stages. However, this yields a DAE (5.44)–(5.50) of *index two*. This can be verified by observing that now V_j can no

longer be determined from equation (5.51) since it has been removed from the DAE. Therefore, (5.50) is differentiated and (5.45) is substituted into the result. One more differentiation of the equation obtained yields $\dot{V}_j$ such that the index is two. If the patterns of this DAE are entered to ALGO and PALG then both obtain $\nu_s = 2$ and $r_s = 1$. Additionally ALGO identifies the vapor flow rate V_j to be the variable that causes the higher-index-problem.

In a first first modeling approach equations (5.47)–(5.49) were replaced by

$$0 = \frac{v_{i,j}}{\sum_{i=1}^{N^c} v_{i,j}} - \frac{v_{i,j+1}}{\sum_{i=1}^{N^c} v_{i,j+1}} - E_{MV_{i,j}}\,(y^*_{i,j} - \frac{v_{i,j+1}}{\sum_{i=1}^{N^c} v_{i,j+1}}) \quad i = 1(1)N^c \tag{5.52}$$

$$0 = V_j - M^v_j \sum_{i=1}^{N^c} v_{i,j}\,, \tag{5.53}$$

which has two major drawbacks:

1. Equation (5.49) which determines the temperature T_j does no longer appear explicitly in the model represented by (5.43)–(5.46),(5.50) (5.52),(5.53). By summing (5.52) over all N^c components one can see that T_j is in this model determined by

$$\sum_{i=1}^{N^c} \left[E_{MV_{i,j}}\,(y^*_{i,j} - \frac{v_{i,j+1}}{\sum_{i=1}^{N^c} v_{i,j+1}}) \right] = 0 \tag{5.54}$$

which only for $E_{MV_{i,j}} \equiv E_{MV_j}$ for all components $i = 1(1)N^c$ is the same as the physically correct relation (5.49). Therefore, this approach computes a wrong temperature if $E_{MV_{i,j}}$ is not he same for all components. For the reason this $E_{MV_{i,j}}$ does not guarantee that the closure condition (5.49) is satisfied.

2. Both, ALGO and PALG terminate with $\nu_s = 1$, although equations (5.43)-(5.46),(5.50),(5.52),(5.53) represent a DAE of *index* two. The reason for this can be seen by considering *pat* F_z and *pat* $F_{\dot{z}}$ for $N^c = 2$ and

$$z_1 = H^l_j\,,\ z_2 = V_j\,,\ z_3 = L_j\,,\ z_4 = m_j\,,\ z_5 = v_{1,j}\,,\ z_6 = v_{2,j},\,,\ z_7 = T_j$$

$$pat\, F_{\dot{z}} = \begin{pmatrix} 0 & 0 & 0 & * & 0 & 0 & 0 \\ * & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$pat\, F_z = \begin{pmatrix} 0 & * & * & 0 & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * & * \\ 0 & * & 0 & 0 & * & * & 0 \\ * & 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Structurally the the two equations in (5.52) appear to be regular with respect to $v_{1,j}, v_{2,j}$. However, considering these equations symbolically one observes that they can only be solved for

$$\frac{v_{1,j}}{\sum_{i=1}^{N^c} v_{1,j}} \quad \text{and} \quad \frac{v_{2,j}}{\sum_{i=1}^{N^c} v_{2,j}}$$

rather than for the absolute values of $v_{1,j}, v_{2,j}$. Therefore, (5.53) cannot be used to determine V_j. By Carrying on these symbolical considerations one sees that V_j has to be determined by differentiating (5.49) and substituting (5.45)[1] such that after one more differentiation $\dot{V}_j$ is obtained and the index is two.

This demonstrates limitations of a structural approach.

For the reasons indicated above this last approach is not good, since it can lead to physically wrong results and additionally is difficult to be characterized by a *structural* analysis.

[1] In order to focus on the major point of this discussion we omitted the dependency of M_j^v on the vapor composition. Physically, M_j^v always depends on this composition. However, the formal discussion here is not affected by it other than making the discussion more complicated than required for our purposes.

Another assumption leading to an index two DAE is to assume constant liquid holdup on a tray, i.e., to replace equation (5.46) by

$$m_j = \text{constant} . \tag{5.55}$$

In order to obtain $\dot{m}_j$ this equation has to be differentiated and substituted into (5.44) such that after one more differentiation $\dot{L}_j$ can be obtained. Both, ALGO and PALG are capable to detect this problem and ALGO additionally identifies L_j as the variable that causes the problem.

So far,this example seems to motivate a general statement saying that neglecting certain physical phenomena such as in this example the pressure drop over the column or the the changing liquid holdup *always* leads to higher-index-problems. However, this is not true in general. If the enthalpy of the liquid in (5.44) of (5.43)–(5.50) is assumed to be in quasi-steady state, i.e., $\dot{H}_j^l \equiv 0$, then the model is of index one. This is true since in this case V_j can be computed from (5.44) and after just one differentiation $\dot{V}_j$ is determined from this equation while $\dot{H}_j^l$ can be determined from (5.50) after one differentiation. The frequently used quasi-steady state assumption above was also suggested by ALGO as one possible way of reducing the index.
Therefore, it is generally not true that simplifications always increase the index. However, the quasi-steady state assumption for H_j^l above is only required if the pressure drop over the column was neglected in the first place.
As a *heuristical* guideline it can be stated that higher-index-problems can generally be avoided by modeling in such a way that one is aware of which equation is intended to be used to determine which variable. An equation that is meant to determine a certain variable should have this variable appearing in it. This means that a situation similar to the one described after replacing (5.47)–(5.49) by (5.52),(5.53) should be avoided.

Example 5.3: Dynamic Design

As already mentioned in the introduction higher-index-systems can also arise from *dynamic design* where one or more variables are assigned desired time courses [38]. Dynamic design can be used to determine the inputs required in order to obtain desired outputs. Especially rather high indices are frequently due to dynamic design. We demonstrate this on three examples:

1. One CSTR with a desired exit concentration
2. A cascade of five CSTR's with a desired exit concentration
3. A distillation column whose bottom pressure is specified.

The last example above shows the relation between dynamic design and coupled systems which can be obtained by connecting several unit operations in a flowsheet.

One CSTR with a desired exit concentration

$$\dot{c} = k_1 (c_0 - c) - R \tag{5.56}$$

$$\dot{T} = k_2 (T_0 - T) - k_3 (T - T_c) \tag{5.57}$$

$$0 = R - k_4 c e^{\frac{k_5}{T}} \tag{5.58}$$

$$0 = c - w(t) , \tag{5.59}$$

where k_i, $i = 1(1)5$, are constants, the subscript 0 denotes feed conditions. T_c is the cooling temperature whose time course has to be designed such that the concentration c equals $w(t)$. This system can be shown to be of index three and to have no degree of freedom. With

$$z_1 = c\,,\ z_2 = T\,,;\ z_3 = R\,,\ z_4 = T_c$$

$$pat\, F_{\dot{z}} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad pat\, F_z = \begin{pmatrix} * & 0 & * & 0 \\ 0 & * & * & * \\ * & * & * & 0 \\ * & 0 & 0 & 0 \end{pmatrix},$$

are the patterns to be entered to ALGO and PALG. Both obtain the same structural index $\nu_s = 3$ and the same structural number of degrees of freedom $r_s = 0$, which were expected for this system.

CSTR-cascade with desired exit concentrations:

For $N_r = 5$ CSTR's the cascade introduced in *Example 1.2.2* can be represented by

$$pat\, F_{\dot{z}} = \begin{pmatrix} 0 & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad pat\, F_z = \begin{pmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix},$$

if it is considered structurally and if $z_i = c_{i-1}$, $i = 1(1)6$, c_0 denotes the feed concentration of the first tank an c_i, $i = 1(1)5$, denotes the concentration in tank i. In Chapter 1 this system is shown to be of index $\nu = N_r + 1 = 6$ and to have no degree of freedom. Both, PALG and ALGO obtain these results.

It should be emphasized that c_0 has to be considered a *state* rather than an *input* variable if dynamic design computations are carried out. If this is not done then the system is *overdetermined* and therefore not solvable. From physical insight this is quite obvious since by specifying the exit concentration of the entire cascade one specifies the only input variable of the system, c_0.

Dynamic Distillation

This model is considered more detailed in [38]. At this point we only present the detailed patterns of the system if two stages are considered and based on this give a more crude representation of the pattern for more stages. The patterns of F_z and $F_{\dot{z}}$

for two stages with specified bottom pressure are shown in Figure 5.3. The representation in this figure are the actual input patterns required by ALGO and PALG. The number at the beginning, here 22, represents the dimension n of the entire DAE. The first pattern is *pat* $F_{\dot{z}}$, the second is *pat* F_z, nonzeros are denoted by "1" since both programs are based on *integer computations.* Each stage is represented by eleven equations. Evidently, the patterns of each stage repeat themselves for each stage added. The entire column of N_s stages with specified bottom pressure is known to be of index $\nu = N_s + 1$. Both programs were tested for $N_{s_1} = 1$, $N_{s_2} = 2$, and $N_{s_3} = 3$ and yielded identical results which were $\nu_{s_1} = 2$, $r_{s_1} = 2$, $\nu_{s_2} = 3$, $r_{s_2} = 4$, and $\nu_{s_3} = 4$, $r_{s_3} = 6$,respectively. For $N_s > 3$ the analogous patterns are of the form

$$\begin{pmatrix} S & C_D & 0 & 0 & 0 & 0 & 0 \\ C_U & S & C_D & 0 & 0 & 0 & 0 \\ 0 & C_U & S & C_D & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ 0 & 0 & C_U & S & C_D & 0 & 0 \\ 0 & 0 & 0 & C_U & S & C_D & 0 \\ 0 & 0 & 0 & 0 & C_U & S & C_D \end{pmatrix}$$

where all entries are $(n \times n)$-matrices and C_D denotes the coupling of a tray j to the next tray $j+1$ below itself and C_u denotes the analog matrix for the upward coupling. S represents relations among variables on the tray. Each of the trays is of index two and has to degrees of freedom. However, by coupling those index-two-systems the index increases by one with every stage added. This is different from the distillation model in *Example 5.2.2*, where each tray is also of index two and so is the entire column *independently* from the number of trays. This shows that the structure of the coupling of subsystems denoted by C_U and C_D above plays an important role if the index of the entire system ought to be determined based on the known indices of its subsystems.

It should be noted that the considerations above also indicate that the index of subsystems of a DAE depends on how the DAE is decomposed in order to obtain

these subsystems. For numerical reasons, not directly related to the index, it is sometimes desirable to carry out such decompositions [44] one has to be aware of the impact of decompositions on the index.

So far there it is only possible to predict the index of a system consisting of subsystems of known index and limited structure [46]. There is no general relation which would allow to compute the index of a joint system just from the knowledge of the indices of its subsystems without taking into account the coupling structure.

Example 5.4: PDAE arising from flow through heated tube

Finally, we want to demonstrate briefly that the structural approach can also be used to determine the (time) index ν_t of a PDAE. Formally, we have not defined any index of a PDAE. It should be emphasized that the method below is just a very first concept how index problems of PDAE's could be tackled. This approach is shown to work in the small example below. However, we do not consider boundary conditions which in this approach would be represented by further algebraic equations. This indicates that further investigations in this area are required.

Since a PDAE basically contains derivatives of with respect to more independent variables, in practice frequently time and up to three spacial coordinates, one can expect that there is more than one type of index. In practical applications one could consider a *time index* ν_t along with up to three spacial indices, e.g. , ν_x, ν_y, ν_z, in cartesian coordinates (x, y, z). However, at this point we do not want to get into a theoretical discussion, but rather want demonstrate how the programs PALG and ALGO can be used for PDAE's. Therefore, we consider the model describing the flow through a heated tube [17]. The modeling equations can be derived from the

equations of motion and continuity along with an energy balance:

$$\frac{\partial \rho}{\partial t} = -w\frac{\partial \rho}{\partial z} - \rho\frac{\partial w}{\partial z} \tag{5.60}$$

$$\frac{\partial w}{\partial t} = -w\frac{\partial w}{\partial z} - \frac{1}{\rho}\frac{\partial p}{\partial z} \tag{5.61}$$

$$\frac{\partial T}{\partial t} = -w\frac{\partial T}{\partial z} + \frac{1}{\rho}(T_H - T) - \frac{p}{\rho c_v}\frac{\partial w}{\partial z} \tag{5.62}$$

Two possible ways of considering the fluid are

1. with pressure and temperature dependent density:

$$\rho = \frac{p}{\mathrm{R}T}\,, \tag{5.63}$$

2. with pressure independent but temperature dependent density:

$$\rho = \rho_0\left[1 + \gamma(T - T_B)\right]\,. \tag{5.64}$$

If now the partial derivatives with respect to the spacial coordinate z are approximated by some finite difference scheme which involves the value of the variable at the current point (t, z) then for structural purposes

$$\frac{\partial \rho}{\partial z} = \rho_{fd}(t, z, \rho) \tag{5.65}$$

$$\frac{\partial w}{\partial z} = w_{fd}(t, z, w) \tag{5.66}$$

$$\frac{\partial T}{\partial z} = T_{fd}(t, z, T) \tag{5.67}$$

$$\frac{\partial p}{\partial z} = p_{fd}(t, z, p) \tag{5.68}$$

can be assumed and substituted into (5.60)–(5.62). With this one can easily verify that the time index ν_t of the DAE arising from the PDAE due to discretization is *one* if (5.63) is used for the density and that it is *two* if (5.64) is used. The number

of degrees of freedom for the first case is three, for the second it is two.
With the discretization and

$$z_1 = \rho\,,\ z_2 = w\,,\ z_3 = T\,,\ z_4 = p$$

the patterns of the DAE arising are

$$pat\,F_{\dot{z}} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\,,\quad pat\,F_z = \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & * \\ * & * & * & * \\ * & 0 & * & * \end{pmatrix}$$

if (5.63) is used and

$$pat\,F_{\dot{z}} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\,,\quad pat\,F_z = \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & * \\ * & * & * & * \\ * & 0 & * & 0 \end{pmatrix}$$

if (5.64) is used. If the patterns above are given to PALG and ALGO then for pressure dependent density both obtain $\nu_s = 1$ and $r_s = 3$. For pressure independent density both obtain $\nu_s = 2$ and $r_s = 2$, which in both cases is consistent with the results given in the beginning.

Concluding this small example it should be noted again that for PDAE's so far there is no formal definition of the index. Motivated by the brief discussion above one can expect that a PDAE has as many indices as there are independent variables involved in it. With discretizations similar to the ones above it is always possible to transfer any PDAE into a DAE whose time index ν_t is well defined and can be determined by ALGO and PALG in the structural sense.

5.3 Comparison of PALG and ALGO

Concluding this chapter on practical applications of the theoretical considerations in preceeding chapters we briefly compare the programs PALG and ALGO with respect

to results and the computational effort involved.

As expected from structural considerations in Chapter 4 ALGO and PALG yield identical results for the structural index ν_s and the structural number of degrees of freedom r_s if they are provided with identical input. This has been demonstrated on several examples in the preceeding section. Both are capable of providing information on the DAE beyond ν_s and r_s. As indicated earlier PALG is based on an equation oriented approach, whereas ALGO is based on a variable oriented approach. Therefore, the ranges of additional information obtained from both programs are slightly different. The information provided by PALG is very well suited to address consistent initialization, whereas ALGO provides further details on what causes the index to exceed one and how it could be reduced by symbolical manipulations. However, it should be emphasized that ALGO can also be used to cope with the problem of consistent initialization to some extend, in the manner PALG also provides information useful for index reduction.

The central component of both programs is based on Duff's "Algorithm to Obtain a Maximal Transversal"[16]. The computational effort of this component is proportional with n^3, where n number of equations and variables in the DAE. There might be slight differences in computational times required by PALG and ALGO depending on the particular structure of the DAE. However, since the components dominating the computational effort in both programs are very similar, we do not expect significant differences. For all examples of the preceding section the computational times on a VAX-workstation where significantly below one minute. This was achieved without even aiming for the most efficient implementations.

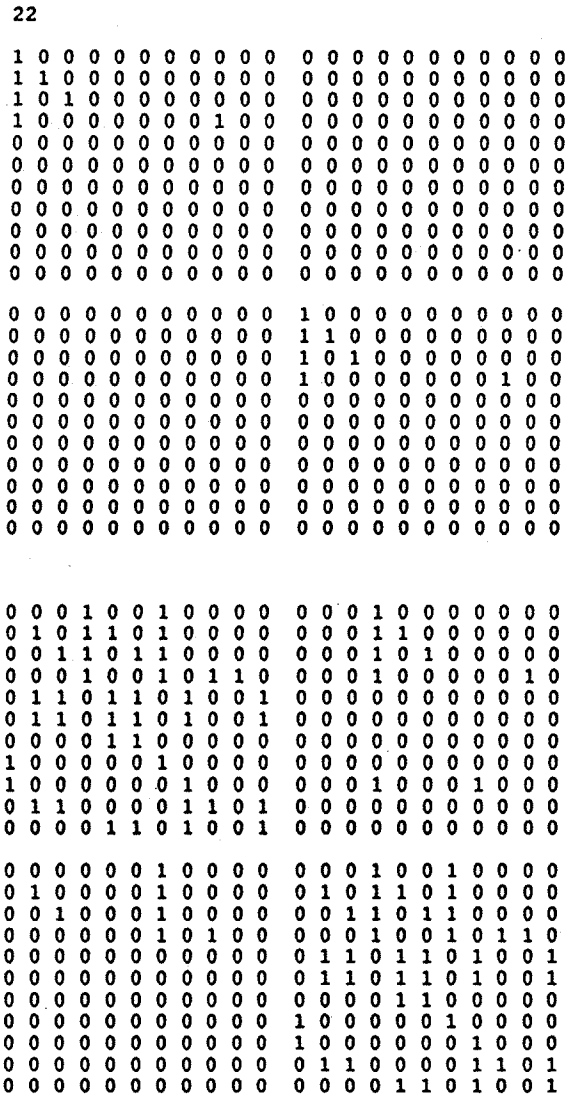


Figure 5.3: Structural information on the distillation column with two stages entering PALG and ALGO. Nonzero entries are denoted by "1":
top: dimension of the system $n = 22$, middle: *pat* $F_{\dot{z}}$, bottom: *pat* F_z.

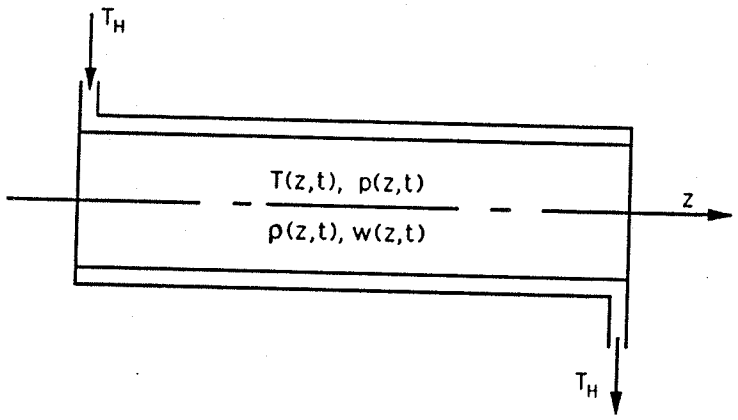


Figure 5.4: Heated tube

Chapter 6

Conclusions and Further Work

6.1 Conclusions

Realistic mathematical models of various applications are quite frequently represented by DAE's. Only recently it has been discovered that their numerical treatment can be significantly more difficult than the numerical integration of ODE's. The key characteristics of a DAE in terms of its numerical solution is its *index*. The numerical integration of index-one-systems is generally not much more complicated than the integration of ODE's [19]. However, if the index of a DAE exceeds one then its numerical treatment is no longer straightforward. In Chapter 3 we indicated three basic strategies how this problem can be addressed. All three of them require in one way or another that the index of the DAE is known. Currently, there is only a limited number of numerical methods and codes that can cope with higher-index-DAE's. Most of them are limited to indices ≤ 3 and/or to DAE's of limited structure, e.g., semi-explicit or in Hessenberg Form. We also indicated that finding consistent initial conditions, which are required by certain methods such as BDF to start the numerical integration, is generally not straightforward, even for certain types of index-one-systems.

Since DAE's are known to be very significant in chemical process modeling and

since they can cause severe numerical problems if not handled appropriately it is desirable to have methods and tools to characterize them. Motivated by the discussion above this work was aiming for two basic goals:

1. With a concise review of the literature on the *theory of DAE's* we wanted to provide the basis for a better understanding of important properties of DAE's.

2. Based on the theory of DAE's we were looking for a manageable way of providing a reliable *characterization of DAE's* prior to any numerical treatment. This characterization should contain all information on the DAE required for a successful and efficient numerical solution.

Based on various publications mostly in the fields of applied mathematics and numerical analysis we provided a review from an *engineering* point of view in Chapter 2. This review revealed important relations among key characteristics of DAE's. We also provided a geometrical illustration of these properties and characteristics of DAE's which is meant to make problems associated with DAE's more vivid.

Motivated by the second goal above and by Pantelides' approach we considered the DAE from a *structural* point of view. The key properties defined in Chapter 2 were redefined in the (new) relaxed structural sense. It could be shown that the *structural index* ν_s represents a *lower bound* on the local index of DAE at any point along solution. In many cases it turned out to be the a very accurate bound which quite frequently equals the (symbolical) index that would be obtained by symbolical computations. Therefore, for practical purposes the structural index is a useful characteristics of a DAE which can be used to decide on an appropriate numerical treatment.

Structural considerations were introduced since they appeared to be the only approach which would yield a fast and computationally tractable characterization of a DAE.

Practical questions arising were, how much and which structural information on the DAE is needed in order to determine its structural properties and how a structural characterization could actually be accomplished. Looking at Pantelides' approach the relation between the effort required and the amount of information provided by it was not obvious. To our knowledge there was no implementation of this method when this work started.

Therefore, we decided to try a different approach and then compare it to Pantelides' method both, theoretically and practically after an implementation of both. In the other approach we considered the structural representation of a conceptual algorithm proposed earlier by Gear [22]. First we formulated this algorithm rigorously. Then structural representations of it were derived.

An important result of this work is that the structural index and the structural number of degrees of freedom of a DAE $F(z, \dot{z}, u) = 0$ only depend on $pat\, F_z$ and $pat\, F_{\dot{z}}$, i.e., they solely depend on the pattern of appearance of variables in equations.

This result makes it possible to actually carry out the structural representations of Gear's algorithm and obtain results that are consistent with structural definitions in Chapter 4. Furthermore, it shows that from a theoretical standpoint this and Pantelides' approach yield identical results for the structural index and the structural number of degrees of freedom. In order to show this Pantelides' approach had to be related to the theoretical considerations in Chapter 2. After establishing this relation it was simpler to accomplish an implementation of this method. In [37] Pantelides uses a pseudo-code representation which involves a recursive component. The implementation of recursive procedures is not possible in FORTRAN 77, which was the intended programming language for pragmatic reasons. However, using some theoretical insight of preceeding Chapters we could replace this component by a modification of Duff's "Algorithm for Obtaining a Maximal Transversal". Additionally, we had to introduce the bookkeeping required for the computation of the structural index ν_s

since this was not in the original publication. Adding this was not straightforward since the algorithm was designed to address consistent initialization and therefore, computes ν_s and r_s only indirectly.

After implementing both, our and Pantelides' structural approach, the programs obtained, called ALGO and PALG, yielded identical results concerning ν_s and r_s for various examples. In principle both approaches are capable of identifying feasible or infeasible sets of variables to be assigned arbitrary initial conditions. The limitations of this examination were indicated in Chapter 4 by means of *Algorithm 4.2.1* and *Algorithm 4.2.1.* So far only ALGO actually contains an implementation of this check. However, a very similar component could be added to PALG.
Due to an *equation* oriented approach in PALG and an *variable* oriented approach in ALGO they provide different types of information beyond ν_s and r_s. Therefore, a combination of PALG and Leimkuhler's method [31] should be be a rather efficient way to provide consistent initialization numerically. However, for index reduction and for detection of causes for higher-index-problems ALGO appears to be better suited. This can be seen from the fact that ALGO can be stopped after having encountered "index > 1". Based on the computations up to this point it can provide information which variables caused the problem and how it could be solved. Due to its approach PALG is not well suited for this task.

Considering presently available numerical methods and software, we think that in many cases the statement "index > 1" and a detailed analysis of possible causes is of great practical value. However, considering recent developments in the area of higher-index-solvers we also think that the problem of consistent initialization, which can be tackled by an efficient and precise implementation of the combination of PALG and Leimkuhler's method mentioned above, will become more and more significant. For index-one-systems and some index-two-systems of limited structure it is already significant today.

We showed that from a the standpoint of computational time and effort both algorithm are in the same reasonable range.

Concluding, it can be said that both implementations, ALGO and PALG, yield information beyond r_s and ν_s. Due to different approaches they view the DAE from different "angles". Therefore, by using both one obtains a more detailed characterization of the DAE than by using just one of them.

6.2 Further Work

There are several theoretical questions that remain open and should be addressed by further work.

For large systems it is desirable to have the input patterns of PALG and ALGO set up automatically. Depending on the representation of the equations this could be done by a code analyzer or a symbolical manipulator such as MACSYMA. This would save a lot of work and would also avoid errors. A similar task could be to set up a program that uses the informations given ALGO above for *symbolical* index reduction by means of a symbolical manipulator.

A second practical task is the implementation and testing of a combination of Pantelides' structural algorithm and the method proposed by Leimkuhler[31] for consistent initialization. This combination would allow an efficient numerical initialization without symbolical computations.

Motivated by the dynamic design examples in Chapter 5 one can see that there is need for further investigations of the effects of connecting or decomposing systems of known index on the index of the resulting system. Closely related to this is the investigation how the distinction between state and input variables affects the index.

In terms of numerical treatment of higher-index-systems, motivated by Section 3.2, one should examine the numerical results obtained from the integration of different

subsystems of the corresponding extended system

$$\dot{z} = \phi(z,u) \tag{6.1}$$

$$0 = K(z,u) \tag{6.2}$$

where $z, \phi \in \mathcal{R}^n$ and $k \in \mathcal{R}^{(n-r)}$, consisting of r differential equations of (6.1) and the $n-r$ algebraic equations (6.2). It would be interested if there is an "optimal" subset and if so how it can be identified.

The current situation that higher index DAE's of certain structure can be solved by appropriately designed higher-index-solvers leads to the question if one can design an algorithm or method that recognizes certain structures, e.g., a Hessenberg Form. Such a method could be used in a simulation environment to support the decision on an appropriate numerical method for the DAE model on hand.

Another question of interest is if there are further heuristics using the information extracted by ALGO and PALG that could be used by a simulation environment to set up an equivalent representation of the DAE that can be solved more easily by available numerical methods and codes. Some rather crude heuristics are currently implemented in ALGO but it would be very convenient to have more of them.

From both, a theoretical and a practical point of view it is interesting to examine the issue of bifurcations of DAE's mentioned in Chapter 2 and Appendix B more detailed. For numerical computations it can be important to know what is going to happen if a solution runs into a singular point or gets close to a singular point. The results of such investigations could be used to develop methods that recognize singular points and the take appropriate actions to cope with them.

Finally, it appears to be possible to extend the concept of the index of a DAE on PDAE's, such that the PDAE has as many different indices as it involves independent variables. Investigations on the significance of this issue and the role of boundary conditions from theoretical and computational points of view seem to be an interesting

task.

Appendix A

Proofs for Propositions in Section 2.2.2

A.1 Proof of Proposition 2.2.2a

In other words

$$\det(\lambda F_{\dot{z}} + F_z) \equiv 0 \tag{A.1}$$

for all $z, \dot{z} \in \mathcal{R}^n$, all $\lambda \in \mathcal{R}$, and all $t \geq 0$ means that this determinant vanishes at all possible points in state space. Generally, a determinant vanishes if it consists of linearly dependent rows or columns:

1. *Assume it vanishes due to row deficiencies:*

   This can be represented by expressing the n^{th} row of the pencil $(\lambda F_{\dot{z}} + F_z)$ by

   $$\text{row}_n(\lambda F_{\dot{z}} + F_z) = \sum_{i=1}^{n} p_i \, \text{row}_i(\lambda F_{\dot{z}} + F_z) \tag{A.2}$$

   which can be abbreviated by

   $$\lambda (F_n)_{\dot{z}} + (F_n)_z = \sum_{i=1}^{n} p_i \, (\lambda (F_i)_{\dot{z}} + (F_i)_z) \tag{A.3}$$

   if $F = (F_1, \ldots, F_{n-1}, F_n)^T$ and $F_i = 0$ denotes the i^{th} equation of the DAE.

   Since $det(\lambda F_{\dot{z}} + F_z) \equiv 0$ holds for *any* $\lambda \in \mathcal{R}$ in

   $$(F_n)_{\dot{z}} = \sum_{i=1}^{n} p_i \, (F_i)_{\dot{z}} \, , \tag{A.4}$$

$$(F_n)_z \;=\; \sum_{i=1}^{n} p_i\,(F_i)_z\,, \tag{A.5}$$

where $p_i = p_i(z, \dot{z}, t)$, $i = 1(1)n-1$, has to be satisfied for all $z, \dot{z} \in \mathcal{R}^n$ and all $t \geq 0$. By using the chain rule "backwards" for an integration with respect to $z, \dot{z}$, respectively, one obtains

$$F_n \;=\; \Psi(F_1, \ldots, F_{n-1}) \tag{A.6}$$

which means

$$p_i \;=\; \frac{\partial \Psi}{\partial F_i} \tag{A.7}$$

because the coefficients p_i have to be the same for (A.4) and (A.5).

Thus, if $det(\lambda F_{\dot{z}} + F_z) \equiv 0$ is due to row deficiencies of the pencil $(\lambda F_{\dot{z}} + F_z)$ then there is at least one redundant equation in the DAE $F(z, \dot{z}, u) = 0$ and it is not solvable by *Proposition 2.1.8.*

2. *Assume the determinant vanishes due to column deficiencies:*

With an analogous argument as above and the additional assumption that columns 1 through ρ where $1 \leq \rho \leq n-1$ are linearly dependent this implies

$$\sum_{i=1}^{\rho} p_i\, F_{\dot{z}_i} \;=\; 0\,, \tag{A.8}$$

$$\sum_{i=1}^{\rho} p_i\, F_{z_i} \;=\; 0\,, \tag{A.9}$$

where $p_i = p_i(z, \dot{z}, t) \neq 0$, $i = 1(1)\rho$, and $F_{\dot{z}_i}$, F_{z_i} denote the i^{th} column of $F_{\dot{z}}$, F_z respectively. As an immediate consequence of this it can be concluded that all equations have identical dependencies with respect to $z_i, \dot{z}_i$, $i = 1(1)\rho$. This means that the terms involving those variables are identical in all equations $F_i(z, \dot{z}, u) = 0$, $i = 1(1)n$. Thus one can formally represent these terms by one (lumped) variable

$$\sigma \;=\; \sigma(z_1, \ldots, z_\rho, \dot{z}_1, \ldots, \dot{z}_\rho) \tag{A.10}$$

in each equation of the DAE. With this lumped variable the DAE can be represented by

$$F(z_{\rho+1}, \ldots, z_n, \dot{z}_{\rho+1}, \ldots, \dot{z}_n, \sigma, u) = 0 \,, \tag{A.11}$$

where $F \in \mathcal{R}^n$, $\sigma \in \mathcal{R}$. Formally, this represents a system of n equations for $(n-\rho)$ unknowns $z_{\rho+1}(t), \ldots, z_n(t), \sigma(t)$ if $u(t)$ is assumed to be the set of *given* input variables. This is an overdetermined system, which might not have any solution at all, since it might contain equations which contradict each other, e.g.

$$z_{\rho+3} + \sigma + \dot{z}_{\rho+9} = 12 \,, \tag{A.12}$$

$$z_{\rho+3} + \sigma + \dot{z}_{\rho+9} = 10 \,. \tag{A.13}$$

However, even if those contradictions can be excluded and the system can actually be solved to determine $z_{\rho+1}(t), \ldots, z_n(t), \sigma(t)$ it is still not possible to determine all $z_1(t), \ldots, z_\rho(t)$ since

$$\sigma = \sigma(z_1, \ldots, z_\rho, \dot{z}_1, \ldots, \dot{z}_\rho) \tag{A.14}$$

represents just one equation for $2 \leq \rho \leq n$ unknowns. For $\rho = 1$, $\sigma \equiv 0$ and neither z_1 nor $\dot{z}_1$ occur in any equation of the DAE which is therefore an overdetermined system which can never yield any result for $z_1(t)$, by the same arguments as for $2 \leq \rho \leq n$. Thus, the system cannot be solved to obtain all components of $z(t)$ and can therefore be considered non-solvable with respect to the variables $z_1(t), \ldots, z_n(t)$.

Thus, it can be stated that $\det(\lambda F_{\dot{z}} + F_z) \equiv 0$ is caused either by redundant equations or by the possibility to obtain an equivalent system by lumping variables which occur in an identical way in all equations of the DAE. Both causes imply that the DAE is not solvable. □

A.2 Proof of Proposition 2.2.3

Consider a DAE $F(z, \dot{z}, u) = 0$, $F, z \in \mathcal{R}^n$ and assume it contains μ, $1 \leq \mu \leq n-1$, redundant equations. Then by the definition of redundant equations these μ equations can be removed from the DAE since they do not place any further restrictions on a solution. Thus, one is left with a DAE

$$\tilde{F}(z, \dot{z}, u) = 0\,, \tag{A.15}$$

where $z \in \mathcal{R}^n$, $\tilde{F} \in \mathcal{R}^{n-\mu}$. Going through *Algorithm 2.1.1* **Steps 1–6** for $i = 0$ yield

$$\dot{x}_0 = \dot{x}_0(z_0, \dot{y}_0)\,, \tag{A.16}$$

$$0 = G_0(z)\,, \tag{A.17}$$

where $x_0 \in \mathcal{R}^{r_0}$, $G_0 \in \mathcal{R}^{n-r_0-\mu}$. Thus, there are only $(n - r_0 - \mu)$ equations G_0 left, which by means of differentiations and substitutions are supposed to yield $(n - r_0)$ expressions for those components of $\dot{z}$ which form $\dot{y}_0$. The following argument shows that therefore, it is not possible to derive a complete extended system:
Whenever it is possible to solve for r_i components $\dot{x}_i$ of $\dot{z}$ after the i^{th} differentiation the number of equations $G_i(z) = 0$ remaining for further differentiations decreases by r_i. Thus, after k differentiations one is left with $(n - \mu - \sum_{i=0}^{k} r_i\,)$ equations for the $(n - \sum_{i=0}^{k} r_i\,)$ components of y_k. If $(n-\mu) - \sum_{i=0}^{k} r_i = 0$ then *Algorithm 2.1.1* cannot continue since G_k is of dimension zero (i.e, there are no more such equations) leaving an incomplete extended system of the form

$$\dot{z}_1 = \dot{z}_1(z, \dot{z}_{n-\mu+1}, \ldots, \dot{z}_n)\,, \tag{A.18}$$

$$\vdots$$

$$\dot{z}_{n-\mu} = \dot{z}_{n-\mu}(z, \dot{z}_{n-\mu+1}, \ldots, \dot{z}_n)\,, \tag{A.19}$$

$$0 = G_0(z)\,, \tag{A.20}$$

$$0 = G_1(z)\,, \tag{A.21}$$

$$\vdots$$

$$0 = G_{k-1}(z)\,, \tag{A.22}$$

which leaves $\dot{z}_{n-\mu+1}, \ldots, \dot{z}_n$ undetermined. □

Appendix B

On the Significance of the Pencil $\left(\lambda F_{\dot{z}} + F_z\right)$

As pointed out in Section 2.2.2 the pencil $(\lambda F_{\dot{z}} + F_z)$ is closely related to solvability. The solvability conditions formulated in Section 2.2.2 considered this pencil *globally*, i.e., for *all* $z, \dot{z} \in \mathcal{R}$, $t > 0$. However, as indicated below there are *local* aspects to it. One of them is that this pencil occurs in codes like DASSL as an iteration matrix for the corrector step. Thus, at points with $det(\lambda F_{\dot{z}} + F_z) \doteq 0$ numerically one can expect convergence problems of the corrector.

In this part of the appendix we consider a (2×2) example which illustrates the local significance of this pencil.

Example B.1:

$$\dot{x} = y \,, \tag{B.1}$$

$$0 = (y - mx - b)(y - \sin x) \,, \tag{B.2}$$

$$\tag{B.3}$$

where m, b are constant parameters and $x, y \in \mathcal{R}$. Possible solutions of the algebraic equation are

$$y_1 = mx + b \,, \tag{B.4}$$

$$y_2 = \sin x \tag{B.5}$$

which can be represented in the following graph:

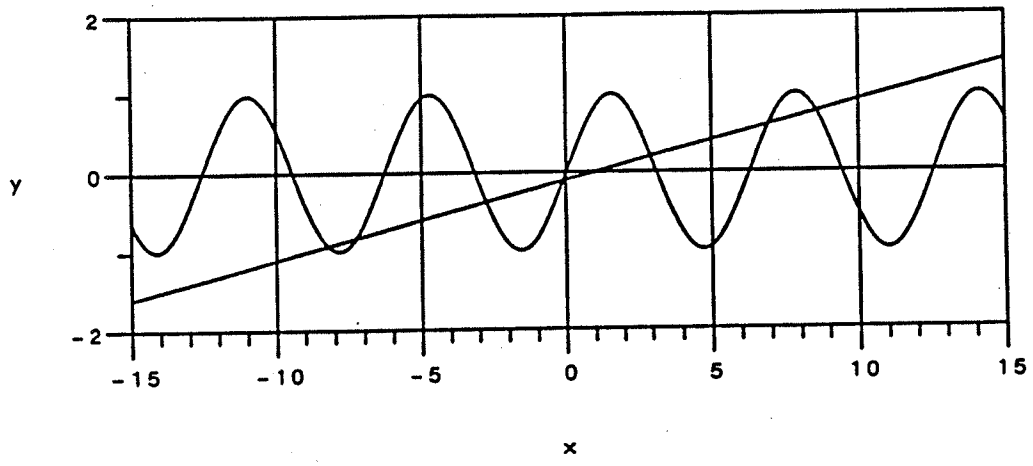


Figure B.1: Graph of the two possible solutions of the algebraic equation

In this case the pencil $(\lambda F_{\dot{z}} + F_z)$ becomes:

$$\begin{pmatrix} \lambda & 1 \\ -(\cos x + m)y + (mx + b)\cos x + m\sin x & 2y - mx - b - \sin x \end{pmatrix} \tag{B.6}$$

If $det(\lambda F_{\dot{z}} + F_z) \equiv 0$ for all λ then the following two relations hold:

$$2y - mx - b - \sin x = 0\,, \tag{B.7}$$

$$-(\cos x + m)y + (mx + b)\cos x + m\sin x = 0\,, \tag{B.8}$$

The first of those equations is satisfied by all points for which $y_1 \equiv y_2$, i.e., the intersection points in Figure B.1, since it can be rewritten:

$$(y - mx - b) + (y - \sin x) = 0. \tag{B.9}$$

With $y = mx + b = \sin x$ the second equation vanishes identically.

The consequence for a solution of the DAE at those points can be seen from the

following considerations:

Assume that the initial values are such that $y_0 = mx_0 + b$, i.e., the solution starts on a point of the straight line. Then the solution moves along the straight line. After running into an intersection point, the solution can continue along the graph of either of the two factors of the algebraic constraint. Without violating the equations two different solutions are possible since both,

$$y_1 = mx + b \tag{B.10}$$

and

$$y_2 = \sin x \tag{B.11}$$

satisfy the algebraic constraint without both factors having to be zero simultaneously. This means there are also two solutions for $x(t)$ since:

$$\dot{x}_1 = mx + b\,, \tag{B.12}$$

$$\dot{x}_2 = \sin x\,. \tag{B.13}$$

If t_s denotes the time when the solution ran into the intersection point then the "initial conditions" for these ordinary differential equations are

$$x_1(t_s) = x_2(t_s) = x_s\,, \tag{B.14}$$

where x_s is the x-value at the intersection, i.e., it satisfies

$$mx_s + b = \sin x_s. \tag{B.15}$$

The solutions of these two equations can be computed by separation of variables and integration:

$$x_1(t) = x_s - \frac{b}{m}(1 - e^{m(t-t_s)})\,, \tag{B.16}$$

$$x_2(t) = \arctan(e^{(t-t_s)} \tan x_s)\,. \tag{B.17}$$

Thus, after an intersection point there are two possible trajectories represented by $(x_1(t), y_1(t))$ and $(x_2(t), y_2(t))$. This phenomena could be called *bifurcation of a trajectory.*

This type of bifurcation is different from the bifurcation known from ODE's. In fact, it cannot occur for ODE's. This is due to the fundamental theorem (for reference see e.g. [20]) which states that an ODE

$$\dot{z} = f(z,t) \tag{B.18}$$

has a *unique* continuously differentiable solution if $f(z,t)$ satisfies a Lipschitz condition

$$\|f(z,t) - f(\hat{z},t)\| \leq C_L \|z - \hat{z}\| , \tag{B.19}$$

for all $t_0 \leq t \leq T$, all $z, \hat{z} \in \mathcal{R}$ and C_L being a constant.

For ODE's bifurcations occur if *parameters* are varied, i.e., the quality of a solution of an DAE might depend significantly on certain parameters. However, for DAE's the effects described above occur *without* changing parameters. This motivates to distinguish two types of bifurcations of DAE's: (i) Bifurcations due to *changes in parameters* as known from ODE's and (ii) bifurcations of trajectories as described above. Motivated by the discussion above we introduce

Definition A.1: *Points* $(z,^* \dot{z}^*, t^*)$ *for which* $det(\lambda F_{\dot{z}} + F_z)|_* \equiv 0$ *for all* $\lambda \in \mathcal{R}$ *are called singular points of the DAE* $F(z, \dot{z}, u) = 0$.

Using this definition and the relation of bifurcation of trajectories with singular points it can be concluded:

Proposition A.1: *Let* Ω *be an open connected subset of* $\mathcal{R}^{2n+1}$ *and* $\mathcal{I} = [t_0, T] \subset \mathcal{R}$ *then the DAE* $F(z, \dot{z}, u) = 0$, $z, F \in \mathcal{R}^n$ *has a unique solution* $z(t)$ *on* $\mathcal{I}$ *in* Ω *for each*

set of feasible initial values $c \in \tilde{\Omega} \subset \mathcal{R}^r$ *if* $det(\lambda F_{\dot{z}} + F_z) \neq 0$ *for all* $(z, \dot{z}, t) \in \Omega$ *and almost every* $\lambda \in \mathcal{R}$.

Reformulated this means that a *sufficient* condition for the DAE having a unique solution in Ω is that it does not have any singular points in Ω. From the standpoint of *Definition 2.1.1* systems that have singular points in Ω are not solvable in Ω. Evidently, this avoids several problems, however it might be of interest to examine systems of that kind closer in order to gain a better understanding of DAE's.

The considerations above are by no means complete. They are meant to indicate that there are several open questions concerning DAE's. A theoretical question is, whether there is need for a "Bifurcation Theory for DAE's ". A more practical question deals with the initialization of a DAE. The results of a Newton iteration which is used to solve nonlinear equations in order to provide consistent initialization depend on the initial guesses. Looking at *Example B.1* this means that if initial guesses for the solution of the algebraic constraint are close to the straight line, then the iteration will converge most likely to an initial point on the straight line. For a choice close to the sinusodial graph it will converge to an initial point that satisfies $y_0 = \sin x_0$. Depending on the starting point there are different solutions. For practical computations it might be of interest that there are several possible solutions, especially if a solution that does not match physical expectations is obtained.

Appendix C

Proof of Propositions 4.1.5 and 4.1.6

A major problem with this proof is that one has to cope with notational details which make it very difficult to follow the general idea. We tried to address this problem by pointing out the key ideas in the beginning. However, we are aware of the fact that it is still very hard to follow the formal argument given below. It might be helpful for the reader to apply the various steps pointed out below on a small example such as the index-three-DAE arising from the pendulum mentioned earlier.

Consider the general nonlinear DAE $F(z, \dot{z}) = 0$ [1], where $F, z \in \mathcal{R}^n$ and differen-

[1] To shorten notation u is omitted since it does not affect the arguments in this proof.

tiated it k times with respect to time t. This leads to

$$\begin{array}{lcll}
F(z,\dot{z}) & = & 0 & (0) \\
F_z\dot{z} \quad + \quad F_{\dot{z}}\ddot{z} & = & 0 & (1) \\
(F_{zz}\dot{z})\dot{z} + (F_{z\dot{z}}\ddot{z})\dot{z} + \{F_z + F_{\dot{z}z}\dot{z} + F_{\dot{z}\dot{z}}\ddot{z}\}\ddot{z} + F_{\dot{z}}z^{(3)} & = & 0 & (2) \\
\vdots & & \vdots & \vdots \\
H_i(z,\dot{z},\ldots,z^{(i-1)}) \; + & & & \\
+ \; (F_{z\dot{z}}z^{(i)})\dot{z} + (F_{\dot{z}\dot{z}}z^{(i)})\ddot{z} + \{F_z + F_{\dot{z}z}\dot{z} + F_{\dot{z}\dot{z}}\ddot{z}\}z^{(i)} + F_{\dot{z}}z^{(i+1)} & = & 0 & (i) \\
\vdots & & \vdots & \vdots \\
H_k(z,\dot{z},\ldots,z^{(k-1)}) \; + & & & \\
+ \; (F_{z\dot{z}}z^{(k)})\dot{z} + (F_{\dot{z}\dot{z}}z^{(k)})\ddot{z} + \{F_z + F_{\dot{z}z}\dot{z} + F_{\dot{z}\dot{z}}\ddot{z}\}z^{(k)} + F_{\dot{z}}z^{(k+1)} & = & 0 & (k)
\end{array} \tag{C.1}$$

which represents a system of $(k+1)\cdot n$ equations and in the remainder is referred to as the *hyper system* $\bar{F}_k$ obtained by differentiating the DAE $F(z,\dot{z}) = 0$ k times with respect to time t. The numbers in braces denote the number of differentiations applied to in order $F(z,\dot{z}) = 0$ in order to obtain these equations. H_i denotes nonlinear functions of the variables in braces and $(F_{z\dot{z}}\dot{z})$ and similar entries denote terms due to second order partial derivatives of F.

Before carrying out more detailed considerations we want to outline the basic idea of this proof:

The first step will show that if the hyper system above contains a subsystem S_k of n equations which is regular with respect to the *highest time derivatives* of all variables z occurring in the DAE, i.e., the Jacobian of the subsystem S_k with respect to these highest derivatives is regular, then further differentiations would only introduce as many "new" variables (higher order time derivatives of z) as new equations. This can be seen by observing that the resulting hyper system $\bar{F}_{k+1}$ would contain a subsystem S_{k+1} which would be regular with respect to highest derivatives which are time differentials of the components of z of degree one higher than the corresponding ones in S_k. Therefore a further differentiation would not introduce further relations between the sets of variables considered before.

It is important to note that S_k can involve each equation $F_j = 0$ only once, either as it occurs in the DAE $F = 0$ without differentiations or as it appears in $\frac{d^i F}{dt^i}$ after the i^{th} differentiation.

In the *second* part we show that if a subsystem S_k is found as described above then one can always solve for $\dot{z} = \Upsilon(z)$ after one more differentiation with respect to time. This implies that the DAE is of index $\nu = k + 1$.

The most important result will be that in the *structural sense* this argument and all steps are determined completely by *pat* F_z and *pat* $F_{\dot{z}}$. This implies that in structural considerations the patterns of higher order partial derivatives of F have no impact on which equations have to be differentiated how many times in order to determine the index according to *Algorithm 2.1.1.* Therefore, the structural index and the structural number of degrees of freedom only depend on *pat* F_z and *pat* $F_{\dot{z}}$. Motivated by the discussion above we introduce:

Definition C.1: *S_k denotes the subsystem containing n equations of the hyper system $\bar{F}_k$ arising if the DAE $F(z, \dot{z}) = 0$ is differentiated k times with respect to time t, which allows to solve for the highest derivatives of all z components occurring in it. $(S_k)_i \subset S_k$ denotes the subset of equations of S_k which originate from $\frac{d^i F}{dt^i} = 0$, $0 \leq i \leq k$.*

For the upcoming formal argument it is convenient to introduce:

Definition C.2: *${}^i z$ denotes the subset of components z_j of z whose highest derivative occurring explicitly in the subsystem S_k as defined above is determined from equations obtained after i differentiations.*

The number of components of ${}^i z$ is r_i, i.e., $dim\, {}^i z = r_i$.

A consequence of this definition is that the degree of differentiation of these highest derivatives of variables $z_j \in^i z$ in S_k is either i or $(i+1)$ since they either occur as z_j or as $\dot{z}_j$ in the original equations $F(z, \dot{z}) = 0$. Since we assumed that after k differentiations S_k is regular with respect to all highest derivatives, we know:

$$\sum_{i=0}^{k} r_i \;=\; n\,. \tag{C.2}$$

With the following argument for ${}^0z$ we now show why each equation of $F(z, \dot{z}) = 0$ appears *exactly once* in S_k, either in its original form or up to k times differentiated: Assume

$${}^0z \;=\; (({}^0z)_1, ({}^0z)_2)^T\,, \tag{C.3}$$

where $({}^0z)_1 \in \mathcal{R}^{r_{01}}$, $({}^0z)_2 \in \mathcal{R}^{r_{02}}$, $r_0 = r_{01} + r_{02}$ such that components of $({}^0z)_1$ occur only undifferentiated in S_k and components of $({}^0z)_2$ occur at most once differentiated in S_k. By *Definition C.2* and the implicit function theorem this implies that a set $(S_k)_0 \equiv F^0$ of r_0 of the original (undifferentiated) equations of the DAE can be used to obtain

$$z_m \;=\; {}^0\Upsilon_1(\overline{{}^0z}, \dot{\overline{{}^0z}})\,, \qquad z_m \in ({}^0z)_1 \tag{C.4}$$

$$\dot{z}_l \;=\; {}^0\Upsilon_2(\overline{{}^0z}, \dot{\overline{{}^0z}})\,, \qquad z_l \in ({}^0z)_2 \tag{C.5}$$

$$\tag{C.6}$$

as functions of variables $\overline{{}^0z}$ which do not belong to the set ${}^0z$ and their time derivatives $\dot{\overline{{}^0z}}$. An immediate consequence of the discussion above is that the derivatives of equations $F^0 = 0$ cannot belong to S_k since they would involve $\dot{z}_m, \ddot{z}_m, \ldots$ and $\ddot{z}_l, z^{(3)}, \ldots$ such that z_m and $\dot{z}_l$ could not represent the *highest* derivative of $z_m \in ({}^0z)_1$ and $z_l \in ({}^0z)_2$ occurring in S_k.

Motivated by the discussion above we introduce another definition which will be useful in later steps:

Definition C.3: *Let* ${}^i z = (({}^i z)_1, ({}^i z)_2)^T$ *such that the highest derivative of* $z_m \in ({}^i z)_1$, $z_l \in ({}^i z)_2$ *is* $z_m^{(i)}$, $z_l^{(i+1)}$, *respectively. Then the elements of the sets* $(z_m, \dot{z}_m, \ldots, z_m^{(i-1)})$, $(z_l, \dot{z}_l, \ldots, z_l^{(i-1)}, z_l^{(i)})$ *are called the lower derivatives of* z_m *and* z_l *in* S_k.

In the argumentation above components of $({}^0 z)_1$ do not have a lower derivative in S_k whereas components of $({}^0 z)_2$ can have lower derivatives z_l in S_k since the highest derivative occurring in S_k is one.

The next step indicates why for *structural* considerations information on *pat* F_z and *pat* $F_{\dot{z}}$ is *sufficient* to determine which equations are used to set up S_k.
Consider the Jacobian $\bar{J}_k$ of $\bar{F}_k$ with respect to $z, \dot{z}, \ldots, z^k, z^{k+1}$

$$\bar{J}_k = \begin{pmatrix} F_z & F_{\dot{z}} & 0 & 0 & 0 & 0 & \ldots & \ldots & 0 \\ * & F_z + D_1 & F_{\dot{z}} & 0 & 0 & 0 & \ldots & \ldots & 0 \\ * & * & F_z + D_2 & F_{\dot{z}} & 0 & 0 & \ldots & \ldots & 0 \\ * & * & * & F_z + D & F_{\dot{z}} & 0 & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & & \ddots & \ddots & \ddots & \vdots \\ * & * & * & \ldots & \ldots & * & F_z + D & F_{\dot{z}} & 0 \\ * & * & * & \ldots & \ldots & * & * & F_z + D & F_{\dot{z}} \end{pmatrix}, \tag{C.7}$$

where $D, D_1, D_2 \in \mathcal{R}^{n \times n}$ denote matrices such that

$$(F_z + D)\, z^{(i)} = (F_{z\dot{z}} z^{(i)})\dot{z} + (F_{\dot{z}\dot{z}} z^{(i)})\ddot{z} + \{F_z + F_{\dot{z}z}\dot{z} + F_{\dot{z}\dot{z}}\ddot{z}\} z^{(i)} \tag{C.8}$$

$$(F_z + D_2)\, \ddot{z} = (F_{z\dot{z}}\ddot{z})\dot{z} + \{F_z + F_{\dot{z}z}\dot{z} + F_{\dot{z}\dot{z}}\ddot{z}\}\ddot{z}\,. \tag{C.9}$$

The matrices D_1 originates from the under-braced terms in

$$(F_z\dot{z} + F_{\dot{z}}\ddot{z})_{\dot{z}} = F_z \underbrace{(F_{z\dot{z}}\dot{z}) + (F_{\dot{z}\dot{z}}\ddot{z})} \tag{C.10}$$

The notation $*$ represents nonzero entries which will be shown to be of no further impact by the considerations below. Obviously, the upper subdiagonal of $\bar{J}_k \in$

$\mathcal{R}^{n(k+1)\times n(k+2)}$ is determined completely by $F_{\dot{z}}$. By the following argument we show that *structurally pat* F_z completely determines the nonzero entries on the diagonal for columns of interest:

To show this we observe that:

1. Terms involving second order partial derivatives $F_{\dot{z}z}$ and $F_{z\dot{z}}$ cannot contribute nonzero entries where *pat* F_z has zeros.

2. Terms involving $F_{\dot{z}\dot{z}}$ may have nonzeros where *pat* F_z has zeros. However, these are not of interest on the diagonal since corresponding to those terms there are nonzeros on the upper subdiagonal due to *pat* $F_{\dot{z}}$. Because we are aiming for the *highest* derivative of occurring, for variables z_j corresponding to those nonzero entries $z_j^{(i+1)}$ is the highest derivative such that that an entry on the diagonal which corresponds to $z_j^{(i)}$ is of no interest.

Summarizing this means that *pat* D, *pat* D_1, *pat* D_2 cannot enter significant nonzeros which are not entered by *pat* F_z. Therefore, if one is only interested in the highest derivatives that occur explicitly in $\bar{F}_k$, then for structural considerations it is *sufficient* to represent *pat* $\bar{J}_k$ by

$$\bar{J}_k = \begin{pmatrix} pat\,F_z & pat\,F_{\dot{z}} & 0 & 0 & 0 & 0 & \dots & \dots & 0 \\ * & pat\,F_z & pat\,F_{\dot{z}} & 0 & 0 & 0 & \dots & \dots & 0 \\ * & * & pat\,F_z & pat\,F_{\dot{z}} & 0 & 0 & \dots & \dots & 0 \\ * & * & * & pat\,F_z & pat\,F_{\dot{z}} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & & \ddots & \ddots & \ddots & \vdots \\ * & * & * & \dots & \dots & * & pat\,F_z & pat\,F_{\dot{z}} & 0 \\ * & * & * & \dots & \dots & * & * & pat\,F_z & pat\,F_{\dot{z}} \end{pmatrix}. \tag{C.11}$$

The nonzeros in the lower triangular of this pattern are not important since they correspond to "lower" derivatives than the entries on the diagonal and the upper

subdiagonal. Therefore, they could also be omitted. The pattern obtained by doing so would be the same as the one that can be derived by differentiating the corresponding linear system

$$M\dot{z} + Lz = 0 \tag{C.12}$$

k times, with $pat\, M \equiv pat\, F_{\dot{z}}$ and $pat\, L \equiv pat\, F_z$.

With the results above and the property that S_k involves each equation of the DAE only once the structural Jacobian $pat\, J_{S_k}$ of S_k with respect to the highest derivatives occurring in it explicitly can be represented by the $(n \times n)$ pattern

$$\begin{pmatrix}
pat\, F_z^0 & pat\, F_{\dot{z}}^0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 \\
* & pat\, F_z^1 & pat\, F_{\dot{z}}^1 & 0 & 0 & 0 & \dots & \dots & 0 \\
* & * & pat\, F_z^2 & pat\, F_{\dot{z}}^2 & 0 & 0 & \dots & \dots & 0 \\
* & * & * & pat\, F_z^3 & pat\, F_{\dot{z}}^3 & 0 & \dots & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\
\vdots & \vdots & \vdots & & & \ddots & \ddots & \ddots & \vdots \\
* & * & * & \dots & \dots & * & pat\, F_z^{k-1} & pat\, F_{\dot{z}}^{k-1} & 0 \\
* & * & * & \dots & \dots & * & * & pat\, F_z^k & pat\, F_{\dot{z}}^k
\end{pmatrix}, \tag{C.13}$$

where F_z^i and $F_{\dot{z}}^i$, $i = 0(1)k$ represent the Jacobians of the r_i equations of the subset (i) in the hyper system $\bar{F}_k = 0$ which can be used to determine the highest derivatives represented by ${}^i z$. Again, for these considerations nonzero entries in the lower triangular are irrelevant and could therefore be omitted.

Using the argument above we briefly indicate that further differentiations of the DAE would not yield any further information about relations among variables. If the entire subsystem is differentiated once with respect to time then n "new" variables (derivatives of higher order) are generated, since S_k was already regular with respect to the highest derivatives occurring in it. Since S_k involved equations of $\frac{d^k F}{dt^k} = 0$ the system S_{k+1} obtained by differentiating S_k will involve equations of $\frac{d^{k+1} F}{dt^{k+1}} = 0$, therefore S_{k+1} is the "same" subsystem in the hyper system $\bar{F}_{k+1} = 0$ as S_k is in $\bar{F}_k = 0$. In other

words the pattern of the Jacobian of S_k with respect to its highest derivatives and pattern of the Jacobian of S_{k+1} with respect to its highest derivative are identical. The highest derivatives in S_{k+1} are just differentiated once more than the ones in S_k. Therefore, differentiating the DAE once more after S_k was found would only introduce as many new highest derivatives as new equations such that it would not yield further relations among the "old" highest derivatives.
This observation is very closely related to ideas of Pantelides which motivated his algorithm [37].

We now enter the *second part* of this proof:
In order to show that the index of the DAE is $\nu = k+1$ we have to show that the hyper system $\bar{F}_k = 0$ can be used to solve for

$$\dot{z} \;=\; \Upsilon(z) \tag{C.14}$$

after a subsystem S_k as described above has been found.
For doing this it is convenient to introduce

Definition C.4: *${}^i\dot{z}$ denotes the set of all derivatives of components of ${}^iz$, $i = 1(1)k$ of order higher than $i, (i+1)$, respectively.*[2]

Using the notations above we formally solve S_k for all highest derivatives occurring in it by applying the implicit function theorem:

$$\begin{array}{lcll}
{}^0z & = & {}^0\Upsilon(\overline{{}^0z}) & , \ \overline{{}^0z} \cap {}^0z = \emptyset \\
{}^1z & = & {}^1\Upsilon(z, \overline{{}^1z}) & , \ \overline{{}^1z} \cap \{{}^0\dot{z} \cup {}^1z\} = \emptyset \\
{}^2z & = & {}^2\Upsilon(z, \overline{{}^1z}, \overline{{}^2z}) & , \ \overline{{}^2z} \cap \{{}^0\dot{z} \cup {}^1\dot{z} \cup {}^2z\} = \emptyset \\
 & \vdots & & \vdots \\
{}^{k-1}z & = & {}^{k-1}\Upsilon(z, \overline{{}^1z}, \ldots, \overline{{}^{k-2}z}, \overline{{}^{k-1}z}) & , \ \overline{{}^{k-1}z} \cap \{{}^0\dot{z} \cup {}^1\dot{z} \cup \ldots \cup {}^{k-2}\dot{z} \cup {}^{k-1}z\} = \emptyset \\
{}^kz & = & {}^k\Upsilon(z, \overline{{}^1z}, \ldots, \overline{{}^{k-1}z}, \overline{{}^kz}) & , \ \overline{{}^kz} \cap \{{}^0\dot{z} \cup {}^1\dot{z} \cup \ldots \cup {}^{k-1}\dot{z} \cup {}^kz\} = \emptyset
\end{array} \tag{C.15}$$

[2]Note that the highest derivative of components of ${}^iz$ in S_k by the arguments following *Definition C.2* is either i or $i+1$.

The lower derivatives (according to *Definition C.3*) of the components of ${}^{i}z$ can always be determined as functions of their lower derivatives, i.e., of derivatives of lower order than their own. This is true since the equations corresponding to the ones used in S_k to determine the highest derivatives of ${}^{i}z$ are regular with respect to those lower derivatives since they are (structurally) identical to those with respect to the lower derivatives. The equations used to compute the lower derivatives cannot belong to S_k since they represent lower derivatives of the equations in S_k and can therefore not belong to S_k. By successive backward substitution of higher derivatives in equations of the hyper system $\bar{F}_k = 0$ one can obtain

$$\dot{\tilde{z}} = \tilde{\Upsilon}(z)\,, \tag{C.16}$$

where $\dot{\tilde{z}}$ denotes the subset of z whose highest derivative in S_k was of order $1 \leq d \leq k+1$. Variables z_j whose highest derivative in S_k was the undifferentiated variable itself form a subset $\tilde{\tilde{z}}$. By analogous arguments as above these variables can be determined as functions of $\tilde{z}$,i.e.,

$$\tilde{\tilde{z}} = \tilde{\tilde{\Upsilon}}(\tilde{z})\,. \tag{C.17}$$

Obviously, this requires one more differentiation with respect to time

$$\dot{\tilde{\tilde{z}}} = \tilde{\tilde{\Upsilon}}_{\tilde{z}}\,\dot{\tilde{z}} = \tilde{\tilde{\Upsilon}}_{\tilde{z}}\,\tilde{\Upsilon}(z) = \hat{\Upsilon}(z) \tag{C.18}$$

to obtain

$$\dot{z} = (\dot{\tilde{z}}, \dot{\tilde{z}})^T = \Upsilon(z)\,. \tag{C.19}$$

Therefore, the DAE is of index $\nu = k+1$ since it took k differentiations to derive S_k plus the last differentiation described above.

The last differentiation above can only be avoided if $detF_{\dot{z}} \neq 0$ which means that it can only be avoided for ODE's which are of index $\nu = 0$. The rather lengthy argument above has indicated that the structural index solely depends on *pat* F_z and

pat $F_{\dot{z}}$. Higher order partial derivatives of F do not play any role in these structural arguments.

Finally, we briefly show that the structural number of degrees of freedom is also determined by *pat* F_z and *pat* $F_{\dot{z}}$ alone.

Above S_k was determined by just examining *pat* F_z and *pat* $F_{\dot{z}}$. With the notations:

$hd(j)$: order of the highest derivative of variable z_j, $j = 1(1)n$

$diff(i)$: is the number of differentiations with respect to time that have been applied to $F_i = 0$, $i = 1(1)n$, in order to obtain the equation that represents the original equation $F_i = 0$ in S_k, i.e., this equation is $\frac{d^{diff(i)}F_i}{dt^{diff(i)}t} = 0$.

one can consider a larger subsystem $\bar{S}_k$ of $\bar{F}_k$. which consists of S_k and all equations that correspond to the ones in S_k such that

$$\bar{S}_{k_{il}} = \frac{d^l F_i}{dt^l}, \tag{C.20}$$

where $0 \leq l < diff(i)$. With these notations $\bar{S}_k$ represents a system of

$$n_{equ} = n + \sum_{i=1}^{n} diff(i) \tag{C.21}$$

equations.

Assume that S_{k_i} could be used to determine $z_j^{(hd(j))}$ then $\bar{S}_{k_{il}}$ can be used to determine $z_j^{(l)}$ if $F_{\dot{z}_j} = 0$ and if $F_{\dot{z}_j} \neq 0$ it can be used to determine $z_j^{(l+1)}$. For consistent initialization $z_{0_j} = z_j(t_0)$ has to be specified. Therefore, from the perspective of consistent initialization $\bar{S}_k$ can be viewed as a system of n_{equ} equations in

$$n_{var} = n + \sum_{j=1}^{n} hd(j) \tag{C.22}$$

variables $z_{0_j}, \dot{z}_{0_j}, \ldots, z_{0_j}^{(hd(j))}$, $j = 1(1)n$. The number of arbitrarily specifiable variable in this underdetermined systems of nonlinear equations is

$$n_{var} - n_{equ} = \sum_{j=1}^{n} hd(j) - \sum_{i=1}^{n} diff(i) , \tag{C.23}$$

which therefore, represents the number of arbitrarily specifiable initial conditions which is defined to be the number of degrees of freedom.

As indicated above both, $hd(j)$ and $diff(i)$, depend on S_k which was shown to be determined completely by $pat\, F_{\dot{z}}$ and $pat\, F_z$ which means that $n_{var} - n_{equ}$ is also only depending on $pat\, F_{\dot{z}}$ and $pat\, F_z$ and can therefore be called the structural number of degrees of freedom r_s.

Since we could show that both, the structural index as well as the structural number of degrees of freedom, depend on $pat\, F_{\dot{z}}$ and $pat\, F_z$ only and since these patterns are identical for $F(z, \dot{z}) = 0$ and its corresponding linearized representation

$$M\,\dot{z} + L\,z = 0 , \tag{C.24}$$

where $pat\, M = pat\, F_{\dot{z}}$ and $pat\, L = pat\, F_z$ these two representations of the DAE have the same structural index and the same structural number of degrees of freedom. □

Appendix D

Documentation on Implementations

D.1 Routines in Common

Both implementations contain the subroutines INPUT and SOLVAB:

INPUT: reads the dimension of the DAE $F(z, \dot{z}, u) = 0$ and
the patterns of the Jacobians F_z and $F_{\dot{z}}$
from a user-specified input file or from manual input.
For manual input the user can specify a file on which
his/her input will be saved.

SOLVAB: examines the structural rank of the merged pattern
$pat\, F_{\dot{z}} + pat\, F_z$ using the subroutines:
- RANKDET
- MC21A

which are described in the documentation on ALGO.

D.2 ALGO

The following brief description of the subroutines in ALGO relates them to steps of *Algorithms 4.2.l/n:*

RANKDET: **Step 1,**
determines the structural rank of a square matrix using
the subroutine MC21A which is based on Duff's
"Algorithm to Obtain a Maximal Transversal"
(ACM Algorithm 575).

PARTITION: **Step 2**

INVERSION: carries out a structural Gauss Elimination (**STEP 3**)

SUBYK1XI: **Step 4**

IND2CAUS: names variables (and equations) which could be the cause
for the index exceeding one. Additionally, it gives
suggestions how the index could be reduced to zero or one.
(output on file 'causout.out')

ALLONIN: creates abbreviated report on results and computations
for higher-index-DAE's (output on file 'causout.out').

SUBYK1CDG: **Step 6**,additionally, it sets up and and saves (on file
'feasfile.inp') the feasibility matrix required by FAIC.

DIFFSUBX: **Steps 7,8**

Partial differentiation according to

- **Rule 5.1** if 'highly nonlinear' is specified
- **Rule 5.1** if 'linear' is specified

FAIC: determines if a set of variables specified to be assigned arbitrary initial conditions allows consistent initialization.
This is done by examining the structural rank of an appropriate of the feasibility matrix on'feasfile.inp'.
Subroutines: - FEAESRANK
- MC21A

A precise report on the computations of the program for a certain example is written to 'algoout.out'. All subroutines are stored on files with their name and the extension 'for', except MC21A, which is stored on 'mc21am.for'. All parameters are defined in 'algopar.par'.

D.3 PALG

This program is based on a graph theoretical approach [37,47]. The description below relates subroutines to steps in *Algorithm 4.3.1*:

NEWNO: adds next equation node to the graph and establishes all appropriate edges to variable nodes (**Steps 1,2**)

AUGMP: Using subroutine AUGAD, which is a modification of MC21A, it tries to create a new assignment of pairs of equations and variables (**Steps 3, 5–8, 11**).

DICOLSUB: carries out "differentiations" of colored subsets of equations and variables and the adjust all edges of the graph (**Steps 9, 10**).

The result is saved on 'palgres.out'. A precise report on the computations along the way is stored on 'palgout.out'. All parameters are defined on 'palgpar.par'.

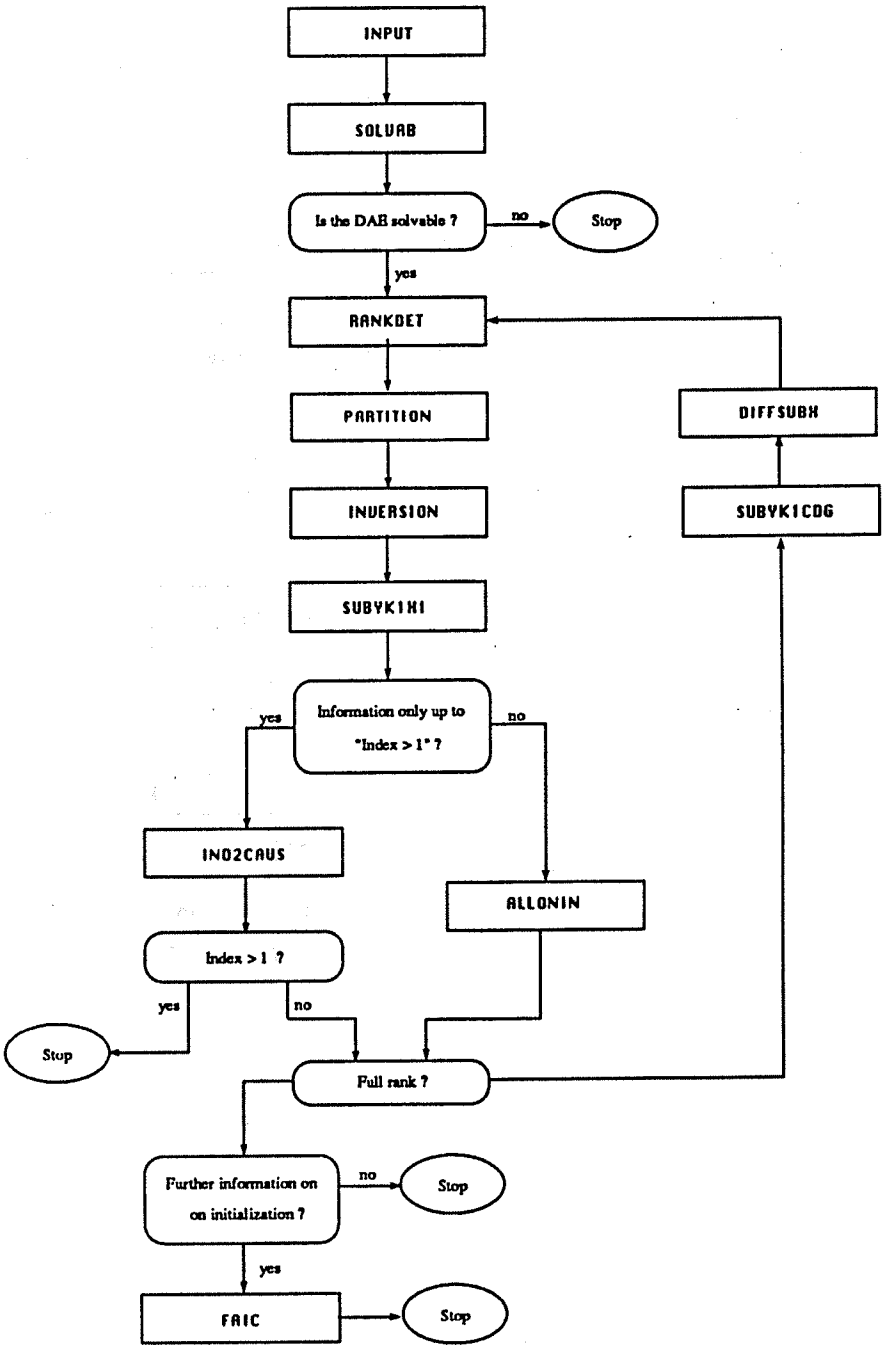


Figure D.1: Structure and Subroutines of ALGO

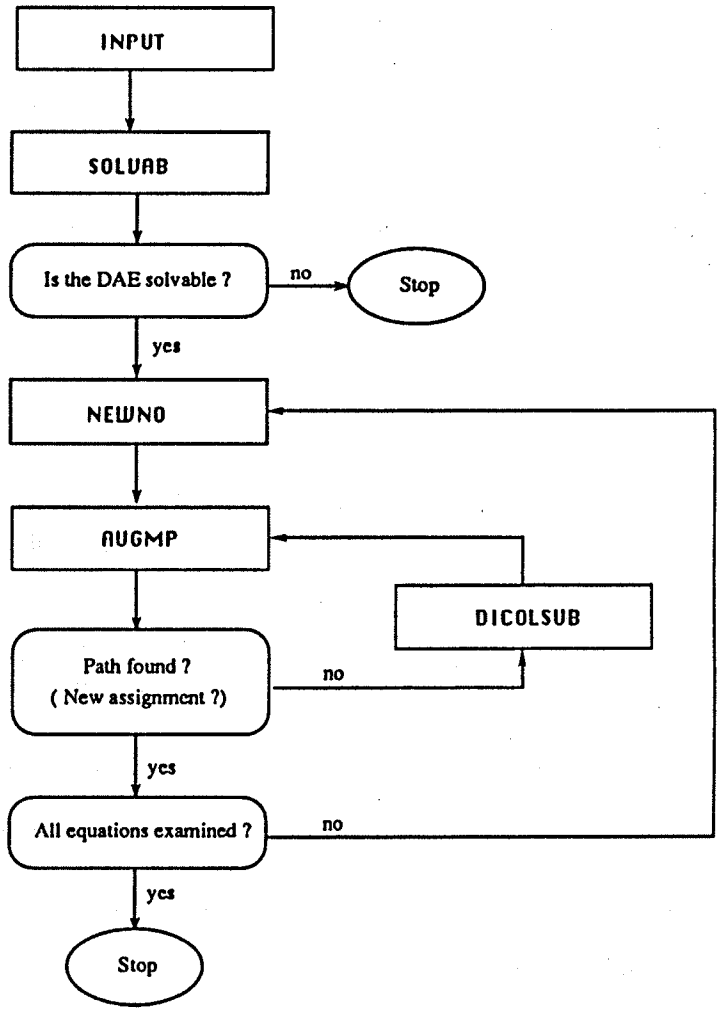


Figure D.2: Structure and Subroutines of PALG

Bibliography

[1] R. Bachmann, L. Brüll, U. Pallaske, Th. Mrziglod, *A contribution to the numerical treatment of differential-algebraic equations arising in Chemical Engineering*, DECHEMA-Monographs, Vol. 116 - VCH Verlagsgesellschaft, 1989, 343-349.

[2] M. Berzins and R.M. Furzeland, *A user's manual for SPRINT: Part 1*, Department of Computer Studies Report 199, Leeds University,1985.

[3] M. Berzins and R.M. Furzeland, *A user's manual for SPRINT: Part 1*, Department of Computer Studies Report 202, Leeds University,1986.

[4] R.K. Biener, *Polycondensation Processes in Distillation Tower Reactors*, Master-Thesis, University of Wisconsin - Madison, 1989.

[5] K.E. Brenan, S.L. Campbell, L.R. Petzold, *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, North-Holland,New York-Amsterdam-London, 1989.

[6] K.E. Brenan and B.E. Enquist, *Backward differentiation approximations of nonlinear differential/algebraic equations*, Math. Comp., 51 (1988), 659-676, S7-S16.

[7] S.L. Campbell, *Singular Systems of Differential Equations I*, Pitman, 1980.

[8] S.L. Campbell, *Singular Systems of Differential Equations II*, Pitman, 1982.

[9] S.L. Campbell, *Consistent initial conditions for singular nonlinear systems*, Circuits Systems Signal Process, 2 (1983), 44-55.

[10] M. Caracotsios, *Estimating consistent initial values in DAE's*, Research Note, AMOCO Chemical Plant, 1989.

[11] Y. Chung and A.W. Westerberg, *A proposed numerical algorithm for solving nonlinear index problems*, Ind. Eng. Chem. Res., 29 (1990), 1234-1239.

[12] J.D. Cobb, *A further interpretation of inconsistent initial conditions in descriptor-variable systems*, IEEE Trans. Aut. Cont., AC-28 (1983), 920-922.

[13] L. Dai, *Singular control systems*, Lecture Notes in Control and Information Science, 118, Springer, Berlin, 1989.

[14] P. Deuflhard, *One step and extrapolation methods for differential-algebraic equation systems*, Numer. Math., 51(1987), 501-516

[15] I.S. Duff and C.W. Gear, *Computing the structural index*, SIAM J. Alg. Discrete Methods, 7, (1986), 594-603.

[16] I.S. Duff, *On algorithms for obtaining a maximal transversal*, ACM Transs. Math. Softw., 7 (1981), 315-330.

[17] K. Fleischer, Universität Stuttgart, personal communication with W. Marquardt, 1990.

[18] F.R. Gantmacher, *The Theory of Matrices*, Vol. 2, Chelsea, 1964.

[19] C.W. Gear, *The simultaneous numerical solution of differential-algebraic equations*, IEEE Trans. Circ. Theory, CT-18 (1971), 89-95.

[20] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, 1971.

[21] C.W. Gear, *Maintaining solution invariants in the numerical solution of ODE's*, SIAM J. Sci. Stat. Comp., 7 (1986), 734-743.

[22] C.W. Gear, *Differential-algebraic equation index transformation* , SIAM J. Sci. Stat. Comp., 9 (1988), 39-47.

[23] C.W. Gear, B. Leimkuhler, and G.K. Gupta, *Automatic integration of Euler-Lagrange equations with constraints*, J. Comp. Appl. Math., 12 & 13 (1985), 77-90.

[24] C.W. Gear and L.R. Petzold, *Differential/algebraic Systems and matrix pencils*, in Matrix Pencils, edited by B. Kagstrom and A. Ruhe, Lecture Notes in Mathematics 973, Springer-Verlag (1983), 75-89.

[25] C.W. Gear and L.R. Petzold, *ODE methods for the solution of differential/algebraic systems*, SIAM J. Num. Anal., 21 (1984), 367-384.

[26] A. Georgiou and C.A. Floudas, *Optimization model for generic rank determination of structural matrices*, Int. J. Contr.,49 (1989), 1633-1644.

[27] E. Hairer, C. Lubich, and M. Roche, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, Universitè de Géneve, September 1988.

[28] F.E. Hohn, *Elementary Matrix Algebra*, 2nd Edt., Macmillan Company New York,1966.

[29] R.D. Johnston, G.W. Barton, and M.L. Brisk, *Determination of the generic rank of structural matrices*, Int. J. Contr.,40 (1984), 257-264.

[30] A.Kröner, P. Holl, W. Marquardt, E.D. Gilles, *Simulation strategy and numerical algorithms in the dynamic simulation environment DIVA*,Universität Stuttgart, F.R.G., AIChE-meeting, Chicago IL., December 1990, (to be submitted for publication).

[31] B.J. Leimkuhler, L.R. Petzold, C.W.Gear, *On the consistent initialization of differential-algebraic systems of equations*, SIAM J. Num. Anal. (to appear).

[32] F.L. Lewis, *A survey of linear singular systems*, Circuits Systems & Signal Processing, 5 (1986), 3-36.

[33] P. Lötstedt and L. Petzold, *Numerical Solution of nonlinear differential equations with algebraic constraints I: Convergences results for backward differentiation formulas*, Math. Comp., 46 (1986), 491-516.

[34] A.K. Louis, *Inverse und schlecht gestellte Probleme*, Teubner, Stuttgart, Studienbücher Mathematik, 1989.

[35] D.G. Luenberger, *Dynamic equations in descriptor form*, IEEE Trans. Aut. Control, AC-22 (1977),312-321.

[36] S.E. Mattson, *On modelling of differential/algebraic systems*, Simulation, January 1989.

[37] C.C. Pantelides, *The consistent initialization of differential -algebraic systems*, SIAM J. Sci. Stat. Comp., 9 (1988), 213-231.

[38] C.C. Pantelides, D. Gritsis, K.R. Morison, and R.W.H. Sargent, *The mathematical modeling of transient systems using differential-algebraic equations*, Comp. in Chem. Eng., 12 (1988), 449-454.

[39] L. Petzold, *Differential/algebraic equations are not ODE's*, SIAM J. Sci. Stat. Comp., 3 (1982), 367-384.

[40] L. Petzold, *A Description of DASSL: A differential/algebraic system solver*, in Scientific Computing, eds. R.S. Stepleman et al., Noth-Holland, Amsterdam, 1983, 65-68.

[41] W.H. Ray et. al., POLYRED *–A CAD package for polymerization processes*, University of Wisconsin Polymerization Reaction Engineering Laboratory, Madison WI, 1990.

[42] W.C. Rheinboldt, *Differential-algebraic systems as differential equations on manifolds*, Math. Comp., 43 (1984), 473-482.

[43] L.M. Silverman, *Inversion of multivariable systems*, IEEE Trans. Aut. Contr. , AC-14 (1969), 270-276.

[44] A. Skjellum, personal communication with W. Marquardt.

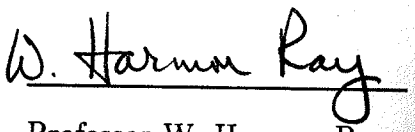
[45] M.A. Stadtherr and A. Lefkopoulos, *Index analysis of unsteady state chemical process simulation problems I: isolated units*, AIChE-meeting, Washington D.C., November 1988, (submitted for publication).

[46] M.A. Stadtherr and A. Lefkopoulos, *Index analysis of unsteady state chemical process engineering problems II: integrated unit system*, (submitted for publication).

[47] R. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comp. 1 (1972), 147-160.

[48] E.L. Yip, R.F. Sincovec, *Solvability, controlability, and observability of continuous descriptor systems*, IEEE Trans. Aut. Control, AC-26 (1981), 702-707.

APPROVED



Professor W. Harmon Ray

December 12, 1990.