Structural algorithms
and perturbations
in differential-algebraic equations

Henrik Tidefelt
This is a Swedish Licentiate’s Thesis.

Swedish postgraduate education leads to a Doctor’s degree and/or a Licentiate’s degree. A Doctor’s Degree comprises 160 credits (4 years of full-time studies). A Licentiate’s degree comprises 80 credits, of which at least 40 credits constitute a Licentiate’s thesis.
To Anna
Abstract

The quasilinear form of differential-algebraic equations is at the same time both a very versatile generalization of the linear time-invariant form, and a form which turns out to suit methods for index reduction which we hope will be practically applicable and well understood in the future.

The shuffle algorithm was originally a method for computing consistent initial conditions for linear time-invariant differential algebraic equations, but has other applications as well, such as the fundamental task of numerical integration. In the prospect of understanding how the shuffle algorithm can be applied to quasilinear differential-algebraic equations that cannot be analyzed by zero-patterns, the question of understanding singular perturbation in differential-algebraic equations has arose. This thesis details an algorithm for index reduction where this need is evident, and shows that the algorithm not only generalizes the shuffle algorithm, but also specializes the more general structure algorithm for system inversion by Li and Feng.

One chapter of this thesis surveys a class of forms of equations, searching less general forms than the quasilinear, to which an algorithm like ours can be tailored. It is found that the index reduction process often destroys structural properties of the equations, and hence that it is natural to work with the quasilinear form in its full generality.

The thesis also contains some early results on how the perturbations can be handled. The main results are inspired by the separate timescale modeling found in singular perturbation theory. While the singular perturbation theory considers the influence of a vanishing scalar in the equations, the analysis herein considers an unknown matrix bounded in norm by a small scalar. Results are limited to linear time-invariant equations of index at most 1, but it is worth noting that the index 0 case in itself holds an interesting generalization of the singular perturbation theory for ordinary differential equations.
Sammanfattning

Den kvasilinjära formen av differential-algebraiska ekvationer är både en mycket allmän-
giltig generalisering av den linjära tidsinvarianta formen, och en form som visar sig läm-
pa sig väl för indexreduktionsmetoder som vi hoppas ska komma att bli både praktiskt
tillämplbara och väl förstådda i framtiden.

Kuperingsalgoritmen (engelska: the shuffle algorithm) användes ursprungligen för att be-
stämma konsistenta initialvillkor för linjära tidsinvarianta differential-algebraiska ekva-
tioner, men har även andra tillämpningar, till exempel det grundläggande problemet nu-
merisk integration. I syfte att förstå hur kuperingsalgoritmen kan tillämpas på kvasilinjära
differential-algebraiska ekvationer som inte låter sig analyseras utifrån mönstret av nol-
lor, har problemet att förstå singulära perturbationer i differential-algebraiska ekvationer
uppstått. Den här avhandlingen presenterar en indexreduktionsmetod där behovet framgår
tydligt, och visar att algoritmen inte bara generaliserar kuperingsalgoritmen, utan även är
e ett specialfall av den mer allmänna strukturalgoritmen (engelska: the structure algorithm)
för att invertera system av Li och Feng.

Ett kapitel av den här avhandlingen söker av en klass av ekvations-former efter former
som är mindre generella än den kvasilinjära, men som en algoritm lik vår kan anpassas till.
Det visar sig att indexreduktionen ofta förstör strukturella egenskaper hos ekvationerna,
och att det därför är naturligt att arbeta med den mest allmänna kvasilinjära formen.

Avhandlingen innehåller också några tidiga resultat gällande hur perturbationerna kan
hanteras. Huvudresultaten är inspirerade av den modellering i skilda tidskalor som görs i
teorin om singulära perturbationer (engelska: singular perturbation theory). Medan teorin
om singulära perturbationer betraktar inverkan av en försvinnande skalär i ekvationerna,
betraktar analysen här en okänd matris vars norm begränsas av en liten skalär. Resultaten
är begränsade till linjära tidsinvarianta ekvationer av index inte högre än 1, men det är
värt att notera att index 0-fallet självt innebär en intressant generalisering av teorin för
singulära perturbationer för ordinära differentialekvationer.
Acknowledgments

An estimated 95% of you who read this know beforehand four persons I would like to thank here. I begin by thanking them for the obviously important reasons you think of. I am grateful to Professor Lennart Ljung, head of the Division of Automatic Control, for his warm acceptance of my wish to join the group, and for his everlasting efforts which make work with the group attractive also after three years. To my supervisor Professor Torkel Glad I am grateful because he lets me conduct research in the interesting field of differential-algebraic equations, and for believing in and helping me develop the few ideas I have had for research topics. Ulla Salaneck resolves the countless practical issues, quicker than anyone can imagine, and adds spirit to the group. The fourth person, (I drop titles from here on) Gustaf Hendeby, makes technical writing a breeze by providing us with document classes, know-how, and more.

I also want to thank Lennart for acting as a co-supervisor; it is in response to your requests that I have soon completed this thesis.

A very important contribution to any thesis is made by those who proofread, and in this case my gratitude goes to three persons, two of which have already been mentioned. Torkel and Johan Sjöberg are experts in my field, and while your comments make me think twice and greatly enhance the actual quality of my work, I take the freedom also to let any absence of comments give me faith in the quality of my work. Torkel is also the inventor of the thesis’ title. Gustaf is always a very thorough proofreader and has an eye for details, and I know it must hurt to see how I disobey many of your kind and well-founded recommendations on style.

I am indebted to the Swedish Research Council for financial support of this work.

Family, friends in the outing club, friends at work, surfers, salseros, computer language visionaries, gamblers, and everybody else sharing my spare time, thank you for giving me the energy to continue studying after so many years in school!

Anna, I know this thesis is useless to you, but it is out of respect — not of irony — my dedication goes to you.

Linköping, May 2007
Henrik Tidfelt
## Contents

1 Introduction ......................................................... 1
   1.1 Differential-algebraic equations in automatic control ............ 1
   1.2 Problem formulation ........................................... 2
   1.3 Contributions ................................................ 2
   1.4 Thesis outline ................................................ 3
   1.5 Notation ........................................................ 4

2 Background .......................................................... 7
   2.1 Models in automatic control ..................................... 7
      2.1.1 Examples .................................................... 8
      2.1.2 Use in estimation .......................................... 9
      2.1.3 Use in control ............................................. 9
      2.1.4 Model classes ............................................. 10
      2.1.5 Model reduction ......................................... 11
      2.1.6 Scaling .................................................... 13
   2.2 Differential-algebraic equations ................................ 14
      2.2.1 Motivation ................................................ 15
      2.2.2 Common forms ............................................. 16
      2.2.3 Indices and their deduction ............................. 19
      2.2.4 Transformation to quasilinear form ..................... 27
      2.2.5 Structure algorithm .................................... 29
      2.2.6 Initial conditions ..................................... 30
      2.2.7 Numerical integration .................................. 32
      2.2.8 Existing software ...................................... 35
   2.3 Singular perturbation theory .................................... 36
      2.3.1 LTI systems .............................................. 37
## Contents

2.3.2 Generalizations .................................................. 38  
2.4 Gaussian elimination ............................................. 38

3 Shuffling quasilinear DAE ........................................ 41
   3.1 Index reduction by shuffling .................................... 42
      3.1.1 The structure algorithm ................................... 42
      3.1.2 Quasilinear shuffling ...................................... 42
      3.1.3 Time-invariant input affine systems .................... 43
      3.1.4 Quasilinear structure algorithm ......................... 46
   3.2 Proposed algorithm ............................................. 47
      3.2.1 Algorithm .................................................. 47
      3.2.2 Zero tests ................................................ 50
      3.2.3 Longevity .................................................. 52
      3.2.4 Seminumerical twist ...................................... 55
      3.2.5 Monitoring ................................................ 55
      3.2.6 Sufficient conditions for correctness ................. 57
   3.3 Algorithm complexity .......................................... 60
      3.3.1 Representations and complexity ......................... 60
      3.3.2 Polynomial quasilinear DAE .............................. 60
   3.4 Consistent initialization ...................................... 62
      3.4.1 Motivating example ....................................... 62
      3.4.2 A bootstrap approach .................................... 64
      3.4.3 Comment .................................................. 64

4 Invariant forms ..................................................... 67
   4.1 Invariant forms ................................................. 67
      4.1.1 Structures and their algorithms ......................... 68
      4.1.2 An algorithm and its structures ......................... 68
   4.2 A class of candidate structures ................................ 69
   4.3 Analysis of candidate forms ................................... 70
      4.3.1 Leading matrix is independent of time ................. 71
      4.3.2 Leading matrix depends on time via driving function ... 72
      4.3.3 Leading matrix is general nonlinear .................... 72
      4.3.4 Example .................................................. 73
   4.4 Discussion ..................................................... 74
      4.4.1 Remarks on the example .................................. 75
      4.4.2 Extensions ................................................. 75

5 Introduction to singular perturbation in DAE ................. 77
   5.1 Motivation ..................................................... 77
      5.1.1 A linear time-invariant example ......................... 78
      5.1.2 Inspiring example ....................................... 79
      5.1.3 Application to quasilinear shuffling .................... 80
      5.1.4 A missing piece in singular perturbation of ODE ....... 81
   5.2 Solution by assumption ....................................... 81
      5.2.1 LTI algorithm ............................................ 81
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2</td>
<td>Making sense of an ill-posed problem</td>
<td>83</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Assumptions</td>
<td>84</td>
</tr>
<tr>
<td>5.3</td>
<td>Analysis</td>
<td>84</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Pointwise approximation</td>
<td>85</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Uniform approximation</td>
<td>88</td>
</tr>
<tr>
<td>5.4</td>
<td>Discussion</td>
<td>89</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Coping without A1</td>
<td>90</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Breaking A2</td>
<td>90</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Nature of the assumptions</td>
<td>91</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Applying the results</td>
<td>91</td>
</tr>
<tr>
<td>6</td>
<td>A different perturbation analysis</td>
<td>93</td>
</tr>
<tr>
<td>6.1</td>
<td>Preliminaries</td>
<td>94</td>
</tr>
<tr>
<td>6.2</td>
<td>Analysis</td>
<td>95</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Singular perturbation in ODE</td>
<td>95</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Singular perturbation in index 1 DAE</td>
<td>101</td>
</tr>
<tr>
<td>6.3</td>
<td>Discussion</td>
<td>105</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Nature of the assumptions</td>
<td>105</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Example</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>Concluding remarks</td>
<td>107</td>
</tr>
<tr>
<td>7.1</td>
<td>Conclusions</td>
<td>107</td>
</tr>
<tr>
<td>7.2</td>
<td>Directions for future research</td>
<td>108</td>
</tr>
<tr>
<td>A</td>
<td>Proofs</td>
<td>115</td>
</tr>
<tr>
<td>A.1</td>
<td>Complexity calculation</td>
<td>115</td>
</tr>
<tr>
<td>B</td>
<td>Notes on the implementation of the structure algorithm</td>
<td>119</td>
</tr>
<tr>
<td>B.1</td>
<td>Caching</td>
<td>119</td>
</tr>
<tr>
<td>B.2</td>
<td>Package symbols</td>
<td>120</td>
</tr>
<tr>
<td>B.3</td>
<td>Data driven interface</td>
<td>120</td>
</tr>
<tr>
<td>B.4</td>
<td>Representation</td>
<td>121</td>
</tr>
<tr>
<td>B.5</td>
<td>Example run</td>
<td>121</td>
</tr>
</tbody>
</table>
This chapter gives an introduction to the thesis by explaining very briefly the field in which it has been carried out, presenting the contributions in view of a problem formulation, and giving some reading directions and explanations of notation.

1.1 Differential-algebraic equations in automatic control

This thesis has been carried out at the Division of Automatic Control, Linköping University, Sweden, within the research area nonlinear and hybrid systems. Differential-algebraic equations is one of a small number of research topics in this area. We shall not dwell on whether these equations are particularly nonlinear or related to hybrid systems; much of the research so far in this group has been on linear time-invariant differential-algebraic equations, although there is now ongoing research also on differential-algebraic equations that are not linear. From here on, the abbreviation DAE will be used for differential-algebraic equation(s).

In the field of automatic control, various kinds of mathematical descriptions are used to build models of the object to be controlled. Sometimes, the equations are used primarily to compute information about the object (estimation), sometimes the equations are used primarily to compute control inputs to the object (control), and sometimes both tasks are performed in combination. From the automatic control point of view the DAE are thus of interest due to their ability to model objects. Not only are they able of modeling many objects, but in several situations they provide a very convenient way of modeling these objects, as is further discussed in section 2.2 In practice, the DAE generally contain
parameters that need to be estimated using measurements on the object; this process is
called identification.

In this thesis the concern is neither primarily with estimation, control, nor identification of
objects modeled by DAE. Rather, we focus on the more fundamental questions regarding
how the equations relate to their solution to so-called initial value problems. It is believed
that this will be beneficial for future development of the other three tasks.

1.2 Problem formulation

The long term goal of the work in this thesis concerns the quasilinear form of DAE,

\[ E(x(t), t) x'(t) + A(x(t), t) = 0 \]  \hspace{1cm} (1.1)

Here \( x(t) \) is a vector of states describing the object being modeled, at time \( t \). The matrix-valued function \( E \) and the vector-valued function \( A \) are assumed to be the result of some
kind of identification process, assigning nominal values to identified parameters, and also
providing information about uncertainties in these values. The long term goal is then to
understand how the corresponding initial value problems can be solved in presence of the
parameter uncertainties. Then, we will also get a better understanding of how one can
handle the problems without parameter uncertainties.

For reasons that will become more clear later, we shall often speak of perturbations rather
than uncertainties, and for consistency with subsequent chapters we will begin using this
nomenclature already here.

The long term goal is, however, just a long term goal; it is far beyond the reach of this
thesis to get to that point. Instead, a small number of problems of manageable size,
depicting more precisely the work in this thesis, are listed below:

- Is the quasilinear form really a good choice for the formulation of long term goals?
- By what approach shall perturbations be dealt with?
- How can the perturbations be understood and handled if the form of the equation is
  restricted to something much simpler than the quasilinear form?

1.3 Contributions

The main contributions in this thesis are, in approximate order of appearance:

- Viewing the shuffle algorithm as a special case of the structure algorithm. See
  section 3.1.
- The seminumerical approach in section 3.2.4, taken to avoid dependence on sym-
  bolic simplification and to allow treatment of inexact equations in index reduction.

\[ \text{The problem of computing the future trajectory of the object state given an initial state and external inputs.} \]
• The asymptotic algorithm complexity expressions for the quasilinear shuffle algorithm applied to polynomial equations. The expressions are given in section 3.3.2.

• Section 3.4 showing how to apply our seminumerical algorithm to find consistent initial conditions for a DAE.

• The systematic approach taken to surveying how various forms of DAE match algorithms. The core of this is found in sections 4.2 and 4.3.

• Highlighting the need to understand perturbations in DAE and how to turn this into a clearly defined problem. This is done in the early sections of chapter 5.

• Extending results from singular perturbation theory to restricted forms of lower index DAE. The analysis is found in section 6.2.

1.4 Thesis outline

The present chapter is completed by introducing some notation in the next section. It defines some non-standard notation, so it might be worth-while skimming through it before proceeding to later chapters.

The background given in chapter 2 is meant to introduce the objects and methods in general that are the subject of this thesis. It contains the previous results needed for the developments in later chapters. Readers familiar with automatic control in general and having some experience with DAE are unlikely to benefit from reading this chapter.

In chapter 3 a method for index reduction of quasilinear DAE is introduced. The chapter also contains some notes on consistent initialization of nonlinear DAE. A reader who is not particularly interested in the slight variation of existing index reduction methods that this chapter presents, might still find it interesting to see how the study of this algorithm raises the perturbation questions approached in the final chapters.

Given the rather general and potentially computation-wise expensive method of chapter 3 developed with a very general form of equations in mind, chapter 4 investigates what other forms that could be worth-while tailoring it to. Chapters 3 and 4 taken together are essentially an extension of Tidefelt [2007b].

Taking a few steps back from the expressive quasilinear form addressed in chapter 3 the problem of understanding singular perturbations in DAE is introduced for LTI DAE in chapter 5. Introducing the problem is the main contribution of the chapter, although it also presents a naïve way of analyzing it. This chapter is an adaptation of Tidefelt [2007a] to the thesis.

A less naïve approach is taken in chapter 6 where existing results from singular perturbation theory is extended to the LTI DAE. This chapter follows Tidefelt [2007c] closely.

Chapter 7 contains conclusions and directions for future research.
1.5 Notation

In accordance with most literature on this subject, equations not involving differentiated variables will often be denoted algebraic equations, although non-differential equations — a better notation from a mathematical point of view — will also be used interchangeably.\(^2\) The matrix-valued function \(E\) in (1.1) will be referred to as the leading matrix, while the function \(A\) will be referred to as the algebraic term.\(^3\) In an LTI ODE,

\[
x'(t) = M x(t) + B u(t)
\]

the matrix \(M\) is referred to as the state-feedback matrix.\(^4\) While \(x\) in the ODE is referred to as the state vector or just the state of the ODE, the elements of \(x\) in the DAE are referred to as the variables of the DAE.

A DAE is denoted square if the number of equations and variables match.

Let \(\lambda(X)\) denote the eigenvalues of \(X\), and let, for instance, \(\text{Re} \lambda(X) < 0\) mean that all eigenvalues have negative real parts.

The symbol \(\overset{\dagger}{=}\) is used to indicate an equality that shall be thought of as an equation. Compare this to the plain \(=\), which is used to indicate that expressions are equal in the sense that one can be rewritten as the other, possibly using context-dependent assumptions. For example, assuming \(x \geq 0\), we may write \(\sqrt{x^2} = x\).

The symbol :\(\overset{\triangle}{=}\) is used to introduce names for values or expressions. The meaning of expressions can be defined using the symbol \(\overset{\triangle}{=}\). Note that the difference between \(f :\overset{\triangle}{=} (x \mapsto x^2)\) and \(f(x) \overset{\triangle}{=} x^2\) is mainly conceptual; in many contexts both would work equally well.

The symbol \(\mathcal{I}\) denotes the identity matrix.

By an initial value problem we refer to the problem of computing trajectories of the variables of a DAE (or ODE), over an interval \([t_0, t_1]\), given sufficient information about the variables and their derivatives at time \(t_0\).

If \(x\) is a function of one variable (typically thought of as time), the derivative of \(x\) with respect to its only argument is written \(x'\). The composed symbol \(\dot{x}\) shall be used to denote a function which is independent of \(x\), but intended to coincide with \(x'\). For example, in numeric integration of \(x'' = u\), where \(u\) is a driving function, we write the ordinary differential equation as

\[
\begin{cases}
\dot{x}' = u \\
x' = \dot{x}
\end{cases}
\]

\(^2\)Seeking a notation which is both short and not misleading, the author would prefer static equations, but this notation is avoided to make the text more accessible.

\(^3\)By this definition, the algebraic term with reversed sign is sometimes referred to as the right hand side of the quasilinear DAE.

\(^4\)This notation is borrowed from [Kailath, 1980]. We hereby avoid the perhaps more commonly used notation system matrix, because of the other — yet related — meanings this term also bears.
Higher order derivatives are denoted \(x''\), \(x'''(3)\), . . . , or \(\ddot{x}, \dot{x}(3), . . .\). Making the distinction between \(x'\) and \(\dot{x}\) this way — and not the other way around — is partly for consistency with the syntax of the Mathematica language, in which our algorithms are implemented.

Gradients (Jacobians), are written using the operator \(\nabla\). For example, \(\nabla f\) is the gradient (Jacobian) of \(f\), assuming \(f\) takes one vector-valued argument. If a function takes several arguments, a subscript on the operator is used to denote with respect to which argument the gradient is computed. For example, if \(f\) is a function of 3 arguments, then \(\nabla_2 f = (x, y, z) \mapsto \nabla (w \mapsto f(x, w, z))(y)\).

For a time series \((x_n)_n\), the forward shift operator \(q\) is defined as \(qx_n \triangleq x_{n+1}\).

In the calculations to come, an uncertain matrix \(E\) will prevail. The set of all possible \(E\) shall be determined by context, and will not be part of our notation. For compactness, we shall write dependence on \(E\) with a subscript. For instance, writing \(y_E(\epsilon)\) means the same as writing \(y(\epsilon, E)\). We also need compact notation for limits that are uniform with respect to \(E\), and those that are not. Writing \(y_E(\epsilon) = O^E(\epsilon)\) means

\[
\exists k^0, \epsilon^* > 0 : \epsilon \in [0, \epsilon^*] \implies \sup_E |y_E(\epsilon)| \leq k^0 \epsilon
\]

while writing \(y_E(\epsilon) = O_E(\epsilon)\) means

\[
\forall E : \exists k^0, \epsilon^* > 0 : \epsilon \in [0, \epsilon^*] \implies |y_E(\epsilon)| \leq k^0 \epsilon
\]

Think of this notation as that \(E\) being a subscript on \(O\) means that the constants of the \(O\) are functions of \(E\); we could have written “\(\forall E : \exists k^0_E, \epsilon^*_E > 0 : \ldots\)” to emphasize this dependency. Also, \(E\) being a superscript can be used as a reminder of the \(\sup_E\) in the definition.

The few abbreviations used are summarized in table 1.1.

### Table 1.1: Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODE</td>
<td>Ordinary differential equation(s)</td>
</tr>
<tr>
<td>DAE</td>
<td>Differential-algebraic equation(s)</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear time-invariant</td>
</tr>
</tbody>
</table>
The intended audience of this thesis is not expected to have prior experience with either automatic control or differential-algebraic equations. For those without background in automatic control, we provide general motivation for why we study equations, and DAE in particular. For those with background in automatic control, but with only very limited experience with DAE, we try to fill that gap. This chapter also contains a discussion of some details of the well-known Gaussian elimination procedure.

This chapter contains all the existing results used in the development in later chapters.

### 2.1 Models in automatic control

Automatic control tasks are often solved by engineers without explicit mathematical models of the controlled or estimated object. For instance, a simple low pass filter may be used to get rid of measurement noise on the signal from a sensor, and this can work well even without saying *Assume that the correct measurement is distorted by zero mean additive high frequency noise*. Speaking out that phrase would express the use of a simple model of the sensor (whether it could be called *mathematical* is a matter of taste). As another example, many processes in industry are controlled by a so-called PID controller, which has a small number of parameters that can be tuned to obtain good performance. Often, these parameters are set manually by a person with experience of how these parameters relate to production performance, and this can be done without awareness of mathematical models. Most advances in control and estimation theory do, however, build on the assumption that a more or less accurate mathematical model of the object is available, and how such models may be used, simplified, and tuned for good numerical properties is the subject of this section.
2.1.1 Examples

The model of the sensor above was only expressed in words. Our first example of a mathematical model will be to say the same thing with equations. Since equations are typically more precise than words, we will lose some of the generality, a price we are often willing to pay to get to the equations which we need to be able to apply our favorite methods for estimation and/or control. Denote, at time $t$, the measurement by $y(t)$, the true value by $x(t)$, and let $e$ be a white noise source with variance $\sigma^2$. Let $v(t)$ be an internal variable of our model:

$$y(t) = x(t) + v(t)$$

(2.1a)

$$v(t) + v'(t) = e'(t)$$

(2.1b)

A drawback of using a precise model like this is that our methods may depend too heavily on that this is the correct model; we need to be aware of how sensitive our methods are to errors in the mathematical model. Imagine, for instance, that we build a device that can remove disturbances at $50 \text{ Hz}$ caused by the electric power supply. If this device is too good at this, it will be useless if we move to a country where the alternate current frequency is $60 \text{ Hz}$, and will even destroy information of good quality at $50 \text{ Hz}$. The model (2.1) is often written more conveniently in the Laplace transform domain, which is possible since the differential equations are linear:

$$Y(s) = X(s) + V(s)$$

(2.2a)

$$V(s) = \frac{s}{1 + s} E(s)$$

(2.2b)

Here, the $s/(1 + s)$ is often referred to as a filter; the white noise is turned into high frequency noise by sending it through the filter.

As a second example of a mathematical model we consider a laboratory process often used in basic courses in automatic control. The process consists of a cylindrical water tank, with a drain at the bottom. Water can be pumped from a reservoir to the tank, and the drain leads water back to the reservoir. There is also a gauge that senses the level of water in the tank. The task for the student is to control the level of water in the tank, and what makes the task interesting is that the flow of water through the drain varies with the level of water; the larger the level of water, the higher the flow. Limited performance can be achieved using for instance, a manually tuned PID controller, but to get good performance at different desired levels of water, a model-based controller is the natural choice. Let $x$ denote the level of water, and $u$ the flow we demand from the pump. A common approximation is that the flow through the drain is proportional to the square root of the level of water. Denote the corresponding constant $c_d$, and let the constant relating the flow of water to the time derivative (that is, the inverse of the bottom area of the tank) be denoted $c_a$. Then we get the following mathematical model with two

---

1White noise and how it is used in the example models is a non-trivial subject, but to read this chapter it should suffice to know that white noise is a concept which is often used as a building block of more sophisticated models of noise.
parameters to be determined from some kind of experiment:

\[ x'(t) = c_a \left( u(t) - c_d \sqrt{x(t)} \right) \]  

(2.3)

The constant \( c_a \) could be determined by plugging the drain, adding a known volume of water to the tank, and measuring the resulting level. The other constant can also be determined from simple experiments.

### 2.1.2 Use in estimation

The first model example above was introduced with a very easy estimation problem in mind. Let us instead consider the task of computing an accurate estimate of the level of water, given a sensor that is both noisy and slow. We will not go into details here, but just mention the basic idea of how the model can be used.

Since the flow we demand from the pump, \( u \), is something we choose, it is a known quantity in (2.3). Hence, if we were given a correct value of \( x(0) \) and the model would be correct, we could compute all future values of \( x \) simply by integration of (2.3). However, our model will never be correct, so the estimate will only be good during a short period of time, before the estimate has drifted away from the true value. The errors in our model are not only due to the limited precision in the experiments used to determine the constants, but more importantly because the square root relation is a rather coarse approximation. In addition, it is unrealistic to assume that we get exactly the flow we want from the pump. This is where the sensor comes into play; even though it is slow and noisy, it is sufficient to take care of the drift. The best of both worlds can then be obtained by combining the simulation of (2.3) with use of the sensor in a clever way. A very popular method for this is the so-called extended Kalman filter (for instance, Jazwinski [1970, theorem 8.1]).

### 2.1.3 Use in control

Let us consider the laboratory process (2.3) again. The task was to control the level of water, and this time we assume that the errors in the measurements are negligible. There is a maximal flow, \( u_{\text{max}} \), that can be obtained from the pump, and it is impossible to pump water backwards from the tank to the reservoir, so we shall demand a flow subject to the constraints \( 0 \leq u(t) \leq u_{\text{max}} \). We denote the desired level of water the set point, symbolized by \( x_{\text{ref}} \). The theoretically valid control law,

\[
  u(t) = \begin{cases} 
    0, & \text{if } x(t) \geq x_{\text{ref}}(t) \\
    u_{\text{max}}, & \text{otherwise} 
  \end{cases}
\]

(2.4)

will be optimal in theory (when changes in \( x_{\text{ref}} \) cannot be foreseen) in the sense that deviations from the set point are eliminated as quickly as possible. However, this type of control law will quickly wear the pump since it will be switching rapidly between off and full speed once the level gets to about the right level. Although still unrealistically naïve, at least the following control law somewhat reduces wear of the pump, at the price
of allowing slow and bounded drift away from the set point. It has three modes, called the 
**drain mode**, the **fill mode**, and the **open-loop mode**:

**Drain mode:**
\[
\begin{cases} 
  u(t) = 0 \\
  \text{Switch to open-loop mode if } x(t) < x_{\text{ref}}(t)
\end{cases}
\]

**Fill mode:**
\[
\begin{cases} 
  u(t) = u_{\text{max}} \\
  \text{Switch to open-loop mode if } x(t) > x_{\text{ref}}(t)
\end{cases}
\]

**Open-loop mode:**
\[
\begin{cases} 
  u(t) = c_d \sqrt{x_{\text{ref}}(t)} \\
  \text{Switch to drain mode if } x(t) > (1 + \delta) x_{\text{ref}}(t) \\
  \text{Switch to fill mode if } x(t) < (1 - \delta) x_{\text{ref}}(t)
\end{cases}
\]

where the parameter $\delta$ is a small parameter chosen by considering the trade-off between 
performance and wear of the pump. In the open-loop mode, the flow demanded from the 
pump is chosen to match the flow through the drain to the best of our knowledge. Note 
that if $\delta$ is sufficiently large, errors in the model will make the level of water settle at the 
wrong level; to each fixed flow there is a corresponding level where the level will settle, 
and errors in the model will make $c_d \sqrt{x_{\text{ref}}(t)}$ correspond to something slightly different 
from $x_{\text{ref}}(t)$. More sophisticated controllers can remedy this.

### 2.1.4 Model classes

When developing theory, be it system identification, estimation or control, one has to 
specify the structure of the models to work with. We shall use the term **model class** to 
denote a set of models which can be easily characterized. A model class is thus a rather 
vague term such as, for instance, a linear system with white noise on the measurements. 
Depending on the number of states in the linear system, and how the linear system is pa-
rameterized, various **model structures** are obtained. When developing theory, a parameter 
such as the number of states is typically represented by a symbol in the calculations — 
this way, several model structures can be treated in parallel, and it is often possible to 
draw conclusions regarding how such a parameter affects some performance measure. In 
the language of system identification, one would thus say that theory is developed for a 
parameterized family of model structures. Since such a family is a model class, we will 
often have such a family in mind when speaking of model classes. The concepts of models, model sets, and model structures are rigorously defined in the standard [Ljung (1999), section 4.5] on system identification, but we shall allow these concepts to be used in a 
broader sense here.

In system identification, the choice of model class affects the ability to approximate the 
true process as well as how efficiently or accurately the parameters of the model may be 
determined. In estimation and control, applicability of the results is related to how likely it 
is that a user will choose to work with the treated model structure, in light of the power of 
the results; a user may be willing to identify a model from a given class if that will enable 
the user to use a more powerful method. The choice of model class will also allow various 
amount of elaboration of the theory; a model class with much structural information will
generally allow a more precise analysis, at the cost of increased complexity, both in terms of theory and implementation of the results.

Before we turn to some examples of model classes, it should be mentioned that models are often describing a system in discrete time. However, this thesis is only concerned with continuous time models, so the examples will all be of this kind.

Continuing on our first example of a model class, in the sense of a parameterized family of model structures, it could be described as all systems in the linear state space form

\[ x'(t) = Ax(t) + Bu(t) \]
\[ y(t) = Cx(t) + Du(t) + v(t) \] (2.6)

where \( u \) is the vector of system inputs, \( y \) the vector of measured outputs, \( v \) is a vector with white noise, and \( x \) is a finite-dimensional vector of states. For a given number of states, \( n \), a model is obtained by instantiating the matrices \( A, B, C, \) and \( D \) with numerical values.

It turns out that the class (2.6) is over-parameterized in the sense that it contains many equivalent models. If the system has just one input and one output, it is well-known that it can be described by \( 2n + 1 \) parameters, and it is possible to restrict the structure of the matrices such that they only contain so many unknown parameters without reducing the possible input-output relations.

Our second and final example of a model class is obtained by allowing more freedom in the dynamics than in (2.6), while removing the part of the model that relates the system output to its states. In a model of this type, all states are considered outputs:

\[ x'(t) = A(x(t)) + Bu(t) \] (2.7)

Here, we might pose various types of constrains on the function \( A \). For instance, assuming Lipschitz continuity is very natural since it ensures that the model uniquely defines the trajectory of \( x \) as a function of \( u \) and initial conditions. Another interesting choice for \( A \) is the polynomials, and if the degree is at most 2 one obtains a small but natural extension of the linear case. Another important way of extending the model class (2.6) is to look into how the system inputs \( u \) are allowed to enter the dynamics.

2.1.5 Model reduction

Sophisticated methods in estimation and control may result in very computationally expensive implementations when applied to large models. By large models, we generally refer to models with many states. For this reason methods and theory for approximating large models by smaller ones have emerged. This approximation process is referred to as model reduction. Our interest in model reduction owes to its relation to index reduction (explained in section 2.2), a relation which may not be widely recognized, but one which this thesis tries bring attention to. This section provides a small background on some available methods.

In view of the DAE for which index reduction is considered in detail in later chapters, we shall only look at model reduction of LTI systems here, and we assume that the large model is given in state space form as in (2.6).
If the states of the model have physical meaning it might be desirable to produce a smaller model where the set of states is a subset of the original set of states. It then becomes a question of which states to remove, and how to choose the system matrices $A$, $B$, $C$, and $D$ for the smaller system. Let the states and matrices be partitioned such that $x_2$ are the states to be removed (this requires the states to be reordered if the states to be removed are not the last components of $x$), and denote the blocks of the partitioned matrices according to

$$
\begin{bmatrix}
x'_1(t) \\
x'_2(t)
\end{bmatrix} =
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix} +
\begin{bmatrix}
B_1 \\
B_2
\end{bmatrix} u(t)
$$

(2.8)

If $x_2$ is selected to consist of states that are expected to be unimportant due to the small values those states take under typical operating conditions, one conceivable approximation is to set $x_2 = 0$ in the model. This results in the \textit{truncated model}

$$
x'_1(t) = A_{11} x_1(t) + B_1 u(t)
$$

(2.9)

$$
y(t) = C_1 x_1(t) + D u(t) + v(t)
$$

Although — at first glance — this might seem like a reasonable strategy for model reduction, it is generally hard to tell how the reduced model relates to the original model. Also, selecting which states to remove based on the size of the values they typically take is in fact a meaningless criterion, since any state can be made small by scaling, see section 2.1.6.

Another approximation is obtained by formally replacing $x'_2(t)$ by 0 in (2.8). The underlying assumption is that the dynamics of the states $x_2$ is very fast compared to $x_1$. A necessary condition for this to make sense is that $A_{22}$ be Hurwitz, which also makes it possible to solve for $x_2$ in the obtained equation $A_{21} x_1(t) + A_{22} x_2(t) + B_2 u(t) \neq 0$.

Inserting the solution in (2.8) results in the \textit{residualized model}

$$
x'_1(t) = \left( A_{11} - A_{12} A_{22}^{-1} A_{12} \right) x_1(t) + \left( B_1 - A_{12} A_{22}^{-1} B_2 \right) u(t)
$$

$$
y(t) = \left( C_1 - C_2 A_{22}^{-1} A_{12} \right) x_1(t) + \left( D - C_2 A_{22}^{-1} B_2 \right) u(t) + v(t)
$$

(2.10)

It can be shown that this model gives the same output as (2.8) for constant inputs $u$.

If the states of the original model do not have interpretations that we are keen to preserve, the above two methods for model reduction can produce an infinite number of approximations if combined with a change of variables applied to the states; applying the change of variables $x = T \xi$ to (2.6) results in

$$
\begin{bmatrix}
\xi'_1(t) \\
\xi'_2(t)
\end{bmatrix} =
\begin{bmatrix}
T^{-1} A T \xi(t) + T^{-1} B u(t) \\
D u(t) + v(t)
\end{bmatrix}
$$

(2.11)

and the approximations will be better or worse depending on the choice of $T$. Conversely, by certain choices of $T$, it will be possible to say more regarding how close the approximations are to the original model. If $T$ is chosen to bring the matrix $A$ in Jordan form,
truncation is referred to as *modal truncation*, and residualization is then equivalent to *singular perturbation approximation* (see section 2.3). [Skogestad and Postlethwaite, 1996]

The change of variables $T$ most well developed is that which brings the system in *balanced form*. When performing truncation or residualization on a system in this form, the difference between the approximation and the original system can be expressed in terms of the system’s *Hankel singular values*. We shall not go into details about what these values are, but the largest defines the *Hankel norm* of a system. Neither shall we give interpretations of this norm, but it turns out that it is actually possible to compute the reduced model of a given order which minimizes the Hankel norm of the difference between the original system and the approximation.

By now we have seen that there are many ways to compute smaller approximations of a system, ranging from rather arbitrary choices to those which are clearly defined as minimizers of a coordinate-independent objective function.

Some model reduction techniques have been extended to linear time-invariant (hereafter LTI) DAE. [Stykel, 2004] However, although the main question in this thesis is closely related to model reduction, these techniques cannot readily be applied in our framework since we are interested in defending a given model reduction (this view should become clear in later chapters) rather than finding one with good properties.

### 2.1.6 Scaling

In section 2.1.5, we mentioned that model reduction of a system in state space form, (2.6), was a rather arbitrary process unless thinking in terms of some suitable coordinate system for the state space. The first example of this was selecting which states to truncate based on the size of the values that the state attains under typical operating conditions, and here we do the simple maths behind that statement. Partition the states such that $x_2$ is a single state which is to be scaled by the factor $a$. This results in

$$
\begin{align*}
(x_1'(t) \\
x_2'(t)
\end{align*}
= \begin{pmatrix}
A_{11} & \frac{1}{a} A_{12} \\
a A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
x_1(t) \\
x_2(t)
\end{pmatrix}
+ \begin{pmatrix}
B_1 \\
a B_2
\end{pmatrix} u(t)
$$

$$
y(t) = (C_1 \frac{1}{a} C_2)
\begin{pmatrix}
x_1(t) \\
x_2(t)
\end{pmatrix}
+ D u(t) + v(t)
$$

(not writing out that also initial conditions have to be scaled appropriately). Note that the scalar $A_{22}$ on the diagonal does not change (if it would, that would change the trace of $A$, but the trace is known to be invariant under similarity transforms).

In the index reduction procedure studied in later chapters, the situation is reversed: it is not a question about which states are small, but which coefficients that are small. The situation is even worse for LTI DAE than for the state space systems considered so far, since in a DAE there is also the possibility to scale the equations independently of the states. Again, it becomes obvious that this cannot be answered in a meaningful way unless the coordinate systems for the state space and the equation residuals are chosen suitably. Just like in model reduction, the user may be keen to preserve the interpretation of the
model states, and may hence be reluctant to use methods that apply variable transforms to the states. However, unlike model reduction of ordinary differential equations, the DAE may still be transformed by changing coordinates of the equation residuals. In fact, changing the coordinate system of the equation residuals is the very core of the index reduction algorithm.

Pure scaling of the equation residuals is also an important part of the numerical method for integration of DAE that will be introduced in section 2.2.7. There, scaling is important not because it facilitates analysis, but because it simply improves numeric quality of the solution. To see how this works, we use the well-known (see, for instance, Golub and Van Loan [1996]) bound on the relative error in the solution to a linear system of equations $A x = b$, which basically says the relative errors in $A$ and $b$ are propagated to $x$ by a factor bounded by the (infinity norm) condition number of $A$. Now consider the linear system of equations in the variable $q x$ (that is, $x$ is given)

$$(\frac{1}{\epsilon} E_1 + A_1 \ A_2) q x = \left( \frac{1}{\epsilon} E_1 \ 0 \right) x \quad (2.13)$$

where $\epsilon$ is a small but exactly known parameter. If we assume that the relative errors in $E$ and $A$ are of similar magnitudes, smallness of $\epsilon$ gives both that the matrix on the left hand side is ill-conditioned, and that the relative error of this matrix is approximately the same as the relative error in $E_1$ alone. Scaling the upper row of equations will hence make the matrix on the left hand side better conditioned, while not making the relative error significantly larger. On the right hand side, scaling of the upper block by $\epsilon$ is the same as scaling all of the right hand side by $\epsilon$, and hence the relative error does not change. Hence, scaling by $\epsilon$ will give a smaller bound on the relative error in the solution. Although the scaling by $\epsilon$ was performed for the sake of numerics, it should be mentioned that, generally, the form (2.13) is only obtained after choosing a suitable coordinate system for the DAE residuals.

Another important situation we would like to mention — when scaling matters — is when gradient-based methods are used in numerical optimization. (Numerical optimization in one form or another is the basic tool for system identification.) Generally, the issue is how the space of optimization variables is explored, not so much the numerical errors in the evaluation of the objective function and its derivatives. It turns out that the success of the optimization algorithm depends directly on how the optimization variables (that is, model parameters to be identified) are scaled. One of the important advantages of optimization schemes that also make use of the Hessian of the objective function is that they are unaffected by linear changes of variables.

### 2.2 Differential-algebraic equations

Differential-algebraic equations (generally written just DAE) is a rather general kind of equations which is suitable for describing systems which evolve over time. The advantage they offer over the more often used ordinary differential equations is that they are generally easier to formulate. The price paid is that they are more difficult to deal with.
The first topic of the background we give in this section is to try to clarify why DAE can be a convenient way of modeling systems in automatic control. After looking at some common forms of DAE, we then turn to the basic elements of analysis and solution of DAE. Finally, we mention some existing software tools. For recent results on how to carry out applied tasks such as system identification and estimation for DAE models, see Gerdin [2006], or for optimal control, see Sjöberg [2006].

2.2.1 Motivation

Nonlinear differential-algebraic equations is the natural outcome of component-based modeling of complex dynamic systems. Often, there is some known structure to the equations, for instance, the long term goal behind the work in this thesis is to better understand a method that applies to equations in quasilinear form,

\[ E(\mathbf{x}(t), t) \mathbf{x}'(t) + A(\mathbf{x}(t), t) \overset{1}{\mathbf{x}'(t)} = 0 \]  

(2.14)

In the next section, we approach this form by looking at increasingly general types of equations.

Within many fields, equations emerge in the form (2.14) without being recognized as such. The reason is that when \( \mathbf{x}'(t) \) is sufficiently easy to solve for, the equation is converted to the state space form which can be written formally as

\[ \mathbf{x}'(t) \overset{1}{=} -E(\mathbf{x}(t), t)^{-1} A(\mathbf{x}(t), t) \]

Sometimes, the leading matrix, \( E \), may be well conditioned, but nevertheless non-trivial to invert. It may then be preferable to leave the equations in the form (2.14). In this case, the form (2.14) is referred to as an implicit ODE or an index 0 DAE. One reason for not converting to state space form is that one may loose sparsity patterns. Hence, the state space form may require much more storage than the implicit ODE, and may also be a much more expensive way of obtaining \( \mathbf{x}'(t) \). Besides, even when the inverse of a sparse symbolic matrix is also sparse, the expressions in the inverse matrix are generally of much higher complexity.

Although an interesting case by itself, the implicit ODE form is not the purpose in this thesis. What remains is the case when the leading matrix is singular. Such equations appear naturally in many fields, and we will finish this section by looking briefly at some examples.

\[ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}^{-1} = \frac{1}{2} \begin{pmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} \]

- Here is an example that shows that the inverse of a sparse matrix may be full:

3If the example above is extended to a 5 by 5 matrix with unique symbolic constants at the non-zero positions, the memory required to store the original matrix in Mathematica [Inc. 2005] is 480 bytes. If the inverse is represented with the inverse of the determinant factored out, the memory requirement is 1400 bytes, and without the factorization the memory requirement is 5480 bytes.
As was mentioned above, quasilinear equations with singular leading matrix is the natural outcome of component-based modeling. This type of modeling refers to the bottom-up process, where one begins by making small models of simple components. The small models are then combined to form bigger models, and so on. Each component, be it small or large, have variables that are thought of as inputs and outputs, and when models are combined to make models at a higher level, this is done by connecting outputs with inputs. Each connection renders a trivial equation where two variables are “set” equal. These equations contain no differentiated variables, and will hence have a corresponding zero row of the leading matrix. The leading matrix must then be singular, but the problem has a prominent structure which is easily exploited.

Our next example is models of electric networks. Here, many components (or sub-networks) may be connected in one node, where all electric potentials are equal and Kirchoff’s Current Law provides the glue for currents. While the equations for the potentials are trivial equalities between pairs of variables, the equations for the currents will generate linear equations involving several variables. Still, the corresponding part of the leading matrix is a zero row, and the coefficients of the currents are ±1, when present. This structure is also easy to exploit.

The previous example is often recognized as one of the canonical applications of the so-called bond graph theory. Other domains where bond graphs are used are mechanical translation, mechanical rotation, hydraulics (pneumatics), some thermal systems, and some systems in chemistry. (Note that the applicability to mechanical systems is rather limited, as objects are required to either translate along a given line, or rotate about a given axis). In the bond graph framework, the causality of a model needs to be determined in order to generate model equations in ODE form. However, the most frequently used technique for assigning causality to the bond graph, named Sequential Causality Assignment Procedure \[\text{[Rosenberg and Karnopp][1983] section 4.3}\], suffers from a potential problem with combinatorial blow-up. One way of avoiding this problem is to generate a DAE instead.

Although some chemical processes can be modeled using bond graphs, this framework is rarely mentioned in recent literature on DAE modeling in the chemistry domain. Rather, equation-based formulations prevail, and according to \[\text{Unger et al.}[1995]\], most models have the quasilinear form. The amount on DAE research within the field of chemistry is remarkable, which is likely due to their extensive applicability in a profitable business where high fidelity models are a key to better control strategies.

### 2.2.2 Common forms

Having presented the general idea of finding suitable model classes to work with in section 2.1.4, this section contains some common cases from the DAE world. As we are moving our focus away from the automatic control applications that motivate our research, towards questions of more generic mathematical kind, our notation changes; instead of using model class, we will now speak of the form of an equation.
Beginning with the overly simple, an autonomous LTI DAE has the form

\[ E x'(t) + A x(t) \overset{!}{=} 0 \] (2.15)

where \( E \) and \( A \) are constant matrices. By autonomous, we mean that there is no way external inputs can enter this equation, so the system evolves in a way completely defined by its initial conditions. Adding driving functions (often representing external inputs) while maintaining the LTI property leads to the general LTI DAE form

\[ E x'(t) + A x(t) + B u(t) \overset{!}{=} 0 \] (2.16)

where \( u \) is a vector-valued function representing external inputs to the model, and \( B \) is a constant matrix. The function \( u \) is always considered known when analyzing the equation, and may be subject to various assumptions.

**Example 2.1**

In automatic control, system inputs are often computed as functions of the system state or an estimate thereof — this is called feedback — but such inputs are not external. To see how such feedback loops may be conveniently modeled using DAE models, let

\[ E_G x'(t) + A_G x(t) + \begin{pmatrix} B_{G1} & B_{G2} \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \overset{!}{=} 0 \] (2.17)

be a model of the system without the feedback control. Here, the inputs to the system has been partitioned into one part, \( u_1 \), which will later be given by feedback, and one part, \( u_2 \), which will be the truly external inputs to the feedback loop. Let

\[ E_H \hat{x}'(t) + A_H \hat{x}(t) + \begin{pmatrix} B_{H1} & B_{H2} \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \overset{!}{=} 0 \] (2.18)

be the equations of the observer, generating the estimate \( \hat{x} \) of the true state \( x \). Finally, let a simple feedback be given by

\[ u_1(t) = L \hat{x}(t) \] (2.19)

Now, it is more of a matter of taste whether to consider the three equations (2.17), (2.18), and (2.19) to be in form (2.16) or not; if not, it just remains to note that if \( u_1 \) is made an internal variable of the model, the equations can be written

\[
\begin{pmatrix} E_G & E_H \\
E_G & E_H \end{pmatrix} \begin{pmatrix} x'(t) \\ \hat{x}'(t) \\ u_1'(t) \end{pmatrix} + \begin{pmatrix} A_G & B_{G1} \\ A_H & B_{H1} \\ -L & I \end{pmatrix} \begin{pmatrix} x(t) \\ \hat{x}(t) \\ u_1(t) \end{pmatrix} + \begin{pmatrix} B_{G2} \\ B_{H2} \end{pmatrix} u_2(t) \overset{!}{=} 0
\] (2.20)

Of course, eliminating \( u_1 \) from these equations would be trivial;

\[
\begin{pmatrix} E_G & E_H \\
E_G & E_H \end{pmatrix} \begin{pmatrix} x'(t) \\ \hat{x}'(t) \end{pmatrix} + \begin{pmatrix} A_G & -B_{G1} L \\ A_H & -B_{H1} L \end{pmatrix} \begin{pmatrix} x(t) \\ \hat{x}(t) \end{pmatrix} + \begin{pmatrix} B_{G2} \\ B_{H2} \end{pmatrix} u_2(t) \overset{!}{=} 0
\]

but the purpose of this example is to show how the model can be written in a form that is both a little easier to formulate and that is better at displaying the logical structure of the model.
One way to generalize the form (2.16) is to remove the restriction to time-invariant equations. This leads to the linear, time-varying form of DAE:

$$E(t) x'(t) + A(t) x(t) + B(t) u(t) = 0$$  \thetag{2.21}

While this form explicitly displays what part of the system’s time variability that is due to “external inputs”, one can, without loss of generality, assume that the equations are in the form

$$E(t) x'(t) + A(t) x(t) = 0$$ \thetag{2.22}

This is seen by (rather awkwardly) writing (2.21) as

$$
\begin{pmatrix}
E(t) \\
I
\end{pmatrix}
\begin{pmatrix}
x'(t) \\
\alpha'(t)
\end{pmatrix} +
\begin{pmatrix}
A(t) & B(t) u(t)
\end{pmatrix}
\begin{pmatrix}
x(t) \\
\alpha(t)
\end{pmatrix} = 0
\alpha(t_0) = 1
$$

where the variable $\alpha$ has been included as an awkward way of denoting the constant 1. Still, the form (2.21) is interesting as it stands since it can express logical structure in a model, and if algorithms exploit that structure one may obtain more efficient implementations or results that are easier to interpret. In addition, it should be noted that the model structures are not fully specified without telling what constraints the various parts of the equations must satisfy. If one can handle a larger class of functions representing external inputs in the form (2.21) than the class of functions at the algebraic term in (2.22), there are actually systems in the form (2.21) which cannot be represented in the form (2.22). The same kind of considerations should be made when considering the form

$$E(t) x'(t) + A(t) x(t) + f(t) = 0$$ \thetag{2.23}

as a substitute for (2.21).

A natural generalization of (2.23) is to allow dependency of all variables where (2.23) only allows dependency of $t$. With the risk of loosing structure in problems with external inputs etc the resulting equations can be written in the form

$$E(x(t), t) x'(t) + A(x(t), t) = 0$$ \thetag{2.24}

The most general form of DAE is

$$f(x'(t), x(t), t) = 0$$ \thetag{2.25}

but it takes some analysis to realize why writing this equation as

$$\begin{cases}
f(\dot{x}(t), x(t), t) = 0 \\
\dot{x}(t) - x'(t) = 0
\end{cases}$$ \thetag{2.26}

does not show that (2.24) is the most general form we need to consider.

Other, less common forms of DAE, obtained by considering various restrictions of (2.24), will be investigated in chapter 3.
So far, we have considered increasingly general forms of DAE without considering how the equations can be analyzed. For instance, modeling often leads to equations which are clearly separated into differential and non-differential equations, and this structure is often possible to exploit. Since discussion of the following forms requires the reader to be familiar with the contents of section 2.2.3, the forms will only be mentioned quickly to give some intuition about what forms with this type of structural properties may look like. What follows is a small and rather arbitrary selection of the forms discussed in Brenan et al. [1996].

The semi-explicit form looks like

\[
x_1'(t) \overset{!}{=} f_1(x_1(t), x_2(t), t) \\
0 \overset{!}{=} f_2(x_1(t), x_2(t), t)
\]

and one often speaks of semi-explicit index 1 DAE (the concept of an index will be discussed further in section 2.2.3), which means that the function \( f_2 \) is such that \( x_2 \) can be solved for:

\[
\nabla_2 f_2 \text{ is square and non-singular}
\]

(2.28)

Another often used form is the Hessenberg form of size \( r \),

\[
x_1'(t) \overset{!}{=} f_1(x_1, x_2, \ldots, x_r, t) \\
x_2'(t) \overset{!}{=} f_2(x_1, x_2, \ldots, x_{r-1}, t) \\
\vdots \\
x_i'(t) \overset{!}{=} f_2(x_{i-1}, x_i, \ldots, x_{r-1}, t) \\
\vdots \\
0 \overset{!}{=} f_r(x_{r-1}, t)
\]

(2.29)

where it is required that

\[
\begin{pmatrix}
\frac{\partial f_r(x_{r-1}, t)}{\partial x_{r-1}} \\
\frac{\partial f_{r-1}(x_{r-2}, t)}{\partial x_{r-2}} \\
\vdots \\
\frac{\partial f_2(x_1, x_2, \ldots, x_{r-1}, t)}{\partial x_1} \\
\frac{\partial f_1(x_1, x_2, \ldots, x_r, t)}{\partial x_r}
\end{pmatrix}
\]

is non-singular.

### 2.2.3 Indices and their deduction

In the previous sections, we have spoken of the index of a DAE and index reduction, and we have used the notions as if they were well defined. This is not the case; there are many definitions of indices. In this section, we will mention some of these definitions, and
define what shall be meant by just index (without qualification) in the remainder of the thesis. We shall do this in some more length than what is needed for the following chapters, since this is a good way of introducing readers with none or very limited experience with DAE to typical DAE issues.

At least three categories of indices can be identified:

- For equations that relate driving functions to the equation variables, there are indices that are equal for any two equivalent equations. In other words, these indices are not a property of the equations per se, but of the abstract system defined by the equations.

- For equations written in particular forms, one can introduce perturbations or driving functions at predefined slots in the equations, and then define indices that tell how the introduced elements are propagated to the solution. Since equivalence of equations generally do not account for the slots, these indices are generally not the same for two equations considered equivalent. In other words, these indices are a property of the equations per se, but are still defined abstractly without reference to how they are computed.

- Analysis (for instance, revealing the underlying ordinary differential equation on a manifold) and solution of DAE has given rise to many methods, and one can typically identify some natural number for each method as a measure of how involved the equations are. This defines indices based on methods. Basically these are a property of the equations, but can generally not be defined abstractly without reference to how to compute them.

The above categorization is not a clear cut in every case. For instance, an index which was originally formulated in terms of a method may later be given an equivalent but more abstract definition.

Sometimes, when modeling follows certain patterns, the resulting equations may be of known index (of course, one has to specify which index is referred to). It may then be possible to design special-purpose algorithms for automatic control tasks such as simulation, system identification or state estimation.

In this thesis, we regard the solution of initial value problems as a key to understanding other aspects of DAE in automatic control. We are not so much interested in the mathematical questions of exactly when solutions exist or how the solutions may be described abstractly, but turn directly to numerical implementation. For equations of unknown, higher index, all existing approaches to numerical solution of initial value problems that we know of perform index reduction so that one obtains equations of low index (typically 0 or 1), which can then be fed to one of the many available solvers for such equations. The index reduction algorithm used in this thesis (described in chapter[3]) deals with the differential index, which we will define in terms of this algorithm. We will then show an equivalent but more abstract definition. See [Campbell and Gear] [1995] for a survey (although incomplete today) of various index definitions and for examples of how different indices may be related.

The index reduction scheme used in this thesis is a so-called elimination-differentiation
approach. These have been in use for a long time, and as is often the case in the area of dynamic systems, the essence of the idea is best introduced by looking at linear time-invariant (LTI) systems, while the extension to nonlinearities brings many subtleties to the surface. The linear case was considered in [Luenberger 1978], and the algorithm is commonly known as the shuffle algorithm.

For convenient notation in algorithm 2.1 (on page 22), introduce the notation

\[ u^{(i)} = \begin{pmatrix} u \\ u' \\ \vdots \\ u^{(i)} \end{pmatrix} \]

In the algorithm, there is a clear candidate for an index: the final value of \( i \). We make this our definition of the differential index.

**Definition 2.1 (Differential index).** The differential index of a square LTI DAE is given by the final value of \( i \) in algorithm 2.1.

While the compact representation of LTI systems makes the translation of theory to computer programs rather straightforward, the implementation of nonlinear theory is not all as straightforward. This seems, at least to some part, to be explained by the fact that there are no widespread computer tools for working with the mathematical concepts from differential algebra. A theoretical counterpart of the shuffle algorithm, but applying to general nonlinear DAE, was used in [Rouchon et al. 1995]. However, its implementation is nontrivial since it requires a computable representation of the function whose existence is granted by the implicit function theorem. For quasilinear DAE, on the other hand, an implicit function can be computed explicitly, and our current interest in these methods owes to this fact. For references to implementation-oriented index reduction of quasilinear DAE along these lines, see for example [Visconti 1999] or [Steinbrecher 2006]. Instead of extending the above definition of the differential index of square LTI DAE to the quasilinear form, we shall make a more general definition, which we will prove is a generalization of the former.

The following definition of the differential index of a general nonlinear DAE can be found in [Campbell and Gear 1995]. It should be mentioned, though, that the authors of [Campbell and Gear 1995] are not in favor of using this index to characterize a model, and define replacements. On the other hand, in the context of particular algorithms, the differential index may nevertheless be a relevant characterization.

Consider the general nonlinear DAE

\[ f(\dot{x}(t), x(t), t) = 0 \] (2.31)

By using the notation

\[ \dot{x}^{(i)}(t) = \left( x(t), \dot{x}(t), \ldots, \dot{x}^{(i)}(t) \right) \] (2.32)
Algorithm 2.1 The shuffle algorithm.

**Input:** A square LTI DAE,

\[ E x'(t) + A x(t) + B u(t) = 0 \]

**Output:** An equivalent non-square DAE consisting of a square LTI DAE with non-singular leading matrix (and redefined driving function) and a set \( C = \bigcup_i C_i \) of linear equality constraints involving \( x \) and \( u'^{(i)} \) for some \( i \).

**Algorithm:**

\[ E_0 := E \]
\[ A_0 := A \]
\[ B_0 := B \]
\[ i := 0 \]

**while** \( E_i \) is singular

Manipulate the equations by row operations so that \( E_i \) becomes partitioned as \( \begin{pmatrix} \bar{E}_i \\ \tilde{E}_i \end{pmatrix} \), where \( \bar{E}_i \) has full rank and \( \tilde{E}_i = 0 \). This can be done by, for instance, Gauss elimination or QR factorization.

Perform the same row operations on the other matrices, and partition the result similarly.

\[ C_i := \begin{pmatrix} \bar{A}_i x + \tilde{B}_i u'^{(i)} = 0 \end{pmatrix} \]

\[ E_{i+1} := \begin{pmatrix} \bar{E}_i \\ \bar{A}_i \end{pmatrix} \]
\[ A_{i+1} := \begin{pmatrix} \bar{A}_i \\ 0 \end{pmatrix} \]
\[ B_{i+1} := \begin{pmatrix} -\tilde{B} & 0 \\ 0 & \tilde{B} \end{pmatrix} \]
\[ i := i + 1 \]

**if** \( i > \text{dim } x \)

**abort** with “ill-posed”

**end**

**end**

**Remark:** The new matrices computed in each iteration simply correspond to differentiating the equations from which the differentiated variables have been removed by the row operations. (This should clarify the notation used in the construction the \( B_i \).) Since the row operations generate equivalent equations, and the equations that get differentiated are also kept unaltered in \( C \), it is seen that the output equations are equivalent to the input equations.

See the notes in algorithm 2.2 regarding geometric differentiation, and note that assumptions about constant Jacobians are trivially satisfied in the LTI case.
the general form can be written \( f_0(\dot{x}^{(1)}(t), t) \overset{!}{=} 0 \). Note that differentiation with respect to \( t \) yields an equation which can be written \( f_1(\dot{x}^{(2)}(t), t) \overset{!}{=} 0 \). Introducing the derivative array

\[
F_i(x'^{(i+1)}(t), t) = \begin{pmatrix}
  f_0(x'^{(1)}(t), t) \\
  \vdots \\
  f_i(x'^{(i+1)}(t), t)
\end{pmatrix}
\] (2.33)

the implied equation

\[
F_i(\dot{x}^{(i+1)}(t), t) \overset{!}{=} 0
\] (2.34)

is called the derivative array equations accordingly.

**Definition 2.2 (Differential index).** Suppose (2.31) is solvable. If \( \dot{x}(t) \) is uniquely determined given \( x(t) \) and \( t \) in the non-differential equation (2.34), for all \( x(t) \) and \( t \) such that a solution exist, and \( \nu_D \) is the smallest \( i \) for which this is possible, then \( \nu_D \) is denoted the differential index of (2.31).

Next, we show that the two definitions of the differential index are compatible.

**Theorem 2.1**

**Definition 2.2** generalizes definition 2.7

**Proof:** Consider the derivative array equations \( F_i(\dot{x}^{(i+1)}(t), t) \overset{!}{=} 0 \) for the square LTI DAE of definition 2.1

\[
\begin{pmatrix}
  A_0 & E_0 \\
  A_0 & E_0 \\
  \vdots & \vdots \\
  A_0 & E_0
\end{pmatrix}
\begin{pmatrix}
  x \\
  \dot{x} \\
  \vdots \\
  \dot{x}^{(i+1)}
\end{pmatrix}
+ \begin{pmatrix}
  B u \\
  B u' \\
  \vdots \\
  B u^{(i)}
\end{pmatrix}
\overset{!}{=} 0
\] (2.35)

Suppose definition 2.1 defines the index as \( i \). Then \( E_i \) in algorithm 2.1 is non-singular by definition. Performing the first row elimination of the shuffle algorithm on (2.35) yields

\[
\begin{pmatrix}
  \bar{A}_0 & \bar{E}_0 \\
  \bar{A}_0 & \bar{E}_0 \\
  \vdots & \vdots \\
  \bar{A}_0 & \bar{E}_0
\end{pmatrix}
\begin{pmatrix}
  x \\
  \dot{x} \\
  \vdots \\
  \dot{x}^{(i+1)}
\end{pmatrix}
+ \begin{pmatrix}
  \bar{B} u \\
  \bar{B} u' \\
  \vdots \\
  \bar{B} u^{(i)}
\end{pmatrix}
\overset{!}{=} 0
\]
Reordering the rows as

\[
\begin{pmatrix}
\bar{A}_0 & E_0 \\
A_0 & E_0 \\
\vdots & \vdots \\
\bar{A}_0 & E_0 \\
\end{pmatrix}
\begin{pmatrix}
x \\
\dot{x} \\
\vdots \\
\dot{x}(i+1) \\
\end{pmatrix}
+ 
\begin{pmatrix}
\bar{B} u \\
\bar{B} u' \\
\vdots \\
\bar{B} u'(i-1) \\
\end{pmatrix} = 0
\]  

and ignoring the last two rows, this can be written

\[
\begin{pmatrix}
A_1 & E_1 & \ldots & E_1 \\
A_1 & E_1 & \ldots & E_1 \\
\vdots & \vdots & \ddots & \vdots \\
A_1 & E_1 & \ldots & E_1 \\
\end{pmatrix}
\begin{pmatrix}
x \\
\dot{x} \\
\vdots \\
\dot{x}(i) \\
\end{pmatrix}
+ \ldots = 0
\]

using the notation in algorithm 2.1. The driving function \(u\) has been suppressed for brevity. After repeating this procedure \(i\) times, one obtains

\[
(A_i \ E_i) \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + \ldots = 0
\]

which shows that definition 2.1 gives an upper bound on the index defined by definition 2.2.

Conversely, it suffices to show that the last two rows of (2.36) does not contribute to the determination of \(\dot{x}\). The last row only restricts the feasible values for \(x\), which is considered a given in the equation. The second last row contains no information than can be propagated to \(\dot{x}\) since it can be solved for any \(\dot{x}(i)\) by a suitable choice of \(\dot{x}(i+1)\) (which appears in no other equation). Since this shows that no information about \(\dot{x}\) was discarded, we have also found that if the index as defined by definition 2.1 is greater than \(i\), then \(E_i\) is singular, and hence the index as defined by definition 2.2 must also be greater than \(i\). That is, definition 2.1 gives a lower bound on the index defined by definition 2.2.

Many other variants of differential index definitions can be found in [Campbell and Gear, 1995], which also provides the relevant references. However, they avoid discussion of geometric definition of differential indices. While not being important for LTI DAE, where the representation by numeric matrices successfully captures the geometry of the equations, geometric definitions turn out to be important for nonlinear DAE. This is emphasized in [Thomas, 1996], as it summarizes results by other authors. [Rabier and Rheinboldt, 1994; Reich, 1991; Szatkowski, 1990, 1992] It is noted that the geometrically defined differential index is bounded by the dimension of the equations, and cannot be computed reliably using numerical methods; the indices which can be computed numerically are not geometric and may not be bounded even for well-posed equations. The presentation in [Thomas, 1996] is further developed in [Reid et al., 2001] to apply also to partial differential-algebraic equations.
Having discussed the differentiation index with its strong connection to algorithms, we now turn to an index concept of another kind, namely the perturbation index. The following definition is taken from [Campbell and Gear 1995], which refers to [Hairer et al. 1989].

**Definition 2.3.** The DAE \( f(\dot{x}(t), x(t), t) = 0 \) has perturbation index \( \nu_P \) along a solution \( x \) on the interval \( I = [0, T] \) if \( \nu_P \) is the smallest integer such that if

\[
f(\dot{x}(t), x(t), t) = \delta(t)
\]

for sufficiently smooth \( \delta \), then there is an estimate\(^4\)

\[
\|\hat{x}(t) - x(t)\| \leq C \left( \|\hat{x}(0) - x(0)\| + \|\delta\|_{\nu_P-1}^t \right)
\]

Clearly, one can define a whole range of perturbation indices by considering various “slots” in the equations, and each form of the equations may have its own natural slots.

There are two aspects of these indices we would like to emphasize. First, they are defined completely without reference to a method for computing them, and in this sense they seem closer to capturing intrinsic features of the system described by the equations, than indices that are defined by how they are computed. Second, and on the other hand, the following example shows that these indices may be strongly related to which set of equations are used to describe a system.

---

**Example 2.2**

Consider computing the perturbation index of the DAE

\[
f(\dot{x}(t), x(t), t) = 0
\]

We must then examine how the solution depend on the driving perturbation function \( \delta \) in

\[
f(\dot{x}(t), x(t), t) = \delta(t)
\]

Now, let the matrix \( K( x(t), t ) \) define a smooth, non-singular transform of the equations, leading to

\[
K( x(t), t ) f(\dot{x}(t), x(t), t) = 0
\]

with perturbation index defined by examination of

\[
K( x(t), t ) f(\dot{x}(t), x(t), t) = \delta(t)
\]

\(^4\)Here, the norm with ornaments is defined by

\[
\|\delta\|_m^t = \sum_{i=0}^{m} \sup_{\tau \in [0, t]} \|\dot{\delta}^{(i)}(\tau)\|, \quad m \geq 0
\]

\[
\|\delta\|_{t-1}^t = \int_0^t \|\dot{\delta}^{(i)}(\tau)\| \, d\tau
\]
Trying to relate this to the original perturbation index, we could try rewriting the equations as

\[ f(x'(t), x(t), t) = K(x(t), t)^{-1} \delta(t) \]

but this introduces \( x(t) \) on the right hand side, and is no good. Further, since the perturbation index does not give bounds on the derivative of the estimate error, there are no readily available bounds on the derivatives of the factor \( K(x(t), t)^{-1} \) that depend only on \( t \).

In the special case when the perturbation index is 0, however, a bound on \( K \) allows us to translate a bound in terms of \( \|K(x(t), t)^{-1} \delta(t)\|^t_{-1} \) to a bound in terms of \( \|\delta(t)\|^t_{-1} \). This shows that, at least, this way of rewriting the equations does not change the perturbation index.

Of course, it is interesting to relate the differential index to the perturbation index, but we have already seen an example of how different index definitions can be related, and shall not dwell more on this. Instead, there is one more index we would like to mention since it is instrumental to a well developed theory. This is the strangeness index, developed for time-varying linear DAE in [Kunkel and Mehrmann 1994], see also [Kunkel and Mehrmann 2006]. Perhaps due to its ability of revealing a more intelligent characterization of a system compared to, for instance, the differentiation index, it is somewhat expensive to compute. This becomes particularly evident in the associated method for solving initial value problems, where the index computations are performed at each step of the solution. This is addressed in the relatively recent [Kunkel and Mehrmann 2004], see also [Kunkel and Mehrmann 2006] remark 6.7 and remark 6.9.

A quite different method which reduces the index is the Pantelides’ algorithm [Pantelides 1988] and the dummy derivatives [Mattsson and Söderlind 1993] extension thereof. This technique is in extensive use in component-based modeling and simulation software for the Modelica language, such as Dymola [Mattsson et al. 2000] and OpenModelica [Fritzson et al. 2006a,b]. A major difference between the previously discussed index reduction algorithms and Pantelides’ algorithm is that the former use mathematical analysis to derive the new form, while the latter uses only the structure of the equations (the equation–variable graph). Since the equation–variable graph does not require the equations to be in any particular form, the technique is applicable to general nonlinear DAE. While the graph-based technique is expected to be mislead by a change of variables and other manipulations of the equations (see section 5.1.1), it is well suited for the equations as they arise in the software systems mentioned above.

Hereafter, when speaking of just the index (without qualification), we refer to the differential index, often thinking of it as the number of steps required to shuffle the equations to an implicit ODE.
2.2.4 Transformation to quasilinear form

In this section, the transformation of a general nonlinear DAE to quasilinear form is considered. This may seem like a topic for section 2.2.2, but since we need to refer to the index concept, waiting until after section 2.2.3 is motivated.

For ease of notation, we shall only deal with equations without explicit dependence on the time variable in this section. This way, it makes sense to write a time-invariant nonlinear DAE as

\[ f(x, x', x'', \ldots) = 0 \]  \hspace{1cm} (2.37)

The variable in this equation is the function \( x \), and the zero on the right-hand side must be interpreted as the mapping from all of the time domain to the real constant 0. We choose to interpret the equality relation of the equation pointwise, although other measure-zero interpretations could be made (we are not seeking new semantics, only a shorter notation compared to (2.25)). Of course, including higher order derivatives in the form (2.37) is just a minor convenience compared to using only first order derivatives in (2.25), but this is a topic for the discussion below.

The time-invariant quasilinear form looks like

\[ E(x) x' + A(x) = 0 \]  \hspace{1cm} (2.38)

Assuming that (2.25) has index \( \nu_D \) but is not in the form (2.38), can we say something about the index of the corresponding (2.38)?

Not being in the form (2.38) can be for two reasons:

- There are higher-order derivatives.
- The residuals are not linear in the derivatives.

To remedy the first, one simply introduces new variables for all but the highest-and-higher-than-1-order derivatives. Of course, one also adds the equations relating the introduced variables to the derivatives they represent; each new variable gets one associated equation. This procedure does not raise the index, since the derivatives which have to be solved for really has not changed. If the highest order derivatives could be solved for in terms of lower-order derivatives after \( \nu_D \) differentiations of (2.25), they will be possible to solve for in terms of the augmented set of variables after \( \nu_D \) differentiations of (2.38) (of course, there is no need to differentiate the introduced trivial equations). The introduced variables’ derivatives that must also be solved for are trivial (that is why the definitions of index does not have to mention solution of the lower-order derivatives).

Turning to the less trivial reason, nonlinearity in derivatives, the fix is still easy; introduce new variables for the derivatives that appear nonlinearly and add the linear (trivial) equations that relate the new variables to derivatives of the old variables; change

\[ f(x, x') = 0 \]  \hspace{1cm} (2.39)
Background to
\[
\begin{align*}
\dot{x}' &= \dot{x} \\
f(x, \dot{x})' &= 0
\end{align*}
\]

Note the important difference to the previous case: this time we introduce new variables for some highest-order derivatives. This might have implications for the index. If the index was previously defined as the number of differentiations required to be able to solve for \( x' \), we must now be able to solve for \( \ddot{x}' = x'' \). Clearly, this can be obtained by one more differentiation once \( x' \) has been solved for, as in the following example.

--- Example 2.3 ---

Consider the index-0 DAE
\[
\begin{align*}
e^{x_2}' &= e^{x_1} \\
x_1' &= -x_2
\end{align*}
\]

Taking this into the form (2.38) brings us to
\[
\begin{align*}
x_2' &= \dot{x} \\
e^{\dot{x}} &= e^{x_1} \\
x_1' &= -x_2
\end{align*}
\]

where \( \dot{x}' \) cannot be solved for immediately since it does not even appear. However, after differentiating the purely algebraic equation once, all derivatives can be solved for;
\[
\begin{align*}
x_2' &= \dot{x} \\
e^{\dot{x}} \dot{x}' &= e^{x_1} x_1' \\
x_1' &= -x_2
\end{align*}
\]

However, the index is not raised in general; it is only in case the nonlinearly appearing derivatives could not be solved for in less than \( \nu_D \) steps that the index will raise. The following example shows a typical case where the index is not raised.

--- Example 2.4 ---

By modifying the previous example we get a system that is originally index-1,
\[
\begin{align*}
e^{x_2}' &= e^{x_1} \\
x_1' &= -x_2 \\
x_3' &= 1
\end{align*}
\]
Taking this into the form (2.38) brings us to

\[
\begin{align*}
  x_2' & = \dot{x} \\
  e^{\dot{x}} & = e^{x_1} \\
  x_1' & = -x_2 \\
  x_3' & = 1
\end{align*}
\]

which is still index-1 since all derivatives can be solved for after one differentiation of the algebraic equations:

\[
\begin{align*}
  x_2' & = \dot{x} \\
  e^{\dot{x}} x' & = e^{x_1} x_1' \\
  x_1' & = -x_2 \\
  x_3' & = 0
\end{align*}
\]

Although the transformation discussed here may raise the index, it may still be a useful tool in case the equations and driving functions are sufficiently differentiable. The transformation is implemented in the software described in appendix B as part of finding the quasilinear structure in equations represented in general form.

### 2.2.5 Structure algorithm

The application of the structure algorithm to DAE described in this section is due to Rou-
chon et al. [1995], which relies on results in Li and Feng [1987].

The structure algorithm was developed for the purpose of computing inverse systems; that is, to find the input signal that produces a desired output. It assumes that the system’s state evolution is given as an ODE and that the output is a function of the state and current input. Since the desired output is a known function, it can be included in the output function; that is, it can be assumed without loss of generality that the desired output is zero. The algorithm thus provides a means to determine \( u \) in the setup

\[
\begin{align*}
  x' & = h( x, u, t ) \\
  0 & = f( x, u, t )
\end{align*}
\]

The algorithm produces a new function \( \eta \) such that \( u \) can be determined from \( 0 \overset{!}{=} \eta( x, u, t ) \). By taking \( h( x, u, t ) = u \), this reduces to a means for determining the derivatives of the variables \( x \) in the DAE

\[
0 \overset{!}{=} f( x, x', t )
\]
In algorithm 2.2 we give the algorithm applied to the DAE setup. It is assumed that \( \dim f = \dim x \), that is, that the system is square.

### 2.2.6 Initial conditions

The reader might have noticed that the shuffle algorithm (on page 22) not only produces an index and an implicit ODE, but also a set of constraints. These constraints the solution at any point in time, and the implicit ODE is only to be used where the constraints are satisfied. The constraints are often referred to as the algebraic constraints which emphasizes that they are non-differential equations. They can be explicit as in the case of non-differential equations in the DAE as it is posed, or implicit as in the case of the output from the shuffle algorithm. Of course, the constraint equations are not unique, and it may well happen that some of the equations output from the shuffle algorithm were explicit in the original DAE.

Making sure that numerical solutions to DAE do not leave the manifold defined by the algebraic constraints is a problem in itself, and several methods to ensure this exist. However, in theory, no special methods are required, since the produced implicit ODE is such that an exact solution starting on the manifold will remain on the manifold. This brings up another practical issue, namely that initial value problems are ill-posed if the initial conditions they specify are inconsistent with the algebraic constraints.

Knowing that a DAE can contain implicit algebraic constraints, how can we know that all implicit constraints have been revealed at the end of the index reduction procedure? If the original DAE is square, any algebraic constraints will be present in differentiated form in the index 0 square DAE. This implies that the solution trajectory will be tangent to the manifold defined by the algebraic constraints, and hence it is sufficient that the initial conditions for an initial value problem are consistent with the algebraic constraints for the whole trajectory to remain consistent. In other words, there exist solutions to the DAE starting at any point which is consistent with the algebraic constraints, and this shows that there can be no other implicit constraints.

We shall take a closer look at this problem in section 3.4. Until then, we just note that rather than rejecting initial value problems as ill-posed if the initial conditions they specify are inconsistent with algebraic constraints, one usually interprets the initial conditions as a guess, and then applies some scheme to find truly consistent initial conditions that are close to the guess in some sense. The importance of this task is suggested by the fact that the influential Pantelides [1988] addressed exactly this, and it is no surprise [Chow, 1998] since knowing where a DAE can be initialized entails having a characterization of the manifold to which all of the solution must belong. Another structural approach to system analysis is presented in Unger et al. [1995]. Their approach is similar to that we propose in chapter 3. However, just as Pantelides’ algorithm, it considers only the equation-variable graph, although it is not presented as a graph theoretical approach. A later algorithm which is presented as graph theoretical, is given in Leitold and Hangos [2001], although a comparison to Pantelides’ algorithm seems missing.

In Leimkuhler et al. [1991], consistent initial conditions are computed using difference
Algorithm 2.2 The structure algorithm.

**Input:** A square DAE,

\[ f( x(t), x'(t), t ) \overset{!}{=} 0 \]

**Output:** An equivalent non-square DAE consisting of a square DAE from which \( x' \) can be solved for, and a set of constraints \( C = \bigwedge_i \left( \Phi_i(x(t), t, 0) \overset{!}{=} 0 \right) \). Let \( \alpha \) be the smallest integer such that \( \nabla_x f_\alpha( x, \dot{x}, t ) \) has full rank, or \( \infty \) if such a number fails to be found.

**Invariant:** The sequence of \( f_k \) shall be such that the solution is always determined by \( f_k(x, \dot{x}, t) = 0 \), which is fulfilled for \( f_0 \) by definition. Reversely, this will make \( f_k(x, \dot{x}, t) = 0 \) along solutions.

**Algorithm:**

\[ f_0 = f \]
\[ i := 0 \]
\[ \text{while } \nabla_x f_i(x, \dot{x}, t) \text{ is singular} \]

Since the rank of \( \nabla_x f_i(x, \dot{x}, t) \) is not full, it makes sense to split \( f_i \) into two parts; \( \tilde{f}_i \) being a selection of components of \( f_i \) such that \( \nabla_x f_i(x, \dot{x}, t) \) has full and maximal rank (that is, the same rank as \( \nabla_x f_i(x, \dot{x}, t) \)), and \( \tilde{f}_i \) being the remaining components.

Locally (and as all results of this kind are local anyway, this will not be further emphasized), this has the interpretation that the dependency of \( \tilde{f}_i \) on \( \dot{x} \) can be expressed in terms of \( \tilde{f}_i(x, \dot{x}, t) \) instead of \( \dot{x} \); there exists a function \( \Phi_i \) such that \( \tilde{f}_i(x, \dot{x}, t) = \Phi_i(x, t, \tilde{f}_i(x, \dot{x}, t)) \).

Since \( \tilde{f}_i(x, \dot{x}, t) = 0 \) along solutions, we replace the equations given by \( \tilde{f}_i \) by the residuals obtained by differentiating \( \Phi_i(x(t), t, 0) \) with respect to \( t \) and substituting \( \dot{x} \) for \( x' \);

\[ f_{i+1} = \left( x, \dot{x}, t \right) \mapsto \nabla_1 \Phi_i(x, t, 0) \dot{x} + \nabla_2 \Phi_i(x, t, 0) \right) \]
\[ i := i + 1 \]
\[ \text{if } i > \text{dim } x \text{ abort with “ill-posed”} \]

**Remark:** Assuming that all ranks of Jacobian matrices are constant, it is safe to abort after \( \text{dim } x \) iterations. [Rouchon et al., 1995] Basically, this condition means that the equations are not used at a single point, but rather as geometrical (algebraic) objects. Hence, in the phrasing of [Thomas, 1996], differentiations are geometric, and \( \alpha \) becomes analogous to the geometric differential index.

In [Rouchon et al., 1995], additional assumptions on the selection of components to constitute \( f_i \) are made, but we will not use those here.
approximation of derivatives, assuming that the DAE is quasilinear and of index 1. Later, Veiera and Biscaia Jr. [2000] gives an overview of methods to compute consistent initial conditions. It is noted that several successful approaches has been developed for specific applications where the equations are in a well understood form, and among other approaches (including one of their own) they mention that the method in Leimkuhler et al. [1991] has been extended by combining it with Pantelides’ algorithm to analyze the system structure rather than assuming the quasilinear index 1 form. Their own method, presented in some more detail in Veiera and Biscaia Jr. [2001], is used to find initial conditions for systems starting in steady state, but allows for a discontinuity in driving functions at the initial time. Of all previously presented methods for analysis of DAE, the one which most resembles that proposed in chapter 3 is found in Chowdhry et al. [2004]. They propose a method similar to that in Unger et al. [1995], but take it one step further by making a distinction between linear and nonlinear dependencies in the DAE. This allows LTI DAE to be treated exactly, which is an improvement over Unger et al. [1995], while performing at least as good in the presence of nonlinearities. In view of our method, the partitioning into structural zeros, constant coefficients, and nonlinearities seems somewhat arbitrary. Still, it is suggested that even more categories could be added to extend the class of systems for which the method is exact. The need for a rigorous analysis of how tolerances affect the algorithm is not mentioned.

2.2.7 Numerical integration

There are several techniques in use for the solution of DAE. In this section, we mention some of them briefly, and explain one in a bit more detail. A classic accessible introduction to this subject is Brenan et al. [1996], which contains many references to original papers and further theory.

The method we focus on in this section is applicable to equations with differential index 1, and this is the one we describe first. It belongs to a family referred to as backward difference formulas or BDF methods. The formula of the method tells how to treat $x'(t)$ in

$$ f(x'(t), x(t), t) = 0 $$

when the problem is discretized. By discretizing a problem we refer to replacing the infinite-dimensional problem of computing the value of $x$ at each point of an interval, to a finite-dimensional problem from which the solution to the original problem can be approximately reconstructed. The most common way of discretizing problems is to replace the continuous function $x$ by a time series which approximates $x$ at discrete points in time:

$$ x_i \approx x(t_i) $$

Reconstruction can then be performed by interpolation. A common approach to the interpolation is to do linear interpolation between the samples, but this will give a function which is not even differentiable at the sample points. To remedy this, interpolating splines can be used. This suggests another way to discretize problems, namely to represent the discretized solution directly in spline coefficients, which makes both reconstruction as
2.2 Differential-algebraic equations

well as treatment of $x'$ trivial. However, solving for such a discretization is a much more intricate problem than to solve for a pointwise approximation.

Before presenting the BDF methods, let us just mention how the simple (forward) Euler step for ODE fits into this framework. The problem is discretized by pointwise approximation, and the ODE $x'(t) \triangleq g(x(t), t)$ is written as a DAE by defining $f(x', x, t) \triangleq -\dot{x} + g(x, t)$. Replacing $x'(t)$ by the approximation $(x_{n+1} - x_n)/(t_{n+1} - t_n)$ then yields the familiar integration method:

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

$$0 \triangleq -\frac{x_{n+1} - x_n}{t_{n+1} - t_n} + g(x_n, t_n)$$

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

The $k$-step BDF method also discretizes the problem by pointwise approximation, but replaces $x'(t_n)$ by the derivative at $t_n$ of the polynomial which interpolates the points $(t_n, x_n), (t_{n-1}, x_{n-1}), \ldots, (t_{n-k}, x_{n-k})$. [Brenan et al., 1996, section 3.1] We shall take a closer look at the 1-step BDF method, which given the solution up to $(t_{n-1}, x_{n-1})$ and a time $t_n > t_{n-1}$ solves the equation

$$f\left(\frac{x_n - x_{n-1}}{t_n - t_{n-1}}, x_n, t_n\right) \triangleq 0$$

to obtain $x_n$. Of course, selecting how far from $t_{n-1}$ we may select $t_n$ without getting too large errors in the solution is a very important question, but it is outside the scope of this background to cover this. A related topic of great importance is to ensure that the discretized solution converges to the true solution as the step size tends to zero, and when it does, to investigate the order of this convergence. Such analyses reveal how the choice of $k$ affects the quality of the solution, and will generally also give results that depend on the index of the equations. The following example is not giving any theoretical insights, but just shows the importance of the index when solving a DAE by the 1-step BDF method.

---

**Example 2.5**

Consider applying the 1-step BDF method to the square index 1 LTI DAE

$$E \dot{x}(t) + Ax(t) + Bu(t) \triangleq 0$$

Discretization leads to

$$E \frac{x_n - x_{n-1}}{h_n} + Ax_n + Bu(t_n) \triangleq 0$$

Where $h_n = t_n - t_{n-1}$. By writing this as

$$(E + h_n A) x_n \triangleq E x_{n-1} - h_n B u(t_n)$$

we see that the 

**iteration matrix**

$$E + h_n A$$

(2.40)
must be non-singular for the solution to be well defined. Recalling that the differential index is revealed by the shuffle algorithm, we know that there exists a non-singular matrix $K$ such that

$$
\begin{pmatrix}
\mathcal{I} \\
\frac{1}{h_n} \mathcal{I}
\end{pmatrix}
K (E + h_n A) = \begin{pmatrix}
\mathcal{I} \\
\frac{1}{h_n} \mathcal{I}
\end{pmatrix}
\begin{pmatrix}
\bar{E} + h_n \begin{pmatrix} A & 0 \\
0 & A
\end{pmatrix}
\end{pmatrix} = \begin{pmatrix}
\bar{E} \\
A
\end{pmatrix} + h_n \begin{pmatrix} A & 0 \\
0 & A
\end{pmatrix}
$$

where the first term is non-singular. This proves the non-singularity of the iteration matrix (2.40) in general, since it is non-singular for $h_n = 0$, and will hence only be singular for finitely many values of $h_n$. Had the index been higher than 1, interpretation of the index via the shuffle algorithm reveals that the iteration matrix is singular for $h_n = 0$, and hence ill-conditioned for small $h_n$. (It can be shown that it is precisely the DAE where the iteration matrix is singular for all $h_n$, that are not solvable at all. [Brenan et al., 1996, theorem 2.3.1]) This shows that this method is limited to systems of index no more than 1.

Note that the row operations that revealed the non-singularity also have practical use, since if applied before solving the DAE, the condition number of the iteration matrix is typically improved significantly, and this condition is directly related to how errors in the estimate $x_{n-1}$ are propagated to errors in $x_n$.

The following example shows how to combine the shuffle algorithm with the 1-step BDF method to solve LTI DAE of arbitrary index.

**Example 2.6**

Consider solving an initial value problem for the square higher-index (solvable) LTI DAE

$$
E x'(t) + A x(t) + B u(t) = 0
$$

After some iterations of the shuffle algorithm (it can be shown that the index is bounded by the dimension of $x$ for well-posed problems, see the remark in algorithm 2.1), we will obtain the square DAE

$$
\begin{pmatrix}
\bar{E}_{\nu D} & -1 \\
0 & -1
\end{pmatrix}
x'(t) + \begin{pmatrix}
\bar{A}_{\nu D} & -1 \\
\bar{A}_{\nu D} & -1
\end{pmatrix}
x(t) + \ldots = 0
$$

where the dependence on $u$ and its derivatives has been omitted for brevity. At this stage, the full set of algebraic constraints has been revealed, which we write

$$
C_{\nu D} x(t) + \ldots = 0
$$

It is known that

$$
\begin{pmatrix}
\bar{E}_{\nu D} & -1 \\
\bar{A}_{\nu D} & -1
\end{pmatrix}
$$

is full-rank, where the lower block is contained in $C_{\nu D}$. This shows that it is possible to construct a square DAE of index 1 which contains all the algebraic constraints, by selecting
as many independent equations as possible from the algebraic constraints, and completing with differential equations from the upper block of the index 0 system.

Note that the resulting index 1 system has a special structure; there is a clear separation into differential and non-differential equations. This is valuable when the equations are integrated, since it allows row scaling of the equations so as to improve the condition of the iteration matrix — compare the previous example.

In the previous example, a higher index DAE was transformed to a square index 1 DAE which contained all the algebraic constraints. Why not just compute the implicit ODE and apply an ODE solver, or apply a BDF method to the index 1 equations just before the last iteration of the shuffle algorithm? The reason is that there is no magic in the ODE solvers or the BDF method; they cannot guarantee that algebraic constraints which are not present in the equations they see remain satisfied even though the initial conditions are consistent. Still, the algebraic constraints are not violated arbitrarily; for consistent initial conditions, the true solution will remain on the manifold defined by the algebraic constraints, and it is only due to numerical errors that the computed solution will drift away from this manifold. By including the algebraic constraints in the index 1 system, it is ensured that they will be satisfied at each sample of the computed solution.

There is another approach to integration of DAE which seems to be gradually replacing the BDF methods in many implementations. These are the implicit Runge-Kutta methods, and early work on their application to DAE include Petzold [1986] and Roche [1989]. Although these methods are basically applicable to DAE of higher index, poor convergence is prohibitive unless the index is low. (Compare the 1-step BDF method which is not at all applicable unless the index is at most 1.) The class of IRK methods is large, and this is where the popular Radau IIA belongs.

Having seen that higher index DAE require some kind of index-reducing treatment, we finish this section by reminding that index reduction and index deduction are closely related, and that both the shuffle algorithm (revealing the differentiation index) and the algorithm that is used to compute the strangeness index may be used to produce equations of low index. In the latter context, one speaks of producing strangeness-free equations.

2.2.8 Existing software

To round off our introductory background of DAE topics, some existing software for the numerical integration of DAE will be mentioned. However, as numerical integration is merely one of the applications of the work in this thesis, the methods will only be mentioned very briefly just to given an idea of what sort of tools there are.

The first report on DASSL [Brenan et al., 1996] was written by Linda Ruth in September 1982. It is probably the best known DAE solver, but has been superseded by an extension called DASPK [Brown et al., 1994]. Both DASSL and DASPK uses a BDF method with dynamic selection of order (1-step through 5-step) and step size, but the latter is
better at handling large and sparse systems, and is also better at finding consistent initial conditions.

The methods in DASPK can also be found in the more recent IDA (dating 2005) \cite{Hindmarsh:2004}, which is part of the software package SUNDIALS \cite{Hindmarsh:2005}. The name of this software package is an abbreviation of \textit{SUite of Nonlinear and Differential/Algebraic equation Solvers}, and the emphasis is on the movement from Fortran source code to C. The IDA solver is the DAE solver used by the general-purpose scientific computing tool \textit{Mathematica}.\footnote{As of version 5. New integration methods are reported being part of the enhancements in version 6, released while this thesis is being written, but it is unclear whether these methods apply to DAE.}

While the BDF methods in the software mentioned so far requires that the user ensure that the index is sufficiently reduced, the implementations built around the strangeness index perform index reduction on the fly. Another interesting difference is that the solvers we find here implement also IRK methods beside BDF. In 1995, the first version of GELDA \cite{Kunkel:1995} (\textit{A GEneral Linear Differential Algebraic equation solver}) appeared. It applies to linear time-varying DAE, and there is an extension called GENDA \cite{Kunkel:2006} which applies to general nonlinear systems. The default choice for integration of the strangeness-free equations is the Radau IIa IRK method implemented in \textsc{Radau5} \cite{Hairer:1991}.

\section{Singular perturbation theory}

Recall the model reduction technique called \textit{residualization} (section\ref{sec:residualization}). In singular perturbation theory, a similar reduction can be seen as the limiting system as some dynamics become arbitrarily fast. \cite{Kokotovic:1986} However, some of the assumptions made in the singular perturbation framework are not always satisfied in the presence of non-structural zeros, and this is a major concern in this thesis. The connection to model reduction and singular perturbation theory is interesting also for another reason, namely that the classical motivation in those areas is that the underlying system being modeled is singularly perturbed in itself, and one is interested in studying how this can be handled in modeling and model-based techniques. Although that framework is built around ordinary differential equations, the situation is just as likely when DAE are used to model the same systems. It is a goal of this thesis to highlight the relation between treatment of small numbers in the leading matrix that are due to stiffness in the system being modeled, and the treatment of small numbers that are artifacts of numerical errors and the like. In view of this, this section not only provides background for forthcoming chapters, but also contains theory with which later development is to be contrasted.

Singular perturbation theory has already been mentioned when speaking of singular perturbation approximation in section\ref{sec:residualization}. However, singular perturbation theory is far more important for this thesis than just being an example of something with reminds of index reduction in DAE. First, it provides a theorem with is fundamental for the analysis in chapter\ref{chap:analysis}. Second, the way it is developed in \cite{Kokotovic:1986} contains the key ideas used in our development in chapter\ref{chap:development}. In this section, a main theorem is stated for
2.3 Singular perturbation theory

LTI systems, and we also indicate that there are generalizations that may be important for future developments of our work.

2.3.1 LTI systems

The following singular perturbation theorem found in Kokotović et al. [1986, chapter 2, theorem 5.1] will be useful. Consider the singularly perturbed LTI ordinary differential equation

\[
\begin{pmatrix}
x'(t) \\
e z'(t)
\end{pmatrix} =
\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}
\begin{pmatrix} x(t) \\ z(t) \end{pmatrix} +
\begin{pmatrix} x(0) \\ z(0) \end{pmatrix} =
\begin{pmatrix} x^0 \\ z^0 \end{pmatrix}
\] (2.41)

where we are interested in small \( \epsilon > 0 \). Define

\[ M_0 := M_{11} - M_{12} M_{22}^{-1} M_{21}, \]

\[ x_s'(t) = M_0 x_s(t) \quad x_s(t_0) = x^0 \] (2.42)

the slow model (obtained by setting \( \epsilon := 0 \) and eliminating \( z \) using the thereby obtained non-differential equations), and denote

\[ z_f'(\tau) = M_{22} z_f(\tau) \quad z_f(0) = z^0 + M_{22}^{-1} M_{21} x^0 \] (2.43)

the fast model (which is expressed in the timescale given by \( \epsilon \tau \sim t - t_0 \)).

**Theorem 2.2**

If \( \text{Re} \lambda( M_{22} ) < 0 \), there exists an \( \epsilon^* > 0 \) such that, for all \( \epsilon \in ( 0, \epsilon^* ] \), the states of the original system (2.41) starting from any bounded initial conditions \( x^0 \) and \( z^0 \), \( \| x^0 \| < c_1 \), \( \| z^0 \| < c_2 \), where \( c_1 \) and \( c_2 \) are constants independent of \( \epsilon \), are approximated for all finite \( t \geq t_0 \) by

\[
\begin{align*}
x(t) &= x_s(t) + O( \epsilon ) \\
z(t) &= -M_{22}^{-1} M_{21} x_s(t) + z_f(\tau) + O( \epsilon )
\end{align*}
\] (2.44)

where \( x_s(t) \) and \( z_f(\tau) \) are the respective states of the slow model (2.42) and the fast model (2.43). If also \( \text{Re} \lambda( M_0 ) < 0 \) then (2.44) holds for all \( t \in [ t_0, \infty ) \).

Moreover, the boundary layer correction \( z_f(\tau) \) is significant only during the initial short interval \( [ t_0, t_1 ] \), \( t_1 - t_0 = O( \epsilon \log \epsilon ) \), after which

\[ z(t) = -M_{22}^{-1} M_{21} x_s(t) + O( \epsilon ) \]

Among the applications of this theorem, numerical integration of the equations is probably the simplest example. The theorem says that for every acceptable tolerance \( \delta > 0 \) in the solution, there exists a threshold for \( \epsilon \) such that for smaller \( \epsilon \), the contribution to the global error from the timescale separation is at most, say, \( \delta/2 \). If the timescale separation is feasible, one can apply solvers for non-stiff problems in the fast and slow model separately, and then combine the results according to (2.44). This approach is likely to be much more efficient than applying a solver for stiff systems to the original problem.

However, note that the conclusion was only that there exists a threshold, as opposed to knowing this threshold. This means that it is not possible to use this result when guaranteeing an error tolerance is important. This leaves us with an ad hoc method for solving
the stiff problem: treat the problem as if \( \epsilon \) is sufficiently small for timescale separation, and just compute a solution without an error estimate.

The way one resorts to the ad hoc procedure here is in fact very similar to how one would apply the results to be presented in this thesis.

### 2.3.2 Generalizations

As an indication of how our results in this thesis may be extended in the future, we devote some space here to listing a few directions in which theorem 2.2 has been extended. All of these extensions are found in Kokotović et al. [1986].

The first extension is that the \( O(\epsilon) \) expressions in (2.44) can be refined so that the first order dependency on \( \epsilon \) is explicit. Neglecting the higher order terms in \( \epsilon \), this makes it possible to approximate the thresholds which are needed to keep track of the global error when integrating the equations in separate timescales. However, it is still not clear when \( \epsilon \) is sufficiently small for the \( O(\epsilon^2) \) terms to be neglected.

The other extension we would like to mention is that of theorem 2.2 to time-varying linear systems. That such results exist may not be surprising, but it should be noted that time-varying systems have an additional source of timescale separation compared to time-invariant systems. This must be taken care of in the analysis, and is a potential difficulty if these ideas are used to analyze a general nonlinear system by linearizing the equations along a solution trajectory (because of the interactions between timescale separation in the solution itself and in the linearized equations that determine it).

### 2.4 Gaussian elimination

Although assumed that the reader is familiar with Gaussian elimination, in this section some aspects of particular interest for the proposed algorithm will be discussed.

The proposed algorithm makes use of row reduction. The most well known row reduction method is perhaps Gaussian elimination, and although infamous for its numerical properties, it is sufficiently simple to be a realistic choice for implementations. In fact, the proposed algorithm makes this particular choice, and among the many variations of Gaussian elimination, a fraction-free scheme is used. This technique for taking a matrix to row echelon form uses only addition and multiplication operations. In contrast, a fraction-producing scheme involves also division. The difference is explained by example. Consider performing row reduction on a matrix of integers of the same order of

\[ \text{A matrix is said to be in row echelon form if each non-zero row has more leading zeros than the previous row. Actually, in order to account for the outcome when full pivoting is used, one should really say that the matrix is in row echelon form after suitable reordering of variables. In the current setting of elimination where it makes sense to speak of structural zeros, the reference to reordering of variables can be avoided by saying that the reduced matrix is such that each non-zero row has more structural zeros than the previous row.} \]
magnitude: 

\[
\begin{pmatrix} 5 & 7 \\ 3 & -4 \end{pmatrix}
\]

A fraction-free scheme will produce a new matrix of integers,

\[
\begin{pmatrix} 5 \\ 5 \cdot 3 - 3 \cdot 5 \\ 5 \cdot (-4) - 3 \cdot 7 \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix}
\]

while a fraction producing scheme generally will produce a matrix of rational numbers,

\[
\begin{pmatrix} 5 \\ 3 - (3/5) \cdot 5 \\ (-4) - (3/5) \cdot 7 \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix}
\]

The fraction-free scheme thus has the advantage that it is able to preserve the integer structure present in the original matrix. On the other hand, if the original matrix is a matrix of rational numbers, both schemes generally produce a new matrix of rational numbers, so there is no advantage in using the fraction-free scheme. Note that it is necessary not to allow the introduction of new integer elements in order to keep the distinction clear, since any matrix of rational numbers can otherwise be converted to a matrix of integers. Further, introducing non-integer scalars would destroy the integer structure. The two schemes should also be compared by the numbers they produce. The number $-41$ in comparison with the original numbers is a sign of the typical blowup of elements caused by the fraction-free scheme. The number $-(41/5) = -8.2$ does not indicate the same tendency.

When the matrix is interpreted as the coefficients of a linear system of equations to be solved in the floating point domain, the blowup of elements implies bad numeric condition, which in turn has negative implications for the quality of the computed solution. Unfortunately, this is not the only drawback of the fraction-free scheme, since the operations involved in the row reduction are ill-conditioned themselves. This means that there may be poor correspondence between the original equations and the row reduced equations, even before attempting to solve them.

Fraction-free Gaussian elimination can also be applied to a matrix of polynomials, and will then preserve the polynomial structure. Note also that the structure is not destroyed by allowing the introduction of new scalars. This can be used locally to drastically improve the numerical properties of the reduction scheme by making it approximately the same as those of the fraction producing scheme. That is, multiplication by scalars is used to locally make the pivot polynomial approximately equal to 1, and then fraction-free operations are used to eliminate below the pivot as usual.

Finally, recall that Gaussian elimination also takes different flavors in the pivoting dimension. However, this dimension is not explored when proposing the algorithm in chapter 3.
Methods for index reduction of general nonlinear differential-algebraic equations are generally difficult to implement due to the recurring use of functions defined only via the implicit function theorem. By adding structure to the equations, these implicit functions may become possible to implement. In particular, this is so for the quasilinear and linear time-invariant (LTI) structures, and it turns out that there exists an algorithm for the quasilinear form that is a generalization of the shuffle algorithm for the LTI form in the sense that, when applied to the LTI form, it reduces to the shuffle algorithm. For this reason, the more general algorithm is referred to as a quasilinear shuffle algorithm.

This chapter is devoted to quasilinear shuffling. The important connection to chapters 5 and 6 is the seminumerical approach presented in section 3.2.4. This chapter also contains a discussion on how the algorithm can be used to find consistent initial conditions, and touches upon the issue of the algorithm complexity.

**Notation.** We use a star to mark that a symbol denotes a constant. For instance, the symbol $E^*$ denotes a constant matrix, while a symbol like $E$ would in general refer to a matrix-valued function. A few times, we will encounter the gradient of a matrix-valued function. This object will be a function with 3 indices, but rather than adopting tensor notation with the Einstein summation convention, we shall permute indices using generalized transposes denoted $(\bullet)^T$ and $(\bullet)^\dagger$. Since their operation will be clear form the context, they will not be defined formally in this thesis.
3.1 Index reduction by shuffling

In section 2.2.3, algorithm 2.1 provided a way of reducing the differential index of LTI DAE. The extension of that algorithm to the quasilinear form is immediate, but to put this extension in a broader context, we will take the view of it as a specialization instead. In this section, we mainly present the algorithm as it applies to equations which are known exactly, and are to be solved exactly.

3.1.1 The structure algorithm

In section 2.2.5 we presented the structure algorithm (algorithm 2.2) as means for index reduction of general nonlinear DAE,

\[ f(x'(t), x(t), t) = 0 \]  \hspace{1cm} (3.1)

This method is generally not possible to implement, since the recurring use of the implicit function theorem often leaves the user with functions whose existence is given by the theorem, but whose implementation is very involved (to the author’s knowledge, there are to date no available implementations serving this need). However, it is possible to implement for the quasilinear form, as was done, for instance, using Gauss-Bareiss elimination [Bareiss, 1968] in Visconti [1999], or outlined in Steinbrecher [2006].

3.1.2 Quasilinear shuffling

Even though algorithms for quasilinear DAE exist, the results they produce may be computationally demanding, partly because the problems they apply to are still very general. This should be compared with the linear, time-invariant (LTI) case,

\[ E x'(t) + A x(t) + A_v v(t) = 0 \]  \hspace{1cm} (3.2)

to which the very simple and certainly tractable shuffle algorithm (see section 2.2.3) applies. Interestingly, the algorithm for quasilinear DAE described in Visconti [1999] is using the same idea, and it generalizes the shuffle algorithm in the sense that, when applied to the LTI form, it reduces to the shuffle algorithm. For this reason, the algorithm in Visconti [1999] is referred to as a quasilinear shuffle algorithm.\(^1\)

In the next two sections, the alternative view of quasilinear shuffling as a specialization of the structure algorithm is taken. Before doing so, we show using a small example what index reduction of quasilinear DAE can look like.

\(^1\)Note that it is not referred to as the quasilinear shuffle algorithm, since there are many options regarding how to do the generalization. There are also some variations on the theme of the LTI shuffle algorithm, leading to slightly different generalizations.
This example illustrates how row reduction can be performed for a quasilinear DAE. The aim is to present an idea rather than an algorithm, which will be a later topic. Consider the DAE

\[
\begin{pmatrix}
2 + \tan(x_1) & x_2 & 4t \\
2 \cos(x_1) & 0 & e^{x_3} \\
\sin(x_1) & x_2 \cos(x_1) & 4t \cos(x_1) - e^{x_3}
\end{pmatrix}
x' + \begin{pmatrix}
5 \\
x_2 + x_3 \\
x_1 e^{x_3} + t^3 x_2
\end{pmatrix} \overset{!}{=} 0
\]

The leading matrix is singular at any point since the first row times \( \cos(x_1) \) less the second row yields the third row. Adding the second row to the third, and then subtracting \( \cos(x_1) \) times the first, is an invertible operation and thus yields the equivalent equations:

\[
\begin{pmatrix}
2 + \tan(x_1) & x_2 & 4t \\
2 \cos(x_1) & 0 & e^{x_3} \\
0 & 0 & 0
\end{pmatrix}
x' + \begin{pmatrix}
5 \\
x_2 + x_3 \\
x_1 e^{x_3} + t^3 x_2 + x_2 + x_3 - 5 \cos(x_1)
\end{pmatrix} \overset{!}{=} 0
\]

This reveals the implicit constraint of this iteration,

\[x_1 e^{x_3} + t^3 x_2 + x_2 + x_3 - 5 \cos(x_1) \overset{!}{=} 0\]

Then, differentiation yields the new DAE

\[
\begin{pmatrix}
2 + \tan(x_1) & x_2 & 4t \\
2 \cos(x_1) & 0 & e^{x_3} \\
e^{x_3} - 5 \sin(x_1) & t^3 + 1 & x_1 e^{x_3} + 1
\end{pmatrix}
x' + \begin{pmatrix}
5 \\
x_2 + x_3 \\
3 t^2 x_2
\end{pmatrix} \overset{!}{=} 0
\]

Here, the leading matrix is generally non-singular, and the DAE is essentially an ODE bundled with the derived implicit constraint.

### 3.1.3 Time-invariant input affine systems

In this section, the structure algorithm is applied to equations

\[0 \overset{!}{=} f(x(t), x'(t), t)\]

where \( f \) is in the form

\[f(x, \dot{x}, t) = E(x) \dot{x} + A(x) + B(x) v(t)\]

with \( v \) being a given driving function. This system is considered time invariant since time only enters the equation via \( v \).

We consider this form only as an introduction to the analysis in section [4.3](#). After one iteration of the structure algorithm, we will see what requirements (3.3) must fulfill in order for the equations after one iteration to be in the same form as the original equations.
This will show that \((3.3)\) is not a natural form for DAE treated by the structure algorithm. In the next section, a more successful attempt will be made, starting from a more general form than \((3.3)\).

The system is rewritten in the form

\[
x'(t) = \frac{1}{\mu} \dot{x}(t), \quad 0 = f(x(t), \dot{x}(t), t) \tag{3.4}
\]

to match the setup in Rouchon et al. \cite{1995} (recall that the notation \(\dot{x}\) is not defined to denote the derivative of \(x\); it is a composed symbol denoting a newly introduced function which is required to equal the derivative of \(x\) by \((3.4)\)). As is usual in the analysis of DAE, the analysis is only valid locally, giving just a local solution. As is also customary, all matrix ranks that appears are assumed to be constant in the neighborhood of the initial point defining the meaning of local solution.

Running the structure algorithm on this system goes like this (compare algorithm 2.2): Let

\[
\tilde{f}_k(x, \dot{x}, t) = \tilde{E}_k(x) \dot{x} + \tilde{A}_k(x) + \tilde{B}_k(x) v(t)
\]

\[
= \varphi_k(x) \tilde{E}_k(x) \dot{x} + \tilde{A}_k(x) + \tilde{B}_k(x) v(t)
\]

\[
= \varphi_k(x) \left( \tilde{E}_k(x) \dot{x} + \tilde{A}_k(x) + \tilde{B}_k(x) v(t) \right) - \varphi_k(x) \tilde{A}_k(x) - \varphi_k(x) \tilde{B}_k(x) v(t) + \tilde{A}_k(x) + \tilde{B}_k(x) v(t)
\]

\[
= \varphi_k(x) \tilde{f}_k(x, \dot{x}, t) + \tilde{A}_k(x) - \varphi_k(x) \tilde{A}_k(x)
\]

\[
+ \left( \tilde{B}_k(x) - \varphi_k(x) \tilde{B}_k(x) \right) v(t)
\]

Define

\[
\hat{A}_k(x) \triangleq \tilde{A}_k(x) - \varphi_k(x) \tilde{A}_k(x)
\]

\[
\hat{B}_k(x) \triangleq \tilde{B}_k(x) - \varphi_k(x) \tilde{B}_k(x)
\]

\[
\Phi_k(x, t, y) \triangleq \varphi_k(x) y + \hat{A}_k(x) + \hat{B}_k(x) v(t)
\]

and note that along solutions,

\[
\Phi_k(x, t, 0) = \Phi_k(x, t, \tilde{f}_k(x, \dot{x}, t)) = \tilde{f}_k(x, \dot{x}, t) = 0
\tag{3.8}
\]
In particular, the expression is constant over time, so it can be differentiated with respect to time to obtain a substitute for the (locally) uninformative equations given by \( \tilde{f}_k \). Thus, let
\[
f_{k+1}(x, \dot{x}, t) \triangleq \left( \frac{\partial}{\partial t} \Phi_k(x(t), t, 0) \right)(t)
\]  
(3.9)

Expanding the differentiation using the chain rule, it follows that
\[
\frac{\partial}{\partial t} \Phi_k(x(t), t, 0) = \Phi_k^{(1,0,0)}(x(t), t, 0) x'(t) + \Phi_k^{(0,1,0)}(x(t), t, 0) \\
= \left( \nabla \hat{A}_k(x(t)) + \left( \nabla^T \hat{B}_k(x(t)) v(t) \right)^T \right) x'(t) \\
+ \hat{B}_k(x(t)) v'(t)
\]  
(3.10)

However, since \( x' = \dot{x} \) along solutions, the following defines a valid replacement for \( f_k \):
\[
f_{k+1}(x, \dot{x}, t) = \\
\left\{ \begin{array}{c}
\hat{E}_k(x) \\
\nabla \hat{A}_k(x)
\end{array} \right\} + \left( \left( \begin{array}{c}
0 \\
\nabla^T \hat{B}_k(x)
\end{array} \right) v(t) \right)^T \hat{x} + \left( \begin{array}{c}
\hat{A}_k(x) \\
0
\end{array} \right) + \left( \begin{array}{c}
\hat{B}_k(x) \\
0
\end{array} \right) \left[ \begin{array}{c}
v(t)
\end{array} \right]
\]  
(3.11)

We have now completed one iteration of the structure algorithm, and turn to finding conditions on (3.3) that make (3.11) fulfill the same conditions.

In (3.11), the product between \( v(t) \) and \( \dot{x}(t) \) is unwanted, so the structure is restricted by requiring
\[
\nabla \hat{B}_k(x(t)) = 0
\]  
(3.12)

that is, \( \hat{B}_k \) is constant; \( \hat{B}_k(x) = \hat{B}_k^* \).

Unfortunately, the conflict has just been shifted to a new location by this requirement. The structure of \( f_{k+1} \) does not match the structure in (3.3) together with the assumption (3.12), since \( \hat{B}_k(x) \) includes the non-constant expression \( \varphi_k(x) \). Hence it is also required that \( E_k \) is constant so that \( \varphi_k(x) \) may be chosen constant. This is written as \( \hat{E}_k(x) = \hat{E}_k^* \).

Then, if structure is to be maintained,
\[
\left( \begin{array}{c}
\hat{E}_k^* \\
\nabla \hat{A}_k(x)
\end{array} \right)
\]  

would have to be constant. Again, this condition is not met since \( \nabla \hat{A}_k(x) \) is generally not constant. Finally, we are led to also requiring that \( \nabla \hat{A}_k(x) \) be constant. In other words, that
\[
\hat{A}_k(x) = \hat{A}_k^* x
\]  
(3.13)

so the structure of (3.3) is really
\[
f(x, \dot{x}, t) = E^* \dot{x} + A^* x + B^* v(t)
\]  
(3.14)

which is a standard constant coefficient linear DAE.
Note that another way to obtain conditions on (3.3) which become fulfilled also by (3.11) is to remove the driving function $v$.

The key point of this section, however, is that we have seen that in order to be able to run the structure algorithm repeatedly on equations in the form (3.3), an implementation that is designed for one iteration on (3.3) is insufficient. In other words, if an implementation that can be iterated exists, it must apply to a more general form than (3.3).

### 3.1.4 Quasilinear structure algorithm

Seeking a replacement for (3.3) such that an implementation for one step of the structure algorithm can be iterated, a look at (3.11) suggests that the form should allow for dependency on time in the leading matrix. Further, since the driving function $v$ has entered the leading matrix, the feature of $v$ entering the equations in a simple way has been lost. Hence it is no longer motivated to keep $A_k(x)$ and $B_k(x)v_k(t)$ separate, but we might as well turn to the quasilinear form in its full generality,

$$f_k(x, \dot{x}, t) = E_k(x, t) \dot{x} + A_k(x, t)$$

The reader is referred to the previous section for the notation used below. This time, the constant rank assumption leads to the existence of a $\varphi_k$ such that

$$\tilde{E}_k(x, t) = \varphi_k(x, t) \bar{E}_k(x, t)$$

Such a $\varphi_k$ can be obtained from a row reduction of $E$, and corresponds to the row reduction performed in a quasilinear shuffle algorithm.

It follows that

$$\tilde{f}_k(x, \dot{x}, t) = \tilde{E}_k(x, t) \dot{x} + \tilde{A}_k(x, t)
= \varphi_k(x, t) \bar{E}_k(x, t) \dot{x} + \bar{A}_k(x, t)
\varphi_k(x, t) \left( \bar{E}_k(x, t) \dot{x} + \bar{A}_k(x, t) \right)
- \varphi_k(x, t) \bar{A}_k(x, t) + \bar{A}_k(x, t) - \varphi_k(x, t) \bar{A}_k(x, t)$$

Define

$$\hat{A}_k(x, t) \triangleq \bar{A}_k(x, t) - \varphi_k(x, t) \bar{A}_k(x, t)$$

$$\Phi_k(x, t, y) \triangleq \varphi_k(x, t) y + \hat{A}_k(x, t)$$

and note that along solutions,

$$\Phi_k(x, t, 0) = \Phi_k(x, t, \tilde{f}_k(x, \dot{x}, t)) = \tilde{f}_k(x, \dot{x}, t) = 0$$

Taking a quasilinear shuffle algorithm perspective on this, we see that $\Phi_k(x, t, 0) = \hat{A}_k(x, t)$ is computed by applying the same row operations to $A$ as were used to find the function $\varphi_k$ above.
The expression $\Phi_k(x, t, 0)$ is constant over time, so it can be differentiated with respect to time to obtain a substitute for the (locally) uninformative equations given by $\tilde{f}_k$. Thus, let

$$f_{k+1}(x, \dot{x}, t) \triangleq \left( \frac{\partial}{\partial t} \Phi_k(x(t), t, 0) \right)(t) \quad (3.19)$$

Expanding the differentiation using the chain rule, it follows that

$$\frac{\partial}{\partial t} \Phi_k(x(t), t, 0) = \Phi_k^{(1,0,0)}(x(t), t, 0) x'(t) + \Phi_k^{(0,1,0)}(x(t), t, 0)$$

$$= \hat{A}_k^{(1,0)}(x(t), t) x'(t) + \hat{A}_k^{(0,1)}(x(t), t) \quad (3.20)$$

Again, since $x' = \dot{x}$ along solutions, $h_k$ may be replaced by

$$h_{k+1}(x, \dot{x}, t) = \left( \frac{E_k(x, t)}{\hat{A}_k^{(1,0)}(x, t)} \right) \dot{x} + \left( \frac{\hat{A}_k(x, t)}{\hat{A}_k^{(0,1)}(x, t)} \right) \quad (3.21)$$

This completes one iteration of the structure algorithm, and it is clear that this can also be seen as the completion of one iteration of a quasilinear shuffle algorithm. As opposed to the outcome in the previous section, this time (3.21) is in the form we started with, so the procedure can be iterated.

### 3.2 Proposed algorithm

Having seen how the structure algorithm can be implemented as an index reduction method for (exact) quasilinear DAE, and that this results in an immediate generalization of the shuffle algorithm for LTI DAE, we now turn to the task of detailing the algorithm to make it applicable in a practical setting. Issues to be dealt with include finding a suitable row reduction method and determining whether an expression is zero along solutions.

The problem of adopting algorithms for revealing hidden constraints in exact equations to a practical setting has previously been addressed in Reid et al. [2002]. The geometrical awareness in their work is convincing, and the work was extended in Reid et al. [2005]. For examples of other approaches to system analysis and/or index reduction which remind of ours, see for instance Unger et al. [1995] or Chowdhry et al. [2004].

#### 3.2.1 Algorithm

The reason to do index reduction in the following particular way is that it is simple enough to make the analysis in section 4.3 easy, and also that it does not rule out some of the candidate forms (see section 4.2) already in the row reduction step by producing a leading matrix outside the form (see the highlighted part of the algorithm description below). If maintaining invariant forms would not be a goal in itself, the algorithm could easily be given better numeric properties (compare section 2.4), and/or better performance in terms of computation time (by reuse of expressions and similar techniques).
We stress again that an index reduction algorithm is typically run repeatedly until a low index is obtained (compare, for instance, algorithm 2.2). Here, only one iteration is described, but this is sufficient since the algorithm output is in the same form as the algorithm input was assumed to be.

Recall the discussion on fraction producing versus fraction-free row reduction schemes in section 2.4. The proposed algorithm uses a fraction-free scheme for two reasons. Most importantly in this paper, it does so in order to hold more invariant forms. Of subordinate importance is that it can be seen as a heuristics for producing simpler expressions. The body of the index reduction loop is given in algorithm 3.1.

**Example 3.2**

Here, one iteration is performed on the following quasilinear DAE:

\[
\begin{pmatrix}
 x_1(t) & x_2(t) & \sin(t) & 0 \\
 e^{x_3(t)} & x_1(t) & 0 & 0 \\
 t & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x'_1(t) \\
x'_2(t) \\
x'_3(t)
\end{pmatrix} + \begin{pmatrix}
x(t)^3 \\
cos(t) \\
4
\end{pmatrix} = 0
\]

The leading matrix is clearly singular, and has rank 2.

For the first step in the algorithm, there’s freedom to pick any two rows as the independent ones. For instance, the rows \{1, 3\} are chosen. The remaining row can then be eliminated using the following series of fraction-free row operations. First:

\[
\begin{pmatrix}
x_1(t) x_2(t) - t \sin(t) & 0 & 0 \\
e^{x_3(t)} - t x_1(t) & 0 & 0 \\
t & 1 & 0
\end{pmatrix}
\begin{pmatrix}
x'_1(t) \\
x'_2(t) \\
x'_3(t)
\end{pmatrix} + \begin{pmatrix}
x(t)^3 - 4 \sin(t) \\
cos(t) - 4 x_1(t) \\
4
\end{pmatrix} = 0
\]

Then:

\[
\begin{pmatrix}
x_1(t) x_2(t) - t \sin(t) & 0 & 0 \\
0 & 0 & 0 \\
t & 1 & 0
\end{pmatrix}
\begin{pmatrix}
x'_1(t) \\
x'_2(t) \\
x'_3(t)
\end{pmatrix} + \begin{pmatrix}
x(t)^3 - 4 \sin(t) \\
e(x(t), t) \\
4
\end{pmatrix} = 0
\]

where the algebraic equation discovered is given by

\[
e(x, t) = (x_1 x_2 - t \sin(t)) (\cos(t) - 4 x_1) - (e^{x_3} - t x_1) (x_2^3 - 4 \sin(t))
\]

Differentiating the derived equation with respect to time yields a new equation with residual in the form

\[
(a_1(x(t), t) a_2(x(t), t) a_3(x(t), t)) \begin{pmatrix}
x'_1(t) \\
x'_2(t) \\
x'_3(t)
\end{pmatrix} + b(x(t), t)
\]

\footnote{It would be of no consequence for the analysis below to require that the set of equations chosen in the first step always include the equations selected in the previous iteration, as is done in [Rouchon et al., 1995].}
Algorithm 3.1 Quasilinear shuffling iteration for invariant forms.

**Input:** A square DAE,

\[ E(x(t), t) \dot{x}(t) + A(x(t), t) x(t) \overset{!}{=} 0 \]

It is assumed that the leading matrix is singular (when the leading matrix is non-singular, the index is 0 and index reduction is neither needed nor possible).

**Output:** An equivalent square DAE of lower index, and additional algebraic constraints.

**Iteration:**

Select a set of independent rows in \( E(x(t), t) \).

Perform fraction-free row reduction of the equations such that exactly the rows that were not selected in the previous step are zeroed. The produced algebraic terms corresponding to the zero rows in the leading matrix, define algebraic equations restricting the solution manifold.

Differentiate the newly found algebraic equations with respect to time, and join the resulting equations with the ones selected in the first step to obtain the new square DAE.

**Remarks:** By far, the most important remark to make here is that the differentiation is not guaranteed to be geometric (recall the remark in algorithm 2.2). Hence, the termination criterion based on the number of iterations in algorithm 2.2 cannot be used safely in this context. If that termination criterion is met, our algorithm aborts with “non-geometric differentiation” instead of “ill-posed”, but no conclusion regarding the existence of solutions to the DAE can be drawn.

Although there are choices regarding how to perform the fraction-free row reduction, a conservative approach is taken by not assuming anything more fancy than fraction-free Gaussian elimination, with pivoting used only when so required and done the most naïve way. This way, it is ensured that the tailoring of the reduction algorithms is really just a tailoring rather than something requiring elaborate extension.

As an alternative to the fraction-free row reduction, the same step may be seen as a matrix factorization [Steinbrecher 2006]. This view hides the reduction process in the factorization abstraction, and may therefore be better suited for high-level reasoning about the algorithm, while the proposed method may be more natural from an implementation point of view and easier to reason about on a lower level of abstraction (like when applying the algorithm to equations with a particular degree of nonlinearity in section 4.3).
where
\[ a_1(x, t) = x_2 \left( \cos(t) - 4x_1 \right) - 4 \left( x_1 x_2 - t \sin(t) \right) + t \left( x_2^3 - 4 \sin(t) \right) \]
\[ a_2(x, t) = x_1 \left( \cos(t) - 4x_1 \right) - 3 x_2^2 \left( e^{x_3} - t x_1 \right) \]
\[ a_3(x, t) = -e^{x_3} \left( x_2^2 - 4 \sin(t) \right) \]
\[ b(x, t) = -\left( \sin(t) + t \cos(t) \right) \left( \cos(t) - 4x_1 \right) - \sin(t) \left( x_1 x_2 - t \sin(t) \right) \]
\[ + x_1 \left( x_2^3 - 4 \sin(t) \right) 4 \cos(t) \left( e^{x_3} - t x_1 \right) \]

Joining the new equation with the ones previously selected yields the following output from the algorithm (dropping some notation for brevity):

\[
\begin{pmatrix}
x_1(t) & x_2(t) & \sin(t) & 0 \\
t & 1 & 0 & 1 \\
a_1 & a_2 & a_3 & 0
\end{pmatrix}
+ \begin{pmatrix}
x_1'(t) \\
x_2'(t) \\
x_3'(t)
\end{pmatrix}
+ \begin{pmatrix}
x_2(t)^3 \\
4 \\
b
\end{pmatrix} = 0
\]

Unfortunately, the expression swell seen in this example is typical for the investigated algorithm. Compare with the neat outcome in example 3.1 where some intelligence was used to find a parsimonious row reduction.

### 3.2.2 Zero tests

The crucial step in algorithm 3.1 is the row reduction, but exactly how this can be done has not been discussed yet. One of the important topics for the row reduction to consider is how it should detect when it is finished. For many symbolic matrices whose rank is determined by the zero-pattern, the question is easy; the matrix is row reduced when the rows which are not independent by construction consist of only structural zeros. This was the case in example 3.2. However, the termination criterion is generally more complicated since there may be expressions in the matrix which are identically zero, although this is hard to detect using symbolic software.

It is proposed that structural zeros are tracked in the algorithm, making many of the zero tests trivially affirmative. An expression which is not a structural zero is tested against zero by evaluating it (and possibly its derivative with respect to time) at the point where the index reduction is being performed. If this test is passed, the expression is assumed rewritable to zero, but anticipating that this will be wrong occasionally, the expression is also kept in a list of expressions that are assumed to be zero for the index reduction to be valid. Numerical integrators and the like can then monitor this list of expressions, and take appropriate action when an expression no longer evaluates to zero.

Note that there are some classes of quasilinear DAE where all expressions can be put in a canonical form where expressions that are identically zero can be detected. For instance, this is the case when all expressions are polynomials.
Of course, some tolerance must be used when comparing the value of an expression against zero. Setting this tolerance is non-trivial, and at this stage we have no scientific guidelines to offer. This need was the original motivation for the research presented in chapters 5 and 6.

The following example exhibits the weakness of the numerical evaluation approach. It will be commented on in section 3.2.5.

--- Example 3.3 ---

Let us consider numerical integration of the mathematical pendulum, modeled by

$$\begin{align*}
x'' &= \lambda x \\
y'' &= \lambda y - g \\
1 &= x^2 + y^2
\end{align*}$$

where we take $g := 10.0$. Index reduction will be performed at two points (the time part of these points is immaterial and will not be written out): one at rest, the other not at rest. Note that it is quite common to begin a simulation of a pendulum (as well as many other systems) at rest. The following values give approximately an initial angle of 0.5 rad below the positive $x$ axis:

$x_{0, \text{rest}} : \{ x(0) = 0.87, \dot{x}(0) = 0, y(0) = -0.50, \dot{y}(0) = 0, \lambda(0) = -5.0 \}$

$x_{0, \text{moving}} : \{ x(0) = 0.87, \dot{x}(0) = -0.055, y(0) = -0.50, \dot{y}(0) = -0.1, \lambda(0) = -4.8 \}$

Clearly, if $\dot{x}$ or $\dot{y}$ constitute an element of any of the intermediate leading matrices, the algorithm will be in trouble, since these values are not typically zero. After two reduction steps which are equal for both points, the equations look as follows (not showing the already deduced algebraic constraints):

$$\begin{pmatrix}
1 & 1 \\
1 & 1 \\
2 \dot{x} & 2x & 2 \dot{y} & 2y
\end{pmatrix} \begin{pmatrix}
x' \\
\dot{x}' \\
y' \\
\dot{y}' \\
\lambda'
\end{pmatrix} + \begin{pmatrix}
\lambda x \\
10 - \lambda y \\
-\dot{x} \\
-\dot{y} \\
0
\end{pmatrix} = 0$$

When row reducing these equations at $x_{0, \text{rest}}$, the algorithm will produce the algebraic equation

$$20 \dot{y} = 2 \left( x^2 + y^2 \right) \lambda$$

but the correct equation, produced at $x_{0, \text{moving}}$ is

$$20 \dot{y} = 2 \left( \left( x^2 + y^2 \right) \lambda + \dot{x}^2 + \dot{y}^2 \right)$$

Our mechanical understanding of the problem gives immediately that $\dot{x}'$ and $\dot{y}'$ are non-zero at $x_{0, \text{rest}}$. Hence, computing the derivatives of all variables using a reduction to index 0 would reveal the mistake.
As a final note on the failure at $x_{0,rest}$, note that $\dot{x}$ and $\dot{y}$ would be on the list of expressions that had been assumed zero. Checking these conditions after integrating the equations for a small period of time would detect the problem, so delivery of an erroneous result is avoided.

### 3.2.3 Longevity

At the point $(x(t_0), t_0)$, the proposed algorithm performs tasks such as row operations, index reduction, selection of independent equations, etc. Each of these may be valid at the point they were computed, but fail to be valid at future points along the solution trajectory. By the longevity of such an operation, or the product thereof, we refer to the duration until validity is lost.

A row operation becomes invalid when its pivot element becomes zero. A selection of equations to be part of the square index 1 system becomes invalid when the iteration matrix looses rank. An index reduction becomes invalid if an expression which was assumed to be zero becomes non-zero. The importance of longevity considerations is shown by an example.

---

**Example 3.4**

A circular motion is described by the following equations (think of $\zeta$ as “zero”):

\[
\begin{align*}
\zeta & = x' x + y' y \\
1 & = (x')^2 + (y')^2 \\
1 & = x^2 + y^2
\end{align*}
\]

This system is square but not in quasilinear form. The trivial conversion to quasilinear form described in section 2.2.4 yields a square DAE of size 5 with new variables introduced for the derivatives $x'$ and $y'$.

By the geometrical interpretation of the equations we know that the solution manifold is one-dimensional and equal the two disjoint sets (distinguished by the sign choices below, of which one has been chosen to work with)

\[
\begin{align*}
\begin{cases}
x = \cos(\beta), & \dot{x} = (\mp) \sin(\beta), \\
\dot{x} = \mp \cos(\beta), & \dot{y} = (\pm) \cos(\beta), \\
\zeta = 0, & \\
\beta \in [0, 2\pi]
\end{cases}
\end{align*}
\]

Let us consider the initial conditions given by $\beta = 1.4$ in the set characterization. The
quasilinear equations are:

\[
\begin{align*}
\dot{x} &= x' \\
\dot{y} &= y' \\
\zeta &= \dot{x} x + \dot{y} y \\
1 &= \dot{x}^2 + \dot{y}^2 \\
1 &= x^2 + y^2
\end{align*}
\]

Note that there are three algebraic equations here. The equations are already row reduced, and after performing one differentiation of the algebraic constraints and one row reduction, the DAE looks like

\[
\begin{pmatrix}
1 & 1 & -1 & x & y \\
\dot{x} & \dot{y} & -1 & x & y \\
2 \dot{x} & 2 \dot{y} & & & \\
\end{pmatrix}
\begin{pmatrix}
x' \\
y' \\
\zeta' \\
\dot{x}' \\
\dot{y}'
\end{pmatrix}
+ \begin{pmatrix}
-x \\
-\dot{y} \\
0 \\
0 \\
2 x \dot{x} + 2 y \dot{y}
\end{pmatrix}
\]

Differentiation of the derived algebraic constraints will yield a full-rank leading matrix, so the index reduction algorithm terminates here. There are now four differential equations,

\[
\begin{pmatrix}
1 & 1 & -1 & x & y \\
\dot{x} & \dot{y} & -1 & x & y \\
2 \dot{x} & 2 \dot{y} & & & \\
\end{pmatrix}
\begin{pmatrix}
x' \\
y' \\
\zeta' \\
\dot{x}' \\
\dot{y}'
\end{pmatrix}
+ \begin{pmatrix}
-x \\
-\dot{y} \\
0 \\
0 \\
2 x \dot{x} + 2 y \dot{y}
\end{pmatrix}
\]

and four algebraic equations

\[
\begin{align*}
\zeta &= \dot{x} x + \dot{y} y \\
1 &= \dot{x}^2 + \dot{y}^2 \\
1 &= x^2 + y^2 \\
0 &= 2 x \dot{x} + 2 y \dot{y}
\end{align*}
\]

with Jacobian

\[
\begin{pmatrix}
\dot{x} & \dot{y} & -1 & x & y \\
2 \dot{x} & 2 \dot{y} & & & \\
2 \dot{x} & 2 \dot{y} & & & \\
\end{pmatrix}
\]

The algebraic equations are independent, so they shall be completed with one of the differential equations to form a square index 1 system. The last two differential equations are linearly dependent on the algebraic equations by construction, but either of the first

\[3\] Another quasilinear formulation would be obtained by replacing the third equation by \( \zeta = x x' + y y' \), containing only two explicit algebraic equations. The corresponding leading matrix would not be row reduced, so row reduction would reveal an implicit algebraic equation, and the result would be the same in the end.
two differential equations is a valid choice. Depending on the choice, the first row of the iteration matrix will be either of

\[
\begin{pmatrix} 1 & 0 & 0 & -h \\ 0 & 1 & 0 & 0 & -h \end{pmatrix}
\]

After a short time, the four other rows of the iteration matrix (which are simply the Jacobian of the algebraic constraints) will approach

\[
\begin{pmatrix}
\sin\left(\frac{\pi}{2}\right) & -\cos\left(\frac{\pi}{2}\right) & -1 & \cos\left(\frac{\pi}{2}\right) & \sin\left(\frac{\pi}{2}\right) \\
2\cos\left(\frac{\pi}{2}\right) & 2\sin\left(\frac{\pi}{2}\right) & 2 & \sin\left(\frac{\pi}{2}\right) & 2\cos\left(\frac{\pi}{2}\right) \\
2\sin\left(\frac{\pi}{2}\right) & -2\cos\left(\frac{\pi}{2}\right) & 2 & 2 & \sin\left(\frac{\pi}{2}\right) \\
2 & 2 & 2 & 2 & 2 \\
\end{pmatrix}
\]

In particular, the third row will be very aligned with

\[
\begin{pmatrix} 0 & 1 & 0 & 0 & -h \end{pmatrix}
\]

which means that it is better to select the differential equation \( \dot{x} = x' \) than \( \dot{y} = y' \). This holds not only on paper, but numerical simulation using widespread index 1 solution software [Hindmarsh et al., 2004, through the Mathematica interface] demands that the former differential equation be chosen.

This example illustrated the fact that, if an implementation performs row reduction and selection of equations to be in the final square system without really caring about the choices it makes, things as the ordering of equations and variables may influence the end result. The usefulness of the reduced equations do depend on implementation details in the algorithm, even though the result does not feature any numerically computed entities.

Even though repeated testing of the numerical conditioning while the equations are integrated is sufficient to detect numerical ill-conditioning, the point made here is that at the point \((x_0, t_0)\) one wants to predict what will be the good ways of performing the row reduction and selecting equations to appear in the square index 1 form.

While it is difficult to foresee when the expressions which are assumed rewritable to zero seizes to be zero (the optimistic longevity estimation is simply that they will remain zero forever), there is more to be done concerning the longevity of the row reduction operations. For each element used as a pivot, it is possible to formulate scalar conditions that are to be satisfied as long as the pivot stay in use. For instance, it can be required that the pivot be no smaller in magnitude than half the magnitude of largest value it is used to cancel.

Using the longevity predictions, each selection of a pivot can be made to maximize longevity. Clearly, this is a non-optimal greedy strategy (since only one pivot selection is considered at a time, compared to considering all possible sequences of pivot selections at once), but it can be implemented with little effort and at a reasonable runtime cost.
3.2 Proposed algorithm

3.2.4 Seminumerical twist

In a previous section, it was suggested that numerical evaluation of expressions (combined with tracking of structural zeros) should be used to determine whether an expression can be rewritten to zero or not. That added a bit of numerics to an otherwise symbolic algorithm, but this combination of symbolic and numeric techniques is more of a necessity than a nice twist. We now suggest that numerical evaluation should also be the basic tool when predicting longevity. While the zero tests are accompanied by difficult questions about tolerances, but are otherwise rather clear how to perform, it is expected that the numeric decisions discussed in this section allow more sophistication while not requiring intricate analysis of how tolerances shall be set.

Without loss of generality, it is assumed that the scalar tests compare an expression, \( e \), with the constant 0. The simplest way to estimate (predict) the longevity of

\[ e(x(t), t) < 0 \]

at the point \((x_0, t_0)\) is to first compute the derivatives \(x'\) at time \(t_0\) using a method that does not care about longevity, and use linear extrapolation to find the longevity. In detail, the longevity, denoted \( L_e(x_0, t_0) \), may be estimated as

\[
\dot{e}(x_0, t_0) := \partial_1 e(x_0, t_0) x'(t_0) + \partial_2 e(x_0, t_0)
\]

\[
\hat{L}_e(x_0, t_0) := \begin{cases} \frac{-e(x_0, t_0)}{\dot{e}(x_0, t_0)} & \text{if } \dot{e} > 0 \\ \infty & \text{otherwise} \end{cases}
\]

In case of several alternatives having infinite longevity estimates by the calculation above, the selection criterion needs to be refined. The natural extension of the above procedure would be to compute higher order derivatives to be able to estimate the first zero-crossing, but that would typically involve more differentiation of the equations than is needed otherwise, and is therefore not a good option. Rather, some other heuristic should be used. One heuristic would be to disregard signs, but one could also altogether ignore derivatives when this situation occurs and fall back on the usual pivot selection based on magnitudes only.

A very simple way to select equations for the square index 1 system is to greedily add one equation at a time, picking the one which has the largest angle to its projection in the space spanned by the equations already selected. If the number of possible combinations is not overwhelmingly large, it may also be possible to check the condition number for each combination, possibly also taking into account the time derivative of the condition number.

3.2.5 Monitoring

Since the seminumerical algorithm may make false judgements regarding what expressions are identically zero, expressions which are not explicitly zero but have passed the
zero-test anyway needs to be monitored. It may not be necessary to evaluate these expressions after each time step, but as was seen in example 3.3 it is wise to be alert during the first few iterations after the point of index reduction.

For the (extremely) basic BDF method applied to equations of index 1, the local integration accuracy is limited by the condition number of the iteration matrix for a time step of size $h$. In the quasilinear index 1 case and for small $h$, the matrix should have at least as good condition as

$$
\left( \begin{array}{c} \bar{E}(x, t) \\ \nabla_1 \bar{A}(x, t) \end{array} \right)
$$

(3.22)

To see where this comes from, consider solving for $x_n$ in

$$
\left( \begin{array}{cc} \bar{E}(x_n, t_n) & x_n - x_{n-1} \\ 0 & h_n \end{array} \right) \frac{h_n}{x_n} + \left( \begin{array}{c} \bar{A}(x_n, t_n) \\ \nabla_1 \bar{A}(x_n, t_n) \end{array} \right) \frac{1}{h_n} = 0
$$

where $x_{n-1}$ is the iterate at time $t_n - h_n$. The equations being index 1 guarantees that this system has a locally unique solution for $h_n$ small enough. Any method of some sophistication will perform row (and possibly column) scaling at this stage to improve numerical conditioning.\cite{Brenan1996, Golub1996} It is assumed that any implementation will achieve at least as good condition as is obtained by scaling the first group of equations by $h_n$.

For small $h_n$, the equations may be approximated by their linearized counterpart for which the numerical conditioning simply given by the condition number of the coefficient matrix for $x_n$. See for example\cite{Golub1996} for a discussion of error analysis for linear equations. This coefficient of the linearized equation is

$$
\left( \begin{array}{c} \nabla_1^T \bar{E}(x_n, t_n) \cdot (x_n - x_{n-1}) \\ 0 \end{array} \right) + \bar{E}(x_n, t_n) + \left( \begin{array}{c} h_n \nabla_1 \bar{A}(x_n, t_n) \\ \nabla_1 \bar{A}(x_n, t_n) \end{array} \right)
$$

Using the approximation

$$
\frac{h_n}{x_n} \approx h_n x'(t_n) \approx h_n \left( \begin{array}{c} \bar{E}(x_n, t_n) \\ \nabla_1 \bar{A}(x_n, t_n) \end{array} \right)^{-1} \left( \begin{array}{c} \bar{A}(x_n, t_n) \\ \nabla_2 \bar{A}(x_n, t_n) \end{array} \right)
$$

gives the coefficient

$$
\left( \begin{array}{c} \bar{E}(x_n, t_n) \\ \nabla_1 \bar{A}(x_n, t_n) \end{array} \right) + h_n \left( \begin{array}{c} \nabla_1^T \bar{E}(x_n, t_n) \left( \begin{array}{c} \bar{E}(x_n, t_n) \\ \nabla_1 \bar{A}(x_n, t_n) \end{array} \right)^{-1} \left( \begin{array}{c} \bar{A}(x_n, t_n) \\ \nabla_2 \bar{A}(x_n, t_n) \end{array} \right) \\ 0 \end{array} \right)^{\top} + \nabla_1 \bar{A}(x_n, t_n)
$$

\footnote{Note that the iteration matrix of example 2.5 was found for an LTI DAE, while we are currently considering the more general quasilinear form.}

\footnote{The notation used is not widely accepted. Neither will it be explained here since the meaning should be quite intuitive, and the terms involving inverse transposes will be discarded in just a moment.}
As $h_n$ approaches 0, the matrix tends to \((3.22)\). This limit will be used to monitor numerical integration in examples, but rather than looking at the raw condition number $\kappa(t)$ as a function of time $t$, a static transform, $\phi$, will be applied to this value in order to facilitate prediction of when the iteration matrix becomes singular. If possible, $\phi$ should be chosen such that $\phi(\kappa(t))$ is approximately linear near a singularity.

Since the $\infty$-norm and 2-norm condition numbers are essentially the same, the static transform is heuristically developed for the 2-norm condition number. Approximating the singular values to first order as functions of time, it follows that, near a singularity, the condition number can be expected to grow as $t \mapsto \frac{1}{t_1-t}$, where $t_1$ is the time of the singularity.

How can $\phi$ be chosen to match the behavior of the condition number near a singularity? The following observation is useful: Suppose $\phi$ is unbounded above, that is, $\phi(\kappa) \to \infty$ as $\kappa \to \infty$. Then every linear approximation of $\phi(\kappa(t))$ will be bad near the singularity, since the linear approximation cannot tend to infinity. Hence, one should consider strictly increasing functions that map infinity to a finite value. A trivial such example is the arctan function. Given the assumed growth of the condition number near a singularity, an expression for $\phi(\kappa)$ can be found by requiring:

$$
\phi\left(\frac{1}{t_1-t}\right) = t - t_1 \iff \\
\phi\left(\frac{1}{\kappa}\right) = -\kappa \iff \\
\phi(\kappa) = -\frac{1}{\kappa}
$$

Since $\kappa$ is always at least 1, this will squeeze the half open interval $[1, \infty)$ onto $[-1, 0)$. As is seen in figure 3.1, the first order approximation is useful well in advance of the singularity. However, further away it is not. For example the prediction based on the linearization at time 2 would be rather optimistic.

### 3.2.6 Sufficient conditions for correctness

It may not be obvious that the seminumerical row reduction algorithm above really does the desired job. After all, it may seem a bit simplistic to reduce a symbolic matrix based on its numeric values evaluated at a certain point. In this section, sufficient (while perhaps conservative) conditions for correctness will be presented. Some new nomenclature will be introduced, but only for the purpose of making the theorem below readable.

Consider the quasilinear DAE

$$
E(x(t), t)x'(t) + A(x(t), t) = 0
$$

Replacing a row

$$
e_i(x(t), t)x'(t) + a_i(x(t), t) = 0
$$
Shuffling quasilinear DAE

\[
\begin{align*}
\omega(x, t) & e_i(x, t) + \eta(x, t) e_j(x, t) \Big) x' + \left[ \omega(x, t) a_i(x, t) + \eta(x, t) a_j(x, t) \right] = 0 \\
E(x(t), t) x'(t) + A(x(t), t) = 0
\end{align*}
\]

where \( \omega \) and \( \eta \) are both continuous at \( (x_0, t_0) \) and \( \omega \) is non-zero at this point, is called a non-singular row operation on the DAE.

Since the new DAE is obtained by multiplication from the left by a non-singular matrix, the non-singular row operation on the quasilinear DAE does not alter the rank of the leading matrix.

Let \( x \) be a solution to the DAE on the interval \( I \), and assume that the rank of \( E(x(t), t) \) is constant as a function of \( t \) on \( I \). A valid row reduction at \( (x_0, t_0) \) of the original (quasilinear) DAE

\[
E(x(t), t) x'(t) + A(x(t), t) = 0
\]

is a sequence of non-singular row operations such that the resulting (quasilinear) DAE

\[
E_{rr}(x(t), t) x'(t) + A_{rr}(x(t), t) = 0
\]

has the following properties:

1. A solution \( x \) is locally a solution to the original DAE if and only if it is a solution to the resulting DAE.

2. \( E_{rr}(x, t) \) has only as many non-zero rows as \( E(x, t) \) has rank.

**Theorem 3.1**

*Consider the time interval \( I \) with \( \inf I = t_0 \), and the DAE with initial condition \( x(t_0) = x_0 \). Assume*

1. *The DAE with initial condition is consistent, and the solution is unique and differentiable on \( I \).*
2. The DAE is sufficiently differentiable for the purpose of running the row reduction algorithm.

3. Elements of $E(x_0, t_0)$ that are zero, are zero in $E(x(t), t)$ for all $t \in I$. Further, this condition shall hold for intermediate results as well.

Then there exists a time $t_1 \in I$ with $t_1 > t_0$ such that the row reduction of the symbolic matrix $E(x, t)$ based on the numeric guide $E(x_0, t_0)$ will compute a valid row reduction where the non-zero rows of the reduced leading matrix $E_{rr}(x(t), t)$ are linearly independent for all $t \in [t_0, t_1]$.

Proof: The first two assumptions ensure that each element of $E(x(t), t)$ is continuous as a function of $t$ at every iteration. Since the row reduction will produce no more intermediate matrices than there are elements in the matrix, the total number of elements in question is finite, and each of these will be non-zero for a positive amount of time.

Further, the non-zero rows of $E_{rr}(x_0, t_0)$ are independent by construction (as this is the reduced form of the guiding numeric matrix). Therefore they will contain a non-singular sub-block. The determinant of this block will hence be non-zero at time $t_0$, and will be a continuous function of time.

Hence, there exists a time $t_1 \in I$ with $t_1 > t_0$ such that all those expressions that are non-zero at time $t_0$ remain non-zero for all $t \in [t_0, t_1]$. In particular, the determinant will remain non-zero in this interval, thus ensuring linear independence of the non-zero reduced rows.

The last assumption ensures the constant rank condition required by the definition of valid row reduction, which is a consequence of each step in the row reduction preserving the original rank, and the rank revealed by the reduced form is already shown to be constant.

Beginning with the part of the definition of valid row reduction concerning the number of zero-rows, note first that the number of non-zero rows will match the rank at time $t_0$ since the row reduction of the numeric guide will precisely reveal its rank as the number of non-zero rows. It then suffices to show that the zero-pattern of the symbolic matrix contains that of the numeric guide during each iteration of the row reduction algorithm. However, this follows quite direct by the assumptions since the zero-pattern will match at $E(x(t_0), t_0)$, and the assumption about zeros staying zero will ensure that no spurious non-zeros appear in the symbolic matrix evaluated at later points in time.

It remains to show that a function $x$ is a solution of the reduced DAE if and only if it is a solution of the original DAE. However, this is trivial since the result of the complete row reduction process may be written as a multiplication from the left by a sequence of non-singular matrices. Hence, the equations are equivalent.

Note that, in example 3.3, the conditions of theorem 3.1 were not satisfied since the expressions $\dot{x}$ and $\dot{y}$ were zero at $(x_0, t_0)$, but does not stay zero. Since their deviation from zero is continuous, they will stay close to zero during the beginning of the solution interval. Hence, it might be expected that the computed solution is approximately correct during the beginning, and this is confirmed by experiments. However, theory for claiming
that this approximation is valid, and quantifying the degree of approximation, we are not aware of — this is the question raised in this thesis.

3.3 Algorithm complexity

Although not a central topic in this work, algorithm complexity will certainly be an issue in the future, would our algorithms be implemented for application use. By giving this question some attention here, it is hoped that basic awareness of the complexity issue will stimulate future developments in the direction of more efficient tools.

It is recommended to skim chapter 4 before reading this section, since we will make some forward references to and use some notions and observations from that chapter.

3.3.1 Representations and complexity

The relation between how models are represented with data structures and the consequences this bear on the algorithms that operate on these structures, is discussed in section 4.1.1. The basic insight is that the more general representations often come with time and space overheads. However, such overheads do typically not contribute to worse asymptotic complexity when the problems become large. They typically increase costs in time and space by some bounded factor, which may be important when squeezing performance in industrial applications, but do not limit applicability in typical development or research applications.

However, the choice of representation may also have more substantial impact on time and space costs. For example, expression reuse may turn out to be an important technique for performing index reduction on quasilinear DAE. Below, another example is given for polynomial quasilinear DAE. Note that it is not uncommon that clever representations that reduce asymptotic costs require more elaborate schemes for transforming the expressions. Hence, for problems of small size, simple representations with small overheads may be preferable to clever schemes with good asymptotic costs.

3.3.2 Polynomial quasilinear DAE

In section 4.3 it is shown that the quasilinear form with polynomials of bounded degree is not invariant under the iterations of the proposed algorithm due to the degree bound getting violated. It is also seen in the example in section 4.3.4 that the increase in expression complexity during index reduction may be substantial. In this section, a more careful analysis of how much the degree of a polynomial DAE may grow is given. To motivate such an analysis, the degree need be related to algorithm/expression complexity, which leads to assuming that the algorithm stores polynomials in expanded form, that is, as the coefficients of the distinct monomials. However, unless working with differential algebra
tools, this is a rather unnatural and inefficient representation that does not generalize to the hierarchical structure which is needed to represent general nonlinear expressions.

The substantial increase in expression complexity was also one of the motivations to develop the symbolic-numeric method for deriving hidden constraints in polynomial DAE in Reid et al. [2002].

Now consider reduction to index 0 of a DAE with degree k polynomial leading matrix and algebraic term, both independent of time, and the leading matrix being 1 short of rank. Further, let the DAE be of index 1, higher indices will be discussed below.

Let the number of equations and variables be n, and assume they are ordered such that elimination leads to an upper triangular matrix. When the first row is used to eliminate the first column of the others, the others are replaced by polynomials of order k + k = 2k, both in the leading matrix and the algebraic term. When the second row is used to eliminate the second column of the rows below it, the new polynomials will be of order 2k + 2k = 4k. The procedure will continue to double the degree of the polynomials, leaving polynomials of degree 2i−1k on row i. Hence, when it is found that the last row is completely zeroed, the algebraic term which gets differentiated has degree 2n−1k. Differentiation then produces a full row of order 2n−1k − 1. This is the degree of the index 0 DAE. This is enough preparation to turn to the higher index case.

**Theorem 3.2**

If the index reduction algorithm is used on an n-variable square DAE, with leading matrix and algebraic term both being polynomials of degree k, and if the differential index is \( \nu \geq 1 \), the degree of the computed index 0 DAE is bounded by

\[
2^{n+\nu-2} k - \nu
\]

This bound is tight for index 1 problems, and off by k for index 2. For higher indices, it is the limit in the sense

\[
\frac{\text{true limit}}{2^{n+\nu-2} k - \nu} \rightarrow 1, \ n \rightarrow \infty
\]

**Proof:** The proof is given in section A.1

**Remark 3.1.** Regarding the proof in section A.1 note that the amount of overestimation is not affected by the number of elimination steps preceding the final differentiation, since there is no associated excursion, only a lowering by one. This means that the worst cases for a given \( \nu \) and n will always occur when the last two differentiations involve only the last equation. This makes it possible to directly compute the true worst case for \( \nu = 1 \) and \( \nu = 2 \). For \( \nu = 3 \), the worst case is still cheap to find by exhaustive search, but this strategy quickly becomes costly when \( \nu \) is further increased.

**Remark 3.2.** In practice, it is not at all meaningful to talk of asymptotic correctness as \( n \) tends to infinity (even if infinity can be as close as 10), since the resulting polynomial degrees are of unmanageable orders.

As an example of the theorem and the remark on how to find the true worst case (in combination with an algorithm that actually computes without approximation), the bound
Table 3.1: Comparing the degree bounds given by the theorem with the worst case computed by exhaustive search. Note that the bound is off by $k$ when $\nu = 2$.

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>$k$</th>
<th>$n$</th>
<th>Bound</th>
<th>True</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>79</td>
<td>79</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>158</td>
<td>153</td>
<td>0.96</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>317</td>
<td>255</td>
<td>0.80</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>15</td>
<td>327677</td>
<td>325631</td>
<td>0.99</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>636</td>
<td>395</td>
<td>0.62</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>15</td>
<td>655356</td>
<td>628831</td>
<td>0.96</td>
</tr>
</tbody>
</table>

given in the theorem is compared with the true upper bound for some values of $\nu$, $k$, and $n$. The result is listed in table 3.1.

3.4 Consistent initialization

The importance of being able to find a point on the solution manifold of a DAE, which is in some sense close to a point suggested or guessed by a user, was explained section 2.2.6. In this section, this problem is addressed using the proposed seminumerical quasilinear shuffle algorithm. While existing approaches (see section 2.2.6) separate the structural analysis from the determination of initial conditions, we note that the structural analysis may depend on where the DAE is to be initialized. The interconnection can readily be seen in the seminumerical approach to index reduction, and a simple bootstrap approach can be used to handle it.

3.4.1 Motivating example

Before turning to discussing the relation between guessed initial conditions and algebraic constraints derived by the seminumerical quasilinear shuffle algorithm, we give an illustration to keep in mind in the sections below.

Example 3.5

Let us return to the mathematical pendulum in example 3.3,

$$\begin{cases}
x'' = \lambda x \\
y'' = \lambda y - g \\
1 = x^2 + y^2
\end{cases}$$

where $g = 10.0$, with guessed initial conditions given by

$x_{0,\text{guess}} : \{ x(0) = \cos(-0.5), \dot{x}(0) = 0, y(0) = \sin(-0.5), \dot{y}(0) = -0.1, \lambda(0) = 0 \}$
Running the seminumerical quasilinear shuffle algorithm at $x_{0,\text{guess}}$ produces the algebraic constraints

$$C_{x_{0,\text{guess}}} = \begin{cases} 
1 & x^2 + y^2 \\
0 & 2x \dot{x} + y \dot{y} \\
0 & 2x^2 \lambda + 2y (g - y \lambda) + 2\dot{y}^2 
\end{cases}$$

The residuals of these equations at $x_{0,\text{guess}}$ are

$$\begin{cases} 
0.0 \\
0.0959 \\
9.61 
\end{cases}$$

so either the algorithm simply failed to produce the correct algebraic constraints although $x_{0,\text{guess}}$ was consistent, or $x_{0,\text{guess}}$ is simply not consistent. Assuming the latter, we try to find another point by modifying the initial conditions for the 3 variables $\dot{x}$, $y$, and $\lambda$ to satisfy the 3 equations in $C_{x_{0,\text{guess}}}$. This yields

$$x_{0,\text{second}} : 
\{ x(0) = \cos(-0.5), \dot{x}(0) = -0.055, y(0) = -0.48, \dot{y}(0) = -0.1, \lambda(0) = -4.804 \}$$

(This point does satisfy $C_{x_{0,\text{guess}}}$. Solving the equations could be difficult, but in this case it was not.) At this point the algorithm produces another set of algebraic constraints:

$$C_{x_{0,\text{second}}} = \begin{cases} 
1 & x^2 + y^2 \\
0 & 2x \dot{x} + y \dot{y} \\
0 & 2x^2 \lambda + 2y (g - y \lambda) + 2\dot{x}^2 + 2\dot{y}^2 
\end{cases}$$

with residuals at $x_{0,\text{second}}$:

$$\begin{cases} 
0.0 \\
0.0 \\
0.0060 
\end{cases}$$

By modifying the same components of the initial conditions again, we obtain

$$x_{0,\text{final}} : 
\{ x(0) = \cos(-0.5), \dot{x}(0) = -0.055, y(0) = -0.48, \dot{y}(0) = -0.1, \lambda(0) = -4.807 \}$$

This point satisfies $C_{x_{0,\text{second}}}$, and generates the same algebraic constraints as $x_{0,\text{second}}$. Further, the algorithm encountered no non-trivial expressions which had to be assumed rewritable to zero, so the index reduction was performed without seminumerical decisions. Hence, the index reduction is locally valid, and the reduced equations provide a way to construct a solution starting at $x_{0,\text{final}}$. In other words, $x_{0,\text{final}}$ is consistent.

---

6 The implementation used here does not compute derivatives to make better zero tests or longevity estimates.
3.4.2 A bootstrap approach

A seminumerical shuffle algorithm maps any guessed initial conditions to a set of algebraic constraints. Under certain circumstances, including that the initial conditions are truly consistent, the set of algebraic constraints will give a local description of the solution manifold. Hence, truly consistent initial conditions will be consistent with the derived algebraic constraints, and our primary objective is to find points with this property. Of course, if a correct characterization of the solution manifold is available, finding consistent initial conditions is easy given a reasonable guess — there are several ways to search a point which minimizes some norm of the residuals of the algebraic constraints. If the minimization fails to find a point where all residuals are zero, the guess was simply not good enough, and an implementation may require a better guess form the user.

If the guessed initial conditions are not consistent with the derived algebraic constraints, the guess cannot be truly consistent either, and we are interested in finding a nearby point which is truly consistent. In hope that the derived algebraic constraints could be a correct characterization of the solution manifold, even though they were derived at an inconsistent point, the proposed action to take is to find a nearby point which satisfies the derived constraints.

What shall be considered nearby is often very application-specific. Variables may be of different types, defying a natural metric. Instead, if the solution manifold is characterized by \( m \) independent equations, a user may prefer to keep all but \( m \) variables constant, and adjust the remaining to make the residuals zero. This avoids in a natural way the need to produce an artificial metric.

No matter how nearby is defined, we may assume that the definition implies a mapping from the guessed point to a point which satisfies the derived algebraic constraints (or fails to find such a point, see the remark above). Noting that a guessed point of initial conditions is mapped to a set of algebraic constraints, which then maps the guessed point to a new point, we propose that this procedure be iterated until convergence or cycling is either detected or suspected.

3.4.3 Comment

The algebraic constraints produced by the proposed seminumerical quasilinear shuffle algorithm are a function of the original equations and a number of decisions (pivot selections and termination criteria) that depend on the point at which index reduction is performed. Since the number of index reduction steps before the algorithm gives up is bounded given the number of equations, and the number of pivot selections and termination criterion evaluations is bounded given the number of index reduction steps, the total number of decisions that depend on the point of index reduction is bounded (although the algorithm has to give up for some problems). Hence, any given original equations can only give rise to finitely many sets of algebraic constraints. The space of initial conditions (almost all of which are inconsistent) can thus be partitioned into finitely many regions according to what algebraic constraints the algorithm produces.
Defining the index of a DAE without assuming that certain ranks are constant in the neighborhood of solutions can be difficult, and motivate the use of so-called uniform and maximum indices, see Campbell and Gear [1995]. To the bootstrap approach above, constant ranks near solutions implies that the algorithm will produce the correct algebraic constraints if only the guessed initial conditions are close enough to the solution manifold. To see how the approach suffers if constant ranks near solutions are not granted, it suffices to note that even finding a point which generates constraints at level 0 which it satisfies can be hard. In other words, consistent initialization can then be hard even for problems of index 1.
Saying that a (seminumerical) quasilinear shuffle algorithm generalizes the shuffle algorithm for LTI DAE, implies that the LTI structure of the equations is maintained after each index reduction step. Hence, it makes sense to say that the LTI form is invariant under the quasilinear shuffle algorithm, and it is expected that the algorithm can be fruitfully tailored to take care of the structural information in any such invariant form.

In this chapter a class of forms ranging from quasilinear to LTI DAE is searched for forms that are invariant under the quasilinear shuffle algorithm, and it is suggested that this kind of survey be extended to a more complete mapping between index reduction algorithms and their invariant forms.

### 4.1 Invariant forms

One of the aims of this chapter is to shed some light on the question of what forms that, besides the LTI form, are invariant under the quasilinear shuffle algorithm. The feature of such forms is that it is expected that the algorithm can be tailored to take advantage of the structural information in the form. (For example, the shuffle algorithm for the LTI form is so simple it can even be implemented easily in MATLAB.) To this end, a class of candidate forms ranging from the LTI to the quasilinear form will be defined and searched for invariants.

Further, while the quasilinear form covers a very wide range of applications [Steinbrecher, 2006; Yang et al., 2005; Unger et al., 1995] (although presence of the quasilinear structure is not always recognized or emphasized), if our search reveals that simpler invariant forms are hard to find, this adds also a pragmatic argument for the development of theory and algorithms for the quasilinear form in its full generality.
4.1.1 Structures and their algorithms

The form (structure) of equations has similar implications for algorithms in mathematics as the choice of data structures has in computer science. The larger the flexibility in the representation, the more difficult it is to draw mathematical conclusions or develop efficient operations on data. Typically, methods are developed with a particular structure in mind. In mathematics, it may be required that equations can be written in a particular way, or that other mathematical objects satisfy some more or less abstract mathematical properties. In computer science, it may be required that a data structure conform to an interface or other abstract specification, which tells how the structure can be manipulated and how data can be accessed (typically together with bounds on the computational cost of performing the operations). For examples from computer science, see for instance Lewis [1997].

In this context, we are more concerned with structures in the sense of the forms which equations can conform to. Thinking of the equations as models, section 2.1.4 gives some background on the importance of the choice of structure, and also some examples. Turning to DAE models and algorithms for index reduction, the recursive nature of various index definitions imply that algorithms that can handle equations with high indices will consist of a loop which returns to equations with the given structure each time the index has been lowered by 1. We shall refer to such a form of equations as invariant (with respect to the iteration body of the algorithm). Clearly, it can be conceived how any given form of equations can give rise to several algorithms for analyzing and solving it.

However, the form of equations is not the only type of structure of interest here. The data structures used to represent equations are very important for what algorithms we may come up with. An early example of this was given in section 3.1.1, where it was remarked that the fully nonlinear structure algorithm could be used for index reduction if only there was a computer implementation of implicit functions (admittedly, this is not only a matter of data structures, but the problem is similar). This matter of representation is also raised in section 3.3.2, noting that various representations of polynomials may lead to different algorithm complexity.

4.1.2 An algorithm and its structures

Having discussed how data structures lead to algorithms in the previous section, we now turn to what is of greater importance in this chapter; namely that a given algorithm can be applied to all the structures that conform to the abstract specification the algorithm was developed for. Further, and as was mentioned above, when an algorithm is applied to a less general structure than it was originally developed for, it may be possible to tailor it to take advantage of the additional structural information. In fact, this is often precisely what making simplifying assumptions is all about; the simplifying assumptions can be seen as adding additional structural information to a problem, although simplifying assumption is a better name for something which is motivated by ease of analysis and/or implementation.
4.2 A class of candidate structures

When an algorithm developed for a general form of equations is specialized for less general forms of equations, one of the important parts of the tailoring is to choose the appropriate data structure — typically, the data structures used by the more general algorithm will have time and space overheads that can be reduced given the additional structural information. In the sequel, the choice of representation will generally not be emphasized, but it is simply assumed that any additional structural information can be utilized to derive improved implementations.

Recall that we are concerned with index reduction algorithms such that some form of equations is invariant under the index reduction step (iteration body) of the algorithm. To investigate how an index reduction algorithm can be tailored to less general structures, we are thus led to search less general forms that are still invariant under the given index reduction step. Clearly, it is meaningful to speak of the invariant forms of an index reduction step (or the induced iterative index reduction algorithm).

4.2 A class of candidate structures

The aim of this section is to define a class of candidate forms. In order to be able to obtain a comprehensive result from the forthcoming investigation of this set, it is necessary that the set can be defined concisely. For instance, an enumeration of all the elements in the set would not be a concise definition. Rather, the class should be defined by a small set of generating rules. Given that the most general form in the class shall be the quasilinear form \( \frac{\partial}{\partial x} \), the rules concern the structure of the two components \( E \) and \( A \), the former evaluating to a matrix and the latter to a vector.

Since the quasilinear form allows the two components to be arbitrary nonlinear functions (we do not consider requirements on differentiability or solvability in this chapter), it is natural to include the following generating rule:

\[ \text{GR1} \quad \text{The class shall contain analogous variations for the two components } E \text{ and } A. \text{ This said, all other generating rules shall be formulated so that they apply to either of the two components.} \]

The following rule includes the quasilinear form in the class (by the last option)\(^1\):

\[ \text{GR2} \quad \text{A component, } G, \text{ evaluated at } (x, t) \text{ may depend jointly on } x \text{ and } t \text{ in one of the following ways:} \]

\[ \begin{align*}
\text{– Dependence on } x \text{ only, so it can be written } G_x(x). \\
\text{– Dependence on } t \text{ only via a function } v \text{ where } v \text{ may be of smaller size than } G_t, \text{ and } G(x, t) \text{ being linear in } v(t). \text{ The } v \text{ after the index reduction step does not have to be the same } v \text{ that was the input to the index reduction, but may only be extended with the derivative of the components of the incoming } v. \text{ That is, this form can be written } G_v(x) v(t). \\end{align*} \]

\(^1\)This rule obviously excludes many conceivable forms. For instance, joint dependence that can be written as a product of two functions instead of a sum, is not included.
– It can be written as a sum of the two forms above, \( G_x(x) + G_y(x) v(t) \).

– No particular structure, simply written \( G(x, t) \).

This said, the remaining generating rules shall define the possible structures of \( G_x \) and \( G_y \). For analogous treatment of the two, they shall both be referred to as \( G_z \) in the following rules.

Note that \( G_y(x) \) is a matrix with 3 indices when its product with \( v(t) \) evaluates to a term in \( E(x, t) \). That is, it has one element for every selection of: first scalar equation, then component of \( x \) (column in \( E \)), and finally component of \( v \).

Turning to the generating rule for the structure of \( G_z \), it is chosen such that the LTI form is included (this requires both constant and linear functions), but to make the story a bit more exciting and to decrease the complexity gap up to the quasilinear form, a few more forms are included:

**GR3** The structures of \( G_z(x) \) considered are the following:

– A constant function, written \( G_0^z \).

– A linear function, written \( G_z x \).

– A polynomial of degree at most \( k \), written \( G_k^z(x) \).

– A polynomial of any degree, written \( G_\infty^z(x) \).

– No particular structure, simply written \( G_z(x) \).

These are all the rules. They generate \( (4 + 4 + 4^2 + 1)^2 = 625 \) elements (the squares due to independent selection of two things), but this number is not very much related to the effort needed to search the class for invariant forms. More important than the number of elements in the class are the simple rules that define it.

### 4.3 Analysis of candidate forms

To structure the search for invariant forms, a partly constructive approach is taken. The search is conducted by considering each of the \( 4 + 4 + 4^2 + 1 \) forms of \( E \) to see what forms of \( A \) they can be matched with. As was mentioned in connection to the description of the index reduction algorithm in section 3.2.1, the algorithm is defined such that the row reduction step per se does not lead out of the form under consideration. Compare this with the index reduction used in Visconti [1999], where the upper triangular set of equations resulting from the row reduction is kept for the next iteration. Then one would also have to analyze the form of the upper triangular part of the leading matrix to see that it matches the form under consideration.

Each step in the row reduction process manipulates the expressions of \( E \) by taking linear combinations with coefficients also from \( E \). Hence, at an arbitrary step of the row reduction process, it can be written as a sum of the two forms above, \( G_y(x) + G_\infty^z(x) v(t) \).

\[ G(x, t) \]

\[ G_\infty^z(x) \]

\[ G_0^z \]

\[ G_z x \]

\[ G_k^z(x) \]

\[ G_\infty^z(x) \]

\[ G_z(x) \]

\[ (4 + 4 + 4^2 + 1)^2 = 625 \]

2 The notation is somewhat unfortunate since it would also be the natural notation for polynomials of infinite degree, such as infinite series expansions.
reduction, the intermediate leading matrix will have elements that are polynomials in the original expressions of $E$. Since each step in the row reduction process also manipulates the elements of the intermediate $A$ by taking linear combinations with coefficients from the intermediate $E$, it can be concluded that the algebraic constraints obtained at the end of the row reduction are linear combinations of the original expressions of $A$ with coefficients being polynomials of the original expressions of $E$. This simple observation is the core of the analysis. The rest is just to observe what happens when the derived algebraic expressions are differentiated with respect to time.

The rest of this section is structured as a systematic search for invariant forms among all of the candidates. Where invariant forms are found, this will be presented as an observation. Hence, the forms that are not invariant are exactly those not mentioned by any of the observations.

### 4.3.1 Leading matrix is independent of time

In case $E(x, t)$ is restricted to the general form $E_x(x)$ in GR3, it is required that differentiation with respect to time of the derived constraints does not yield $t$-dependent coefficients in front of $x'$. This would, in general, be the case if $A(x, t)$ would contain terms that depend on both $x$ and $t$. Since the algebraic constraints are linear combinations with coefficients that may depend on $x$ (in any nonlinear fashion), it leads to the conclusion that $A(x, t)$ may not depend on $t$. Next, it must be investigated which of the possible further restrictions of $A_x$ that yield invariant forms. However, since time enters the derived algebraic constraints only through the variables $x$, the derivative with respect to time produces only algebraic terms that are zero. Now, zeros match all of the candidate forms, leading to the following result:

**Observation 1.** The time-invariant (autonomous) restriction of the quasilinear form,

$$E_x(x(t)) x'(t) + A_x(x(t)) = 0$$

is invariant under the index reduction algorithm. Further, but not very interestingly, any restrictions of $A_x$ generates additional invariant forms.

The case of a constant function, $E^0_x$ requires that the terms in the derived algebraic equations that depend on $x$ be linear in $x$ and independent of $t$. This will be the case if and only if the terms in $A(x, t)$ that depend on $x$ be linear in $x$ and independent of $t$. This implies the structure $A(x, t) = A_x x + A_v v(t)$. If there would be no dependence on $x$ at all there would be no solution to the equations since the output of the index reduction would have a leading matrix containing only zeros except for the selected independent rows. No dependence on $x$ in $A(x, t)$ thus yields an irreducible form. Clearly, if there would be no dependence on $t$, that is $A_v = 0$, this yields another invariant. Conclusion:

**Observation 2.** The linear time-invariant restriction of the quasilinear form,

$$E_x x'(t) + A_x x(t) + A_v v(t) = 0$$

(4.2)
is invariant under the index reduction algorithm. Further, the subset of all reducible candidate forms where $E$ has this form has only one more invariant element, being

$$E_x x'(t) + A_x x(t) \overset{!}{=} 0$$

(4.3)

The case of $E(x, t)$ being in the form $E_x x$ is not part of any invariant form since forming polynomials with these expressions and then differentiating will generally lead to polynomials of higher degree in front of $x'$.

A similar argument rules out the case of $E(x, t)$ being polynomial with a predetermined bound on the degree.

It remains to consider the case of arbitrary polynomials of finite degree. The analysis of this case is similar to the constant coefficient case, but in this case one cannot permit dependence on $t$ in $A(x, t)$ at all:

Observation 3. The polynomial time-invariant restriction of the quasilinear form,

$$E_x^\infty (x(t)) x'(t) + A_x^\infty (x(t)) \overset{!}{=} 0$$

(4.4)

is invariant under the index reduction algorithm. Further, the subset of all candidate forms where $E$ has this form is obtained by taking all possible restrictions of $A_x^\infty$.

\begin{flushright}
\Box
\end{flushright}

4.3.2 Leading matrix depends on time via driving function

There are no invariant forms to discover in this section. To see this, note that a polynomial in the expressions of $E_v(x(t), v(t))$ typically contains expressions such as $v_2(t)^2$. This requires the full nonlinear form of $A$ in order to cater for the derivative. In turn, this would require the full nonlinear form of $E$, which means that any form of this kind is not invariant.

4.3.3 Leading matrix is general nonlinear

Forming polynomials of the expressions in $E(x, t)$ and differentiating with respect to time requires the general quasilinear form in order to cater for the derivative. This is all there is to the analysis in this section:

Observation 4. The quasilinear form,

$$E(x(t), t) x'(t) + A(x(t), t) \overset{!}{=} 0$$

(4.5)

is the only which is invariant among the considered forms where $E$ is general nonlinear.

\begin{flushright}
\Box
\end{flushright}
4.3 Analysis of candidate forms

4.3.4 Example

To give a little more life to the arguments in the previous section, the index reduction algorithm is here applied to one of the forms claimed not to be invariant. Also note that in connection to the description of the index reduction algorithm in section 3.2.1, another example was given. That example was actually an illustration of the fact that the general quasilinear form is invariant.

Example 4.1

Now consider the form (recall that one may think of this simply as the functions being linear)

\[ E x(t) x'(t) + A_x x(t) + A_v x(t) v(t) \doteq 0 \]  \hspace{1cm} (4.6)

instantiated as

\[
\begin{pmatrix}
  x_2(t) & 0 \\
  0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
  x'_1(t) \\
  x'_2(t) \\
\end{pmatrix}
+ 
\begin{pmatrix}
  x_1(t) v_1(t) \\
  x_2(t) \\
\end{pmatrix}
\doteq 0
\]

Choosing to keep the first equation and reducing results in

\[
\begin{pmatrix}
  x_2(t) & 0 \\
  0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
  x'_1(t) \\
  x'_2(t) \\
\end{pmatrix}
+ 
\begin{pmatrix}
  x_1(t) v_1(t) \\
  x_2(t)^2 - x_1(t) v_1(t) \\
\end{pmatrix}
\doteq 0
\]

Differentiating yields

\[
\begin{pmatrix}
  x_2(t) & 0 \\
  -v_1(t) & 2 x_2(t) \\
\end{pmatrix}
\begin{pmatrix}
  x'_1(t) \\
  x'_2(t) \\
\end{pmatrix}
+ 
\begin{pmatrix}
  x_1(t) v_1(t) \\
  -x_1(t) v'_1(t) \\
\end{pmatrix}
\doteq 0
\]

which is not in the form (4.6) as \( v_1(t) \) is not allowed in the leading matrix. However, the algebraic term is in the form of (4.6).

Note that by extending this example with the variable \( x_3 \) and, for instance, the equation

\[ x_3(t) x_2'(t) + x_1(t) \doteq 0 \]

the same index reduction step could be used, but the resulting leading matrix would still be singular. After reducing the derived equation, the following algebraic constraint is revealed:

\[ x_1(t) x_2(t) x_3(t) v_1(t)^2 - x_1(t) x_2(t)^2 (2 x_2(t) + x_3(t) v_1'(t)) \doteq 0 \]

This depends on \( t \) in such a way that differentiation will lead to a leading row which is not linear in \( x \) (it even depends on \( v \)), and an algebraic term that, although affine in \( v \), has a coefficient in front of \( v \) that is no longer linear in \( x \) (as in (4.6)).

\[^3\text{Note that this equation is an intermediate result of the index reduction step, and that as such, we are not concerned with its form.}\]
It is not difficult to construct examples of even higher index in the form (4.6), which after repeated index reduction has an algebraic term that is no longer linear in $v$, and hence only match the most general form (1.1). Thus, it is a property of the form (4.6) that if the index is sufficiently high, index reduction will, in general, eventually produce equations in the form (1.1).

### 4.4 Discussion

All invariant forms found in the class of candidate forms under consideration, were forms whose invariance — under this or similar types of index reduction algorithms — have already been used in the literature. The special methods that apply to polynomial equations have their own advantages and disadvantages, the main advantage being that there is plenty of theory on how to reduce the equations into forms that display useful properties or allow index reduction. The main disadvantage is that these methods are very expensive computationwise. However, it is beyond the scope of this thesis to compare quasilinear shuffle algorithms with other algorithms. Instead, the expectation that any invariant form would allow for a fruitful tailoring of the algorithm should be commented upon. Polynomials can be represented by properly structured coefficients, which means that the algorithm applied to polynomials can be implemented as a computation on such coefficient structures. Further, differentiation of polynomials is also easily expressed in terms of these structures. Compared to a fullblown implementation for nonlinear functions, the lightness of such a representation and ease of differentiation would surely lead to improved computation time. In addition, the algorithm would be relatively easy to implement even without a computer algebra foundation to start from.

The invariant forms with constant leading matrix are fairly well understood and efficient implementations for index reduction and numerical solution exist.

The two invariant forms allowing for general nonlinear dependence of $E(x, t)$ on $x$ are distinguished by $t$ being present or not in both parts of the quasilinear form. In other words, one of these forms is the general quasilinear form, the other being its autonomous counterpart. However, the autonomous form can cater for the general form by including the equation

$$s'(t) = 1$$

with initial condition $s(t_0) = t_0$, and then using $s$ in place of $t$ in the equations. Hence, an algorithm that can handle the autonomous form is just as powerful as an algorithm for the general form, and therefore it is expected that it makes little sense to discuss the differences between the two. On the other hand, the $s$-trick is possible due to the form being general enough to allow for arbitrary expressions in the states. Compare, for instance, the case of the polynomial $A_\infty$. Using this form one would have to resort to polynomial approximation of both the driving functions and the way such functions enter the equations. This is a rather limited and inefficient kind of time-dependency, however,

---

4 We stress again that we consider analysis of small perturbations a missing piece, motivating our efforts in chapters 5 and 6.
and means that in the considered class of candidate forms, one has to go all the way to the quasilinear form (autonomous or not) to find an element which is more expressive than the LTI form and at the same time in a reasonable way allows for explicit dependency on time in the equations.

### 4.4.1 Remarks on the example

Example 4.1 showed how a quasilinear DAE that was not in any of the invariant forms by repeated index reduction would lead to the most general form. Further, other candidate forms were visited during the index reduction process. This sort of information about the various forms can be conservatively captured by the graph describing the partial ordering of all candidate forms. That is, the candidate forms can be placed in a graph, where there’s an edge from form $a$ to form $b$ if index reduction of $a$ in general leads to a form which match $b$ but non of the possible restrictions of $b$ (then $a$ is considered a predecessor of $b$, which means that $b$ is invariant if and only if it is a maximal element).

This graph can be used if modeling in a certain application domain is known to lead to one of the non-invariant forms, $a$, with a known bound, $\tilde{\nu}$ on the required number of index reduction steps. It then suffices to have (efficient) implementations for the form $a$ and those that can be reached from $a$ in $\tilde{\nu} - 1$ steps. Note though, that even if $b$ is a predecessor of $a$, there may be equations in the form $a$ which are not the result of performing index reduction on any equations in the form $b$. Hence, if $\tilde{\nu} - 1 > 1$, this number of steps may be overestimating how far from $a$ index reduction may lead.

### 4.4.2 Extensions

A study similar to that in section 4.3 could be conducted based on differential algebra methods (Gröbner bases, Ritt’s algorithm, and the like) applied to polynomial DAE. However, due to the fact that these methods have very high computational complexity, the outcome of such a study would not have as immediate value to implementations designed for real-world examples.

It would be interesting, though, to see the outcome of a study similar to this one, but considering a much larger class of candidate systems. Inspired by the polynomial structure where driving functions would be approximated by polynomials, it would be interesting to search structures which would match other ways of approximating driving signals. Would it be meaningful to investigate periodic DAE, searching a structure that can take advantage of the Fourier series representation of driving signals? It would also be interesting to consider both fraction-producing and more elaborate index reduction methods. If such a method would be designed for a form not in the class of candidate forms considered in this paper, this would require extension of this class in a natural way.

Yet a better option for those who are skilled in mathematics would be to see if this kind of survey can be conducted in a more abstract framework, not limiting the results to particular index reduction algorithms. The works by Reinboldt and others (for instance, [Rheinboldt, 1984]) seem to provide the needed foundation.
In chapter 3 an algorithm for index reduction of quasilinear DAE was introduced. At several occasions, we turned the light on the small perturbations which we need to understand in order in order to give the proposed algorithm a theoretical foundation. In this chapter we take a closer look at that problem.

Before we begin, let us remark that the issue with perturbations in DAE has been considered previously in [Mattheij and Wijckmans 1998]. While their setup is different to ours, we share many of their observations. However, their way of approaching the issue — even their way of formulating the problem — differs from ours. Consequently, their results are not immediately competing with ours.

**Notation.** Note how the presence or absence of a decimal point in numbers are used to distinguish exact integers from non-exact numbers. With this notation 0 is a structural zero, while 0. is not.

### 5.1 Motivation

In this section, we try to motivate the study of singular perturbations in DAE, both by showing connections to the previously described seminumerical quasilinear shuffle algorithms, and by showing why we think this in an interesting topic by itself.
5.1.1 A linear time-invariant example

The example of this section is meant to show the need for a seminumerical algorithm like the one proposed; it will not show how the proposed algorithm is used to solve the problem.

Example 5.1

Starting from an index 0 DAE in two variables,

\[
\begin{pmatrix}
1.2 & 4.9 \\
1.0 & 2.1 \\
0 & 0
\end{pmatrix} x'(t) + \begin{pmatrix}
3.2 & 0 \\
2.1 & 0 \\
0 & 1
\end{pmatrix} x(t) = 0
\]

an index 1 DAE in three variables is formed by making a copy of the second variable. In the leading matrix, the second variable is replaced to 70% by the new variable.

\[
\begin{pmatrix}
1.21 & 7 \cdot 0.49 \\
1.09 & 0.21 \\
0 & 1
\end{pmatrix} x'(t) + \begin{pmatrix}
3.2 & 0 \\
2.1 & 0 \\
0 & 1
\end{pmatrix} x(t) = 0
\]

(5.1)

To perform index reduction of this DAE, it suffices to note that the first two rows of the leading matrix are independent, and as the last row is completely zeroed, the last equation is differentiated. This leads to

\[
\begin{pmatrix}
1.21 & 7 \cdot 0.49 \\
1.09 & 0.21 \\
0 & 1
\end{pmatrix} x'(t) + \begin{pmatrix}
3.2 & 0 \\
2.1 & 0 \\
0 & 1
\end{pmatrix} x(t) = 0
\]

Now, instead of performing the index reduction on (5.1) directly, begin by applying a well-conditioned change of equations given by the matrix:

\[
T := 4 \cdot \begin{pmatrix}
2. & -9. & 0. \\
8. & -5. & 3. \\
1. & -5. & 7.
\end{pmatrix}^{-1}
\]

It is natural to expect that this should not make a big difference to the difficulty in solving the DAE via reduction to index 0, but when the computation is performed on a computer, the picture is not quite as clear. The new DAE has the matrices \( T^{-1} E \) and \( T^{-1} A \). By computing a QR factorization (using standard computer software) of the leading matrix, a structurally upper triangular leading matrix was obtained together with an orthogonal matrix \( Q_1 \) associated with this form. The corresponding matrix of the algebraic term is computed by multiplication by \( Q_1 \) from the left. This leads to

\[
\begin{pmatrix}
-0.62 & -0.95 & -2.2 \\
0 & 0.62 & 1.4 \\
0 & 0 & 3.4 \cdot 10^{-16}
\end{pmatrix} x'(t) + \begin{pmatrix}
-1.6 & -0.53 & -0.41 \\
0.51 & 0.56 & -0.048 \\
-7.2 \cdot 10^{-17} & 0.46 & -0.46
\end{pmatrix} x(t) = 0
\]

Although looking like an implicit ODE, this view is unacceptable for two reasons. First, the system of equations is extremely stiff. (Even worse, the stiff mode happens to be
unstable this time, not at all like the original system.) Second, considering numerical precision in hardware, it would not make sense to compute a solution that depends so critically on a coefficient that is not distinctly non-zero.

The ad hoc solution to the problem in the example is to replace the small coefficient in the leading matrix by zero, and then proceed as usual, but suppose ad hoc is not good enough. How can one then determine if \(3.4 \cdot 10^{-16}\) is tiny, or just looks tiny due to equation and variable scalings? What is the theoretical excuse for the replacement of small numbers by zeros? What assumptions have to be made?

### 5.1.2 Inspiring example

In this section, an example suggesting that the ill-posedness may be possible to deal with is given. The assumptions made here are chosen theoretically insufficient on purpose — the point is that making even the simplest assumptions seems to solve the problem.

**Example 5.2**

Having equations modelling a two-timescale system (recall section [2.3](#)) where the slow dynamics is known to be stable, we now decide that unstable fast dynamics is unreasonable for the system at hand. In terms of assumptions, we assume that the fast dynamics of the system are stable, and claim it natural to make this assumption for the system at hand. We then generate random perturbations in the equation coefficients that we need to zero, discarding any instantiations of the equations that disagree with our assumption, and use standard software to solve the remaining instantiations. Two algebraic terms were used, given by selecting \(\delta\) from \(\{1, 10^{-2}\}\) in the pattern

\[
A = \begin{pmatrix}
0.29 & 0.17 & 0.046 \\
0.34 & \delta & 0.66 \\
0.87 & \delta & 0.14
\end{pmatrix}
\]

and then scaling the rows according to \(P_1\) (see section [5.2.1](#)). Let the four numbers \((?_{11}, ?_{12}, ?_{21}, ?_{22})\) be generated by first taking four independent samples from a uniform distribution centered at 0, and then scaling to make the biggest number have the magnitude 1. Then \(E\) was generated according to the pattern

\[
E := \begin{pmatrix}
1 & 1 & 1 \\
0 & \epsilon ?_{11} & \epsilon ?_{12} \\
0 & \epsilon ?_{21} & \epsilon ?_{22}
\end{pmatrix}
\]

for some chosen \(\epsilon > 0\). The example is chosen such that \(\epsilon = 0\) yields a stable slow system. Thus the perturbations of interest are those that make all modes of the stiff system stable. The initial conditions are chosen with \(x_1(0) = 1\) and consistent with \(\epsilon = 0\).

Simulation results are shown in figure [5.1](#). By choosing a threshold for \(\epsilon\) based on visual appearance, the threshold can be related to \(\delta\). Finding that \(1 \cdot 10^{-2}\) and \(1 \cdot 10^{-5}\) could be
Figure 5.1: Solutions for $x_1$ obtained by generating 50 random perturbations of given magnitudes. Details are given in the text. Left: $A$ defined by $\delta = 1$. Right: $A$ defined by $\delta = 10^{-2}$. Top: $\epsilon = 1 \cdot 10^{-1}$. Middle: $\epsilon = 1 \cdot 10^{-3}$. Bottom: $\epsilon = 1 \cdot 10^{-5}$.

reasonable choices for $\delta$ being 1 and $10^{-2}$, respectively, it is tempting to conclude that it would be wise to base the scaling of the last two rows on $A_{22}$ alone.

5.1.3 Application to quasilinear shuffling

In theory, index reduction of equations in the quasilinear form is simple: Manipulate the equations using invertible row operations so that the leading matrix becomes separated into one block which is completely zeroed, and one block with independent rows. Differentiate the discovered algebraic equations, and repeat until the leading matrix gets full rank. As examples of the in-theory ramifications of this description, consider the following list:

- It may be difficult to perform the row reduction in a numerically well-conditioned way.
- The produced equations could involve very big expressions.
- Testing whether an expression is zero is highly non-trivial.

The forthcoming discussion applies to the last of these ramifications. Typical examples in the literature have leading matrices whose rank is determined solely by a zero-pattern. For instance, if some variable does not appear differentiated in any equation, the corresponding column of the leading matrix will be zero. It is then easy to see that this column
will remain zero after arbitrarily complex row operations, so if the operations are chosen to create structural zeros in the other columns at some row, it will follow that the whole row is structurally zero. Thus an algebraic equation is revealed, and when differentiating this equation, the presence of variables in the equation determines the zero-pattern of the newly created row in the leading matrix, and so the index reduction may be continued.

Now, recall how the zero-pattern was lost by a seemingly harmless transform of the equations in example 5.1. Another situation when linear dependence between rows in the leading matrix are not visible in a zero-pattern, is when a user happens to write down equations that are dependent up to available accuracy. It must be emphasized here that available accuracy is often not a mere question of floating point number representation in numerical hardware (as in our example), but a consequence of uncertainties in estimated model parameters.

In chapter 3, it was proposed that a numerical approach is taken to zero-testing whenever tracking of structural zeros does not hold the answer, where an expression is taken for being (re-writable to) zero if it evaluates to zero at some trial point. Clearly, a tolerance will have to be used in this test, and showing that a meaningful threshold even exists is the main topic of this note. The analysis below will be restricted to linear time-invariant (LTI, hereafter) DAE, and then the choice of trial point will be of no consequence.

5.1.4 A missing piece in singular perturbation of ODE

Our final attempt of convincing the reader that our topic is interesting is to remark that singular, unstructured, perturbations are not only a delicate problem in the world of DAE. At least, the author knows of no result for analyzing the implications of having unstructured perturbations even when the leading matrix of a quasilinear DAE is only near-singular, that is, the equations are an implicit ODE.

5.2 Solution by assumption

By the title of this section, we wish to emphasize how we think the question of understanding the singular perturbations can be handled. Considering example 5.1 it is evident that no convergence results can follow without making some assumptions regarding the structure of the perturbations. We take this further in our first attempts at this problem; we are content dealing with it, not by listing a set of reasonable assumptions to make, but by adding assumptions as they are needed in the analysis.

5.2.1 LTI algorithm

This section will detail the index reduction algorithm proposed in chapter 3 tailored to LTI DAE. The restriction to LTI DAE is for the sake of the analysis below. Extending the
index reduction algorithm as such to quasilinear systems is almost immediate; it is the analysis of it that becomes more difficult.

Consider the square DAE

$$E x'(t) + A x(t) \frac{1}{\epsilon} = 0$$

where $E$ and $A$ are no longer considered functions, but constant matrices. To ease the notation, it is assumed that the variables are ordered to suit the presentation.

The algorithm (described below) applies row operations represented by the matrix $K_0$ to the equations, maintaining a form,

$$
\begin{align*}
K_0 E &= \begin{pmatrix} E_{11} & E_{12} \\ 0 & \epsilon E_{22} \end{pmatrix} \\
K_0 A &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}
\end{align*}
$$

where the scalar $\epsilon \geq 0$ is chosen such that $E_{22}$ has element-wise max-norm 1. The matrix $E_{11}$ is upper triangular and non-singular.

As long as $E_{22}$ contains elements that are distinct non-zero (taking numerical precision into account), the algorithm proceeds with the row reduction until either $E_{11}$ spans all of the leading matrix, or the small size of $\epsilon$ is suspected to indicate a stiffness that needs to be taken care of by model reduction. In the first case, index reduction is complete since the underlying ODE has been revealed. The case $\epsilon = 0$ is standard in theoretically oriented contexts; when this happens the purely algebraic equations are differentiated (shuffled), which replaces the zeroed block in the leading matrix by the corresponding rows from the matrix in the algebraic term. The differential index has then been lowered by 1, and the index reduction process can start over from the beginning. The last case, and when no elements are distinct non-zero, will be discussed shortly, with the goal of approximating $\epsilon$ by zero. Values for $\epsilon$ admitting such approximation will be referred to as tiny from here on.

Besides the assumptions that are the goal of this chapter, determining whether $\epsilon$ is tiny cannot be done looking at $\epsilon$ alone. Clearly, the threshold must be a function of the involved matrices (with the exception of $E_{22}$). In particular, note that $\epsilon$ can be made arbitrarily small by scaling of equations or variables. Although answering the tinyness-question is outside the scope of this chapter, equation and variable scalings need to be specified to some extent for the forthcoming analysis.

In order to define equation and variable scaling without too much inconvenience, one of the assumptions needed in the analysis below is made now:

**A1** It is assumed that $A_{22}$ is well-conditioned whenever tinyness of $\epsilon$ is investigated.

This assumption can be relaxed, but that requires a more sophisticated algorithm that will not be detailed here. In particular, this assumption implies that the rows in $(A_{21}, A_{22})$ are non-zero, and it is possible to augment the row reduction scheme with row scalings such that the following property holds:
P1 The norm of the smallest row in \((A_{21} \ A_{22})\) equals the norm of the biggest row in \((A_{11} \ A_{12})\), unless zero.

Note that the act of ensuring this property may effectively change the value of \(\epsilon\).

If the row reduction stopped because \(E_{22}\) contained only elements that are not distinctively non-zero, the problem is ill-posed unless \(\epsilon\) may be considered tiny. This relates the problem of lack of numerical precision to the problem of stiff systems. In view of this and the close resemblance to the singular perturbation framework, it is natural to say that row reduction stopped due to singular uncertainty.

Having to decide whether \(\epsilon > 0\) may be considered tiny, the algorithm is faced with several alternatives:

- Ignorantly replace the small numbers by structural zeros (thereby making \(\epsilon = 0\)), and proceed as above.
- Reject the problem with a message of the kind not implemented.
- Make an informed decision whether it is right to replace the small numbers by structural zeros. If so, proceed as above, and if not, reject the problem as hopelessly ill-conditioned.

This thesis pursues the last of these.

Although not the issue here, one more termination condition should be mentioned for completeness. Besides the error situations and the case of successful completion when the leading matrix has obtained full rank, the algorithm terminates when the differentiation step does not add rank to the leading matrix. (In this case the solution is not well defined.)

Before ending this section, the choice of row reduction algorithm will be discussed briefly. There are several standard algorithms that maintain the upper triangular form of \(E_{11}\) with zeros below, for instance Gaussian elimination and QR factorization. Any such algorithm can be combined with the equation and variable scalings prescribed here, although that may break good numerical properties. If extension to quasilinear DAE would not be a concern, using QR factorization methods would be a good choice for their good numerical properties (trying to ensure that the solution to the row reduced equations is close to the solution of the original equations), but with the quasilinear form in mind, Gaussian elimination is proposed.

### 5.2.2 Making sense of an ill-posed problem

Consider the form (5.2), when the algorithm is faced with the question whether \(\epsilon = 0\) is a good approximation (example 5.1 showed that uncertainties generally make this question ill-posed). The analysis of this chapter will produce additional assumptions (the first one being A11) that will make the approximation errors \(O(\epsilon)\). Although not providing a way to compute a threshold, this shows that it is meaningful to speak of \(\epsilon\) small enough. There will be assumptions both regarding robust features of the equations and those reflecting the intended meaning of the equations. Thus the assumptions will not be possible to verify
in a purely mathematical setting, but only when the user has some idea of what object the equations are a model of, and the user also has some knowledge about how the underlying object could possibly behave. Let these assumptions (to be determined) be referred to as the assumptions about the underlying object.

Introduce an \( \epsilon \)-uncertainty in the non-structural elements of \( E_{22} \), so that \( E_{22} = 0 \) is considered a possible model for the underlying object. Then the DAE is really a family of DAE, obtained by considering the possible perturbations while excluding any equations that do not conform to the assumptions about the underlying object.

It is clear that the proposed index reduction scheme is useless in a numeric setting if the solutions to the family of equations are not close. If it can be shown that the solutions approach a limit as the degree of uncertainty gets smaller, then taking any of the solutions as the solution to the original equations should be considered just as good. In particular, the solutions should approach that obtained by setting to zero any numbers that cannot be distinguished from zero under the given uncertainty.

5.2.3 Assumptions

Two additional assumptions about the underlying object are added from start.

A2 It is assumed that the given DAE is meant to be of differential index 1.

A3 It is assumed that for every very fast mode of the underlying ODE, the duration of the boundary layer is bounded by a common time \( t_1 \), which is no further into the future than that it is acceptable to have potentially large transient errors until then.

Admittedly, A3 is an odd bird, but the discussion of this is deferred until section 5.4.

For simplicity, the study is restricted to finite times. This way, it will not be necessary to assume stability of the slow model.

5.3 Analysis

The method used to understand the problem in this chapter reminds of the analysis in Kunkel and Mehrmann [2006, chapter 3], where equation and variable transforms are applied to a time-varying linear DAE to obtain a form where solvability and other questions are easily answered. However, it is a method which neither assumes nor makes approximations. The method of linearizing along a solution trajectory in Campbell [1995] comes with an analysis of the error caused by small errors in the initial conditions. This method, with its classic solvability conditions, is pushed further in Campbell and Griepentrog [1995] to make it amenable to numerical implementation. In contrast, the analysis carried out here is motivated by the uncertainties which are present in any equations which are an approximation of some unknown underlying dynamic behavior.
Notation. Unlike other places in this thesis, the prime sign is not used to denote the derivative of a function of one argument in this section. Instead, it is used to construct composed symbols. For example, the symbols $E$, $E'$, and $E''$ have no mathematical relation implied by the notation, but the notation is mnemonic.

5.3.1 Pointwise approximation

In this section, the analysis is concerned with perturbations that only differ in magnitude, not in direction. It will be shown that the deviations in the solution after a short boundary layer decay at the same rate as the magnitude of the perturbation.

The goal will be to rewrite the DAE in standard singular perturbation form, where the approximation in setting $\epsilon := 0$ is reasonably well understood, and then revert the rewriting process to see how the approximation relates to the original formulation. Rewriting of the equations will be performed in two ways, namely equation transforms (or equivalently, row operations on the matrices), and variable transforms (or equivalently, column operations on the matrices). Since setting $\epsilon := 0$ can be seen as a row operation on the leading matrix, this will interact with the other equation transforms when reverting the rewriting process. Hence, the equation transforms must be well conditioned in certain ways in order to end up not too far from the original equations. On the other hand, the variable transforms are unaffected by the row operation $\epsilon := 0$, and need not be well-conditioned in order for the reversed rewriting process to yield useful results.

The form (5.2) requires that $E_{22}$ have norm 1, but it may be singular. In particular, it may be singular due to structural properties (a variable not appearing in any equation, or some equation being non-differential), but it can also be singular due to numerical coincidence.

Since we are now concerned with a particular instance of the family, it makes sense to compute the QR factorization $Q_1 E_{22} = \begin{pmatrix} R_0^T & 0 \end{pmatrix}^T$. Extend the orthogonal matrix such that it can be applied to all equations;

$$\bar{Q}_1 K_0 E = \begin{pmatrix} E_{11} & E_{12} \\ 0 & \epsilon R_0 \end{pmatrix}$$

The corresponding matrix in the algebraic term,

$$\bar{Q}_1 K_0 A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21}' & A_{22}' \\ A_{31}' & A_{32}' \end{pmatrix}$$

must have full rank in the last block row, or the equations would not have a well-defined solution. However, the analysis benefits from the stronger $A_{11}'$ which ensures that (the square)

$$\begin{pmatrix} A_{22}' \\ A_{32}' \end{pmatrix} = Q_1 A_{22}$$

In other words, it will be shown that the deviations are $O_{E_{22}}(\epsilon)$ after a short boundary layer.
is well-conditioned in itself. Let $A'_{32} Q^T_2 = (0 \quad R_1)$ be another QR factorization. Then, the change of variables $x = K_1^T y$ with

$$
K_1^T = \begin{pmatrix}
I & 0 \\
-(A'_{22}^{-1} A'_{32}^{-1}) & Q^T_2
\end{pmatrix}
$$

is well defined (although the condition number may be large) and brings the equations in the form

$$
\begin{align*}
\bar{Q}_1 K_0 E K_1^T &= \begin{pmatrix}
E''_{11} & E''_{12} & E''_{13} \\
\epsilon E''_{21} & \epsilon E''_{22} & \epsilon E''_{23} \\
0 & 0 & 0
\end{pmatrix} \\
\bar{Q}_1 K_0 A K_1^T &= \begin{pmatrix}
A''_{11} & A''_{12} & A''_{13} \\
0 & A''_{22} & A''_{23} \\
0 & 0 & R_1
\end{pmatrix}
\end{align*}
$$

where

$$
E''_{11} = E_{11} - E_{12} A_{22}^{-1} A_{21}
$$

Since this is a DAE of index 1 (by $A^2$), differentiating the last equation must yield a full-rank leading matrix, and by the block triangular form of that matrix, it may be concluded that

$$
\begin{pmatrix}
E''_{11} & E''_{12} \\
\epsilon E''_{21} & \epsilon E''_{22}
\end{pmatrix}
$$

is non-singular. At the same time, $R_1 y_3 = 0$ implies $y_3 = 0$, and hence $y_3$ together with the last group of equations can be removed without changing the solution for $y_1$ and $y_2$. Partition $\bar{Q}_1$ and $K_1$ to reflect this. The equations given by the following matrices then determine $y_1$ and $y_2$.

$$
\begin{align*}
\bar{Q}_{1,1} K_0 E K_{1,1}^T &= \begin{pmatrix}
E''_{11} & E''_{12} \\
\epsilon E''_{21} & \epsilon E''_{22}
\end{pmatrix} \\
\bar{Q}_{1,1} K_0 A K_{1,1}^T &= \begin{pmatrix}
A''_{11} & A''_{12} \\
0 & A''_{22}
\end{pmatrix}
\end{align*}
$$

where the leading matrix is now known to be non-singular. Even better, since (5.6) shows that $E''_{11}$ does not depend on the perturbed $E_{22}$, it makes sense to make an assumption regarding it.

A4 It is assumed that $E_{11} - E_{12} A_{22}^{-1} A_{21}$ is well-conditioned with eigenvalues much bigger than $\epsilon$.

This way, the equations can soon be turned into the standard singular perturbation form (2.41).

By well-conditioned row operations represented by

$$
K_2 = \begin{pmatrix}
I \\
-\epsilon E''_{21} E''_{11}^{-1} I
\end{pmatrix}
$$
5.3 Analysis

and a well defined change of variables, \( y = K_3^T z \), with

\[
K_3^T = \begin{pmatrix}
I & -E''_{11} E''_{12} \\
0 & I
\end{pmatrix}
\]

we reach

\[
K_2 \bar{Q}_{1,1} K_0 E K_{1,1}^T K_3^T = \begin{pmatrix}
E''_{11} & 0 \\
0 & \epsilon E''_{22}
\end{pmatrix}
(5.7a)
\]

\[
K_2 \bar{Q}_{1,1} K_0 A K_{1,1}^T K_3^T = \begin{pmatrix}
A''_{11} & A''_{12} \\
A''_{21} & A''_{22}
\end{pmatrix}
(5.7b)
\]

where the leading matrix is still non-singular. Hence, the standard singular perturbation form is

\[
\begin{pmatrix}
\dot{z}_1(t) \\
\epsilon \dot{z}_2(t)
\end{pmatrix} = \begin{pmatrix}
E''_{11} A''_{11} & E''_{11} A''_{12} \\
E''_{22} A''_{21} & E''_{22} A''_{22}
\end{pmatrix} \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix}
(5.8)
\]

Here, \( A_3 \) grants that \( E''_{22} A''_{22} \) is Hurwitz so that theorem 2.2 applies. Hence, after the short boundary layer, the solution is close to the solution of

\[
\begin{pmatrix}
\dot{z}_1(t) \\
0
\end{pmatrix} = \begin{pmatrix}
E''_{11} A''_{11} & E''_{11} A''_{12} \\
E''_{22} A''_{21} & E''_{22} A''_{22}
\end{pmatrix} \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix}
\]

or equivalently, the DAE in \( z \) with matrices

\[
\begin{pmatrix}
I \\
0
\end{pmatrix} K_2 \bar{Q}_{1,1} K_0 E K_{1,1}^T K_3^T
\]

\[
K_2 \bar{Q}_{1,1} K_0 A K_{1,1}^T K_3^T
\]

Now, undoing the change of variables from \( y \) to \( z \), reintroducing \( y_3 \), and undoing the change of variables from \( x \) to \( y \), we find that \( x \) is approximately given by the DAE with matrices

\[
\begin{pmatrix}
I \\
0 \\
0 \\
0
\end{pmatrix} \begin{pmatrix}
I & 0 & 0 & 0 \\
0 & Q_{1,11} & Q_{1,12} & 0 \\
0 & Q_{1,21} & Q_{1,22} & 0 \\
-\epsilon E''_{21} E''_{11}^{-1} & I & 0 & 0
\end{pmatrix} K_0 E
\]

\[
\begin{pmatrix}
I \\
0 \\
0 \\
0
\end{pmatrix} \begin{pmatrix}
I & 0 & 0 & 0 \\
0 & Q_{1,11} & Q_{1,12} & 0 \\
0 & Q_{1,21} & Q_{1,22} & 0 \\
0 & 0 & 0 & I
\end{pmatrix} K_0 A
\]

Using (5.3) and (5.4), these matrices evaluate to

\[
\begin{pmatrix}
E_{11} & E_{12} \\
0 & 0 \\
0 & 0 \\
A'_{21} - \epsilon E''_{21} E''_{11}^{-1} A_{11} & A'_{22} - \epsilon E''_{21} E''_{11}^{-1} A_{12} & A_{12} & A_{12}
\end{pmatrix}
\]
It is seen that the row operation $K_2$ is not to be undone, only $Q_1$ is. This does not change the leading matrix, while the lower part of the matrix in the algebraic term can be written

$\begin{pmatrix} A_{21} & A_{22} \end{pmatrix} - \epsilon \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} Q_1^T E''_{21} E''_{11}^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix}$ \hspace{1cm} (5.9)

By $A_4$, the second of these terms is small and tends to zero at the rate of $\epsilon$. The problem of understanding a tiny perturbation in the leading matrix has thus been shifted to understanding a small perturbation in the matrix of the algebraic term. Since $A_{22}$ is well-conditioned by $A_1$, also the slightly perturbed matrix in the final form will be well-conditioned.

Since the DAE now readily can be written as an LTI ODE after elimination of $x_2$, and the associated matrix is like $E''_{11}$ (see $A_4$) but with a small perturbation, we are close to a conclusion. How the solution depends on the size of the perturbation is related to the perturbed matrix itself in a non-trivial manner, as is shown in Van Loan [1977]; the statements about perturbations in the matrix exponential in this chapter are all from this source. To begin with, at any fixed time of the solution, the relative error due to the perturbation vanishes at the same rate as the size of the perturbation, as the size of the perturbation tends to zero. It may be surprising then, that $E''_{11}$ being Hurwitz is not enough to ensure that the absolute perturbation error tends to zero with time, even for very small perturbations. However, such conditions exist, and one is to be presented shortly. It makes use of the logarithmic norm of a matrix, see for instance Strom [1975] or Söderlind [2006] for a discussion. The reason this condition is not made an assumption here is that, by looking at the solution during a fixed time interval, the decay of the perturbation error over time is not really needed. In addition, the condition to be presented here is just one of many conditions, and it recommended that the reader refer to Van Loan [1977] rather than checking only this: If the logarithmic norm of $E''_{11}$ is negative, in addition to $E''_{11}$ being Hurwitz, then the absolute error due to sufficiently small perturbations will eventually tend to zero as time tends to infinity.

### 5.3.2 Uniform approximation

In this section, it will be established that the size of the deviations in the solution are $O(\epsilon)$ maximizing both over time and the uncertainty-perturbations of $E_{22}$. While the time dimension was “handled” by only considering a finite time interval, it remains to ensure that the results are not affected by the exact content of $E_{22}$; knowing that the norm is 1 must be enough, so that $\epsilon$ alone defines the size of the perturbation. Using our compact notation, this is to say that an $O^{E_{22}}(\epsilon)$ bound — after a short boundary layer — will be established.

Clearly, the analysis takes different paths depending on the rank of $E_{22}$, but this causes no problems since there are only finitely many values of the rank to consider. Hence, it is sufficient to consider the rank as given.

The perturbation $E_{22}$ enters the analysis above in two places, first when setting $\epsilon := 0$, and then when estimating the error caused by the non-singular perturbation at the end of the analysis.
Beginning with the latter, note that the smallness of the second term in (5.9) requires that $E''_{21}$ be bounded under the perturbations (this is the only part of the expression that depends on $E_{22}$). This can be seen in (5.5), where the left hand side safely can be maximized with respect to unitary $Q_1$ and $Q_2$ ($Q_2$ being the part of $K_1$ which depends on $E_{22}$). Boundedness of $K_0 E$ then gives boundedness of everything in the right hand side of (5.5).

Note that $A''_{21} = 0$ and that $A''_{11} = A_{11} - A_{12} A_{22}^{-1} A_{21}$ is $O^{E_{22}}(1)$, and hence that it has been shown that $A''_{21}$ is $O^{E_{22}}(\epsilon)$. In addition, boundedness of $E''_{12}$ gives boundedness of $K_3$, and hence it is clear that all matrices in (5.7) are $O^{E_{22}}(1)$. Further, note that the block $Q_1 A_{22} Q_2^T$, but on the right hand side of (5.5), shows that $A''_{22}$ cannot have arbitrarily bad condition number without violating $A_{11}$; in other words, the condition number is bounded independently of $E_{22}$. Since $A''_{22}$ is just a small perturbation of $A_{22}$, also $A'''_{22}$ has bounded condition for sufficiently small $\epsilon$.

Turning to the singular perturbation part, it needs to be investigated how the form (5.8) depends on $E_{22}$, and then theorem 2.2 needs to be refined so as to confirm that the convergence is uniform with respect to $E_{22}$. Without going into details about how the theorem can be refined, we will just make a few remarks that should help the reader who is interested in revisiting the proofs in Kokotović et al. [1986].

First, we establish that the initial conditions after the changes of variables used in the proof of theorem 2.2 are bounded irrespectively of $E_{22}$. Avoiding reference to notation used in the proof, we just remark that this follows if $M_{22}^{-1} M_{21}$ and $M_{12} M_{22}^{-1}$ can be bounded. The first of these holds since $E_{22}'' - 1$ cancels out, and $A''_{22}$ has bounded norm and condition number. The second additionally uses that $E_{11}'' - 1$ is bounded.

The proof of theorem 2.2 also considers two matrix exponentials, one of a small perturbation of $M_0 = M_{11} - M_{12} M_{22}^{-1} M_{21}$, and one of a small perturbation of $M_{22}$. The latter is fortunately related to the duration of the bounding layer, and needs no further inspection by $A_3$. Note that the former is close to $E_{11}'' - 1 A_{11}''$, which is independent of $E_{22}$, but it remains to show that the derivative with respect to $\epsilon$ can be bounded regardless of $E_{22}$. Still avoiding reference to notation in the proof, this requires that $M_{22}^{-2} M_{21} M_0$ be $O^{M_{22}}(1)$, which follows by notes we have already made. This concludes our remarks on how theorem 2.2 needs to be refined.

This shows that the assumptions already listed before this section are enough to obtain uniform convergence with respect to $E_{22}$. However, due to the very strong nature of $A_3$, we refrain from highlighting this result in the form of a theorem.

### 5.4 Discussion

Before concluding this chapter, it is motivated to include a brief discussion of the analysis and its results. We begin, however, by giving two more examples that illustrate the lack of necessity in the assumptions made in this chapter.
5.4.1 Coping without $A_1$

The following two DAE represent different trivial cases when $A_1$ fails.

First, let $\epsilon$ be considered tiny and consider the DAE with matrices

$$
\begin{pmatrix}
1 & 0 & 1 \\
0 & \epsilon^? & \epsilon^? \\
0 & \epsilon^? & \epsilon^?
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}
$$

The equation contains a completely useless equation and is clearly ill-posed. If input data is assumed to be well-posed, this situation should not happen.

Second, keeping the same $A_{22}$ but adding structural zeros in the leading matrix and a non-zero element in $A_{21}$,

$$
\begin{pmatrix}
1 & 0 & 1 \\
0 & \epsilon^? & \epsilon^? \\
0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{pmatrix}
$$

it is found that $x_1 = 0$, and it follows that $\dot{x}_3 = 0$. Hence, $x_2$ is a stiff variable that quickly approaches zero.

5.4.2 Breaking $A_2$

When $A_2$ fails, it will generally not be possible to reach the standard singular perturbation form (2.41). Consider the DAE with matrices

$$
\begin{pmatrix}
1. & 0 & 1. \\
0 & \epsilon^? & \epsilon^? \\
0 & \epsilon^? & \epsilon^?
\end{pmatrix}
\begin{pmatrix}
1 & 1 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{pmatrix}
$$

Note that $A_1$ implies that $A_4$ must fail completely here.

Consider the case when the perturbed matrix has full rank. The algorithm then applies a variable transform to make the leading matrix block diagonal:

$$
\begin{pmatrix}
1. & 0 & 0. \\
0 & \epsilon^? & \epsilon^? \\
0 & \epsilon^? & \epsilon^?
\end{pmatrix}
\begin{pmatrix}
1 & 1 & -1 \\
0 & 1 & 0 \\
1. & 0 & 0.
\end{pmatrix}
$$

Inverting the perturbation is no good this time, since

$$
\begin{pmatrix}
? & ? \\
? & ?
\end{pmatrix}^{-1}
\begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix}
$$

is not full-rank for any perturbation, and hence singular perturbation theory cannot be applied here immediately. However, since the two rows are dependent it is possible to proceed, although doing so is not included in the algorithm proposed here.
5.4.3 Nature of the assumptions

\( A1 \) and \( A4 \) can easily be checked during index reduction. If they do not hold, the proposed index reduction scheme cannot be used to solve the \( \text{DAE} \), and an implementation may choose to report this as an error. \( A3 \) might sometimes be possible to validate using, for example, physical insight about the object that the \( \text{DAE} \) is supposed to describe. However, the judgment will become more difficult if \( \epsilon \) is just small, but not tiny, since that is the distinction between unreasonable and reasonable stiffness of the object being modeled. \( A2 \) typically requires a leap of faith, but it may also be the case that it is known that the equations have been obtained by equation and variable transforms applied to equations which are seen to be index 1 from their zero-patterns. Note that \( A2 \) is very similar to \( A4 \).

5.4.4 Applying the results

In [Kokotović et al. 1986], it is argued that theorem 2.2 is often used to motivate the ad hoc method of just setting \( \epsilon := 0 \) because that seems like a reasonable thing to do. However, the cautious user will also need to define smallness of \( \epsilon \) in terms of the allowable errors in the solution, and this will require a refined formulation of theorem 2.2 where any “\( O(\epsilon) \)” expressions are explicitly bounded. Without this, setting \( \epsilon := 0 \) is still an ad hoc procedure, and it must be emphasized that this remark applies just as much to the \( \text{DAE} \) index reduction scheme presented here. Admittedly, it seems unsatisfactory to make this kind of refinement based on \( A3 \) but detailed analysis was never the aim of this chapter; the point made is that one has to make assumptions in order to make sense of these otherwise ill-posed equations, and that it indeed is possible to find assumptions that help in the analysis.

In case the threshold for \( \epsilon \) is well above the threshold obtained by tracking numerical precision, the user could choose to first use a lower threshold to do away with the really tiny numbers first, since \( A3 \) can then be motivated using physical insight. Doing so would separate treatment of the ill-posed from index reduction as a means to handle stiffness. Thus, when tiny numbers are not so tiny any more, and physical insight may no longer be able support \( A3 \), the assumption must — and can — instead be checked by computing \( E''''^{-1} A'''' \) and verify that its poles are sufficiently far into the complex left half plane.
A different perturbation analysis

Understanding sensitivity with respect to small parameters of the leading matrix of an LTI DAE is an interesting topic in itself. However, it is also the foundation from which we aim to develop understanding of index reduction of quasilinear DAE. This outset was explained in more detail in chapter 5, but the analysis therein was based on arguably inconvenient assumptions. It is the aim of this chapter to make a parallel development, but at the same time more strict and based on more convenient assumptions.

Consider the LTI DAE

$$E x'(t) + A x(t) = 0$$

with uncertain matrices

$$K_0 E = \begin{pmatrix} E_{11} & E_{12} \\ 0 & \epsilon E_{22} \end{pmatrix}$$

$$K_0 A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

Assuming, among other, that the true equations are of differential index 1, it was shown in chapter 5 that there exists a threshold for $\epsilon$ such that smaller $\epsilon$ may be neglected. The threshold was shown to exist by providing an $O^{E_{22}}(\epsilon)$ bound (recall the definition of this notation from section 1.5) on the difference between the solution for $\epsilon = 0$ and any solution for a possible perturbation. Note that neglecting $\epsilon$ is generally the only way to proceed with the equations if $E_{22}$ contains only numbers that cannot be told apart from zero.

The question of how the solution depends on the small parameter $\epsilon$ is related to the so-called singular perturbation theory, well developed in Kokotović et al. [1986]. In that setup, $E_{11} = I$, $E_{12} = 0$, and $E_{22} = I$, so one essentially deals with an ODE with time-scale separation. The important difference to our setting is that $E_{22}$ is considered known and of perfect condition in singular perturbation theory.

93
**Notation.** In this chapter, new symbols are constructed by adding “over-bars” to existing symbols. For instance, this means that $\bar{E}$ does not refer to some selection of rows from $E$ or any other construction, but is just a mnemonic way of constructing a symbol that should remind of $E$. In section 5.3, the sleeker prime sign was used in the same way, but that would be too confusing in this chapter since the symbols often denote functions of one argument (for which the prime sign is defined to denote differentiation).

### 6.1 Preliminaries

In this section, we state two bounds on the norm of the matrix exponential. They are much more simple than tight.

**Lemma 6.1**

Let $A$ be a linear map from an $n$-dimensional space to itself. Let $\alpha(A)$ denote the largest real part of the eigenvalues of $A$. Then

$$\|e^{At}\| \leq e^{\alpha(A)t} \sum_{i=0}^{n-1} \left( \frac{2\|A\|}{i!} t^i \right)$$  \hspace{1cm} (6.2)

**Proof:** Let $Q^H A Q = D + N$ be a Schur decomposition of $A$, meaning that $Q$ is unitary, $D$ diagonal, and $N$ nilpotent. The following bound, derived in [Van Loan 1977],

$$\|e^{At}\| \leq e^{\alpha(A)t} \sum_{i=0}^{n-1} \frac{\|N\|^i t^i}{i!}$$

readily gives the result since $\|N\| = \|Q^n A Q - D\| \leq \|A\| + \|A\|$. \hfill $\Box$

**Lemma 6.2**

If the map $A$ is Hurwitz, that is, $\alpha(A) < 0$, then for $t \geq 0$,

$$\|e^{At}\| \leq e^{\left( \frac{2\|A\|}{-\alpha(A) - 1} \right)t}$$

**Proof:** Let $f(t) := \|e^{At}\|$. From lemma 6.1 we have that

$$f(t) \leq \sum_{i=0}^{n-1} \left( \frac{2\|A\|}{i!} t^i \right) e^{\alpha(A)t} =: \sum_i f_i(t)$$

Each $f_i(t)$ can easily be bounded globally since they are smooth, tend to 0 from above as $t \to \infty$, and the only stationary point is given by

$$f_i'(t) = 0 \iff e^{\alpha(A) t} \left( \frac{2\|A\|}{i!} t^{i-1} \right) (t \alpha(A) + i)^{-1} = 0$$

$$\iff t = -\frac{i}{\alpha(A)}$$
That is,
\[ f_i(t) \leq f_i \left( -\frac{i}{\alpha(A)} \right) = \left( \frac{2\|A\|}{-\alpha(A)} \right)^i \frac{i^i}{i!} e^{-i} \leq \left( \frac{n \cdot 2\|A\|}{-\alpha(A)} \right)^i \frac{i^i}{i!} e^{-n} \]

Hence,
\[ f(t) \leq \sum_{i=0}^{n-1} \left( \frac{n \cdot 2\|A\|}{-\alpha(A)} \right)^i \frac{i^i}{i!} e^{-n} \leq e^{-n} \sum_{i=0}^{\infty} \left( \frac{n \cdot 2\|A\|}{-\alpha(A)} \right)^i \frac{i^i}{i!} = e^{\left( \frac{2\|A\|}{-\alpha(A)} - 1 \right) n} \]

\[\square\]

6.2 Analysis

The analysis in this chapter is limited to DAE of index at most 1. We first consider equations of index 0, before taking on the slightly more involved systems of index at most 1.

6.2.1 Singular perturbation in ODE

The derivation in this section follows the structure in Kokotović et al. [1986]. In their analysis, results come in two flavors; one where approximations are valid on any finite time interval, and one where stability of the slow dynamics in the system make the approximations valid without restriction to finite time intervals. In the present treatment, it is from here on assumed that only finite time intervals are considered, but the other case is treated just as easily.

**Lemma 6.3**

Let \( E \) be a constant (possibly singular) matrix with \( \|E\| \leq 1 \) but otherwise unknown. Assume \( M_{22} \) is non-singular. Then there exists a change of variables,
\[
\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} I & 0 \\ LE(\epsilon) & I \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix}
\]

such that the ODE-looking DAE
\[
\begin{pmatrix} I \\ \epsilon E \end{pmatrix} \begin{pmatrix} x'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} x(t) \\ z(t) \end{pmatrix}
\]

can be written
\[
\begin{pmatrix} I \\ \epsilon E \end{pmatrix} \begin{pmatrix} \xi'(t) \\ \eta'(t) \end{pmatrix} = \begin{pmatrix} M_{11} + M_{12} LE(\epsilon) \\ 0 \\ M_{12} \\ M_{22} - \epsilon E LE(\epsilon) M_{12} \end{pmatrix} \begin{pmatrix} x(t) \\ 0 \\ \eta(t) \end{pmatrix}
\]

and
\[ LE(\epsilon) = L(0) + O(\epsilon) \]
Proof: Applying the change of variables and then performing row operations on the equations to eliminate $x'$ from the second group of equations, lead to the condition defining $L_E(\epsilon)$:

$$0 = M_{21} + M_{22} L_E(\epsilon) - \epsilon E L_E(\epsilon) \left( M_{11} + M_{12} L_E(\epsilon) \right)$$

This shows that

$$L(0) = -M^{-1}_{22} M_{21}$$

and it remains to show that $L$ is a differentiable function of $\epsilon$, and that the derivative at 0 can be bounded independently of $E$. Differentiating the equation with respect to $\epsilon$ yields

$$0 = M_{22} L_E'(\epsilon) - \left( E L_E(\epsilon) + \epsilon E L_E'(\epsilon) \right) \left( M_{11} + M_{12} L_E(\epsilon) \right) - \epsilon E L_E(\epsilon) M_{12} L_E'(\epsilon)$$

In particular,

$$0 = M_{22} L_E'(0) + E M^{-1}_{22} M_{21} \left( M_{11} - M_{12} M^{-1}_{22} M_{21} \right)$$

can be solved with respect to $L_E'(0)$, and the solution is bounded independently of $E$ since we know $\|E\| \leq 1$.

Lemma 6.4

For the derivative of order $n \geq 1$,

$$L_E^{(n)}(0) = M^{-1}_{22} E Y_n$$

where $Y_n$ is a matrix bounded independently of $E$. In addition, $Y_1$ is independent of $E$.

Proof: A complete proof is not given here, but the procedure is outlined. By differentiating (6.6) with respect to $\epsilon$ repeatedly, and setting $\epsilon = 0$ in the resulting equations, one obtains equations where increasing orders of the derivative of $L_E$ can be solved for at $\epsilon = 0$. It can be observed that the highest order derivative of $L_E$ appearing in the equations always appear in a single term where it is multiplied by $M_{22}$ from the left, and all other terms are multiplied by $E$ from the left, which completes the outline.

Lemma 6.5

If the initial conditions for (6.4) are consistent with $\epsilon = 0$, then under the change of variables (6.3), the initial conditions for $\eta$ satisfy

$$\eta_E^0(\epsilon) = O^E(\epsilon)$$

Proof: That the initial conditions are consistent with $\epsilon = 0$ means that

$$0 = M_{21} x^0 + M_{22} z^0$$

and from $z = L_E(\epsilon) x + \eta$ it follows that

$$\eta_E^0(\epsilon) = z^0 - L_E(\epsilon) x^0$$

Multiply by the invertible $M_{22}$ and apply lemma 6.3 to find

$$M_{22} \eta_E^0(\epsilon) = M_{22} z^0 + M_{21} x^0 + O^E(\epsilon) x^0 = O^E(\epsilon) x^0$$
Corollary 6.1

It holds that

$$\eta_0^E(\epsilon) = M_{22}^{-1} E \sum_{i=1}^{\infty} \frac{\epsilon^n}{n!} Y_n x^0$$

Proof: Follows by using lemma 6.4 in the proof of lemma 6.5

Lemma 6.6

In addition to the assumptions of lemma 6.5, assume $E$ is known to be non-singular and that there exist $R_0 > 0$ and $\phi_0 < \pi/2$ such that for $\lambda$ being a pole of (6.4),

$$|\lambda| > R_0 \implies |\arg(-\lambda)| < \phi_0$$

Also assume that the DAE is not close to index 1 in the sense that there exists a bound $\kappa_0$ on the condition number of $E$.

Then, for any fixed $t_1 \geq t_0$, for all $t \in [t_0, t_1]$,

$$|\eta_E(t, \epsilon)| = O^E(\epsilon)$$

Proof: The isolated system in $\eta$ has the state matrix

$$M_\eta := \frac{1}{\epsilon} E^{-1} M_{22} - L_E(\epsilon) M_{12}$$

The condition number bound gives $\|E^{-1}\| \leq \kappa_0$, and hence $\|M_\eta\| < \frac{\kappa_0 + 1}{\epsilon} \|M_{22}\|$ for sufficiently small $\epsilon$. By lemmas 6.2 and 6.5, it only remains to show that $\epsilon \alpha(M_\eta)$ can be bounded by a negative constant. By showing that the there exists a constant $k_1 > 0$ such that any eigenvalue $\lambda$ of $\epsilon M_\eta$ is larger in magnitude than $k_1$ as $\epsilon \to 0$, it follows that all eigenvalues of $M_\eta$ approaches infinity like $\frac{k_1}{\epsilon}$, as $\epsilon \to 0$. It then follows that they will all satisfy the argument condition for sufficiently small $\epsilon$, and that $\alpha(M_\eta) < -\frac{k_1}{\epsilon} \cos(\phi_0)$.

This is shown by using that all eigenvalues of $\epsilon M_\eta$ are greater than $\| (\epsilon M_\eta)^{-1} \|^{-1}$, where

$$\| (\epsilon M_\eta)^{-1} \|^{-1} = \| (E^{-1} (M_{22} - \epsilon E L_E(\epsilon) M_{12}) )^{-1} \|^{-1} \geq \| (M_{22} - \epsilon E L_E(\epsilon) M_{12})^{-1} \|^{-1}$$

Here, it is clear that the limit is positive since $M_{22}$ is non-singular, but to ensure that there is an $\epsilon^* > 0$ such that $\| (\epsilon M_\eta)^{-1} \|^{-1}$ is greater than some positive constant for all $\epsilon \in [0, \epsilon^*]$, we must also show that the derivative with respect to $\epsilon$ is finitely bounded independently of $E$. By differentiability of the matrix inverse and matrix norm, this follows if the derivative of the inverted matrix is bounded independently of $E$, which is readily seen.
Lemma 6.7
Under the assumptions of lemma 6.3 there exists a change of variables,
\[
\begin{pmatrix}
x \\
\eta
\end{pmatrix} = \begin{pmatrix}
I & \epsilon H_E(\epsilon) E \\
0 & I
\end{pmatrix} \begin{pmatrix}
\xi \\
\eta
\end{pmatrix}
\] (6.8)
such that the implicit ODE (6.5) can be written
\[
\begin{pmatrix}
I & \epsilon E \\
0 & 0
\end{pmatrix} \begin{pmatrix}
\xi'(t) \\
\eta'(t)
\end{pmatrix} = \begin{pmatrix}
M_{11} + M_{12} L_E(\epsilon) & 0 \\
0 & M_{22} - \epsilon E L_E(\epsilon) M_{12}
\end{pmatrix} \begin{pmatrix}
\xi(t) \\
\eta(t)
\end{pmatrix}
\] (6.9)
and for sufficiently small \(\epsilon\), \(\|H_E(\epsilon)\|\) is bounded by a constant independently of \(E\).

Proof: Applying the change of variables and then performing row operations on the equations to eliminate \(\eta'\) from the first group of equations, lead to the condition defining \(H_E(\epsilon)\):
\[
0 = \begin{pmatrix}
M_{11} + M_{12} L_E(\epsilon) \\
0
\end{pmatrix} \epsilon H_E(\epsilon) E + M_{12} - H_E(\epsilon) \begin{pmatrix}
0 \\
M_{22} - \epsilon E L_E(\epsilon) M_{12}
\end{pmatrix}
\]
It follows that
\[
H_E(0) = M_{12} M_{22}^{-1}
\]
which is clearly bounded independently of \(E\). The equation is linear in \(H_E(\epsilon)\) and the coefficients depend smoothly on \(\epsilon\), so the solution is differentiable at \(\epsilon = 0\). It thus remains to show that the derivative of \(H_E(\epsilon)\) with respect to \(\epsilon\) at 0 can be bounded independently of \(E\). Differentiating the equation and looking at \(\epsilon = 0\) reveals that
\[
0 = \begin{pmatrix}
M_{11} + M_{12} L_E(0) \\
0
\end{pmatrix} H_E(0) E - H_E'(0) M_{22} + H_E(0) E L_E(0) M_{12}
\]
\[= \begin{pmatrix}
M_{11} + M_{12} L_E(0) \\
0
\end{pmatrix} M_{12} M_{22}^{-1} E - H_E'(0) M_{22} + M_{12} M_{22}^{-1} E L_E(0) M_{12}
\]
where it is seen that \(H_E'(0)\) is bounded as desired. \(\square\)

Theorem 6.1
Consider the following variation of the standard singular perturbation setup:
\[
x'(t) \overset{\frac{1}{\epsilon}}{=} M_{11} x(t) + M_{12} z(t)
\]
\[
\epsilon E z'(t) \overset{\frac{1}{\epsilon}}{=} M_{21} x(t) + M_{22} z(t)
\]
(6.10)
where \(E\) is a constant non-singular matrix with \(\|E\| \leq 1\) but otherwise unknown. Let the solution at time \(t\) be denoted \(x_E(t, \epsilon)\), and let us write \(x_E(t, 0) = x(t, 0)\) to emphasize that \(E\) does not matter if \(\frac{1}{\epsilon} \overset{=}{\rightarrow} 0\).

Assume \(M_{22}\) is non-singular, and that there exist \(R_0 > 0\) and \(\phi_0 < \pi/2\) such that for \(\lambda\) being a pole of (6.10),
\[
|\lambda| > R_0 \implies |\arg(-\lambda)| < \phi_0
\]
Then
\[
|x_E(t, \epsilon) - x(t, 0)| = \mathcal{O}^E(\epsilon) \] (6.11)
\[
|z_E(t, \epsilon) + M_{22}^{-1} M_{21} x(t, 0)| = \mathcal{O}^E(\epsilon) \] (6.12)
**Proof:** Define $L_E(\epsilon)$ and $H_E(\epsilon)$ as above, and consider the solution expressed in the variables $\xi$ and $\eta$. Lemma 6.6 shows how $\eta$ is bounded uniformly over time and with respect to $E$. Note that $x(t, 0)$ coincides with $\xi(t, 0)$, so the left hand side of (6.11) can be bounded as

$$|x_E(t, \epsilon) - x(t, 0)| = |\xi_E(t, \epsilon) + \epsilon H_E(\epsilon) \eta_E(t, \epsilon) - \xi(t, 0)| \leq |\xi_E(t, \epsilon) - \xi(t, 0)| + O_E(\epsilon^2)$$

To see that the first of these terms is $O_E(\epsilon)$, note first that lemmas 6.5 and 6.7 give that the initial conditions for $\xi$ are only $O_E(\epsilon^2)$ away from $x^0$. Hence, the restriction to a finite time interval gives that the contribution from initial conditions is negligible. The contribution from perturbation of the state matrix for $\xi$ depends on the perturbed matrix in a non-trivial manner, but useful bounds exist. Since lemma 6.7 shows that the size of the perturbation is $O_E(\epsilon)$, it follows that the contribution is $O_E(\epsilon)$ at any fixed time $t$. The constants of the $O_E(\epsilon)$ bounds will of course be a continuous function of time, and since the time interval of interest is compact, it follows that a dominating constant exists.

Concerning $z$,

$$|z_E(t, \epsilon) + M_{22}^{-1} M_{21} x(t, 0)| \leq |z_E(t, \epsilon) + M_{22}^{-1} M_{21} x_E(t, \epsilon)| + |M_{22}^{-1} M_{21} (x(t, 0) - x_E(t, \epsilon))|$$

$$= |\eta_E(t, \epsilon)| + O_E(\epsilon) |x_E(t, \epsilon)| + \|M_{22}^{-1} M_{21}\| O_E(\epsilon)$$

since $|x_E(t, \epsilon)|$ can be bounded over any finite time interval. □

**Theorem 6.2**

*Theorem 6.1* can be extended to the setup

$$x'(t) \overset{1}{=} M_{11}(\epsilon) x(t) + M_{12}(\epsilon) z(t)$$
$$\epsilon E(\epsilon) z'(t) \overset{1}{=} M_{21}(\epsilon) x(t) + M_{22}(\epsilon) z(t)$$

(6.13)

if there exist $\epsilon^* > 0$ and $k^1$ such that for all $\epsilon \in [0, \epsilon^*]$ and $E$ consistent with the assumptions of lemma 6.6

$$||M_{ij}(\epsilon)|| \leq k^1$$
$$||E'(\epsilon)|| \leq k^1$$

(6.14)

The approximations then take the form

$$|x_E(t, \epsilon) - x(t, 0)| \leq k_0 \epsilon$$

(6.15)

$$|z_E(t, \epsilon) - z(t, 0)| \leq k_0 \epsilon$$

(6.16)

where

$$z(t, 0) = -M_{22}^{-1}(0) M_{21}(0) x(t, 0)$$

(6.17)
Proof: Note that the requirement on bounded derivatives imply

$$M_{ij}(\epsilon) = M_{ij}(0) + O^E(\epsilon)$$

The result then follows by repeating, with minor changes, the proofs of the lemmas and theorem above. □

**Lemma 6.8**

Theorem 6.2 can be extended to the index 0 square DAE setup

$$0 = \begin{pmatrix} E_1(\epsilon) & A_1(\epsilon) \\ \epsilon E_2(\epsilon) & A_2(\epsilon) \end{pmatrix} x(t) + \begin{pmatrix} A_1(\epsilon) \\ A_2(\epsilon) \end{pmatrix} x(t)$$

(6.18)

Here $\|E_2\| \leq 1$ bounds the lower block of the leading matrix, which is unknown although with known bound on the condition number. The lower block may be a function of $\epsilon$, but the bound on the condition number shall not depend on $\epsilon$. It is further required that the limiting case is index 1, that is, the matrix

$$\begin{pmatrix} E_1(0) \\ A_2(0) \end{pmatrix}$$

(6.19)

is non-singular. The matrix $E_1(\epsilon)$ shall have bounded derivative with respect to $\epsilon$, just as in (6.14).

Since the variables are not partitioned in any particular fashion here, the approximation is simply written

$$|x_E(t, \epsilon) - x(t, 0)| \leq k_0 \epsilon$$

and (6.17) makes no sense any more since $A_{22}$ need not be a selection of independent columns of $A_2$.

Note that (6.18) being index 0 implies that both blocks of the leading matrix are of full rank.

Proof: By the results in Chang et al. [1997] it follows that if $E_1(\epsilon)$ is full-rank at $\epsilon = 0$ and has bounded derivative for $\epsilon$ small enough, then it has a QR factorization with bounded derivative for $\epsilon$ small enough. Also note that if $E(\epsilon)$ is non-singular at $\epsilon = 0$ and has bounded derivative for $\epsilon$ small enough, then its inverse will also have bounded derivative.

Applying the change of variables given by the QR factorization

$$x = Q(\epsilon) \begin{pmatrix} \bar{x} \\ \bar{z} \end{pmatrix}$$

brings the equation in the form

$$0 = \begin{pmatrix} \bar{E}_{11}(\epsilon) & 0 \\ \epsilon \bar{E}_{21}(\epsilon) & \epsilon \bar{E}_{22}(\epsilon) \end{pmatrix} \begin{pmatrix} \bar{x}'(t) \\ \bar{z}'(t) \end{pmatrix} + \begin{pmatrix} \bar{A}_{11}(\epsilon) & \bar{A}_{12}(\epsilon) \\ \bar{A}_{21}(\epsilon) & \bar{A}_{22}(\epsilon) \end{pmatrix} \begin{pmatrix} x'(t) \\ z'(t) \end{pmatrix}$$

Finally, the matrix $\bar{E}_{21}(\epsilon)$ is of full rank, so $\bar{E}_{21}(\epsilon)$ is invertible. Therefore, we can write

$$\bar{x}'(t) = \bar{E}_{11}(\epsilon)^{-1} \left( \bar{E}_{21}(\epsilon) \bar{x}'(t) + \bar{A}_{11}(\epsilon) x'(t) + \bar{A}_{12}(\epsilon) z'(t) \right)$$

(6.20)

which implies

$$\bar{x}'(t) = \left( \bar{E}_{11}(\epsilon)^{-1} \bar{E}_{21}(\epsilon) \right) \bar{x}'(t) + \left( \bar{E}_{11}(\epsilon)^{-1} \bar{A}_{11}(\epsilon) \right) x'(t) + \left( \bar{E}_{11}(\epsilon)^{-1} \bar{A}_{12}(\epsilon) \right) z'(t)$$

(6.21)

Since $\bar{E}_{11}(\epsilon)^{-1}$ is non-singular, we can solve for $\bar{x}'(t)$ in terms of $x'$ and $z'$.
with the norm of each row unchanged, and it is seen that both \( E'_{11} \) and \( E'_{22} \) are non-singular. The \( O^E(\epsilon) \) approximations are also still valid, and continues to be so if the first group of equations is manipulated by row operations leading to

\[
0 \overset{\perp}{=} \begin{pmatrix} I & 0 \\ \epsilon \bar{E}_{21}(\epsilon) & \epsilon \bar{E}_{22}(\epsilon) \end{pmatrix} \begin{pmatrix} \bar{x}'(t) \\ \bar{z}'(t) \end{pmatrix} + \begin{pmatrix} \bar{A}_{11}(\epsilon) & \bar{A}_{12}(\epsilon) \\ \bar{A}_{21}(\epsilon) & \bar{A}_{22}(\epsilon) \end{pmatrix} \begin{pmatrix} x'(t) \\ z'(t) \end{pmatrix}
\]

Finally eliminating \( \epsilon E'_{21} \) from the leading matrix by row operations will only change the lower part of the matrix in the algebraic term by

\[
\epsilon \bar{E}_{21}(\epsilon) \begin{pmatrix} \bar{A}_{11}(\epsilon) & \bar{A}_{12}(\epsilon) \end{pmatrix}
\]

which does not put the bounded derivative condition of theorem 6.2 at risk. Since \( \|E'_{22}\| \leq 1 \), it remains to verify that the limiting matrix \( \bar{A}_{22}(0) \) is non-singular. This follows by applying \( Q(0) \) to the non-singular matrix (6.19), which shows that

\[
\begin{pmatrix} \bar{E}_{11}(0) & 0 \\ \bar{A}_{21}(0) & \bar{A}_{22}(0) \end{pmatrix}
\]

(6.20)

is non-singular.

Hence, \( \bar{x} \) and \( \bar{z} \) are approximated according to (6.15) and (6.16). The approximation can be written as just one norm bound, which carries over to \( x \) without loss since \( Q(\epsilon) \) defines an isometry.

\[
\square
\]

6.2.2 Singular perturbation in index 1 DAE

With the exceptions of lemmas 6.3, 6.5, and 6.7, the theorems so far require, via lemma 6.6, that \( E \) (or \( (E_{21} \, E_{22}) \)) have bounded condition number. However, it is possible to proceed also when some singular values are exactly zero, if assuming that the DAE is not close to index 2. Next, the results of the previous section will be extended to this situation by revisiting the relevant proofs.

Common to the proofs in this section is the observation that the uncertain perturbation there is a non-empty interval including 0 of positive \( \epsilon \) values in which the perturbation has constant rank. Since there are only finitely many possible values for the rank to take, proving an \( O^E(\epsilon) \) result for the case when the rank is known immediately leads to the corresponding \( O^E(\epsilon) \) for the case of unknown rank.

Lemma 6.9

(Compare lemma 6.6)

In addition to the assumptions of lemma 6.5 assume the perturbed DAE is known to have index no more than 1, and that there exist \( R_0 > 0 \) and \( \phi_0 < \pi/2 \) like in lemma 6.5. Also assume that the ratio between the largest and smallest non-zero singular value of \( E \) is bounded by some constant \( \kappa_0 \). Then, for any fixed \( t_1 \geq t_0 \), for all \( t \in [t_0, t_1] \),

\[
|E \eta_E(t, \epsilon)| = O^E(\epsilon)
\]
**Proof:** The case of index 0, when $E$ is full-rank, was treated in lemma 6.6, so it remains to consider the case of index 1. When the rank is zero, $E = 0$ and it is immediately seen from (6.5) that $\eta$ must be identically zero and the conclusion follows trivially. Hence, assume that the rank is neither full nor zero and let

$$E = (U_1 \quad U_2) \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

be an SVD of where $\Sigma$ is of known dimensions and has condition number less than $\kappa_0$. Applying the unknown change of variables $\eta = V \begin{pmatrix} \eta_1' \\ \eta_2' \end{pmatrix}$ and the row operations represented by $U^T$, (6.5) turns into

$$\begin{pmatrix} I & 0 & 0 \\ \epsilon \Sigma & 0 & 0 \end{pmatrix} \begin{pmatrix} \xi(t) \\ \eta_1(t) \\ \eta_2(t) \end{pmatrix} = \begin{pmatrix} M_{11} + M_{12} L_E(\epsilon) & M_{12} V_1 & M_{12} V_2 \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} \xi(t) \\ \eta_1(t) \\ \eta_2(t) \end{pmatrix}$$

where, for instance and in particular,

$$A_{33} := U_2^T M_{22} V_2 - \epsilon U_2^T E L_E(\epsilon) M_{12} V_2 = U_2^T M_{22} V_2$$

Since the DAE is known to be index 1, differentiation of the last group of equations shows that $A_{33}$ is non-singular, and hence the change of variables

$$\begin{pmatrix} \tilde{\eta}_1(t) \\ \tilde{\eta}_2(t) \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \\ -A_{33}^{-1} A_{32} & I \end{pmatrix} \begin{pmatrix} \bar{\eta}_1(t) \\ \bar{\eta}_2(t) \end{pmatrix}$$

leads to the DAE in $(\xi, \tilde{\eta}_1, \tilde{\eta}_2)$ with matrices

$$\begin{pmatrix} I & 0 & 0 \\ 0 & \epsilon \Sigma & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} M_{11} + M_{12} L_E(\epsilon) & M_{12} V_1 - M_{12} V_2 A_{33}^{-1} A_{32} & M_{12} V_2 \\ 0 & A_{22} - A_{23} A_{33}^{-1} A_{32} & A_{23} \\ 0 & 0 & A_{33} \end{pmatrix}$$

It is seen that $\tilde{\eta}_2 = 0$ and that $\tilde{\eta}_1$ is given by an ODE with state matrix

$$M_{\tilde{\eta}_1} = \frac{1}{\epsilon} \Sigma^{-1} (A_{22} - A_{23} A_{33}^{-1} A_{32})$$

Just like in lemma 6.6 it needs to be shown that the eigenvalues of this matrix tend to infinity as $\epsilon \to 0$, independently of $E$, but here we need to recall that $E$ is not only present in $\Sigma$, but also in the unknown unitary matrices $U$ and $V$. Again, we do this by showing

$$\lim_{\epsilon \to 0} \sup_E \left\| (\epsilon M_{\tilde{\eta}_1})^{-1} \right\|^{-1} > 0$$

Using $\|\Sigma\| = \|E\| \leq 1$, and that

$$\begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} = \left( A_{22} - A_{23} A_{33}^{-1} A_{32} \right)^{-1}$$

and

$$\begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} \geq \left( A_{22} - A_{23} A_{33}^{-1} A_{32} \right)^{-1}$$
we find
\[
\left\| (\epsilon M_{\tilde{\eta}_1})^{-1} \right\|^{-1} = \left\| (A_{22} - A_{23} A_{33}^{-1} A_{32})^{-1} \Sigma \right\|^{-1}
\geq \left\| (A_{22} - A_{23} A_{33}^{-1} A_{32})^{-1} \right\|^{-1}
\geq \left\| (U^T (M_{22} - \epsilon E L_E(\epsilon) M_{12}) V)^{-1} \right\|^{-1}
= \left\| V^T (M_{22} - \epsilon E L_E(\epsilon) M_{12})^{-1} U \right\|^{-1}
= \left\| (M_{22} - \epsilon E L_E(\epsilon) M_{12})^{-1} \right\|^{-1}
\]
and just like in lemma 6.6 the expression gives that the eigenvalues tend to infinity uniformly with respect to \(E\), and hence that \(\epsilon\) can be chosen sufficiently small to make \(|\tilde{\eta}_1|\) bounded by some factor times \(|\tilde{\eta}_1(0)|\). Further,
\[
|\tilde{\eta}_1(0)| = \left| \begin{pmatrix} \tilde{\eta}_1(0) \\ \eta_2(0) \end{pmatrix} \right| = \left| \begin{pmatrix} \tilde{\eta}_1(0) \\ 0 \end{pmatrix} \right| \leq \left| \begin{pmatrix} \tilde{\eta}_1(0) \\ \eta_2(0) \end{pmatrix} \right| = |\eta_E(\epsilon)| = O^E(\epsilon)
\]
Using this, the conclusion finally follows by taking such a small \(\epsilon\):
\[
|E \eta_E(t, \epsilon)| = \left| EV \begin{pmatrix} \mathcal{I} & 0 \\ -A_{33}^{-1} A_{32} & \mathcal{I} \end{pmatrix} \begin{pmatrix} \tilde{\eta}_1(t) \\ 0 \end{pmatrix} \right|
\leq \left\| U \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^T \begin{pmatrix} \mathcal{I} & 0 \\ -A_{33}^{-1} A_{32} & \mathcal{I} \end{pmatrix} \right\| O^E(\epsilon)
= \left\| \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \right\| O^E(\epsilon) = O^E(\epsilon)
\]

**Corollary 6.2**

Lemma 6.9 can be strengthened when \(z\) has only two components. Then, just like in lemma 6.6, the conclusion is
\[
|\eta_E(t, \epsilon)| = O^E(\epsilon)
\]

**Proof:** The only rank of \(E\) that needs to be considered is 1, and then \(A_{33}^{-1} A_{32}\) will be a scalar. From (6.21) it follows that \(A_{33}^{-1} A_{32} \tilde{\eta}_1(0) = O^E(\epsilon)\), which is then extended to all later times \(t\), and hence
\[
\left| \begin{pmatrix} \tilde{\eta}_1(t) \\ \eta_2(t) \end{pmatrix} \right| = \left| \begin{pmatrix} \tilde{\eta}_1(t) \\ -A_{33}^{-1} A_{32} \tilde{\eta}_1(t) \end{pmatrix} \right| = O^E(\epsilon)
\]
\]

\(\Box\)
Theorem 6.3 can be extended as follows.

**Theorem 6.3**
Consider the setup (6.10), but rather than assuming that $E$ be of bounded condition, it is assumed that $E$ is a constant matrix with $\|E\| \leq 1$, bounded ratio between the non-zero singular values, and that the perturbed equation has index no more than 1. Except regarding $E$, the same assumptions that were made in theorem 6.1 are made here. Then
\[
|x_E(t, \epsilon) - x(t, 0)| = O_E(\epsilon) \quad (6.22)
\]
\[
|z_E(t, \epsilon) + M_{22}^{-1} M_{21} x(t, 0)| = O_E(\epsilon) \quad (6.23)
\]
where the rather useless second equation is included for comparison to theorem 6.1.

**Proof:** Define $L_E(\epsilon)$ and $H_E(\epsilon)$ as above, and consider the solution expressed in the variables $\xi$ and $\eta$. Lemma 6.9 shows how $E \eta$ is bounded uniformly over time. Note that $x(t, 0)$ coincides with $\xi(t, 0)$, so the left hand side of (6.22) can be bounded as
\[
|x_E(t, \epsilon) - x(t, 0)| = |\xi_E(t, \epsilon) + \epsilon H_E(\epsilon) E \eta_E(t, \epsilon) - \xi(t, 0)| 
\leq |\xi_E(t, \epsilon) - \xi(t, 0)| + O_E(\epsilon^2)
\]
The conclusion concerning $x$ then follows by an identical argument to that found in the proof of theorem 6.1. The weak conclusion regarding $z$ follows by noting that, in lemma 6.9 given $E$, $\|A_{33} A_{32}\|$ approaches some finite value as $\epsilon \to 0$, since $A_{33}$ must approach a non-singular matrix.

**Corollary 6.3**
Theorem 6.3 can be strengthened in case $z$ has only two components. Then (6.23) can be written with $O_E(\epsilon)$ on the right hand side.

**Proof:** Follows by using corollary 6.2 in the proof of theorem 6.3.

Theorem 6.2 can be extended as follows.

**Theorem 6.4**
Consider the setup (6.13), but rather than assuming that $E$ be non-singular, let it satisfy the assumption of theorem 6.3. Except regarding $E$, the same assumptions that were made in theorem 6.2 are made here. Then the conclusion of theorem 6.3 still holds.

**Proof:** When repeating, with minor changes, proofs of lemmas and theorems above, a singular value decomposition with bounded derivatives will be needed in lemma 6.9. The existence of such a factorization follows by modifying Steinbrecher [2006, theorem 2.4.1] to suit our needs.

**Theorem 6.5 (Main theorem)**
Lemma 6.8 can be extended to index 1 if there is a known bound on the ratio between the non-zero singular values of $E_2$.

**Proof:** It suffices to note that the proof of lemma 6.8 never uses the condition number bound on $E$ directly, and that the lemmas and theorems it builds upon (that is, the indirect use of the condition number bound) have all been extended to index 1.
6.3 Discussion

To conclude this chapter, we make some remarks on the assumptions used, and include an example that indicate a direction for future research.

6.3.1 Nature of the assumptions

Requiring a bound on the ratio between non-zero singular values of the uncertain singular perturbation seems rather limiting, since we typically do not think of the objects described by our equations in terms of differential indices. Compare this to the condition involving the eigenvalues; the eigenvalues determine the system poles, and their magnitude and argument have nice interpretations in time domain, and hence it is not too far-fetched to require the user to make assumptions regarding them. Trying to eliminate the unnatural requirement on the condition number — or to find a counter-example — will be an important topic in our future research.

The bound of lemma 6.2 may be very conservative, and although more precise bounds can readily be extracted from the proof, easily obtained bounds will not be good enough. Having excluded the possibility of bounding \( \eta \) by looking at the matrix exponential alone, it remains to explore the fact that we are actually not interested in knowing the maximum gain from initial conditions to later states of the trajectory of \( \eta \), but the initial conditions are a function of \( E \), and hence it might be sufficient to maximize over a subset of initial conditions. Here, it is expected that corollary 6.1 will come to use.

6.3.2 Example

In this section we follow up the discussion on the condition number in the previous section by providing an example which should shed some more light on — and stimulate future research on — the problem of singular perturbation in DAE.

---

Example 6.1

In this section, the bounding of \( \eta \) over time is considered in case \( \eta \) has two components. For simplicity, we shall assume that \( \eta \) is given by

\[
\eta'(t) = \frac{1}{\epsilon} E^{-1} M_{22} \eta(t)
\]

where \( M_{22} = I \), and we set \( \epsilon = 1 \). By selecting \( E \) as

\[
E = \begin{pmatrix}
-\delta & 1 - \delta \\
0 & -\delta
\end{pmatrix}
\]

where \( \delta > 0 \) is a small parameter we ensure \( \| E \| \leq 1 \), and since

\[
E^{-1} = \begin{pmatrix}
-1/\delta & 1/\delta^2 - 1/\delta \\
0 & -1/\delta
\end{pmatrix}
\]
we see that both eigenvalues are perfectly stable and far into the left half plane, while the
off-diagonal element is at the same time arbitrarily big. It is easy to verify using software
that the maximum norm of the matrix exponential grows without bound as $\delta$ tends to zero.
This shows that using only the norm of the initial conditions is not enough if we would
like to find a bound on $|\eta(t)|$ which does not depend on the condition number of $E$. 
Concluding remarks

This chapter concludes the thesis by summarizing conclusions and pointing out directions for future research.

7.1 Conclusions

Structural algorithms for the analysis and solution of DAE manipulate the equations to obtain formulations where features of interest become prominent. In particular, the structure algorithm reveals equations describing the solution manifold and an ODE that determines the solution trajectories on this manifold — in theory. In a practical setting, equations are not exactly known and implicit functions are generally difficult to compute with, but for equations in quasilinear form the situation seems brighter. Indeed, many before us have proposed structural algorithms which generalize how LTI DAE are analyzed using the shuffle algorithm. We provide an alternative view of these algorithms (which we call quasilinear shuffle algorithms) as specializations of the more general structure algorithm, but more importantly, we stress the need for an analysis of how sensitive the solution is to small perturbations in the equations. What makes the analysis of DAE fundamentally different from that of ODE is that small perturbations of a singular or near-singular matrix can cause arbitrarily large changes in the derivative of the solution. To understand such behavior, assumptions must be made that ensure that the arbitrarily large derivatives point in directions which make the solution converge quickly to a manifold where non-differential equations replace the arbitrariness. Numerical solution will then restrict the solution to this manifold, but this raises the question of how large errors are caused by this restriction. Answering this question is far outside the scope of this thesis, but we consider stressing this question a contribution in itself. As a first step in the direction of answering this question, however, we consider the simpler problem of showing that the errors in the
solution vanish with the size of the perturbations. Our analysis has been restricted to LTI DAE of low index, and has resulted in two alternative sets of assumptions which ensure the convergence.

As a sidetrack, a concisely defined class of forms of equations has been searched for forms which are invariant under the iterations of a particular quasilinear shuffle algorithm. This revealed no new forms that could be worth-while tailoring the algorithm to.

7.2 Directions for future research

Some directions for future research have been mentioned in earlier chapters, but the following short list contains some which we are particularly interested in developing:

- Searching for weaker conditions for the index 1 case.
- Extending the results to higher indices.
- Moving to time-variant forms, and eventually the quasilinear form.
- Quantifying the perturbation analysis, so that the ad hoc tuning of the smallness threshold for $\epsilon$ can be replaced by a threshold deriving from error tolerances that a user can understand.
- Formulating the perturbation problem for non-square systems.
Bibliography


Peter Fritzson, Peter Aronsson, Adrian Pop, David Akhvlediani, Bernhard Bachmann, David Broman, Anders Fernström, Daniel Hedberg, Elmin Jagudin, Håkan Lundvall, Kaj Nyström, Andreas Remar, and Anders Sandholm. OpenModelica system documentation — preliminary draft, 2006-12-14, for OpenModelica 1.4.3 beta. Technical report, Programming Environment Laboratory — PELAB, Department of Computer and Information Science, Linköping University, Sweden, 2006a.

Peter Fritzson, Peter Aronsson, Adrian Pop, David Akhvlediani, Bernhard Bachmann, David Broman, Anders Fernström, Daniel Hedberg, Elmin Jagudin, Håkan Lundvall, Kaj Nyström, Andreas Remar, and Anders Sandholm. OpenModelica users guide — preliminary draft, 2006-09-28, for OpenModelica 1.4.2. Technical report, Programming Environment Laboratory — PELAB, Department of Computer and Information Science, Linköping University, Sweden, 2006b.


A.1 Complexity calculation

Theorem A.1  
If the index reduction algorithm is used on an n-variable square DAE, with leading matrix and algebraic term both being polynomials of degree k, and if the differential index is $\nu \geq 1$, the degree of the computed index 0 DAE is bounded by

$$2^{n+\nu-2} k - \nu$$

This bound is tight for index 1 problems, and off by k for index 2. For higher indices, it is the limit in the sense

$$\frac{\text{true limit}}{2^{n+\nu-2} k - \nu} \rightarrow 1, \ n \rightarrow \infty$$

Proof: Adopt the convention that the degree of the zero polynomial is $-\infty$.

That the bound is tight for index 1 was shown in section [3.3.2]. Recall that argument. To show the bound for higher indices, two invariants of the DAE of the intermediate steps will be used. First note that the leading matrix will return to a form in which it can be divided into an upper part which is in row echelon form, and a lower part which is zeroed except for a full square block to the right. The only type of excursion from this form is during the steps following a differentiation, when the lower part, from being a full matrix, is zeroed column by column from the left until it attains the square form.

The first invariant is that for each row, the degree bound will be the same wherever it is not $-\infty$. Further, it will be the same also over the rows in the lower part of the DAE. Finally, with the exception of the step immediately following a differentiation, the degree of the algebraic terms will not be $-\infty$. This invariant holds initially as the degree bound,
in all of the leading matrix as well as in the algebraic term, is $k$. In case the lower part of the DAE is completely zeroed, differentiation of the corresponding algebraic terms will not break the invariant. In the immediately following step, the first row of the whole DAE will be used to eliminate the first column of the lower part, causing the degree bound of the algebraic terms to become the degree of the differentiated block plus the degree of the algebraic term of the first equation. Since the degree of the first equation’s leading row is the same as that of its algebraic term, the same degree bound will obtain for the lower leading matrix when the first column has been zeroed. For the remaining cases, it may be assumed that the degree bound is the same in the algebraic term as in the corresponding rows of the leading matrix. In these cases one row, be it taken from the upper part or be it the first row of the lower block, is used to eliminate the first non-zero column of the rows that will constitute the lower part in the next step. All these rows will have the same degree bounds before the elimination, and after the elimination the degree bound will be the sum of the old bound and the bound found in the pivot row. This concludes the proof of the first invariant.

The second invariant applies to the intermediate steps where the lower part of the DAE has the square form in the leading matrix. The proof will therefore consider two cases, depending on whether lower part is completely zeroed or being square. The zeroed case will not be considered complete until the square form is reobtained. The property in question is that after $\mu \geq 1$ differentiations, the degree in row $i$ is bounded by

$$2^{i+\mu-1} k - \mu + 1$$  \hspace{1cm} (A.1)

Note that once this has been shown for a row that has been made part of the upper triangular group of independent rows, subsequent increases in $\mu$ will not break the invariant since the bound is increasing with $\mu$ (that is, if the bound was good for some $\mu$, a higher value for $\mu$ will also yield an upper bound on the degree) However, the bound will not be tight when $\mu$ has been increased. To show that the invariant holds after the first differentiation, recall from the calculation in the introduction to section 3.3.2 that the degree bound is $2^{i-1} k$ just before the differentiation (note that this will satisfy (A.1) for $\mu > 1$). It remains to study the excursion. Assuming that the first zeroed row was number $i$, the algebraic terms to be differentiated will have the degree bound $2^{i-1} k$. Differentiation yields the bound $2^{i-1} k - 1$ (which proves the theorem for $\nu = 1$), and the elimination of column $j$ using row $j < i$ will add $2^{j-1} k$ to the bound, so once the square structure is reobtained, the bound will be

$$2^{i-1} k - 1 + \sum_{j=1}^{i-1} 2^{j-1} k = 2^{i-1} k - 1 + \left( 2^{i-1} - 1 \right) k = 2^i k - 1 - k$$

Using $\mu = 1$, this is $k + 1$ less than (A.1) (this gives the special result for index 2). To see that subsequent row elimination steps maintain the invariant, recall that this will make the bound in row $i + 1$ twice that for $i$, and for $\mu \geq 1$ it holds that $2 \cdot \left( 2^{i+\mu-1} k - \mu + 1 \right) < 2^{(i+1)+\mu-1} k - \mu + 1$. It remains to show that the invariant is maintained also after the excursion following differentiation. It is a similar calculation to the one above, but based on the non-tight bounds of (A.1) rather than the tight $2^{i-1} k$ used above. Let $\mu$ be the
number of differentiations before the current one. Then the degree bound is computed as

\[ 2^{i+\mu-1} k - \mu + 1 - 1 + \sum_{j=1}^{i-1} \left( 2^{j+\mu-1} k - \mu + 1 \right) \]

\[ \leq 2^{i+\mu-1} k + \left( 2^{i+\mu-1} - 1 \right) k - 2\mu + 1 \]

\[ \leq 2^{i+(\mu+1)-1} k - (\mu + 1) + 1 \]

The general degree bound result now follows by noting that the degree bound for the index 0 DAE will be obtained by taking \( \mu = \nu - 1 \) and maximizing \( i \) in the latter invariant, since this will give the degree of the algebraic term which will be differentiated to yield the row that makes the leading matrix full rank.

To see the limit part, note that \( \nu \) is fixed, so the total relative amount of overestimation can be written as a product of relative over-estimation in the different differentiation excursions times the product of over-estimation coming from the ordinary elimination steps. By taking \( n \) large enough, it can be ensured that the first \( \lfloor \phi n \rfloor \), \( \phi < 1 \) elimination steps are performed without over-estimation. The overestimation from ordinary elimination steps can thus be bounded by

\[ \prod_{i=\lfloor \phi n \rfloor + 1}^{n} \frac{2^{(i+1)+\mu-1} k - \mu + 1}{2 \cdot \left( 2^{i+\mu-1} k - \mu + 1 \right)} \]

\[ = \prod_{i=\lfloor \phi n \rfloor + 1}^{n} \left( 1 + \frac{\mu - 1}{2 \cdot \left( 2^{i+\mu-1} k - \mu + 1 \right)} \right) \]

\[ \leq \left( 1 + \frac{\mu - 1}{2 \cdot \left( 2^{\lceil \phi n \rceil +1+\mu-1} k - \mu + 1 \right)} \right)^{n-\lfloor \phi n \rfloor} \]

\[ < 1 + \epsilon \]

for any \( \epsilon > 0 \) if \( n \) is large enough and \( \phi \) close enough to 1. Since the number of differentiation excursions does not increase with \( n \), it only remains to show that the relative over-estimation in each such excursion can be made small. This basically follows by noting that the bound computed above will approach relative tightness as the number of elimination steps (doubling the degree bound) following the last increase in \( \mu \) tends to infinity, since most of the error stems from bounds computed for a lower \( \mu \) now being overestimated using a larger \( \mu \).
Notes on the implementation of the structure algorithm

The structure algorithm and the construction of a square set of index 1 equations described in chapter 3 has been implemented in Mathematica. This appendix describes that implementation from a software design perspective.

Notation. The notation used in this appending is a mix of the notation used in the rest of the thesis and elements from the Mathematica language. Mathematica’s use of equality and assignment signs has not been adopted; := is still used to denote usual assignments, and the sign = is avoided. The arrow \(\rightarrow\) is used to indicate what an expression evaluates to in Mathematica; that is, the arrow’s right hand side represents program output.

B.1 Caching

The implementation relies on cached values for many expressions, which requires clearing of these caches when turning to a new problem. The cache is cleared by invoking a single function, and this is normally done without the user noticing. Nevertheless, the user needs to be aware that caching is used in case the working DAE is changed in a non-standard fashion. This is considered a deficiency of Mathematica, since a more clever use of scopes would likely solve the problem by demanding that the user solve only one problem in each scope.
B.2 Package symbols

The implementation consists of, as is typical for Mathematica programs, a bunch of pattern-matching rules. These are placed in a package context as they should, and the complete list of exported symbols is

\(\text{thex}\) This is a list with typical “names” for the variables of the problem. It is very useful when one wants to see what the equations look like. Note that this is not a list of true Mathematica symbols.

\(\text{thexSymbols}\) This is like \(\text{thex}\), but the elements are symbols, although not as pretty.

\(h\) Function giving the equation residuals (after each index reduction step, and for every considered initial conditions).

\(a\) The leading matrix.

\(b\) The algebraic term, besides driving functions that enter affinely.

\(c\) The matrix of coefficients for how the driving functions that are not in \(b\) enter the equations.

\(v\) Driving functions.

\(\alpha\) Relative order, at a given point of initial conditions.

\textbf{automaticSetup} This brings general nonlinear DAE on the quasilinear form (see (2.14)).

\textbf{parameterBindings} A set of rules that assign values to symbolic constants in the equations.

\textbf{algebraicConstraints} These are all the algebraic constraints.

\textbf{usefulConstraints} This is \(\bar{\Phi}\).

\textbf{improveInitialPoint} A function providing a convenient way of adjusting guessed initial conditions to satisfy a set of derived algebraic constraints.

\textbf{statespaceEquations} This is \(\hat{h}\).

\textbf{index1Equations} This is the residuals of the produced square index 1 form.

\textbf{iterationMatrix} The iteration matrix for a simple 1-step BDF method.

\textbf{clearManualCache} This is a function for clearing the expression cache. It is invoked by \textbf{automaticSetup}.

B.3 Data driven interface

The entry to the implementation is data-driven; the symbol \(\alpha\) is defined so that evaluation of it causes the whole machinery to run until \(\alpha\) is found and can be returned.
B.4 Representation

In the choice between expressions or functions as the basic representation, functions were preferred because of the better modularity they offer. For example, the symbol $a$ is the function $a$, not the expression $a(x, t)$ for some globally defined symbols $x$ and $t$.

B.5 Example run

A typical use can look like this:

1. First, the problem is set up by defining the symbols $a$, $b$, $c$, and $v$ (these equations describe a mathematical pendulum, using Newton’s second law):

$$\text{automaticSetup[Flatten[\begin{pmatrix}
x'' = \lambda x \\
y'' = \lambda y - g \\
x^2 + y^2 = 1
\end{pmatrix}], \{x, y, \lambda\}, t, \{g \mapsto 10.0\}] }$$

By now, among other things, the expression cache is cleared, and we have (without invoking the structure algorithm machinery)

$$\text{thex} \longrightarrow \{x, \dot{x}, y, \dot{y}, \lambda\}$$

2. Guess initial conditions

$$t_0 := 0.0$$
$$x_{0,\text{guess}} := \{0.88, -0.01, 0.48, -0.1, -5.0\}$$

3. Compute algebraic constraints as if $x_{0,\text{guess}}$ was consistent, and evaluate the residuals:

$$\text{usefulConstraints} [x_{0,\text{guess}}, t_0, x_{0,\text{guess}}, t_0] \longrightarrow \{0.0048, -0.11, -20.\}$$

This shows that the guessed initial conditions need improvement. We must then tell which of the initial conditions we are most keen to keep unchanged, in the form of a list variables in decreasing order of importance:

$$x_{0,\text{second}} := \text{improveInitialPoint} [x_{0,\text{guess}}, t_0, x_{0,\text{guess}}, \text{thex}, \{x, \dot{y}, y, \dot{x}\}]$$

---

In real Mathematica, this would rather be done in a way that does not depend on the order of the components, but that requires some additional syntax which we prefer to keep out.
Here, the implementation reports that the initial conditions for \{ x, \dot{y} \} were unaltered. Now, \( x_{0,\text{second}} \rightarrow \{ 0.88, 0.054, 0.47, -0.1, 4.7 \} \). We may verify that this point satisfies the equations derived at \( (x_{0,\text{guess}}, t_0) \):

\[
\text{usefulConstraints}[x_{0,\text{guess}}, t_0, x_{0,\text{second}}, t_0] \rightarrow \{-2.8 \cdot 10^{-17}, 0.0, -8.9 \cdot 10^{-16}\}
\]

Hopefully, the new point generates the same algebraic constraints, because then the bootstrap procedure for finding initial conditions terminates. It does, since

\[
\text{usefulConstraints}[x_{0,\text{second}}, t_0, x_{0,\text{second}}, t_0] \rightarrow \{-2.8 \cdot 10^{-17}, 0.0, -8.9 \cdot 10^{-16}\}
\]

4. Request the index-1 form derived at the consistent initial conditions:

\[
\text{index1Equations}[x_{0,\text{second}}, t_0, \text{thex}, \text{Thread}[\text{thex'}], t] \rightarrow \begin{cases} 
-x\lambda + \dot{x}' = 0 \\
\dot{x} - x' = 0 \\
-1 + x^2 + y^2 = 0 \\
2x\dot{x} + 2y\dot{y} = 0 \\
2x^2\lambda - 2y(g - y\lambda) + 2\dot{x}^2 + 2\dot{y}^2 = 0 
\end{cases}
\]

Here,

\[
\text{Thread}[\text{thex}] \rightarrow \{ \dot{x}, \dot{x}', \dot{y}, \dot{y}', \dot{\lambda} \}
\]

is just a reasonable way of naming the \( u \)-variables. Thus note in the first equation that \( \dot{x}' \) is the time derivative of the variable \( \dot{x} \). Also note how the second equation then says that the time derivative of the variable \( x \) is the variable \( \dot{x} \).

5. In case one wants to know the index, it has been computed previously and now immediately evaluates. It is also possible to examine what expressions have been used for pivoting in the row reductions, and list the expressions which have been assumed rewritable to zero:

\[
\alpha[x_{0,\text{second}}, t_0] \rightarrow 3 \\
\text{pivots}[x_{0,\text{second}}, t_0, \text{thex}, t] \rightarrow \{1, 1, 1, 1, 1\} \\
\text{beZeros}[x_{0,\text{second}}, t_0, \text{thex}, t] \rightarrow \{\}
\]

6. The produced equations can readily be integrated numerically, one just has to write out the time parameter in the equations. Since \( \text{thex} \) is not a list of true Mathematica symbols, we use the predefined

\[
\text{thexSymbols} \rightarrow \{ x, Dx, y, Dy, \lambda \}
\]
instead. Equations specifying initial conditions are entered as

\[ eqns_{init} := \text{Thread}[\text{Through}[\text{thexSymbols}[0]]] = x_{0,\text{second}} \]

and the dynamic equations are entered as (the last part of the right hand side simply applies the rule defining a numerical value for \( g \))

\[ eqns_{ind1} := \text{index1Equations}[x_{0,\text{second}}, t_0, \text{Through}[\text{thexSymbols}[t]], \text{Through}[\text{Thread}[\text{thexSymbols}'[t]], t]] /. \text{parameterBindings} \]

This yields

\[
\begin{align*}
\quad & x[t] \lambda[t] \doteq Dx'[t] \\
\quad & 1.0 + Dy'[t] \doteq y[t] \lambda[t] \\
\quad & x[t]^2 + y[t]^2 \doteq 1 \\
\quad & Dx[t] x[t] + Dy[t] y[t] \doteq 0 \\
\quad & Dx[t]^2 + Dy[t]^2 + (x[t]^2 + y[t]^2) \lambda[t] \doteq 10.0 y[t]
\end{align*}
\]

which is in the form accepted by Mathematica’s NDSolve. The following requests a numerical solution from time \( t_0 \) to \( t_0 + 4 \):

\[ sol := \text{First}[\text{NDSolve}[\text{Join}[eqns_{init}, eqns_{ind1}], \text{thexSymbols}, \{t, t_0, t_0 + 4\}]] \]

7. Finally, the numerical quality of the solution may be investigated. Where the condition of iterationMatrix is OK, the solution is expected to be good, while singularity points may hint where the numerical solver could be in trouble. It should be mentioned that no numerical problems were reported when \( sol \) was computed. The condition number of the 1-step basic BDF method’s iteration matrix is shown in figure B.1. At the point where our iteration matrix looses rank, the jump in the solution’s error is more than one order of magnitude larger than the requested accuracy of 8 significant digits. Monitoring of the iteration matrix’ condition could have stopped the integration before this happened, but to compute a solution with error analysis past this point, we need a better understanding of perturbations in DAE.
Figure B.1: Quality of the computed solution for the pendulum equations. Upper: Condition of the basic iteration matrix. Lower: Difference between DAE solution and a high precision solution to an ODE formulation.
algebraic constraints, 30
algebraic equation, 4
algebraic term, 4
autonomous, 17
backward difference formulas, 32
balanced form, 13
BDF method, 32
bond graph, 16
boundary layer, 37
component-based model, 16
connection, 16
DAE, 5, 14
quasilinear, 15
DASPK, 35
DASSL, 35
derivative array, 23
differential index, 21
differential-algebraic equations, 14
drift, 35
driving function, 17
dummy derivatives, 26
elimination-differentiation index reduction, 20
equation
algebraic, 4
non-differential, 4
forms
balanced, 13
implicit ODE, 15
quasilinear, 15
state space, 11
fraction-free, 38
fraction-producing, 38
GELDA, 36
GENDA, 36
Hankel
norm, 13
singular value, 13
IDA, 36
implicit ODE, 15
index, 19
(qualifed), 26
differential, 21
perturbation, 25
strangeness, 26
index reduction, 20
initial value problem, 4
invariant form, 66
iteration matrix, 33
leading matrix, 4

longevity, 51

LTI, 5

model, 7
  component-based, 16
  residualized, 12
  truncated, 12
model class, 10
model reduction, 11
model structure, 10
non-differential equation, 4

ODE, 5
  implicit, 15

Pantelides’ algorithm, 26
perturbation index, 25
quasilinear form, 15

RADAU5, 36
residualization, 12
residualized model, 12
shuffle algorithm, 21
singular perturbation approximation, 15
square (DAE), 4
state (vector), 4
state-feedback matrix, 4
strangeness index, 26
structural zero, 38
structure algorithm, 29

truncated model, 12
truncation, 12
variable, 4


