University of Twente

Faculty of Electrical Engineering

On the relation between equation formulation of constrained systems and implicit numerical integration and optimization

P.A.J. Koenders

M.Sc. Thesis

Supervisors: Prof. dr. ir. J. van Amerongen

Dr. ir. P. C. Breedveld

Ir. G. Golo

June 2002

007CE2002

Control Laboratory

Electrical Engineering Department

University of Twente

P.O. Box 217

7500 AE Enschede

The Netherlands

Summary

The aim of this study is to establish the relation between the implicit numerical integration process and control or optimization problems. The iteration process of the implicit numerical methods and minimization of an error can be seen as control problems. An inventory of different formulations of constrained nonlinear systems is made. Their dynamical properties are analyzed and applicable numerical techniques for their simulation are investigated. After studying the available nonlinear control strategies, we propose to use index reduction by means of feedback linearization as an alternative for implicit integration methods. Since a system controlled by means of feedback linearization is marginally stable the state vector will drift off the constraint manifold. Thus, we always have to use the projection algorithm to stabilize the system. Feedback linearization and the projection algorithm can be applied straightforwardly and the solution can be found by means of simple explicit solvers. The implementation of the feedback linearization and projection algorithm approach is relatively simple. The basic numerical techniques, the implementations and the performance are compared. The latter comparison is made by simulation of a model of the Flyball governor (James Watt's regulator for steam engines). The simulations are performed in the software package 20-Sim. The results of the study show that feedback linearization and projection algorithm are a suitable alternative for the implicit integration method.

Table of contents

Nomenclature

$\boldsymbol{z}$	Vector of algebraic variables.	
$\boldsymbol{u}$	Vector of exitation variables.	
$\boldsymbol{\lambda}$	Vector of semi-state variables. (Lagrange multiplier)	
$\boldsymbol{x}$	State vector of energy variables: volume, charge, flux, displacement, momenta, etc.	
$H(\boldsymbol{x})$	Total energy of system.	[J]
$\bar{\boldsymbol{f}}^R, \bar{\boldsymbol{e}}^R$	Vectors power variables corresponding to R-ports.	
$\boldsymbol{\Phi}$	Vector of constraint variables.	
$\boldsymbol{M}$	Mass	
$\boldsymbol{p}$	Momentum of mass M	
$\boldsymbol{q}$	Position of mass M	

Preface

This project is the last part of my masters education at the university of Twente carried out at the department of Electrical Engineering in the Control Laboratory group. I followed the education part-time. Four days a week I was working at Witteveen+Bos in Deventer and one day a week I was working on my masters education. This is the reason why it took two years to complete this project. Peter Breedveld is the supervisor of the project. The first year we almost weekly discussed the progress of the project. The second year Goran Golo joined the progress discussions. The discussions where instructive and pleasant. I would like to thank them for their critical view and ideas that improved the work.

Peter Koenders
Deventer, 6 June 2002

1 Introduction

Computer simulation is an important engineering tool because it can be used to analyze engineering problems and it can be used for control design. Computer simulation is used in areas like chemical process engineering, robotic arm manipulators, ground vehicles, space aircraft and electrical circuits engineering. In general nonlinear constrained systems can be formulated in different ways depending on the numerical techniques the modeler wants to use. When the nonlinear constrained system is described by means of Differential Algebraic Equations (DAE) we can use an implicit integration methods to find the solution (Gear, 1971), (Petzold, 82). When the DAE is transformed into Ordinary Differential Equations (ODE) we can use explicit integration methods in order to fined the solution (Ascher, 1996), (Golo, 2000). The applied numerical technique, the way of implementation and the way that the equations are formulated determine the numerical efficiency, the reliability and the accuracy of the computer simulation.

The aim of this study is to formulate the implicit numerical integration process, i.e. the iteration process that minimizes the error, in terms of a control or rather optimization problem. The quantity to be integrated need to be adapted in such a way that the output satisfies the imposed constraint that causes the dependence between the storage ports. By comparing and evaluating the possible formulation techniques a relation is to be found between this problem and the way in which the discipline of modern control theory solves optimization problems.

Chapter 2 contains an inventory of the state of art ways of formulation of constrained nonlinear systems. The dynamical properties of the different ways of formulation are investigated and the applicable numerical techniques are investigated. Next the basic numerical techniques and implementation strategies of implicit integration methods are studied. The results are reported in chapter 3. The obtained knowledge will be used to develop control strategies. In chapter 4 the available nonlinear control strategies are studied in order to find a suitable method to solve the control and optimization problem of the implicit integration method. In chapter 5 the proposed control and optimization strategy is compared with the implicit integration method. After studying the available nonlinear control strategies we find that Index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm is a suitable alternative for implicit integration methods. Because a system that is controlled by means of feedback linearization is marginally stable, the state vector will drift off the constraint manifold. So, we always have to use the projection algorithm to stabilize the system. Feedback linearization and the projection algorithm can be applied straightforwardly and the solution can be found by means of simple explicit solvers. The implementation of the feedback linearization and projection algorithm approach is relatively simple. In chapter 4 we propose a systematic procedure to obtain and to implement the corresponding equations.

2 State of art in formulating systems with constraints

2.1 Introduction

There are a number of ways to make a graphical description of a physical system (Ideal Physical Model (IPM), Bond Graph Model (BGM), Block Diagram Model (BDM)). In this study we will take the bond graph model representation as an example. Using the bond graph technique we can make a graphical representation of physical systems that is called bond graph model. The bond graph model represents a multiport system involving energy flows (Paynter, 1961), (Breedveld, 1985). When we have obtained the bond graph model we can write down the algebraic state space equations of the junctions. The equations describing the bond graph model correspond to implicit port-controlled Hamiltonian systems (Golo, 2000). In this study we will describe the bond graph models as port-controlled Hamiltonian systems (PCH) (Van der Schaft, 2000). In this study non-linear constrained physical systems are concerned. Systematic procedures for constructing graphical models from constrained physical systems lead to models containing dependent states. The equations that are derived are Differential Algebraic Equations (DAE). DAE can be solved by means of implicit integration methods. Another option is to eliminate the dependent states from the graphical model. In case of a bond graph model the dependent states can be eliminated by transformation of junction structure (Golo, 2000). In general the equations that are derived from this modified bond graph model is an Ordinary Differential Equation (ODE) with an invariant (Ascher, 1996). This invariant can cause inaccuracy of the numerical simulations. In this chapter we will discuss methods to formulate and modify bond graph models that are obtained from constrained mechanical systems and how equations can be derived and modified.

The way that the bond graph model and the equations are formulated determines the dynamical properties of the system. How a constrained system is formulated depends on the needs of the modeler. For example, if we need a simulation model for real-time simulation, the constrained system should be transformed into an explicit system.

The dynamical properties are usually expressed by the eigenvalues of the linearized system. In other words, the position of the eigenvalues determines what type of numerical techniques should be used. For example, if the real parts of the eigenvalues differ largely (such type of system is called *stiff system*) then the explicit integration techniques don't give good results.

Another way to characterize a constrained system is the index of a system. The index of a system is a measure how far a constrained system is from an ODE. If the index of a system is either

one or two, and some additional conditions are satisfied, then implicit numerical techniques give good results. Otherwise, techniques for index reduction have to be applied.

In this chapter we will discuss the definition of the index of a system and the definition of a *stiff system*. In order to give some insight in the dynamical properties of the systems in relation to the method of formulation we will use a linear mechanical system as an example.

2.2 Implicit port-controlled Hamiltonian description of systems

When we have obtained a bond graph model from a constrained mechanical system, we can write down the equations. The port-controlled Hamiltonian (PCH) description of a bond graph model is given by (Golo, 2000):

$$\dot{\boldsymbol{x}} = \boldsymbol{J}^{C}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^{R} + \boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.1}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}), \tag{2.2}$$

$$\bar{\boldsymbol{e}}^{R} = -(\boldsymbol{G}^{C,R}(\boldsymbol{x}))^{T}\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}^{R,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.3}$$

$$\overline{\Omega}(\bar{\boldsymbol{f}}^{R},\bar{\boldsymbol{e}}^{R},\boldsymbol{x}) = 0, \tag{2.4}$$

$$\boldsymbol{y} = -(\boldsymbol{G}^{C,U}(\boldsymbol{x}))^{T}\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) - (\boldsymbol{G}^{R,U}(\boldsymbol{x}))^{T}\bar{\boldsymbol{f}}^{R} + \boldsymbol{J}^{U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.5}$$

where $\boldsymbol{J}^{C}(\boldsymbol{x})$ and $\boldsymbol{J}^{U}(\boldsymbol{x})$, are skew-symmetric matrices, $\boldsymbol{G}(\boldsymbol{x})$ is a full rank matrix for $\forall \boldsymbol{x} \in \boldsymbol{\chi}$. $\boldsymbol{x}$ is the state vector (vector of energy variables: volume, charge, flux, displacement, momenta etc.). $\boldsymbol{\chi}$ is the manifold of variables. $H(\boldsymbol{x})$ is the total energy of system, $\bar{\boldsymbol{f}}^{R},\bar{\boldsymbol{e}}^{R}$ are the power variables corresponding to R-type of ports and $\overline{\boldsymbol{\Omega}}$ is an algebraic function such that every pair $\bar{\boldsymbol{f}}^{R},\bar{\boldsymbol{e}}^{R}$ satisfying (2.4) satisfies also the inequality $\bar{\boldsymbol{f}}^{R}(\bar{\boldsymbol{e}}^{R})^{T} \geq 0$.

In many cases the map $\overline{\Omega}$ is linear and when we have an energy dissipating system we can express (2.4) as

$$\bar{\boldsymbol{f}}^{R} = \boldsymbol{R}(\boldsymbol{x})\bar{\boldsymbol{e}}^{R}$$

Where $\boldsymbol{R}(\boldsymbol{x})$ is a positive definite matrix, then system (2.1)-(2.4) becomes

$$\begin{aligned}\dot{\boldsymbol{x}} = (\boldsymbol{J}^{C}(\boldsymbol{x}) - \boldsymbol{G}^{C,R}(\boldsymbol{x})\boldsymbol{R}(\boldsymbol{x})(\boldsymbol{G}^{C,R}(\boldsymbol{x}))^{T})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \\ (\boldsymbol{G}^{C,U}(\boldsymbol{x}) - \boldsymbol{G}^{C,R}(\boldsymbol{x})\boldsymbol{R}(\boldsymbol{x})\boldsymbol{G}^{R,U}(\boldsymbol{x}))\boldsymbol{u},\end{aligned} \tag{2.6}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) \tag{2.7}$$

Let $\boldsymbol{G}^{\perp}(\boldsymbol{x})$ be a maximum rank matrix so that $\boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}(\boldsymbol{x}) = 0$. Sine $\boldsymbol{G}(\boldsymbol{x})$ is a constant rank matrix then $\boldsymbol{G}^{\perp}(\boldsymbol{x})$ is also a constant rank matrix. Premultiplying (2.1) by $\boldsymbol{G}^{\perp}(\boldsymbol{x})$ the following system of equations is obtained.

$$\boldsymbol{G}^{\perp}(\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{J}^{C}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^{R} + \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.8}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) \tag{2.9}$$

$$\bar{\boldsymbol{e}}^{R} = -(\boldsymbol{G}^{C,R}(\boldsymbol{x}))^{T}\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}^{R,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.10}$$

$$\overline{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \bar{\boldsymbol{e}}^{R}, \boldsymbol{x}) = 0. \tag{2.11}$$

Similarly if $\bar{\boldsymbol{f}}^{R} = \boldsymbol{R}(\boldsymbol{x})\bar{\boldsymbol{e}}^{R}$ the system (2.8)-(2.11) becomes

$$\begin{aligned}\boldsymbol{G}^{\perp}(\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{G}^{\perp}(\boldsymbol{x})(\boldsymbol{J}^{C}(\boldsymbol{x}) - \boldsymbol{G}^{C,R}(\boldsymbol{x})\boldsymbol{R}(\boldsymbol{x})(\boldsymbol{G}^{C,R}(\boldsymbol{x}))^{T})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \\ \boldsymbol{G}^{\perp}(\boldsymbol{x})(\boldsymbol{G}^{C,U}(\boldsymbol{x}) - \boldsymbol{G}^{C,R}(\boldsymbol{x})\boldsymbol{R}(\boldsymbol{x})\boldsymbol{G}^{R,U}(\boldsymbol{x}))\boldsymbol{u},\end{aligned} \tag{2.12}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) \tag{2.13}$$

2.3 Index of system and stiffness of system

Index of system:

The index of constrained system characterizes the behavior of the numerical solution and is a measure of the singularity, how much constrained system differs from an ODE. In the following text the definition of the *local differential index*, *index two system* and *index one system* is given (Ascher, 1998);

Local differential index *Consider a system described by the following implicit equation*

$$\boldsymbol{F}(\dot{\boldsymbol{z}}, \boldsymbol{z}, \boldsymbol{u}) = 0 \tag{2.14}$$

Equation (2.14) *has a local differential index m at the point* $(\boldsymbol{z}_0, \boldsymbol{u}_0)$ *if m is the minimal number such that there exists a neighborhood of the point* $(\boldsymbol{z}_0, \boldsymbol{u}_0)$ *in which the system of equations*

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

$$\frac{d\boldsymbol{F}(\dot{\boldsymbol{z}}, \boldsymbol{z}, \boldsymbol{u})}{dt} = 0,$$

$$\vdots$$

$$\frac{d^{m}\boldsymbol{F}(\dot{\boldsymbol{z}}, \boldsymbol{z}, \boldsymbol{u})}{dt^{m}} = 0,$$

can be uniquely solved for $\dot{\boldsymbol{z}}$, *as a function of* $\boldsymbol{z}$ *and* $\boldsymbol{u}$ *only.*

When (2.3) is inserted in (2.4), the set of equations (2.1)-(2.4) becomes

$$\dot{\boldsymbol{x}} = \boldsymbol{J}^{C}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^{R} + \boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.15}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}), \tag{2.16}$$

$$\tilde{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \boldsymbol{x}, \boldsymbol{u}) = 0. \tag{2.17}$$

Proposition [Index two system] *Consider system described by (2.15)-(2.17). Assume that for* $\boldsymbol{x} = \boldsymbol{x}_0$ *and* $\boldsymbol{u} = \boldsymbol{u}_0$ *equation (2.16) is satisfied and that there exists* $\bar{\boldsymbol{f}}_0^{R}$ *such that (2.17) is satisfied. If*

$$det\left(\frac{\partial}{\partial \boldsymbol{x}^{T}}\left(\boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x})\right)\boldsymbol{G}(\boldsymbol{x})\right)_{x=x_0} \neq 0, \tag{2.18}$$

$$det\left(\frac{\partial \tilde{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \boldsymbol{x}, \boldsymbol{u})}{\partial (\bar{\boldsymbol{f}}^{R})^{T}}\right)_{\substack{x=x_0 \\ \bar{f}^{R}=\bar{f}_0^{R} \\ u=u_0}} \neq 0, \tag{2.19}$$

then the system (2.15)-(2.17) has a differential index two at the point $(\boldsymbol{x}_0, \bar{\boldsymbol{f}}_0^{R}, \boldsymbol{\lambda}_0, \boldsymbol{u}_0)$.

Proof: In this case $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{f}^{R})$. Differentiating of (2.17) gives

$$\frac{\partial \tilde{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \boldsymbol{x}, \boldsymbol{u})}{\partial (\bar{\boldsymbol{f}}^{R})^{T}}\dot{\bar{\boldsymbol{f}}}^{R} + \boldsymbol{L}_1(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \bar{\boldsymbol{f}}^{R}) = 0$$

Now it is clear that if the condition (2.19) is satisfied then the last equation can be solved for $\dot{\bar{\boldsymbol{f}}}^{R}$.

Differentiating *(2.16)* gives

$$\frac{\partial}{\partial \boldsymbol{x}^{T}}\left(\boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x})\right)\boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{L}_2(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \dot{\boldsymbol{f}}^{R}) = 0$$

If condition (2.18) is satisfied then the last equation can be uniquely solved for $\boldsymbol{\lambda}$. Therefor for the given value of $\boldsymbol{x}_0, \bar{\boldsymbol{f}}_0^{R}, \boldsymbol{u}_0, \boldsymbol{\lambda}_0$ can be uniquely computed. Differentiating of the last equation gives

$$\frac{\partial}{\partial \boldsymbol{x}^{T}}\left(\boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x})\right)\boldsymbol{G}(\boldsymbol{x})\dot{\boldsymbol{\lambda}} + \boldsymbol{L}_3(\boldsymbol{x}, \boldsymbol{\lambda}, \bar{\boldsymbol{f}}^{R}, \boldsymbol{u}, \dot{\boldsymbol{u}}) = 0.$$

It is clear that if the condition (2.18) is satisfied then the last equation can be uniquely solved for $\dot{\boldsymbol{\lambda}}$.

So the index of (2.15)-(2.17) is two.

Similarly, we can consider the following system ($\boldsymbol{\lambda}$ is eliminated).

$$\boldsymbol{G}^{\perp}(\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{J}^{C}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^{R} + \boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{2.20}$$

$$0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) \tag{2.21}$$

$$\tilde{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \boldsymbol{x}, \boldsymbol{u}) = 0. \tag{2.22}$$

Proposition [Index one system] *Consider the system described by (2.20)-(2.22). Assume that for* $\boldsymbol{x} = \boldsymbol{x}_0$ *and* $\boldsymbol{u} = \boldsymbol{u}_0$ *equation (2.21) is satisfied and that there exists* $\bar{\boldsymbol{f}}_0^{R}$ *such that (2.22) is satisfied. If the conditions (2.18) and (2.19) are satisfied then the system (2.20)-(2.22) has a differential index one at the point* $(\boldsymbol{x}_0, \bar{\boldsymbol{f}}_0^{R}, \boldsymbol{u}_0)$.

Proof: In this case $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{f}^{R})$. The part of the proof regarding the calculation of $\dot{\bar{\boldsymbol{f}}}^{R}$ is the same as in the proof of the index-two system. Now, we concentrate on calculation of $\dot{\boldsymbol{x}}$. Differentiating (2.21) and regrouping the derivatives of the left side gives the following

$$\begin{bmatrix} \boldsymbol{G}^{\perp}(\boldsymbol{x}) \\ \dfrac{\partial}{\partial \boldsymbol{x}^{T}}\left(\boldsymbol{G}^{T}(\boldsymbol{x})\dfrac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x})\right) \end{bmatrix}\dot{\boldsymbol{x}} = \boldsymbol{L}_4(\boldsymbol{x}, \bar{\boldsymbol{f}}^{R}, \boldsymbol{u}).$$

If the condition (2.18) is fulfilled then the rank of the matrix on the left side is full $\forall \boldsymbol{x} \in \boldsymbol{\chi}$. For proving this, the following identity is used

$$rank\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} = rank(\boldsymbol{A}) + rank\left(\boldsymbol{B}\boldsymbol{A}^{\perp}\right)$$

where $\boldsymbol{A}^{\perp}$ is the maximal rank matrix such that $\boldsymbol{A}\boldsymbol{A}^{\perp} = \boldsymbol{0}$

Stiffness of system:

The *stiffness* of a system is defined by Lambert (1980). This definition is modified by Van Dijk, (1994).

Lambert (1980), defines *stiff systems* in the following way:

Consider the common system of ODE of the form:

$$\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{y},\ \boldsymbol{\phi}(x) \in \Re^m \tag{2.23}$$

This system is said to be *stiff* if:

1. $Re(\lambda_i) < 0, \quad i = 1,2,..,n$

and

2. $$\frac{\max_i \left| Re(\lambda_i) \right|}{\min_i \left| Re(\lambda_i) \right|} = S >> 1 ,$$

where λ_i is the eigenvalue of matrix $\boldsymbol{A} \in \Re^m$, $Re(\lambda_i)$ is the real part of λ_i and S is called the *"stiffness ratio"*.

A nonlinear system $\dot{\boldsymbol{y}} = f(x, \boldsymbol{y})$ is said to be stiff, on an interval I of x, if for $\forall x \in I$ the eigenvalues $\lambda(x)$ of the Jacobian $\partial \boldsymbol{f} / \partial \boldsymbol{y}$ satisfy *1* and *2* above.

The previous definition is refined by Van Dijk, as follows:

A system described by (2.23) is *stiff* if $\left| h.\partial \boldsymbol{f} / \partial y \right| >> 1$ where h represents the step size of the applied integration method.

2.4 Formulation, dynamical behavior and computational efficient

Systematic procedures for constructing bond graph models from physical systems lead to bond graph models containing dependent states. The dependent states can be eliminated by means of model modification or symbolic reduction of the constraint. The equation that is derived from the bond graph model with constraints is usually index 1 or 2. The equation that is derived from the bond graph model without dependent states is index 0. An explicit integration method can be used is the index is reduced to zero. When the index is one or two we must use an implicit integration method. The *stiffness* is also a property that determines the choice of integration method. When the index of the system is zero and the system is *stiff* we must use an implicit integration method. Figure 1 shows the graphical representation of the formulation of constrained systems.

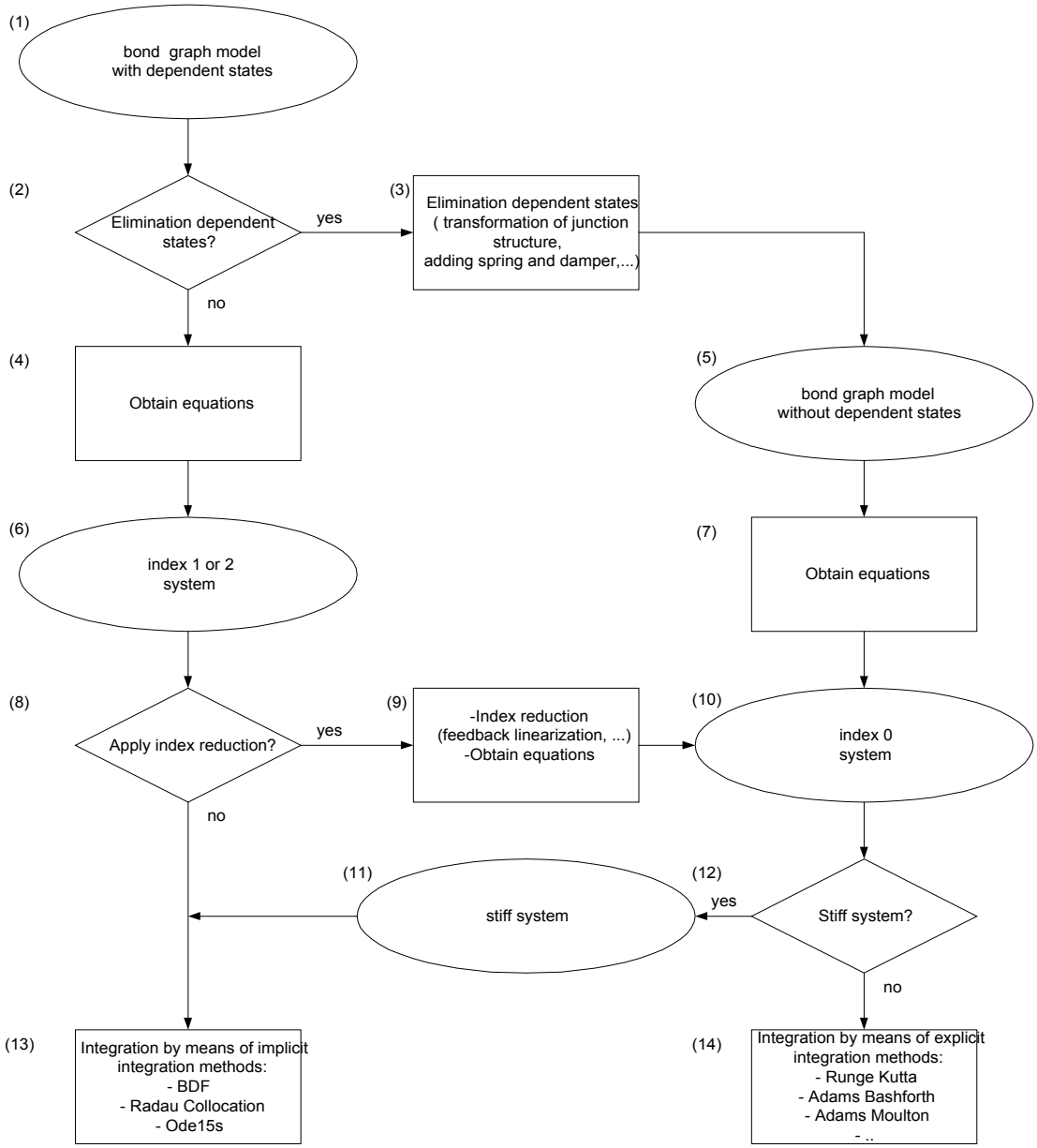


Figure 1 : Graphical representation of the formulation of constrained system

The formulation of constrained mechanical systems.

- (1) Bond graph model with dependent states:

  The kinematc constraints will cause dependent states in the bond graph model.

- (5) Bond graph model without dependent states:

  The kinematic constraints that are present in the bond graph model can be eliminated by means of, for example, geometric junction transformation (Golo, 2000) or adding spring and damper (3).

- (6) Index 1 or 2 system:

  The equation that is derived from the bond graph model with dependent states is usually index 1 or 2. This DAE can be solved by means of implicit integration (13) methods like:

- BDF (Gear, 1971) the Dassl (Petzold, 1982) implementation is used in 20 Sim and the Ode15s implementation (Shampine,2000) is used in Matlab.
- Radau Collocation (Ascher, 1998).

- (10) Index 0 system:
  The equation obtained from the bond graph model without a dependent state is index 0. The index 1 or 2 system (6) can be transferred into an index 0 PCH system by means of index reduction (9). An example of index reduction is feedback linearization that will be discussed in chapter 4. The index 0 system can be solved by means of explicit integration methods (14) like Runge Kutta, Adams Bashforth, Adams Moulton (Ascher, 1998), (Gear, 1971).
- (11) *Stiff* system:
  A *stiff* index 0 system can't be solved by means of an explicit integration methods (Ascher, 1998). A *stiff* index 0 system has to be solved by means of an implicit integration method.

The Flow chart that is presented in Figure 1 can help the modeler in order to find the needed formulation strategy. The flow chart could be implemented in a Decision Support System (DSS).

2.5 Examples

In this subchapter we will show how to obtain and modify a bond graph model and how to derive equations. We will use a gear wheels system as an example. The bond graph model, which is obtained from the concerned Gear wheels system, has a dependent state. The derived equations can be formulated as an index 1 or as an index 2 system. The dependent states can be eliminated by adding spring and damper or by symbolic reduction of the constraint. The dynamical behavior will be investigated by analyzing the location of the eigenvalues.

2.5.1 Bond graph model with dependent state, derive index 1 and 2 system

Figure 2 shows the linear mechanical system that consists of winches. The winches are transitionally fixed and can rotate only. The masses are subject to gravity and can be translated vertically. There is an external force *F* that acts on M_2. The winches and the cables are modeled as massless. The loads are modeled as point masses. The masses are kinematically coupled. There is one kinematic constraint. The difference between the velocities of the masses M_1 and M_2 is $\frac{R_1}{r_1}v_1 = -\frac{R_2}{r_2}v_2$ and therefore the length of the cable between the masses is constant.

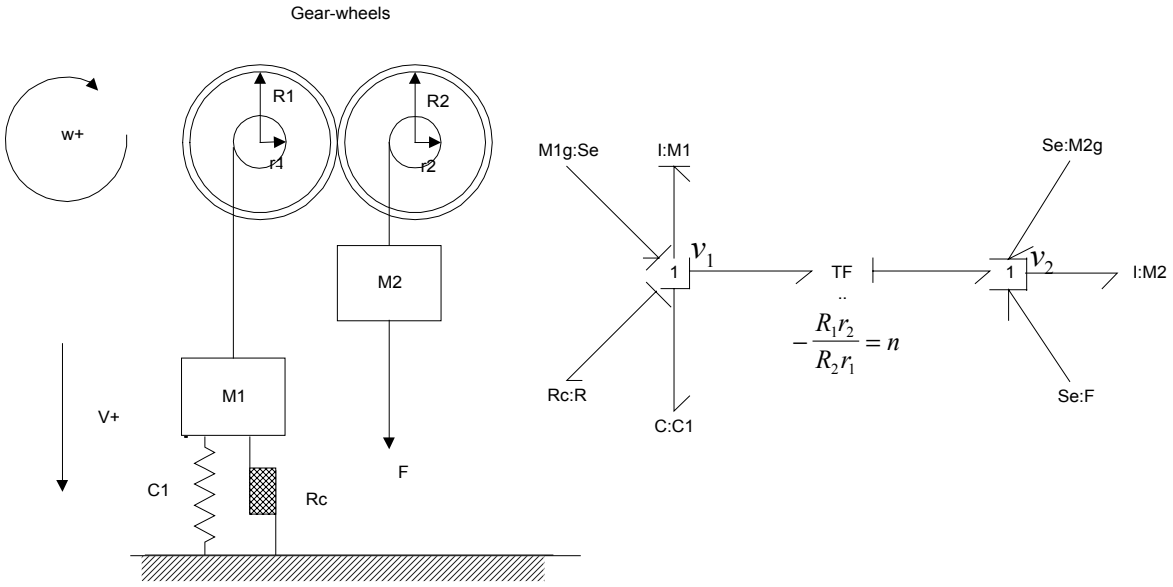


Figure 2 : Gear Wheels: Ideal physical model and bond graph model.

The state vector is

$$\boldsymbol{x}^T = \begin{bmatrix} p_1 & x & p_2 \end{bmatrix} \in \boldsymbol{\chi},$$

where p_1 and p_2 are the momenta of masses M_1 and M_2, respectively x is the position of mass M_1 and $\boldsymbol{\chi} = \Re^3$. The state x is independent. One of the states p_1 and p_2 is dependent and the other is independent. In this case the state p_1 is chosen independent and p_2 is chosen dependent.

The total energy is given by.

$$H(\boldsymbol{x}) = \frac{1}{2}\frac{p_1^2}{M_1} + \frac{1}{2}\frac{p_2^2}{M_2} + \frac{1}{2}\frac{1}{C_1}x^2$$

The equations (2.1)-(2.4) describe this system where,

$$\boldsymbol{J}^C(\boldsymbol{x}) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{G}(\boldsymbol{x}) = \begin{bmatrix} n \\ 0 \\ -1 \end{bmatrix}, \boldsymbol{G}^{C,R}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{2.24}$$

$$\boldsymbol{G}^{R,U}(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \boldsymbol{G}^{C,U}(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} M_1 g \\ M_2 g \\ F \end{bmatrix}, \boldsymbol{G}^{\perp}(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 & n \\ 0 & 1 & 0 \end{bmatrix} \tag{2.25}$$

$$\overline{\Omega}(\bar{\boldsymbol{f}}^R, \bar{\boldsymbol{e}}^R, \boldsymbol{x}) = \bar{\boldsymbol{e}}^R - R_c^{-1}\bar{\boldsymbol{f}}^R. \tag{2.26}$$

Index 2 system:

Because (2.18) and (2.19) are satisfied $\forall \boldsymbol{x} \in \not\chi$ the equations can be formulated as an index 2 system, given by

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -R_c & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{\partial H(\boldsymbol{x})}{\partial \boldsymbol{x}} + \begin{bmatrix} n \\ 0 \\ -1 \end{bmatrix} \boldsymbol{\lambda} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} M_1 g \\ M_2 g \\ F \end{bmatrix}$$

$$0 = \begin{bmatrix} n & 0 & -1 \end{bmatrix} \frac{\partial H(\boldsymbol{x})}{\partial \boldsymbol{x}}.$$

Index 1 system:

Since $\boldsymbol{G}^{\perp}(\boldsymbol{x})$ is a maximum rank matrix such that $\boldsymbol{G}^{\perp}(\boldsymbol{x})\boldsymbol{G}(\boldsymbol{x}) = 0$ then the previous system can be rewritten as

$$\begin{bmatrix} 1 & 0 & n \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{x} \\ \dot{p}_2 \end{bmatrix} = \begin{bmatrix} -R_c & -1 & 0 \\ 1 & 0 & 0 \\ n & 0 & -1 \end{bmatrix} \begin{bmatrix} \frac{p_1}{M_1} \\ \frac{x}{C_1} \\ \frac{p_2}{M_2} \end{bmatrix} + \begin{bmatrix} 1 & n & n \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} M_1 g \\ M_2 g \\ F \end{bmatrix} \tag{2.27}$$

Here, the first two rows represent the differential equations and the third row represents the algebraic equation.

Eigenvalues:

System (2.27) is described by $\boldsymbol{E}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$. Suppose the rank of $\boldsymbol{E}$ is *m* and $m \leq n$ where *n* is the dimension if the matrix $\boldsymbol{A}$. Then *m* eigenvalues of the system $\boldsymbol{E}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ are the root of the following equation

$$det(\boldsymbol{\lambda E} - \boldsymbol{A}\boldsymbol{x}) = 0$$

The rest *n-m* are placed in $-\infty$.

In this case

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 & n \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{A} = \begin{bmatrix} \frac{-R_c}{M_1} & -\frac{1}{C_1} & 0 \\ \frac{1}{M_1} & 0 & 0 \\ \frac{M_2}{M_1} n & 0 & -1 \end{bmatrix}$$

The eigenvalues of the system are the solution of the polynomial equation

$$\left(1+n^2\right)\lambda^2+\left(\frac{R_c}{M_1}\right)\lambda+\frac{1}{M_1C_1}=0$$

Since *m=2* and *n=3* the third eigenvalue is $-\infty$.

$\lambda_1=-\infty \quad \lambda_{2,3}=-0{,}0025\pm 1{,}118.i$

The numerical values, which are used to calculate the eigenvalue, are; $n=-\frac{R_1r_2}{R_2r_1}=1$,

$M_1=M_2=$ *5 kg, Rc = 0.05 N/*m^2*, C1 = 0.08 N/m, Cp = 1E-5 N/m and Rp = 1000N/*m^2*, F=1 N, g=1 N/kg..*

Stiffness ratio:

We can calculate the *stiffness ratio* by means off

$$S=\frac{\max\limits_i\left|Re(\lambda_i)\right|}{\min\limits_i\left|Re(\lambda_i)\right|}=\frac{\infty}{0{,}0025}=\infty$$

Because *S>>1*, (2.27) is a *stiff system*.

2.5.2 Elimination dependent state by adding spring and damper

In this case the added spring force F_{Cp} and the added damping forces F_{Rp} control the constraints. The forces $F_{Cp}+F_{Rp}$ control in such a way that the velocity constraints are satisfied. A system can contain several dependent states, for every dependent state that we want to eliminate we must add a spring-damper combination. Every spring that we add causes an extra state variable, $\boldsymbol{x}_A$ is the vector of additional state variables. Now the total state $\boldsymbol{x}_T$ vector becomes

$\boldsymbol{x}_T^T=\left[\boldsymbol{x} \quad | \quad \boldsymbol{x}_A\right]\in\chi_T$,

where $\boldsymbol{x}$ is the original state vector and $\chi_T=\Re^4$. When the energy of the spring is added, the total energy becomes

$H_T(\boldsymbol{x}_T)=H(\boldsymbol{x})+H_A(\boldsymbol{x}_A)$,

where

$$H_A(\boldsymbol{x}_A)=\frac{1}{2}\boldsymbol{x}_{Cp}^TC_p^{-1}\boldsymbol{x}_{Cp}$$

and where C_p^{-1} is the vector of spring constants. Now, the control input λ of system (2.1)-(2.5) can be written as

$$\lambda=\frac{\partial H_A(\boldsymbol{x}_A)}{\partial x_A}+R_{Cp}\dot{\boldsymbol{x}}_A=\boldsymbol{x}_{Cp}C_p^{-1}+\dot{\boldsymbol{x}}_{Cp}R_{Cp} \tag{2.28}$$

This controller is equivalent to a PD controller with respect to position x_{Cp}, where C_p^{-1} is equal with the proportional factor and R_{Cp} (vector of damping constants) is equal with the differential factor. The spring constants C_p should be chosen small enough to obtain a acceptable violation of the velocity constraint.

Example:

We will eliminate dependent state p_2 from the Gear-wheel model by means of adding spring and damper. The Gear wheel model contains only one dependent state. The forces $F_{Cp} + F_{Rp}$ control in such a way that $-nv_1+v_2$ is constrained to zero, where $n = -\frac{R_1 r_2}{R_2 r_1}$. Figure 3 shows that the dependent state is eliminated.

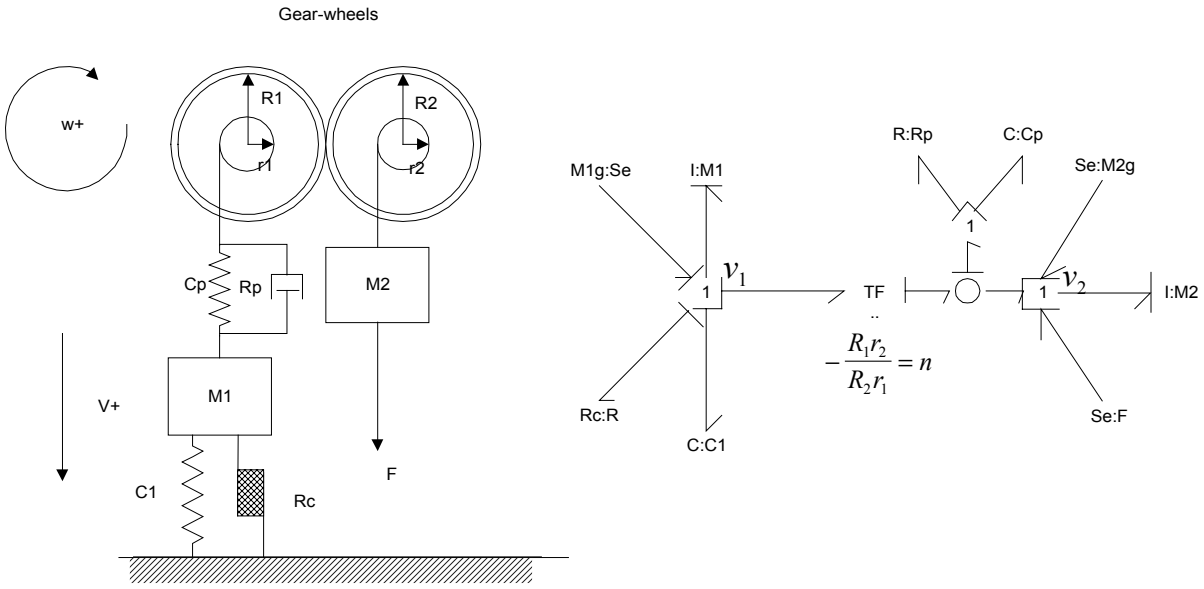


Figure 3 : Gear-wheels: Parasitic-elements approach used to eliminate the constraint

Index 0 system:

When we add a spring and damper to eliminate dependent state p_2 the total state vector becomes

$$\boldsymbol{x}_T^T = \begin{bmatrix} p_1 & x & p_2 & x_{Cp} \end{bmatrix} \in \mathcal{X}$$

where $\mathcal{X} = \Re^4$. When we use (2.28) to calculate control input $\boldsymbol{\lambda}$, (2.1)-(2.4) becomes an index-0 system, given by

$$\begin{bmatrix} \dot{p}_1 \\ \dot{x} \\ \dot{p}_2 \\ \dot{x}_{Cp} \end{bmatrix} = \begin{bmatrix} \frac{-Rc - n^2 R_{cp}}{M_1} & -\frac{1}{C_1} & \frac{nR_{cp}}{M_2} & n\frac{1}{Cp} \\ \frac{1}{M_1} & 0 & 0 & 0 \\ \frac{n^2 R_{cp}}{M_1} & 0 & -\frac{nR_{cp}}{M_2} & -\frac{1}{C_p} \\ \frac{n}{M_1} & 0 & -\frac{1}{M_2} & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ x \\ p_2 \\ x_{Cp} \end{bmatrix} + \begin{bmatrix} M_1 g \\ 0 \\ M_2 g + F \\ 0 \end{bmatrix}. \tag{2.29}$$

Eigenvalues:

The eigenvalues of (2.29) can be calculated by solving the following equation

$det(\boldsymbol{\lambda I}_n - \boldsymbol{Ax}) = 0$.

Where,

$$\boldsymbol{A} = \begin{bmatrix} \frac{-Rc - n^2 R_{cp}}{M_1} & -\frac{1}{C_1} & \frac{nR_{cp}}{M_2} & n\frac{1}{Cp} \\ \frac{1}{M_1} & 0 & 0 & 0 \\ \frac{n^2 R_{cp}}{M_1} & 0 & -\frac{nR_{cp}}{M_2} & -\frac{1}{C_p} \\ \frac{n}{M_1} & 0 & -\frac{1}{M_2} & 0 \end{bmatrix}.$$

When these matrices are substituted and when the numerical values of the Gear-wheel model are substituted the polynomial becomes.

$$\lambda^4 + 400\lambda^3 + 40004{,}5\lambda^2 + 700\lambda + 50000 = 0$$

The roots of this polynomial are,

$$\lambda_{1,2} = -200 \pm 0{,}5i \quad \lambda_{3,4} = -0{,}0025 \pm 1{,}118i$$

Stiffness ratio:

We can calculate the *stiffness ratio* by means off

$$S = \frac{\max_i \left| Re(\lambda_i) \right|}{\min_i \left| Re(\lambda_i) \right|} = \frac{200}{0{,}0025} = 80.000$$

Because *S>>1*, (2.29) is a *stiff system*.

2.5.3 Elimination dependent state by symbolic index reduction

In Figure 4 the inertia with the derivative causality is transferred to the inertia with integral causality and the two masses are replaced by an equivalent element. In the case of this simple example the transformation can be done by hand. For more complex models it is possible to use mathematical software to manipulate equations. In both cases extra manipulation of the equation is needed.

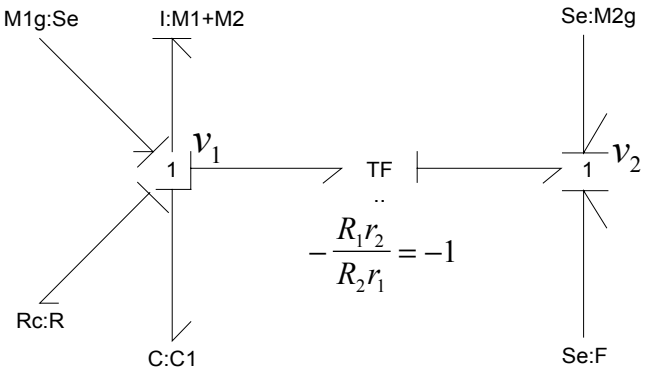


Figure 4 : Gear-wheels: The transfer of dependent storage elements is used to eliminate the constraint

Index 0 system:

The general equation is given by

$$\begin{bmatrix} \dot{p}_n \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{-Rc}{M_1 + M_2} & -\frac{1}{C_1} \\ \frac{1}{M_1 + M_2} & 0 \end{bmatrix} \begin{bmatrix} p_n \\ x \end{bmatrix} + \begin{bmatrix} M_1 g + M_2 g + F \\ 0 \end{bmatrix}.$$

Eigenvalues:

The eigenvalues can be calculated by solving the following equation

$det(\lambda \boldsymbol{I}_n - \boldsymbol{Ax}) = 0$.

Where,

$$\boldsymbol{A} = \begin{bmatrix} \frac{-Rc}{M_1 + M_2} & -\frac{1}{C_1} \\ \frac{1}{M_1 + M_2} & 0 \end{bmatrix}.$$

When these matrices are substituted and when the numerical values of the Gear-wheel model are substituted the polynomial becomes.

$$\lambda^2 + 0{,}005\lambda + 1{,}25 = 0$$

The roots of this polynomial are,

$$\lambda_{1,2} = -0{,}0025 \pm 1{,}118i$$

Stiffness ratio:

We can calculate the *stiffness ratio* by means off

$$S = \frac{\max_i \left| Re(\lambda_i) \right|}{\min_i \left| Re(\lambda_i) \right|} = \frac{0{,}0025}{0{,}0025} = 1$$

Because *S=1*, the system is a non-*stiff system*.

2.6 summary

The way the system is formulated determines the eigenvalues and it determines the index of the system and therefore the integration methods that can be used to solve the problem. The Flow chart that is presented in Figure 1 can help the modeler in order to find the needed formulation strategy. The flow chart could be implemented in a Decision Support System (DSS). In this chapter a number of formulations of a constrained system are shown. In Table 1 formulation method, the index, the eigenvalues, the system dynamics and the suitable integration methods are shown.

Table 1 : System dynamics Gear-wheel model

Formulation method	**Index**	**Eigenvalues**	**Dynamic system behavior**	**Suitable Integration methods**
Straight forward	1	$\lambda_1 = -\infty$, $\lambda_{2,3} = -0{,}0025 \pm 1{,}118i$	stiff	Implicit integration methods: BDF, Radau collocation
Elimination dependent state (adding spring *Cp* and damping *Rp*)	0	$\lambda_{1,2} = -200 \pm 0.5i$ $\lambda_{3,4} = -0{,}0025 \pm 1{,}118i$	stiff	- Implicit integration methods: BDF, Radau collocation. - Large C_p: explicit integration method; Runge Kutta, Adams Moulon, Adams Bashforth.
Symbolic index reduction (transferring dependent storage elements	0	$\lambda_{1,2} = -0{,}0025 \pm 1{,}118i$	non-stiff	- Explicit integration methods; Runge Kutta, Adams Moulon, Adams Bashforth

.

Discussion Table 1 and results examples subchapter 2.5.

- The bond graph model that is obtained from the gear wheels system has a dependent state. The equation can be formulated as an index 1 or index 2 system. When the index is reduced by means of adding spring and damper the obtained equation has index 0.
- The straightforward approach and elimination of the dependent state by adding spring and damper approach obtain *stiff systems* that have to be solved by implicit integration methods.
- The elimination of the dependent state by adding spring and damper approach obtains an index-0 system, so it is an ODE. However, because the system is *stiff* it has to be solved by mean of an implicit integration method.
- The symbolic index reduction approach obtains *non-stiff systems* that can be solved by explicit integration methods.

3 Implicit integration methods

3.1 Introduction

The goal of this chapter is to explain implicit integration methods. Implicit integration methods are stable for index 1 systems and semi explicit index 2 systems when some conditions are satisfied (Ascher, 1998). In chapter 4 we will discuss index reduction by means of feedback linearization and stabilization. In chapter 5 we will compare implicit integration method with the feedback linearization method. Examples of implicit integration methods are Backward Differential Formula (BDF) and Radau collocation (Ascher, 1998) (Petzold, 1982)

3.2 Numerical techniques for implicit integration methods

The important numerical parts of implicit integration methods are: discretization by polynomial basis, the modified Newton iteration and step size control. Discretization is possible by interpolating polynomials with fixed coefficients, variable coefficients or fixed leading coefficients. This choice dominates the computer load. For example, with a variable coefficient strategy the coefficients change when the step size changes and in this case a new Jacobian must be calculated. The calculation of the Jacobian determines the computational border. However, because of the error, due to interpolation, it can be necessary to change the step size to get a stable polynomial. A good alternative for the variable coefficient strategy is the fixed leading coefficient polynomial. With this polynomial only the leading coefficient is adopted when the order changes. The rest of the coefficients remain constant. Every time step, the DAE is solved by Newton iteration. By only calculating the iteration matrix when necessary the computer load becomes less. The accuracy of the modified Newton iteration determines the amount of the drift on the manifold. The step size and order are adapted in such a way that the computer code can solve the DAE while satisfying the tolerance with the minimum computer load.

3.2.1 Polynomial basis

Suppose that a function $\boldsymbol{x}(t)$ is know at time instants t_i for $i = 0, 1, .., k$, for example $\boldsymbol{x}(t_{n-i}) = \boldsymbol{x}_{n-i}$. It can be estimated by constructing an interpolating polynomial. For example, the variable coefficient k-order polynomial (Newton form) is given by

$$\boldsymbol{x}(t) = \boldsymbol{x}_n + (t - t_n)[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}] + (t - t_n)(t - t_{n-1})[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}] + \ldots . \tag{3.30}$$

Where

$$[\boldsymbol{x}_n] = \boldsymbol{x}_n$$

$$[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots, \boldsymbol{x}_{n-k},] = \frac{[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots, \boldsymbol{x}_{n-k+1}] - [\boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}, \ldots, \boldsymbol{x}_{n-k}]}{t_n - t_{n-k}} .$$

We can estimate $\dot{\boldsymbol{x}}(t)$ at time step t_n by differentiating this polynomial. In our case

$$\dot{\boldsymbol{x}}(t_n) = [\boldsymbol{x}_n, \boldsymbol{x}_{n-1}] + (t_n - t_{n-1})[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}] + \ldots\ldots \tag{3.31}$$

The general DAE is given by,

$$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, t) = \boldsymbol{0}, \tag{3.32}$$

Substituting (3.31) into (3.32) gives,

$$\boldsymbol{F}([\boldsymbol{x}_n, \boldsymbol{x}_{n-1}] + (t_n - t_{n-1})[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}] + \ldots, \boldsymbol{x}_n, t_n) = \boldsymbol{0}.$$

This is a non-linear equation with respect to $\boldsymbol{x}_n$, its solution gives a new value $\boldsymbol{x}_n$.

Example: First- order BDF

When $\dot{\boldsymbol{x}}(t_n) = \boldsymbol{f}(t_n, \boldsymbol{x}_n)$, the first-order variable coefficient BDF is given by

$$h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = \boldsymbol{x}_n - \boldsymbol{x}_{n-1}, \tag{3.33}$$

Multistep equations are generally presented as follows

$$\sum_{j=0}^{k} \boldsymbol{\alpha}_j \boldsymbol{x}_{n-j} = h \sum_{j=0}^{k} \boldsymbol{\beta}_j \boldsymbol{f}_{n-j}. \tag{3.34}$$

The coefficients of the first-order BDF are, $k = 1$, $\boldsymbol{\alpha}_0 = 1$, $\boldsymbol{\alpha}_1 = -1$ and $\boldsymbol{\beta}_0 = 1$.

Example: Second-order BDF

When $\dot{\boldsymbol{x}}(t_n) = \boldsymbol{f}(t_n, \boldsymbol{x}_n)$, second-order variable coefficient BDF is given by

$$h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = \boldsymbol{x}_n - \boldsymbol{x}_{n-1} + \frac{h_n^2}{h_n + h_{n-1}} \left(\frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{h_n} - \frac{\boldsymbol{x}_{n-1} - \boldsymbol{x}_{n-2}}{h_{n-1}} \right), \tag{3.35}$$

When the step sizes are taken fixed and equally spaced this equation becomes

$$h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = \frac{3}{2} \left(\boldsymbol{x}_n - \frac{4}{3} \boldsymbol{x}_{n-1} + \frac{1}{3} \boldsymbol{x}_{n-2} \right). \tag{3.36}$$

When this BDF is written in the general multistep form, (3.34), the coefficients are $k = 2$, $\boldsymbol{\alpha}_0 = 1$, $\boldsymbol{\alpha}_1 = -4/3$, $\boldsymbol{\alpha}_2 = 1/3$ and $\boldsymbol{\beta}_0 = 2/3$.

(3.35) shows that the coefficients depend on the step size and on the previous step sizes. The fixed leading coefficient BDF is given by

$$h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = -\hat{\boldsymbol{\alpha}}_n^0 \boldsymbol{x}_n + \sum_{j=1}^{k} \boldsymbol{\alpha}_j \boldsymbol{x}_{n-j}.$$

Where the leading coefficient is given by

$$\hat{\boldsymbol{\alpha}}_0 = -\sum_{j=1}^{k} \frac{1}{j}. \tag{3.37}$$

and the rest of the coefficients are constant. So, the polynomial only depends on order and not on step size. This can save Jacobian calculations when this polynomial is used in Newton iterations.

3.2.2 Modified Newton iteration

The following equation has to be solved

$\boldsymbol{F}(\boldsymbol{x}_n^m) = 0$.

Suppose that in step *m*, $\boldsymbol{x}_n^m$ is obtained. We would like to derive $\boldsymbol{x}_n^{m+1}$ so that $\boldsymbol{F}(\boldsymbol{x}_n^{m+1}) = 0$.

The problem will be solved by expanding $\boldsymbol{F}(\boldsymbol{x}_n^m)$ into Taylor's series around the point $\boldsymbol{x}_n^{m+1}$, for example

$$0 = \boldsymbol{F}(\boldsymbol{x}_n^{m+1}) \approx \boldsymbol{F}(\boldsymbol{x}_n^m) + \left(\frac{\partial F}{\partial \boldsymbol{x}}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_n^m} (\boldsymbol{x}_n^{m+1} - \boldsymbol{x}_n^m).$$

Solving the last equation for $\boldsymbol{x}_n^{m+1}$, so called Newton iteration formula is obtained

$$\boldsymbol{x}_n^{m+1} = \boldsymbol{x}_n^m - \left(\frac{\partial F}{\partial \boldsymbol{x}}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_n^m} \boldsymbol{F}(\boldsymbol{x}_n^m). \tag{3.38}$$

In this case *m* is the number of the iteration step. Now we will take the first order fixed coefficient BDF polynomial as an example. The problem to be solved is

$$\boldsymbol{F}(\boldsymbol{x}_n) = \boldsymbol{x}_n - \boldsymbol{x}_{n-1} - h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = 0, \tag{3.39}$$

Substituted in the Newton equation. The iteration formula is given by

$$\boldsymbol{x}_n^{m+1} = \boldsymbol{x}_n^m - \left(\boldsymbol{I} - h_n \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_m^m} (\boldsymbol{x}_n^m - \boldsymbol{x}_{n-1}^m - h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n^m), \tag{3.40}$$

The matrix $\left(\boldsymbol{I} - h_n \partial \boldsymbol{f} / \partial \boldsymbol{x}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_n^m}$ is called the iteration matrix. Usually the iteration can be started with $\boldsymbol{x}_n^0 = \boldsymbol{x}_{n-1}$. DAE solvers use more complex strategies like the fixed leading coefficient strategy. In this case the initial guess is done by evaluating the *variable coefficient* BDF and the Newton iteration is done by a *fixed leading coefficient* BDF. In this case the BDF polynomial is given by

$$\boldsymbol{F}(\boldsymbol{x}_n) = \hat{\boldsymbol{\alpha}}_n^0 \boldsymbol{x}_n - \sum_{j=1}^{k} \boldsymbol{\alpha}_j \boldsymbol{x}_{n-j} + h_n \boldsymbol{f}(t_n, \boldsymbol{x}_n) = 0. \tag{3.41}$$

Where the iteration matrix looks like

$$\hat{\boldsymbol{G}} = \left(\hat{\boldsymbol{\alpha}}_n^0 \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}} + h_n \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{\boldsymbol{x}=\boldsymbol{x}_n^m}. \tag{3.42}$$

(3.38) is solved by LU decomposition.

3.2.3 Step size control

Because the step size and the order are variable, the non-linear DAE can be solved with a good accuracy and with relatively less computer load. The order and step size depend on the local truncation error. The order of the next step is chosen in such a way that it is as large as possible and the step size is chosen in such a way that the error estimate satisfies the tolerance.

3.2.4 Flow chart

The software code has three iteration loops; the time update loop, the Newton iteration loop and the stepsize minimization loop.

1. **The Time update loop**:

   The Time update loop is done by the main loop, the iteration variable is ***n***. The iteration is updated during the simulation time.

2. **The Newton iteration loop:**

   The iteration variable is ***m***. The iteration is executed every time step *n*. Before the Newton iteration is started the polynomial is constructed by means of previous calculated states $\boldsymbol{x}_{n-i}$ for *i = 0, 1, .., k*. In order to minimize computer load the iteration matrix (Jacobian) and the LU decomposition will only be evaluated when necessary. The iteration matrix will be evaluated when:

   - The iteration fails to converge.

     The Newton iteration is considered to converge when

     $$\frac{\rho}{1-\rho}\left|\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right| < 0.33ETOL$$

     Where ETOL is the user tolerance, ρ is an indication of the rate of convergence of the iteration, which can be estimated by

     $$\rho = \left(\frac{\left|\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right|}{\left|\boldsymbol{x}_n^{1}-\boldsymbol{x}_n^{0}\right|}\right)^{\frac{1}{m}}.$$

     The norm is a root mean square norm and

     $ETOL = RTOL|\boldsymbol{x}| + ATOL$.

   - When the new step size or order largely differ from the previous ones. In this case $\boldsymbol{\alpha}_n^0 \neq \hat{\boldsymbol{\alpha}}_n^0$.

     Where the true leading coefficient is

     $$\boldsymbol{\alpha}_0 = -\sum_{j=1}^{k}\frac{h_n}{t_n - t_{n-1}},$$

     and the estimated leading coefficient can be calculated by

$$\hat{\alpha}_0 = -\sum_{j=1}^{k} \frac{1}{j}.$$

To decide if a new iteration matrix must be calculated the eigenvalues ($\overline{\lambda}$) of $\boldsymbol{G}$ are estimated. When c a new Jacobian must be calculated. In DAE solver DASSL $\mu = 0{,}25$. When some previous evaluated Jacobian $\hat{\boldsymbol{G}}_{previous}$ is used, the variable *c* is added to the Newton iteration formula. The iteration formula becomes $\boldsymbol{x}_n^{m+1} = \boldsymbol{x}_n^m - c\hat{\boldsymbol{G}}_{previou}{}^{-1}\boldsymbol{F}(\boldsymbol{x}_n^m)$ and the condition $\boldsymbol{B} = \boldsymbol{I} - c\hat{\boldsymbol{G}}_{previous}^{-1}\boldsymbol{G} < 1$ must be satisfied. When ODE $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ will be calculated the eigenvalues are λ . When these eigenvalues are substituted in $\boldsymbol{B}$ the eigenvalues of $\boldsymbol{B}$ are $\overline{\lambda}$. Because the eigenvalues of $\boldsymbol{G}$ can't be calculated for a DAE, they are taken in such a way that the eigenvalues of $\boldsymbol{B}$ are minimized in the left half of the complex half plane. This leads to $c = 2\hat{\alpha} / (\alpha + \hat{\alpha})$ and the estimated eigenvalues of $\boldsymbol{G}$ are $\overline{\lambda} = (\hat{\alpha} - \alpha) / (\alpha + \hat{\alpha})$.

- A certain number of steps have passed.
  When the iteration number of the Newton iteration loop is bigger then 4 and the iteration doesn't converge a new iteration matrix will be calculated.

3. **Step size minimization loop:**
   The iteration variable is ***l***. The iteration is executed when the Newton iteration fails to converge after two LU decompositions. In this case the step size is made smaller by a factor 0,25.

Figure 5 shows the flow chart of the implicit integration method.

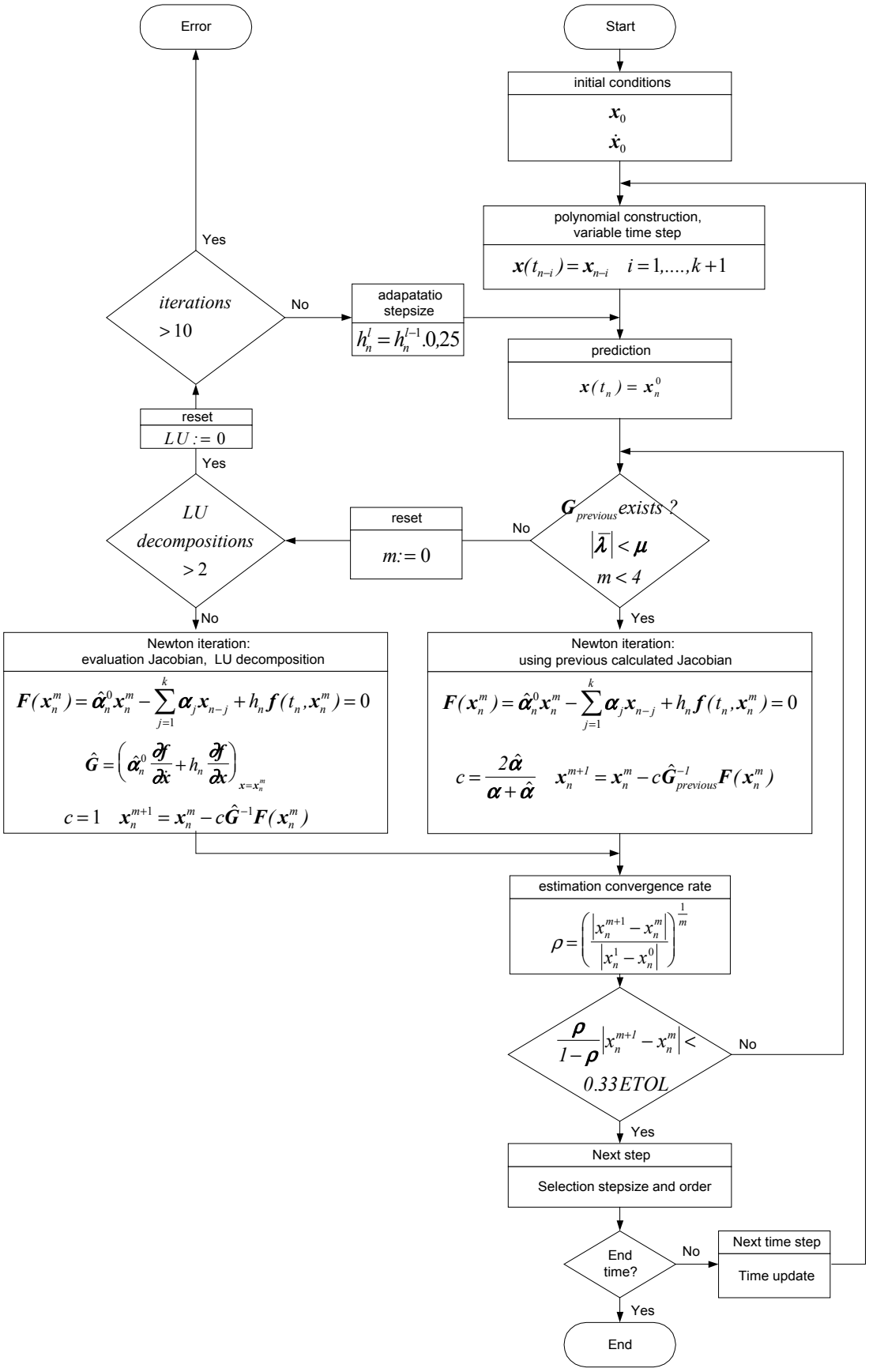


Figure 5 : Flow chart of DAE solver

3.3 Summary

The implicit integration method is based on the Newton iteration method. Every time step the state vector and rate vector are estimated by means of BDF and then solved by means of the Newton iteration method. In order to increase computational efficiency the Jacobian is not calculated every time step. When the vector of the estimated eigenvalues satisfies some criterion $|\bar{\lambda}| > \mu$ then the previous Jacobian is used $\hat{\boldsymbol{G}}_{previous}$. Another method to increase computational efficiency is to adopt step size and order. The step size is chosen as large as possible and the order is chosen as small as possible. The Newton iteration is considered to converge when some criterion of the *state vector* is satisfied. This is an important property, the method satisfies some criterion of the *state vector* and not of the *rate vector*.

4 Index reduction, a control approach

4.1 Introduction

The topic of this study is the solution of the following general model equations that describe nonlinear constrained systems in the form of a control problem

$$\dot{\boldsymbol{x}} = \boldsymbol{J}^{C}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^{R} + \boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{4.43}$$

$$\boldsymbol{\Phi}(\boldsymbol{x}) = 0 = \boldsymbol{G}^{T}(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}). \tag{4.44}$$

$$\bar{\boldsymbol{e}}^{R} = -(\boldsymbol{G}^{C,R}(\boldsymbol{x}))^{T}\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}^{R,U}(\boldsymbol{x})\boldsymbol{u}, \tag{4.45}$$

$$\bar{\boldsymbol{\Omega}}(\bar{\boldsymbol{f}}^{R}, \bar{\boldsymbol{e}}^{R}, \boldsymbol{x}) = 0. \tag{4.46}$$

These equations are introduced in chapter 2. The control law $\boldsymbol{\lambda}$ must be developed in such a way that (4.44) remains close to the desired trajectory $\boldsymbol{\Phi}_{desired}(\boldsymbol{x}) = 0$. This is called control of motion (Slotine, 1991). In this subchapter we will investigate the available nonlinear control and stabilization techniques in order to find a suitable method to control this problem. We want the method to be straight forward and suitable for the solution of complex systems. The control law will be evaluated with respect to the following characteristics: stability, accuracy, robustness, cost. The stability of motion will be investigated by means of the Lyapunov stability theory. The proposed control law and stability investigation will be explained by means of a planar pendulum. This is a relatively small-constrained nonlinear system, which can be solved analytically too.

4.2 Non-linear control

There is no general method for designing nonlinear controllers (Slotine,1991). In this section we give a short description of the available methods. We will select the methods that are suitable to control the described nonlinear constrained systems.

Trial and error:

The idea is to use analysis tools (phase plane method, describing function method, Lyapunov analysis) to guide the search for a controller. The constructed controller can be justified by means of simulation or analysis. For complex systems this method often fails.

Feedback linearization:
The basic idea is to first transform a nonlinear system into a linear system and then use linear design techniques to design a controller. This method requires full state measurement. The method doesn't guarantee robustness in the case of parameter uncertainty or disturbance.

Robust control:
In robust nonlinear control (sliding mode control) the controller designed is based on both the nominal model and some characterization of the model uncertainties (for example unknown load masses).

Adaptive control:
Adaptive control design can be applied to systems with known dynamic structure, but unknown constant or relatively slowly varying parameters.

Gain scheduling:
The idea is to select a number of operating points, which cover the range of the system operation. At each point the designer makes a linear, time invariant approximation of the plant and designs a linear controller for this linearized plant.

A Selection of a suitable method to control the constrained nonlinear system (4.43)-(4.46) can be made as follows:
As complex systems are to be solved in a straightforward way, the trial and error method and the gain scheduling method seem less suitable. Feedback linearization will be because it is a straightforward method. The full state measurement demand of the feedback linearization, is automatically satisfied, because we will use it in numerical simulation where all the states are calculated anyway and chosen parameters can be considered "certain". For post-stabilization we will use robust control or adaptive control. Post stabilization with respect to the manifold at the end of each step is a suitable method (Ascher U.A, 1996).

4.3 Control design

4.3.1 Feedback linearization

The basic idea of feedback linearization is to cancel the nonlinearities in a nonlinear system so that the closed loop dynamics is a linear form (Slotine,1991). We can generate a linear input-output relation by means of differentiation of the output and then formulate a controller by using linear control. This is called the *input-output linearization* approach. The relation between the output $\Phi(\boldsymbol{x})$ and the input $\boldsymbol{\lambda}$ of system (4.43)-(4.46) can also be found by repeated differentiation of the output. In this case the violation of the constraint $\Phi(\boldsymbol{x})$ is called *output* of the system. If we need to differentiate the output of the system *r* times in order to generate a unique explicit relationship between the output $\Phi(\boldsymbol{x})$ and the input $\boldsymbol{\lambda}$, the system is said to have *relative degree r*. The *relative degree* is equal with the *index* of a system, so we can use the same definition. The *index* of a system is equal to the number of times we need to differentiate the output of the system in order to obtain a unique solution, the definition is given in section 2.3. If we differentiate the output of the concerned system (4.44), the following equation is obtained

$$\dot{\Phi}(\boldsymbol{x}) = \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\dot{\boldsymbol{x}} = 0,$$

substitution of equation (4.43) gives

$$\dot{\Phi}(\boldsymbol{x}) = \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\left(\boldsymbol{J}^C(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^R + \boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}\right) = 0.$$

Suppose that the dimension of the generalized state vector $\boldsymbol{x}$ is *k* and the dimension of the Lagrange multiplier $\boldsymbol{\lambda}$ is *n*. As $n<k$, we speak of an *underactuated* system. Solving the previous equation for $\boldsymbol{\lambda}$ gives

$$\boldsymbol{\lambda} = -\left(\frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\boldsymbol{G}(\boldsymbol{x})\right)^{-1}\left(\frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\boldsymbol{J}^C(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^R + \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}\right). \quad (4.47)$$

Figure 6 shows the block diagram of the closed-loop system (4.43) controlled by control law (4.47).

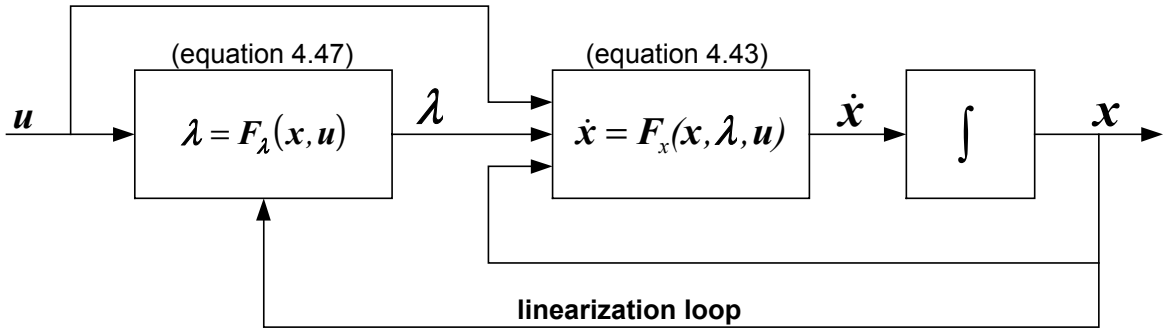


Figure 6 : Feedback linearization

4.3.2 Nonlinear controller

In this study constraints at the velocity level are considered. In this section we will take a closer look at the velocity constraints and we will construct the nonlinear controller equivalent with the feedback linearization.

In order to be able to investigate the constraints at the velocity constraint, we consider a mechanical system described by position state vector $\boldsymbol{q}$. When the constraints are velocities $\dot{\boldsymbol{q}}$ described by $\boldsymbol{G}^T(\boldsymbol{q})\dot{\boldsymbol{q}}$ this will lead to the following constraint equation

$$\Phi(\boldsymbol{p},\boldsymbol{q}) = \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p} = 0 \tag{4.48}$$

and the following Hamiltonian constraint equation

$$\Phi(\boldsymbol{p},\boldsymbol{q}) = \boldsymbol{G}^T(\boldsymbol{q})\frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}} = 0\,. \tag{4.49}$$

Where $\boldsymbol{p}$ represents the momentum of mass $\boldsymbol{M_p(q)}$. This equation is the equivalent with (4.44) concerning the velocity constraint. Now we will apply the feedback linearization control law (4.47). First (4.48) is differentiated

$$\dot{\Phi}(\boldsymbol{p},\boldsymbol{q}) = \frac{\partial\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p}\right)}{\partial t} =$$

$$\frac{\partial\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p}\right)}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p}\right)}{\partial \boldsymbol{p}}\dot{\boldsymbol{p}} =$$

$$\frac{\partial\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p}\right)}{\partial \boldsymbol{q}}\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p} + \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\dot{\boldsymbol{p}} =$$

$$\frac{\partial\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{M}_p^{-1}(\boldsymbol{q})\boldsymbol{p}\right)}{\partial \boldsymbol{q}}\boldsymbol{p} + \boldsymbol{G}^T(\boldsymbol{q})\dot{\boldsymbol{p}} = 0 \tag{4.50}$$

In chapter 2 it is explained how energy dissipation can be represented by means of general equations (4.43)-(4.44). Next we rewrite (4.43)-(4.46) as a function of $\boldsymbol{(p,q)}$ as well as the equation of the energy dissipating elements. Suppose (4.46) is expressed as $\bar{\boldsymbol{f}}^R = \boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\bar{\boldsymbol{e}}^R$ where $\boldsymbol{R}^T(\boldsymbol{p},\boldsymbol{q}) = \boldsymbol{R}(\boldsymbol{p},\boldsymbol{q}) > 0$, when this dissipation matrix and (4.46) are substituted in (4.43) the following equation can be obtained

$$\begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{p}}\end{bmatrix} = \begin{bmatrix}0 & \boldsymbol{J}_{12}\\ -\boldsymbol{J}_{12}^T & \boldsymbol{J}_{22}\end{bmatrix}\begin{bmatrix}\frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{q}}\\ \frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}}\end{bmatrix} + \begin{bmatrix}0\\ \boldsymbol{G}(\boldsymbol{p},\boldsymbol{q})\end{bmatrix}\boldsymbol{\lambda} - \begin{bmatrix}0 & 0\\ 0 & \boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\right)^T\end{bmatrix}\begin{bmatrix}\frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{q}}\\ \frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}}\end{bmatrix} +$$

$$\left(\boldsymbol{G}^{C,U}(\boldsymbol{p},\boldsymbol{q}) - \boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{R,U}(\boldsymbol{p},\boldsymbol{q})\right)^T\right)\boldsymbol{u} \tag{4.51}$$

Where $\left(\boldsymbol{J}_{22},\boldsymbol{J}_{12},-\boldsymbol{J}_{12}^{T},\boldsymbol{G}(\boldsymbol{p},\boldsymbol{q})\right)$ represents the non-energy dissipating interconnection structure, $\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\right)^{T}$ represents the energy dissipating interconnection structure and $\left(\boldsymbol{G}^{C,U}(\boldsymbol{p},\boldsymbol{q})-\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{R,U}(\boldsymbol{p},\boldsymbol{q})\right)^{T}\right)$ is the input force matrix of the mechanical system. The interconnection structures and the input force matrix can be obtained from the bond graph model. Matrix $\boldsymbol{J_{22}}$ is skew symmetric and $\boldsymbol{J}_{22}=-\boldsymbol{J}_{12}^{T}$. When we substitute (4.51) in (4.50), control law $\boldsymbol{\lambda}$ can be obtained

$$\boldsymbol{\lambda}=-\left(\boldsymbol{G}^{T}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q})\right)^{-1}\left(\begin{array}{l}\boldsymbol{G}^{T}(\boldsymbol{q})\left(\begin{array}{l}-\boldsymbol{J}_{12}^{T}\dfrac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{q}}+\boldsymbol{J}_{22}\dfrac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}}- \\ \boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\right)^{T}\dfrac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}}+\boldsymbol{G}^{*}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{u}\end{array}\right)+ \\ \dfrac{\partial\left(G^{T}(\boldsymbol{p},\boldsymbol{q})\left(\partial H(\boldsymbol{p},\boldsymbol{q})/\partial \boldsymbol{p}\right)\right)}{\partial \boldsymbol{q}}\boldsymbol{M}_{p}\dfrac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}}\end{array}\right) \tag{4.52}$$

Where

$$\boldsymbol{G}^{*}(\boldsymbol{p},\boldsymbol{q})=\left(\boldsymbol{G}^{C,U}(\boldsymbol{p},\boldsymbol{q})-\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{R,U}(\boldsymbol{p},\boldsymbol{q})\right)^{T}\right)$$

Now we can construct the nonlinear controller equivalent with the feedback control law (4.47) considering velocity constraints.

$$\boldsymbol{\lambda}=-\boldsymbol{K}_{q}\frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{q}}-\boldsymbol{K}_{p}\frac{\partial H(\boldsymbol{p},\boldsymbol{q})}{\partial \boldsymbol{p}} \tag{4.53}$$

Where

$$\boldsymbol{K}_{q}=-\left(\boldsymbol{G}^{T}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q})\right)^{-1}\boldsymbol{G}^{T}(\boldsymbol{q})\boldsymbol{J}_{12}^{T},$$

$$\boldsymbol{K}_{p}=\left(\boldsymbol{G}^{T}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q})\right)^{-1}\left(\begin{array}{l}\boldsymbol{G}^{T}(\boldsymbol{q})\left(\boldsymbol{J}_{22}+\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q})\left(\boldsymbol{G}^{C,R}(\boldsymbol{p},\boldsymbol{q})\right)^{T}\right) \\ +\dfrac{\partial\left(G^{T}(\boldsymbol{p},\boldsymbol{q})\left(\partial H(\boldsymbol{p},\boldsymbol{q})/\partial \boldsymbol{p}\right)\right)}{\partial \boldsymbol{q}}\boldsymbol{M}_{p}\end{array}\right)$$

4.3.3 Stabilization investigation

A common method to investigate the stability of feedback linearized constrained systems is the investigation of the zero dynamics (Chiou, 1998). This method is suitable for small systems but difficult to use on large systems. In this section we will discuss a stability investigation method that is based on the Lyapunov methods. The Lyapunov stability theory can be used to investigate stability of nonlinear systems (Slotine,1991). The direct method can be used to investigate local and global stability and the linearization method can be used to investigate local stability. In this section we will investigate stability of the open loop system, the closed loop system and stability on the velocity manifold. First we will discuss the definitions of the direct Lyapunov method for global and local stability.

Global stability (Lyapunovs direct method) *Assume that there exists a scalar function V of the state , with continuous first order derivative such that*

- *$V(\boldsymbol{x})$ is positive definite*
- *$\dot{V}(\boldsymbol{x})$ is negative definite*
- *$V(\boldsymbol{x}) \to \infty$ as $\|\boldsymbol{x}\| \to \infty$*

then the equilibrium at the origin is globally asymptotically stable

Local stability (Lyapunovs direct method) *If, in a ball Ω_c, there exists a scalar function $V(\boldsymbol{x})$ with continuous first partial derivatives such that*

- *$V(\boldsymbol{x})$ is positive definite (locally in Ω_c)*
- *$\dot{V}(\boldsymbol{x})$ is negative semi-definite (locally in Ω_c)*

then the equilibrium point is stable. If, actually, the derivative $\dot{V}(\boldsymbol{x})$ is locally negative definite in Ω_c, then the stability is asymptotic.

Stability on the constraint manifold:

(4.44) shows the differential equation of the constraint manifold. In order to know if is the system is stable on the manifold will use the following Lyapunov function

$$\boldsymbol{V(x)} = \boldsymbol{\Phi}^T\boldsymbol{(x)}\,\boldsymbol{\Phi(x)} + \boldsymbol{K} \tag{4.54}$$

where $\boldsymbol{K}$ is a vector with constant positive values such that $\boldsymbol{V(x)}$ is positive definite. The derivative of the Lyapunov function is given by

$$\dot{\boldsymbol{V}}\boldsymbol{(x)} = 2\frac{\partial \boldsymbol{\Phi(x)}}{\partial \boldsymbol{x}}\,\dot{\boldsymbol{x}}\,\boldsymbol{\Phi(x)} = 2\dot{\boldsymbol{\Phi}}\boldsymbol{(x)}\,\boldsymbol{\Phi(x)}\,.$$

Because system (4.43) is controlled in such a way that $\dot{\Phi}(x) = 0$ the derivative of the Lyapunov function $\dot{V}(x) = 0 \quad \forall x \in \chi$. *So, the system is marginally stable on the constraint manifold. This can cause drift of the constraint manifold. If we want the system to be stable we need to stabilize the system with respect to the constraint manifold.*

4.3.4 Stabilization of the constraint manifold

From the stabilization investigation discussed in the previous section we know that nonlinear mechanical systems that are controlled by means of feedback linearization are not stable on the constraint manifold. So we need to stabilize the system with respect to the constraint manifold. In this section we will discuss stabilization by means of a *projection algorithm* (Slotine, 1991) that will be explained by means of a self-tuning adaptive controller. The operation of a self-tuning adaptive controller is as follows: at each time step, the estimator sends to the controller a set of estimated parameters, which are computed on the basis of past input and output signals of the plant. The controller computes new plant input that causes new plant output. The estimator will measure the plant input and output and the cycle will be repeated. Figure 7 shows stabilization of constraint manifold explained by means of the self-tuning adaptive controller.

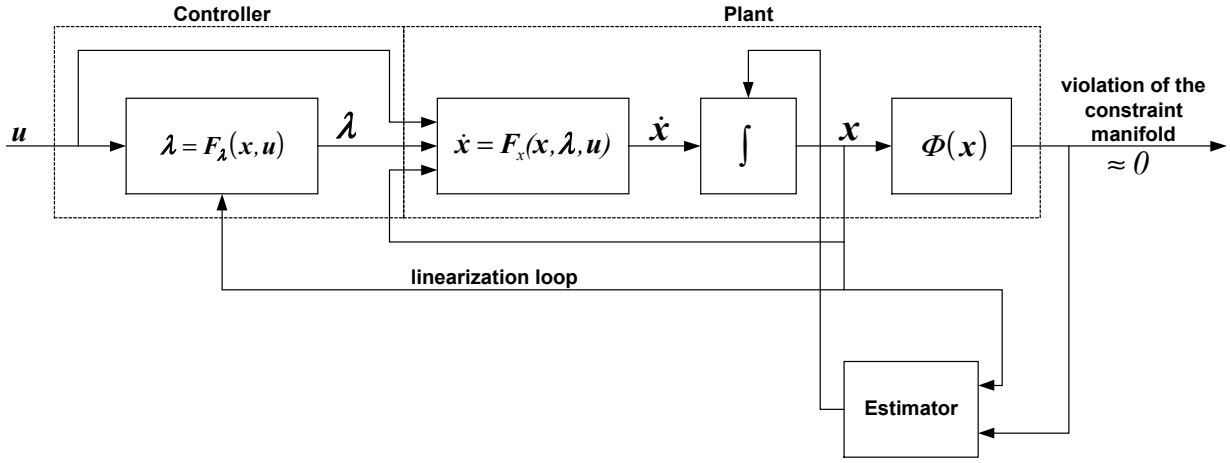


Figure 7 : A self-tuning controller

In this case the plant is system (4.43)-(4.46) that is controlled by means of feedback linearization (4.47). The estimator measures the *unstabilized state vector* x_n^m and the violation of the *constraint manifold* $\Phi\left(x_n^m\right)$. Every time step the estimator computes *stabilized states vector* x_n^{m+1} such that the constraint manifold $\Phi\left(x_n^{m+1}\right) = 0$ will be satisfied.

The projection method is defined as follows:

1. $\Phi\left(x_n^{m+1}\right) = 0$,
2. *and the distance* $\left\|x_n^{m+1} - x_n^m\right\|$ *has the minimum value.*

This can be seen as an optimization problem. Consider the cost function

$$\boldsymbol{J}\left(\boldsymbol{x}_n^{m+1},\boldsymbol{\psi}\right)=\frac{1}{2}\left(\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right)^T\left(\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right)+\boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m+1}\right)\boldsymbol{\psi}.$$

The optimal point has to satisfy the following two equations

$$\frac{\partial \boldsymbol{J}\left(\boldsymbol{x}_n^{m+1},\boldsymbol{\psi}\right)}{\partial \boldsymbol{x}_n^{m+1}}=0\Rightarrow \boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}+\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m+1}\right)}{\partial \boldsymbol{x}_n^{m+1}}\boldsymbol{\psi}=0, \tag{4.55}$$

$$\frac{\partial \boldsymbol{J}\left(\boldsymbol{x}_n^{m+1},\boldsymbol{\psi}\right)}{\partial \boldsymbol{\psi}}=0\Rightarrow \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m+1}\right)=0. \tag{4.56}$$

This represents a nonlinear system of equation and the solution can be found by using some numerical techniques. If the point $\boldsymbol{x}_n^{m+1}$ is *close enough* to a surface

$$\boldsymbol{S}=\left\{\boldsymbol{x}:\boldsymbol{\Phi}(\boldsymbol{x})=0\right\},$$

then $\boldsymbol{\Phi}(\boldsymbol{x})$ is approximated as

$$\boldsymbol{\Phi}(\boldsymbol{x})\approx\boldsymbol{\Phi}\left(\boldsymbol{x}_n^{m}\right)+\left(\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\right)^T\left(\boldsymbol{x}-\boldsymbol{x}_n^{m}\right). \tag{4.57}$$

Therefor equation (4.55) and equation (4.47) become

$$\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}+\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\boldsymbol{\psi}=0,$$

$$\boldsymbol{\Phi}\left(\boldsymbol{x}_n^{m}\right)+\left(\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\right)^T\left(\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right)=0.$$

If $\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}$ is a full rank matrix then the unique solution of the previous system of equations is

$$\boldsymbol{x}_n^{m+1}=\boldsymbol{x}_n^{m}-\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\left(\left(\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\right)^T\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}\right)^{-1}\boldsymbol{\Phi}\left(\boldsymbol{x}_n^{m}\right). \tag{4.58}$$

Projection algorithm (4.58) is equivalent with the *Kaczmarz projection algorithm* (Löhnberg, 2000) and with the *Post-stabilization method* (Ascher, 1996).

Stability:

Löhnberg (2000) shows that

- $\boldsymbol{\Phi}\left(\boldsymbol{x}_n^{m+1}\right)\to 0$ for bounded $\frac{\partial \boldsymbol{\Phi}^T\left(\boldsymbol{x}_n^{m}\right)}{\partial \boldsymbol{x}_n^{m}}$;
- $\left\|\boldsymbol{x}_n^{m+1}-\boldsymbol{x}_n^{m}\right\|\to 0\forall m<\infty$.

Ascher (1996) gives the following accuracy theorem:

Assume sufficient smoothness near the compact manifold χ *defined by* $\Phi(\boldsymbol{x}) = 0$. *Assume that* $\|\boldsymbol{I} - \boldsymbol{HF}\| \le \rho < 1$ *holds as well, and that either* $\rho = O(k)$ *or* χ *is decreasing in (4.43). Then the bounds* $\boldsymbol{x}_n - \boldsymbol{x}(t_n) = O(k^p)$ *and* $\|\Phi(\boldsymbol{x}_n)\| \le K\left(\rho k^{p+1} + k^{2(p+1)}\right)$ *hold. If* $\boldsymbol{HF} = \boldsymbol{I}$ *then* $\|\Phi(\boldsymbol{x}_n)\| \le O\left(k^{2(p+1)}\right)$. *Where p is the order and k the step size of the integration method.*

In our case $\boldsymbol{H} = \dfrac{\partial \Phi^T(\boldsymbol{x}_n^m)}{\partial \boldsymbol{x}_n^m}$ *and* $\boldsymbol{F} = \boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}$.

4.3.5 Implementation

When we want to use the *feedback linearization* to control the constraint manifold and when we want to use the *projection algorithm* to stabilize the constraint manifold we can use the following steps for implementation.

1. First we have to describe the constrained nonlinear mechanical system by means of PCH system (4.43)-(4.46).
2. Then we can obtain the control input by means of one analytical differentiation of the constraint equation (4.47) (*feedback linearization*). The analytical differentiation can for example be done with Maple.
3. We can stabilize the constraint manifold by means of the *projection algorithm* (4.58).
4. Write the code of an integration method (for example Runge Kutta 4) or use an available integration method to integrate the problem.

We can write the equations in mathematical simulation software, for example Matlab or 20Sim. When we use Matlab we can write the equations in m-files. We can choose to use the available integration method or to write the code of an integration method. When we use 20Sim we can write the equations in the equation editor. The end of every integration step can be detected by means of the Boolean variable *major*. The *projection algorithm* must be written in the if-then statement that is activated by means of the *major* variable. The appendix shows an example of the PCH system of the pendulum (discussed in the next section) controlled by feedback linearization and stabilized by the projection algorithm.

4.4 Example

In this chapter we will apply the proposed feedback linearization control law, the stability investigation and the poststabilization to a pendulum model. Figure 8 shows the physical model of a pendulum and Figure 9 shows the bond graph model of the pendulum. The rod has mass M_{rod} and the inertia can be calculated by $J = J_0 + \frac{1}{4} M_{rod} r^2$. The end of the rod is connected to mass *M*. The rod is rigid, so $\dot{r} = 0$.

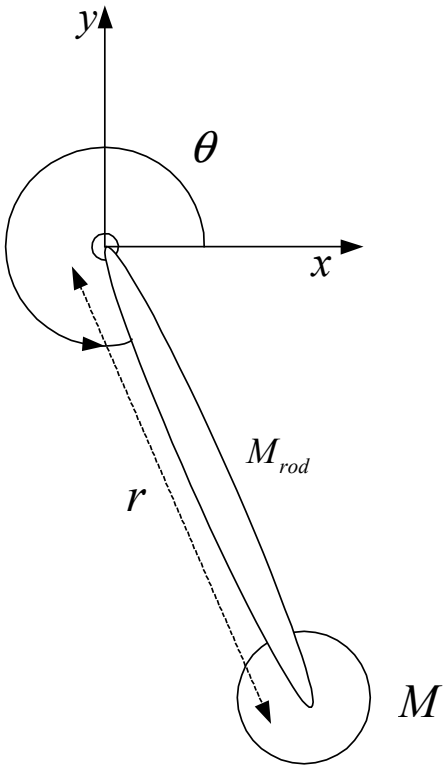


Figure 8 : Pendulum

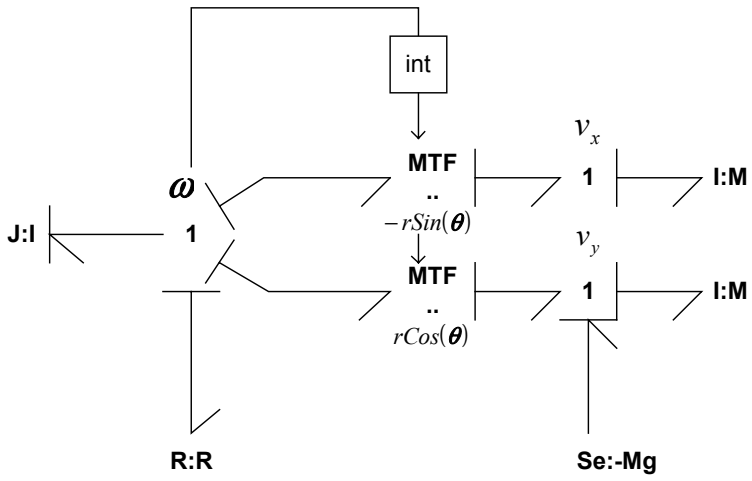


Figure 9 : Bond graph model of pendulum

The state vector is

$$\boldsymbol{x}^T = \left[\theta, p_J, p_x, p_y\right]^T \in \boldsymbol{\chi},$$

where θ is the angle between the *x*-vector and the rod, $\dot{\theta}$ is the angular momentum, p_x is the momenta of the point mass of the rod in *x* direction and p_y is the momenta of the point mass of the rod in *y* direction. The total energy is given by

$$H(\boldsymbol{x}) = \begin{bmatrix} p_J \\ p_x \\ p_y \end{bmatrix}^T \begin{bmatrix} J^{-1} & 0 & 0 \\ 0 & M^{-1} & 0 \\ 0 & 0 & M^{-1} \end{bmatrix} \begin{bmatrix} p_J \\ p_x \\ p_y \end{bmatrix} - MgrSin(\theta)$$

The system can be described by (4.43)-(4.46) where

$$Jc = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.59}$$

$$\frac{\partial H}{\partial \boldsymbol{x}^T}(\boldsymbol{x}) = \begin{bmatrix} M_y grCos(\theta) & J^{-1} p_J & M^{-1} p_x & M^{-1} p_y \end{bmatrix}^T, \boldsymbol{G}^T(\boldsymbol{x}) = \begin{bmatrix} 0 & rSin(\theta) & 1 & 0 \\ 0 & -rCos(\theta) & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{G}^{C,R} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; R^{-1}(\boldsymbol{p},\boldsymbol{q}) = R, \; \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}.$$

The parameters are

J=0,1;

M=1;

r=1;

u*=0;*

R=0,1;

The initial conditions are given by

$$\boldsymbol{x}^T = \left[\theta_0, 0, 0, 0\right]^T$$

4.4.1 Stabilization investigation

In this subchapter we will investigate the stability of the system on the constraint manifold. The pendulum that is described in the previous section will be controlled by means of feedback linearization, control law (4.47). The system is made stable by adding bearing friction *R*.

Hardware set-up:

The simulations are executed on a Pentium I computer, (Commodore evolution) 166 Mhz, 32 MB RAM. The computer is not connected to a network.

Software set-up:

The simulation package is 20 Sim and the system software is Windows 95. The integration methods are standard available in 20 Sim. The BDF method Dassl is implemented in 20 Sim. Figure 10 shows the violation of the velocity constraint manifold.

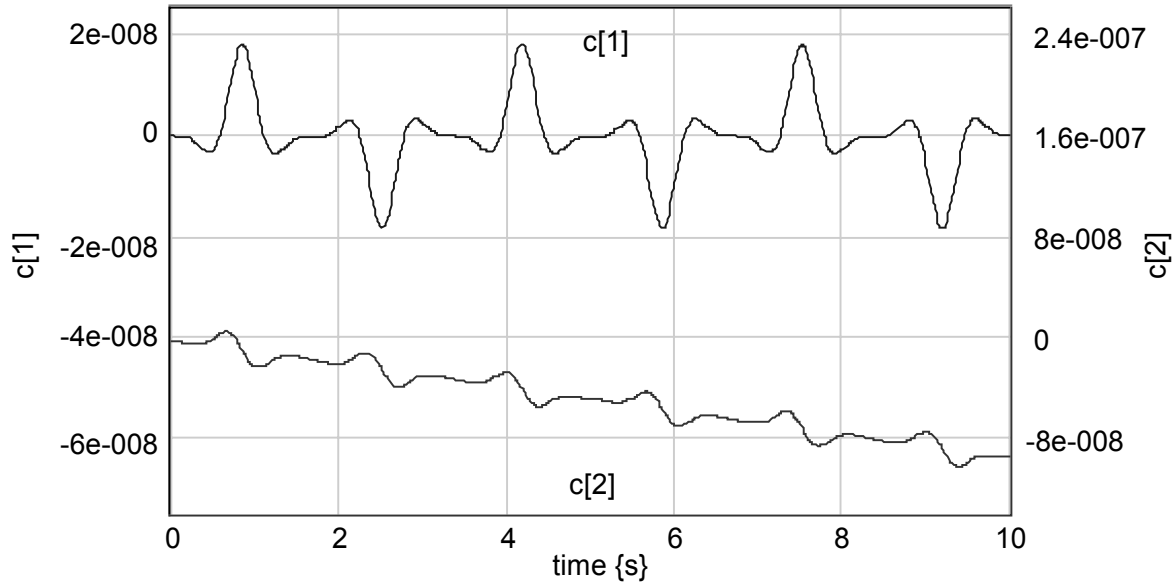


Figure 10 : Violation of the velocity constraint manifold

The first constraint *c[1]* is stable, the second constraint *c[2]* is not stable. This is what we expect, because in subchapter 4.3.3 it is shown that the state vector will drift of the manifold. Projecting the state vector on the constraint manifold will stabilize the system. The system is stabilized by means of equation (4.58). Figure 11 shows the violation of the stabilized constraint manifold.

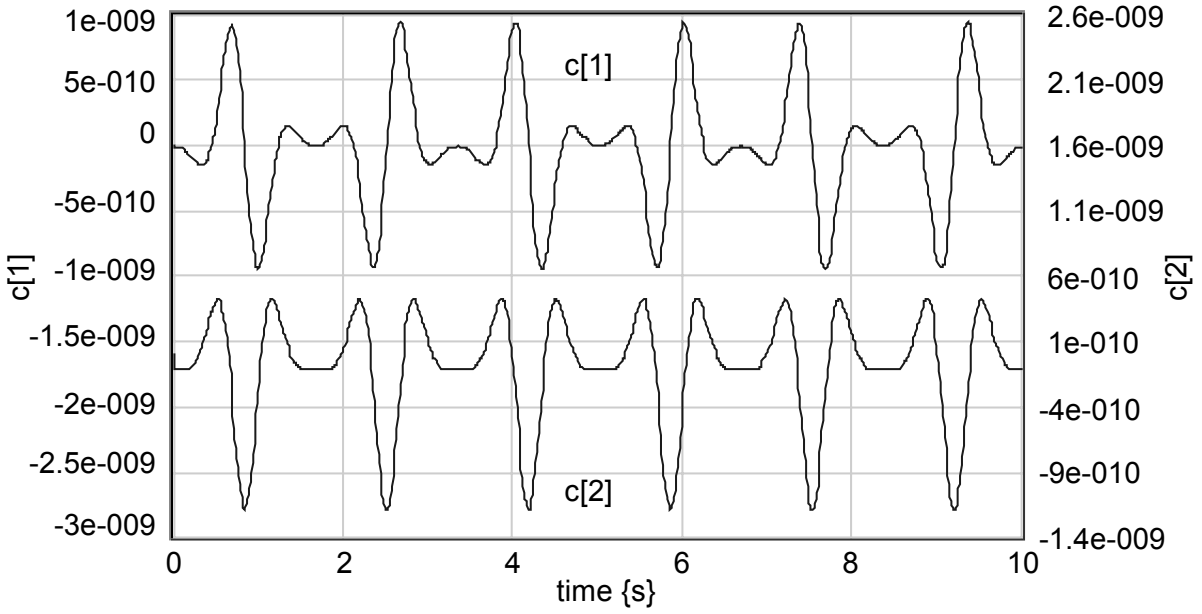


Figure 11 : Violation of the velocity constraint manifold, stabilized manifold

Now the amplitude of the first constraint *c[1]* is smaller and the second constraint *c[2]* is stabilized. The accuracy of the stabilization can be calculated by (Ascher, 1996). Because $\boldsymbol{HF} \neq \boldsymbol{I}$ the accuracy can be calculated by

$$\left\| \Phi(\boldsymbol{x}_n) \right\| \leq K\left(\rho k^{p+1} + k^{2(p+1)} \right).$$

The step size *k=0.01* and the order of the Runge Kutta integration method *p=4*, this gives the accuracy

$$\left\| \boldsymbol{g}(\boldsymbol{x}_n) \right\| \leq O(10^{-10})$$

4.5 Conclusions

When we want to apply control techniques for index reduction *feedback linearization* in combination with the *projection algorithm* are suitable techniques. We can use *feedback linearization* to compute the control input and we can use the *projection algorithm* to stabilize the constraint manifold. The *feedback linearization* can be characterized by means of the *relative degree r* of a system, which is equal to the *index* of a system. It is shown that *feedback linearization* can be described as a nonlinear controller. Because a system that is controlled by means of *feedback linearization* is marginally stable, the state vector will drift off the constraint manifold. So, we have to use the *projection algorithm* to stabilize the system. *Feedback linearization* and the *projection algorithm* can be applied straightforward and the solution can be found by means of simple explicit solvers.

5 Implicit numerical methods and Index reduction, a comparison

5.1 Introduction

In this chapter we will compare the implicit integration method with the index reduction method (index reduction by means of feedback linearization). First we will compare the numerical techniques and then the numerical efficiency. The numerical techniques will be compared by means of analysis of the theory that is discussed in chapters 3 and 4. The numerical efficiency will be compared by numerical simulation of a flyball governor. The flyball governor is a relatively large non-linear mechanical system. It is difficult to compare the methods because they depend for a big deal on the way of implementation. We chose the Dassl (Petzold, 1982) implementation of the implicit integration method and the index reduction method (index reduction by means of feedback linearization) that is discussed in chapter 4. The index-0 system that is obtained by applying index reduction will be integrated by means of the Runge Kutta integration method. In order to get a fair comparison we will use 20 Sim for all the simulations. The equations will be written in the equation editor of 20 Sim. The language equation editor of 20 Sim is SIDOPS+.

5.2 Comparison of numerical techniques

When we want to compare numerical techniques we can look at the basic numerical techniques and at the implementation strategies. When we use feedback linearization we can write the code of the integration method ourselves. So, we can choose the basic numerical technique and the way of implementation. When we use an implicit solver we depend on the basic numerical technique and the implementation of the available software. Table 2 shows the comparison of the basic numerical techniques.

Table 2 : Basic numerical techniques

Numerical technique	**Implicit integration method**	**Index reduction (feedback linearization)**
property	implicit technique	explicit technique
basic numerical technique	• constructing polynomial of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$. • estimation of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ at the next time step (by means of interpolating polynomial) • Implementation in solver.	• calculation of control input $\boldsymbol{\lambda}$ by means of feedback linearization. • evaluation $\boldsymbol{x}$ at the next time step (by means of explicit integration method). • Implementation in equation editor.
stabilization technique	• Newton iteration $\boldsymbol{x}_n^{m+1} = \boldsymbol{x}_n^m - \left(\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{x}}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_n^m} \boldsymbol{Q}(\boldsymbol{x}_n^m)$ • Implementation in solver.	• Projection algorithm $\boldsymbol{x}_n^{m+1} = \boldsymbol{x}_n^m - \left(\frac{\partial F}{\partial \boldsymbol{x}^T}\frac{\partial F}{\partial \boldsymbol{x}}\right)^{-1}_{\boldsymbol{x}=\boldsymbol{x}_n^m} \left.\frac{\partial F}{\partial \boldsymbol{x}^T}\right|_{\boldsymbol{x}=\boldsymbol{x}_n^m} \boldsymbol{F}(\boldsymbol{x}_n^m)$ When $\partial F/\partial \boldsymbol{x}\big|_{\boldsymbol{x}=\boldsymbol{x}_n^m}$ is square the *projection algorithm* is equal to *Newton iteration*. • Implementation in equation editor.

Discussion:

- The implicit integration method estimates $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ based on interpolating polynomial, the polynomial will be constructed based on the numerical results of the previous steps. The index reduction method calculates the control input based on the numerical result of one previous step.
- In case of non-linear problems we have to use a stabilization technique. The Newton iteration method is used to stabilize the implicit integration method and the projection algorithm is used to stabilize the feedback linearization method. When $\partial F/\partial \boldsymbol{x}\big|_{\boldsymbol{x}=\boldsymbol{x}_n^m}$ is square the projection algorithm is equal to the Newton iteration method.

The Dassl implementation of the implicit integration method BDF has a lot of features to make the software code numerical efficient. Table 3 the comparison of the implementation strategies.

Table 3 : Implementation strategies

Numerical technique	**Implicit integration method BDF (Dassl)**	**Index reduction (feedback linearization with Runge Kutta integration method)**
Jacobian evaluation	the Jacobian isn't calculated every time step	the Jacobian is calculated every time step
variable step size control	yes	The 20 Sim implementation of RK-4 is fixed step size and RK-8 is variable step size.
variable order control	yes	no
possible implementation in 20Sim	yes	yes
possible implementation in Matlab	only when the constraint equation is linear or time dependent	yes

Discussion

- Dassl doesn't calculate the Jacobian every time step, this makes the method numerical efficient.
- Dassl is made more numerical efficient by means of variable step size control and variable order control.
- The implicit solver of Matlab (Ode15s) can only solve DAE when the constraint equation is constant or depends on time. Because the constraint equation of the flyball governor model (5.62) depends on $\boldsymbol{x}(t)$, Matlab isn't able to solve the system.

5.3 Comparison by means of simulation

5.3.1 Flyball governor model

In Figure 12 is shown mechanism, flyball governor, used to control the speed of a driven shaft (Golo G, 2000). The body B_0 fixed to ground is connected to the body B_2 by a massless shaft of which the length is L. The shaft and the body B_2 are rigidly connected and they rotate with respect to B_0. The body B_1 (collar) is connected to the body B_2 by a cylindrical joint and by a spring whose coefficient of rigidity is k_s (it is assumed that constitutive relation for the spring is linear). The connection rod B_7 (B_8) is rigidly connected to the body B_3 (B_4) and is connected to the body B_2 (B_2) by means of a revolute joint. The lengths of B_7, B_8 are $L_1 + L_2$. The connection rod B_5 (B_6) links the body B_1 by a spherical joint and links the rod B_7 (B_8) by means of a revolute joint. The distance of connection point from B_2 is L_1. The mechanism is constructed in such way that the set of bodies B_3, B_4, B_5, B_6 remains in the same plane. It is assumed that the bodies are rigid and the joints ideal. The masses and inertias of rods B_5, B_6, B_7, B_8 are neglected. The collar has a mass M_s. Its distance from B_0 is q_s. The inertia of body B_2 around the axis Z is M_Z. The viscous force opposes the motion around Z axis. The coefficient of the viscosity is R_Z (it is assumed that constitutive relation is linear). The mass of the body B_3 (B_4) is m. The axis z_1 is parallel to the rod B_7. The angle between axes z_1 and Z is q_{x_1}. The axis y_1 is chosen in such a way that axes z_1, Z, y_1 remain in the same plane. In a similar way the frame for the body B_4 (x_2, y_2, z_2) is chosen (for the sake of clarity it is not drawn in Figure 12. The tensor of inertia of B_3 with respect to the frame $x_1 y_1 z_1$ is $\boldsymbol{M} = diag(M_x, M_y, M_z)$ and $M_x = M_y$. The gravitational constant is denoted by g and torque applied to the axis Z is denoted by T_Z. (see Favre W, 1996 for the derivation of the bond graph model).

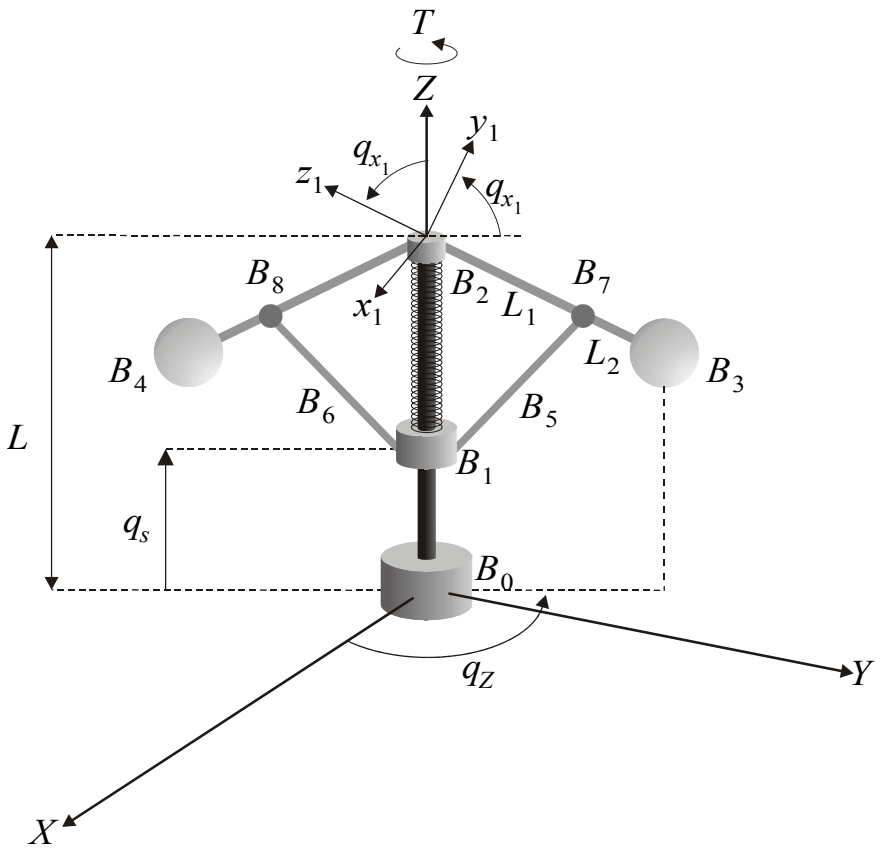


Figure 12 : Flyball governor: Ideal physical model

The state vector is given by

$$x^T = [p_{x1}, p_{y1}, p_{z1}, p_{x2}, p_{y2}, p_{z2}, p_z, p_s, q_{x1}, q_{x2}, q_s]$$

Where

$p_{x1}, p_{y1}, p_{z1}, p_{x2}, p_{y2}, p_{z2}$ are dependent states and $p_z, p_s, q_{x1}, q_{x2}, q_s$ are independent states.

Here, $\left(p_{x1}, p_{y1}, p_{z1}\right)$ is the angular momentum of the body B_3 with respect to the frame $x_1 y_1 z_1$, $\left(p_{x2}, p_{y2}, p_{z2}\right)$ is the angular momentum of the body B_4 with respect to the frame x_2, y_2, z_2, p_Z is the angular momentum B_2 with respect to axis Z and p_s is momentum of the collar. The function $r(q,s)$ is defined as follows

$$r(q,s) = \frac{1}{L_1 Sin(q)} - \frac{Cot(q)}{L - s}.$$

Total energy is given by

$$H(\boldsymbol{x}) = H_p(\boldsymbol{x}) + H_q(\boldsymbol{x})$$

$$H_p(\boldsymbol{x}) = \frac{1}{2}\begin{bmatrix} p_{x_1} \\ p_{y_1} \\ p_{z_1} \end{bmatrix}^T \begin{bmatrix} M_x^{-1} & 0 & 0 \\ 0 & M_y^{-1} & 0 \\ 0 & 0 & M_z^{-1} \end{bmatrix} \begin{bmatrix} p_{x_1} \\ p_{y_1} \\ p_{z_1} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} p_{x_2} \\ p_{y_2} \\ p_{z_2} \end{bmatrix}^T \begin{bmatrix} M_x^{-1} & 0 & 0 \\ 0 & M_y^{-1} & 0 \\ 0 & 0 & M_z^{-1} \end{bmatrix} \begin{bmatrix} p_{x_2} \\ p_{y_1} \\ p_{z_2} \end{bmatrix} + \frac{p_Z^2}{2M_Z} + \frac{p_s^2}{2M_s}$$

$$H_q(\boldsymbol{x}) = \frac{k_s\left(L - L_s - q_s\right)^2}{2} + M_s g q_s + 2mgL - \left(L_1 + L_2\right)\left(Cos\left(q_{x1}\right) + Cos\left(q_{x2}\right)\right)mg$$

The Flyball Governor can be described by the general equations, which are discussed in chapter 2, given by

$$\dot{\boldsymbol{x}} = \boldsymbol{J}^C(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{\lambda} + \boldsymbol{G}^{C,R}(\boldsymbol{x})\bar{\boldsymbol{f}}^R + \boldsymbol{G}^{C,U}(\boldsymbol{x})\boldsymbol{u}, \tag{5.60}$$

$$0 = \boldsymbol{G}^T(\boldsymbol{x})\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) \tag{5.61}$$

Where

$$\boldsymbol{J}^C(\boldsymbol{x}) = \begin{bmatrix} 0 & -p_{z1} & p_{y1} & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ p_{z1} & 0 & -p_{x1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -p_{y1} & p_{x1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -p_{z2} & p_{y2} & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & p_{z2} & 0 & -p_{x1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -p_{y2} & p_{x1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -R_Z & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{\lambda} = \begin{bmatrix} \lambda_{x1} \\ \lambda_{y1} \\ \lambda_{z1} \\ \lambda_{x2} \\ \lambda_{y2} \\ \lambda_{z2} \end{bmatrix},$$

$$\boldsymbol{G}(\boldsymbol{x}) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & Sin\left(q_{x1}\right) & Cos\left(q_{x1}\right) & 0 & Sin\left(q_{x2}\right) & Cos\left(q_{x2}\right) \\ r\left(q_{x1}, q_s\right) & 0 & 0 & r\left(q_{x1}, q_s\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{G}^{C,R}(\boldsymbol{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The parameters are the following

$r = 0{,}025$

$\rho = 7600$

$m = \frac{4}{3} r^3 \pi \rho$

$M_s = 1$

$M_Z = 0{,}1$

$\theta_0 = \frac{\pi}{4}$

$L_1 = 0{,}2$

$L_2 = 0{,}2$

$L = 0{,}4$

$k_s = 100$

$L_s = L - 2\frac{L_1 + L_2}{L_1}\frac{L - L_1 Cos(\theta_0)}{Lk_s} mg - \frac{M_x g}{k_s}$

$M_x = \frac{2}{5} mr^2 + (L_1 + L_2)^2 m$

$M_y = \frac{2}{5} mr^2 + (L_1 + L_2)^2 m$

$M_z = \frac{2}{5} mr^2$

$R_Z = 1$

$u = 10$

The initial conditions are given by

$x_0^T = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \theta_0 \quad \theta_0 \quad 0]$.

When these matrices are substituted in equation (5.60) and (5.61) an index 2 DAE is obtained. The constraint equation (5.61) is given by

$$\boldsymbol{\Phi r(x)} = \begin{bmatrix} \frac{p_{x1}}{M_x} - \left(\frac{1}{L_1 Sin(q_{x1})} - \frac{Cot(q_{x1})}{L - q_s} \right) \frac{p_s}{M_s} \\ \frac{p_{y1}}{M_y} - Sin(q_{x1}) \frac{p_Z}{M_Z} \\ \frac{p_{z1}}{M_z} - Sin(q_{x1}) \frac{p_Z}{M_Z} \\ \frac{p_{x2}}{M_x} - \left(\frac{1}{L_1 Sin(q_{x2})} - \frac{Cot(q_{x2})}{L - q_s} \right) \frac{p_s}{M_s} \\ \frac{p_{y2}}{M_y} - Sin(q_{x2}) \frac{p_Z}{M_Z} \\ \frac{p_{z2}}{M_z} - Sin(q_{x2}) \frac{p_Z}{M_Z} \end{bmatrix} \tag{5.62}$$

The constraint is a vector of 6 elements. The connection between the state variables is non-linear and depends on the state variables.

In order to get an index 1 DAE we will write the equation (5.60) and (5.61) as (2.20)-(2.22).

The ODE can be calculated by index reduction by means of analytical differentiation. The control input $\boldsymbol{\lambda}$ can be calculated by the following equation

$$\boldsymbol{\lambda} = -\left(\frac{\partial \boldsymbol{\Phi}(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \boldsymbol{G}(\boldsymbol{x}) \right)^{-1} \left(\frac{\partial \boldsymbol{\Phi}(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \boldsymbol{J}^C(\boldsymbol{x}) \frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}) + \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \boldsymbol{G}^{C,R}(\boldsymbol{x}) \bar{\boldsymbol{f}}^R + \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \boldsymbol{G}^{C,U}(\boldsymbol{x}) \boldsymbol{u} \right).$$

The analytical differentiation $\partial \boldsymbol{\Phi}(\boldsymbol{x}) / \partial \boldsymbol{x}^T$ is calculated by Maple. The rest of the calculation including the invertation of the matrix, $\left(\left(\partial \boldsymbol{\Phi}(\boldsymbol{x}) / \partial \boldsymbol{x}^T \right) \boldsymbol{G}(\boldsymbol{x}) \right)^{-1}$ is done numerically.

Violation of the position constraint:

The length of rod B_5, which is the same as the length of B_6, is used to calculate the violation of the position constraint. The violation of the position is the difference between the static calculated length and the dynamic calculated length.

$violation \quad position \quad constraint = L_{static} - L_{dynamic}$.

Where

$L_{static} = L_1^2 + L^2 - 2 L_1 L Cos(\boldsymbol{\theta}_0)$;

$L_{dynamic} = L_1^2 + (L - q_s)^2 - 2 L_1 (L - q_s) Cos(q_{x1})$.

Violation of the velocity constraint:

The velocity constraint (5.62) is a vector of 6 elements. The symmetry of the model causes $\Phi_1(x) = \Phi_4(x)$, $\Phi_2(x) = \Phi_5(x)$ and $\Phi_3(x) = \Phi_6(x)$. Because vector elements $\Phi_1(x)$ and $\Phi_4(x)$ realize the largest violation of the constraint, $\Phi_1(x)$ is presented in the simulation results.

5.3.2 Simulation results

Introduction:

In this section the implicit integration method BDF will be compared with the explicit, fourth order, Runge-Kutta method (RK-4) and eight order Runge-Kutta method (RK-8). The BDF method is of variable order and a variable step size. The RK-4 method is of fixed step size and fixed order and the RK-8 method is of fixed order and variable step size. The RK-4 method and RK-8 method are used stabilized and unstabilized.

For constrained mechanical systems, violation of the velocity and position constraint and drift off the constraint manifold indicate the performances of *explicit integration methods* (Ascher U.1998), (Brenan K. 1989). These key properties are measured for different step sizes, order of the integration method and simulation times. When there is drift the simulation will be repeated for a larger simulation time to investigate the behavior of the drift.

For constrained mechanical systems the violation of the velocity and position constraint, the number of steps, number of differential equations evaluations, number of Jacobian matrices, number of failed corrector evaluations and the number of error test failures indicate the performance of *implicit integration methods* (Ascher U.1998), (Brenan K. 1989) and (Jeng Yen, 1995). These key properties are measured for different relative tolerance (RTOL), absolute tolerance (ATOL), initial step size, maximum step size and simulation times.

Method of simulation:

In order to be able to compare the fixed step size methods with the variable step size methods we propose to compare performance at average step sizes with the fixed step sizes. The standard deviation shows if the step size control is active. When the standard deviation is large the step size control is active and when the standard deviation is small the step size control is not active.

We start with the simulation of the BDF method. We set tolerance at demanded values and then measure: average step size, standard deviation, violation of the position and velocity constraint, slope, number of model calculations, number of output points and the simulation time. The package 20 Sim only gives the number of model calculations and the number of output point. The number of differential equations evaluations, number of Jacobian matrices, number of failed corrector evaluations and the number of error test failures can't be presented.

Secondly we simulate the unstabilized RK-4 method. We set the step sizes to the average step sizes we measured simulating the BDF method and we measure: violation of the position and velocity constraint, slope, number of model calculations, number of output point and the simulation time. The unstabilized RK-4 method will probably give drift off the constraint manifold.

The simulation of the stabilized RK-4 method is equal to the simulation of the unstabilized RK-4 method. We expect that the drift will be stabilized.

Simulation of the unstabilized RK-8 method. The tolerance, maximum step size and minimum step size will be set in such a way that the measured average step size is equal to the average step size of the BDF method.

The simulation of the stabilized RK-8 method is equal to the unstabilized RK-8 method.

Evaluation numerical results:

The numerical results of the different integration methods will be presented in a table. The numerical results evaluated, the subjects of discussion are: limits of the integration method, violation of the constraint and comparison wit the other integration methods.

Calculation average step size and standard deviation:

The average step size and the standard deviation are calculated with the following equations

$$average \quad step \quad size = \mu_h = \frac{1}{k}\sum_{n=1}^{k} h_n ,$$

$$standard \quad deviation = \sigma_h = \sqrt{\frac{1}{k-1}\sum_{n=1}^{k}\left(h_n - \mu_h\right)^2} ,$$

where

h : step size,

n : step number,

k : maximum step number.

Hardware set-up:

The simulations are executed on an ADM Athlon 800 computer (equivalent with Pentium III), 800 Mhz, 512 MB RAM. The computer is not connected to a network.

Software set-up:

The simulation package is 20 Sim and the system software is Windows XP. The integration methods are standard available in 20 Sim. The BDF method Dassl is implemented in 20 Sim.

5.3.2.1 Numerical results BDF method

Method for selection of initial step size, maximum step size and tolerance:
We have to take a closer look at the relation between the step size and tolerance in order to make a proper selection. When we want to select the step sizes in such a way that the method is stable. When the BDF method is used to solve an index-1 problem the stability can be defined as (Ascher U.1998).

The k-step BDF for fixed step sizes h for k<7 converges to $O(h^k)$ *if all initial values are correct to* $O(h^k)$ *and if the Newton iterations on each step is solved to accuracy* $O(h^{k+1})$. *This convergence result has been extended to variable step size BDF methods when the method is stable.*

The Newton iteration is considered to converge when

$$\frac{\rho}{1-\rho}\left|\boldsymbol{x}_n^{m+1} - \boldsymbol{x}_n^{m}\right| < 0.33 ETOL$$

Where ETOL is the user tolerance, ρ is an indication of the rate of convergence of the iteration, the norm is a root mean square norm and
$ETOL = RTOL|\boldsymbol{x}| + ATOL$.

In order to find the relation with step size we take a closer look at the step size selection of the BDF method. The new step $h_{n+1} = rh_n$ size is taken conservatively so that the estimated error (*EST*) is a fraction of the desired integration error tolerance *ETOL*. Where

$$r = \left(\frac{fracETOL}{EST}\right)^{\frac{1}{p+1}} \tag{5.63}$$

and *p* is the estimated order.

Now we know that the user error *ETOL* is used to terminate the Newton iteration step and to select new step sizes.

We propose to select the tolerance by the following rule of thumb (Brenan K. 1989).
$RTOL = 10^{-(m+1)}$, *where m is the number of significant digits required for solution.* $ATOL$ *is set to the values at which* $\boldsymbol{x}$ *is essentially insignificant. RTOL, ATOL and the initial step size are set..*

The initial step size will be selected approximately 10 % of the average step size. The maximum step size will be selected in such a way that the integration method is stable.

The following table shows the numerical solutions of the BDF method. The average step size, standard deviation, violation of the position and velocity constraint, slope of drift, number of model calculations, number of output point and the simulation time are measured for different settings of RTOL, ATOL, the initial step size and the maximum step size are set.

Table 4 : Simulation results Flyball governor model

Integration method: BDF (initial step size 0.001)						
RTOL = ATOL	**max. step size**	**step size** $\mu_h \pm \sigma_h$	**position violation**	**number model calculations**	**number output points**	**sim. time [sec.]**
1.5	1.5	1.000 ± 0.667	0.027	179	31	30
1.5	1.5	1.46 ± 0.25	0.027	805	344	500
0.5	0.09	0.0117 ± 0.0885	0.0015	1319	606	30
0.5	0.05	0.0496 ± 0.0046	0.0025	1319	606	30
10^{-1}	0.05	0.0496 ± 0.0046	0.0028	1319	606	30
10^{-2}	0.09	0.065 ± 0.021	0.0023	1096	467	30
10^{-3}	0.09	0.041 ± 0.012	0.0001	1770	733	30
10^{-4}	10	0.027 ± 0.011	$3.3.10^{-5}$	2742	1124	30
10^{-5}	10	0.021 ± 0.011	$9.4.10^{-6}$	3501	1406	30
10^{-6}	10	0.012 ± 0.0036	$7.9.10^{-7}$	5786	2601	30
10^{-10}	10	0.0031 ± 0.0012	$5.4.10^{-10}$	20449	9244	30
10^{-14}	1	0.00075 ± 0.0021	$9.6.10^{-14}$	98787	39819	30
10^{-15}	1	BDF-message: about 500 steps are taken, continue?				

Discussion numerical results:

1. Limits of the integration method.
   - Maximum step size:

     When the tolerances are set to large values (0.5-1.5) the average step size is approximately equal to the maximum step size. This is what we expect. The estimated new step size is calculated by $h_{n+1} = rh_n$ where *r*. depends on tolerance (5.63). When the tolerance is too large, the estimated new step size will be too large and the Newton iteration is not able to

converge anymore. The size of the estimated new step size can be limited by the maximum step size.

- Minimum step size:

  When the tolerance is made smaller then 10^{-14} the solver generates the following error message "about 500 steps are taken, continue?".

2. Constraint violation:
   - Violation of the velocity constraint:

     The value of the velocity constraint violation that is presented by 20Sim is zero in all cases. This is the result of the relative accuracy of the double precision calculation, which is $2{,}2\ 10^{-16}$. So, the velocity constraint violation is smaller then $2{,}2\ 10^{-16}$ in all cases.
   - Violation of the position constraint:

     The numerical results satisfy the tolerance settings, for example when RTOL=ATOL=10^{-2} the violation of the position constraint is 0.0023.
   - Drift off the constraint manifold:

     The slope of the position violation isn't presented by 20Sim, because it is smaller then the relative accuracy of double precision calculation ($2{,}2\ 10^{-16}$).

5.3.2.2 Numerical result RK-4 method

The following table shows the numerical solutions of the unstabilized RK-4 method. The step sizes are set equal to the average step sizes of the BDF method. The violation of the position and velocity constraint, slope of drift, number of model calculations, number of output point and the simulation time are measured for different step sizes.

Table 5: Simulation results Flyball governor model

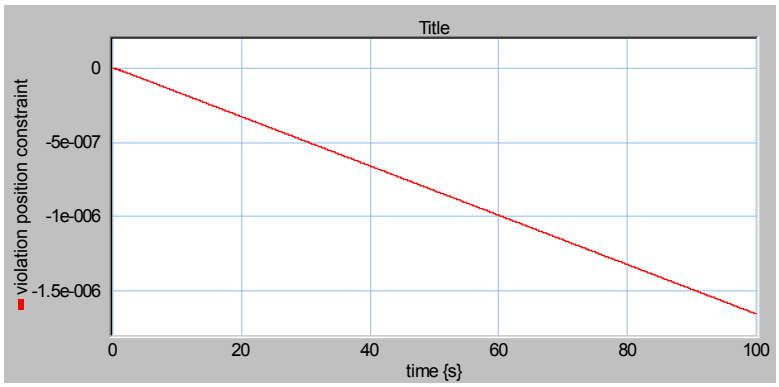
Integration method: RK-4 (unstabilized)						
step size	**position violation**	**slope position violation**	**velocity violation**	**number model calculations**	**number output points**	**sim. time [sec.]**
0.2	Error: zero determinant in determining inverse matrix					
0.15	42.10^{-3}	14.10^{-4}	0.008	805	202	30
0.1	1.10^{-4}	$3.3.10^{-6}$	0.006	1201	301	30
0.065	6.10^{-4}	2.10^{-5}	9.10^{-5}	1849	463	30
0.0496	2.10^{-4}	$6.6.10^{-6}$	4.10^{-5}	2421	606	30
0.041	$1.2.10^{-4}$	4.10^{-6}	$2.1.10^{-5}$	2929	733	30
0.041	2.10^{-3}	4.10^{-6}	$2.1.10^{-5}$	48800	12200	500
0.027	$2.4.10^{-5}$	8.10^{-7}	$4.4.10^{-6}$	4449	1113	30
0.021	$9.1.10^{-6}$	3.10^{-7}	$1.7.10^{-6}$	5717	1430	30
0.012	1.10^{-6}	3.310^{-8}	$1.9.10^{-7}$	10001	2501	30
0.0031	$4.6.10^{-9}$	$1.5.10^{-10}$	9.10^{-10}	38713	9679	30
0.0001	$7.7.10^{-14}$	$2.5.10^{-15}$	$3.9.10^{-14}$	$1.2.10^{6}$	$0.3.10^{6}$	30

Discussion numerical results:

1. Limits of the integration method:
   - When the step size is 0.2 the following error message is obtained from the solver "Error: zero determinant in determining inverse matrix".
2. Constraint violation:
   - Drift off the constraint manifold:

     There is drift off the position constraint manifold. In order to check if the slope is constant the slope is measured after 30 seconds simulation time and after 500 seconds simulation time (step size is 0.041). The slope of drift is 4.10^{-6} after 30 seconds of simulation time and 4.10^{-6} after

500 seconds of simulation time. The following picture shows that the drift has a constant slope.



The drift can cause a relatively large violation off the position constraint manifold after long-time simulation.

The following table shows the numerical solutions of the stabilized RK-4 method.

Table 6 : Simulation results Flyball governor model

Integration method: RK-4 (stabilized)					
step size	**position violation**	**velocity violation**	**number model calculations**	**number output points**	**sim. time [sec.]**
0.24	Error: zero determinant in determining inverse matrix				
0.1	0.0001	0.006	1201	301	30
0.065	$2.8.10^{-5}$	$1.6.10^{-4}$	1849	463	30
0.0496	$7.6.10^{-6}$	2.10^{-7}	2421	606	30
0.041	$2.8.10^{-6}$	$1.4.10^{-6}$	2929	733	30
0.027	$3.8.10^{-7}$	$4.3.10^{-8}$	4449	1113	30
0.021	$6.1.10^{-8}$	$5.7.10^{-9}$	5717	1430	30
0.012	$3.6.10^{-10}$	5.10^{-11}	10001	2501	30
0.0031	$3.2.10^{-11}$	$2.9.10^{-14}$	38713	9679	30
0.0001	$4.8.10^{-15}$	$1.2.10^{-15}$	$1.2.10^{6}$	$0.3.10^{6}$	30

Discussion numerical results:

1. Limits of the integration method:
   - When the step size is 0.24 the following error message is obtained from the solver "Error: zero determinant in determining inverse matrix".
2. Constraint violation:
   - Violation of the velocity constraint:
     The value of the velocity constraint violation that is presented by 20Sim is zero in all cases. This is the result of the relative accuracy of the double precision calculation, which is $2,2\ 10^{-16}$. So, the velocity constraint violation is smaller then $2,2\ 10^{-16}$ in all cases.
   - Drift off the constraint manifold:
     The slope of the position violation isn't presented by 20Sim, because it is smaller then the relative accuracy of double precision calculation ($2,2\ 10^{-16}$).

5.3.2.3 Numerical result RK-8 method

The following table shows the numerical solutions of the stabilized RK-8 method. The tolerances, minimum step size sand maximum step sizes are set equal to the BDF method.

Table 7: Simulation results Flyball governor model

Integration method: RK-8 (unstabilized, initial step size 0.001)								
RTOL = ATOL	**max. step size**	**step size** $\mu_h \pm \sigma_h$	**position violation**	**slope position violation**	**velocity violation**	**number model calculations**	**number output points**	**sim. time [sec.]**
0.5	0.05	0.0495 ± 0.0044	$5.3.10^{-9}$	1.160^{-10}	$1.1.10^{-8}$	7842	604	30
0.1	0.05	0.0495 ± 0.0044	$5.3.10^{-9}$	1.160^{-10}	$1.1.10^{-8}$	7842	604	30
0.1	0.09	0.09 ± 0.01	$8.4.10^{-8}$	$2.8.0^{-9}$	$2.1.10^{-8}$	4371	337	30
10^{-2}	0.09	0.09 ± 0.01	$8.4.10^{-8}$	$2.8.0^{-9}$	$2.1.10^{-8}$	4371	337	30
10^{-10}	0.09	0.09 ± 0.01	$8.4.10^{-8}$	$2.8.0^{-9}$	$2.1.10^{-8}$	4371	337	30
10^{-7}	0.041	0.041 ± 0.003	1.10^{-9}	$3.3.10^{-11}$	2.10^{-10}	9532	734	30
10^{-7}	0.041	0.041 ± 0.003	1.10^{-9}	$3.3.10^{-11}$	2.10^{-10}	9532	734	30
10^{-7}	0.041	0.041 ± 0.006	$1.8.10^{-8}$	$3.6.10^{-11}$	2.10^{-10}	$0.16.10^{6}$	12200	500
10^{-7}	0.027	0.027 ± 0.001	$3.5.10^{-11}$	$1.2.10^{-12}$	$7.1.10^{-12}$	14500	1113	30
10^{-7}	0.0031	0.0031 ± 10^{-6}	$2.8.10^{-14}$	$3.3.10^{-14}$	6.10^{-15}	$0.13.10^{6}$	9677	30
10^{-10}	0.0031	0.0031 ± 10^{-6}	$2.8.10^{-14}$	$3.3.10^{-14}$	6.10^{-15}	$0.13.10^{6}$	9677	30

Discussion numerical results:

1. Limits of the integration method:
   - Maximum step size:

     For all the tolerance settings the maximum step size must be made smaller in order to keep the method stable. The RK-8 method is stable until the step size becomes larger then 0.09, this is larger then the BDF method (0.05). This is what we expect. The RK-8 method has a larger order (8) and therefor a better accuracy then the BDF method (maximum order is 5).

     For all the tolerance settings the step size is equal to the maximum step size and the standard deviation is small. We can conclude that this method can't control step size in this case.
   - Minimum step size:

When the tolerance is made smaller then 10^{-10} the computer isn't able to solve the problem anymore.

2. Constraint violation:
   - Violation of the velocity and position constraint:

     The method has a large accuracy. This is what we expect because the local error is $O(h^9)$.
   - Drift off the constraint manifold:

     There is a small drift off the position constraint manifold.

Stabilization RK-8 method:

The numerical results of the stabilized RK-8 method are equal to the results of the unstabilized RK-8 method. Because the accuracy of the unstabilized RK-8 method is large the stabilization method isn't able to stabilize the integration method. The following figure shows the violation of the position constraint.

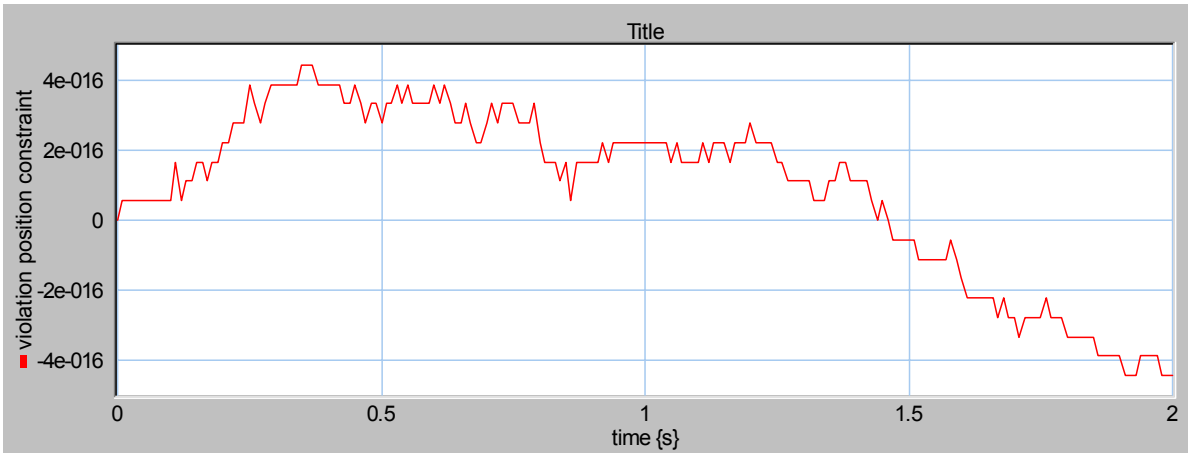


Figure 13 : RK-8 method (stabilized): Violation of position constraint

The post-stabilization method projects the state vector on the constraint manifold. Because the error of the RK-8 method is small, the post-stabilization acts on the numerical noise. This is why the post-stabilization method can't stabilize the RK-8 method.

The following figure shows the violation of the position constraint in detail (maximum step size is 0.01). The post-stabilization acts on time position 0,55 seconds and on time position 0,54 seconds. The subtraction is 0.01, which is equal to the maximum step size.

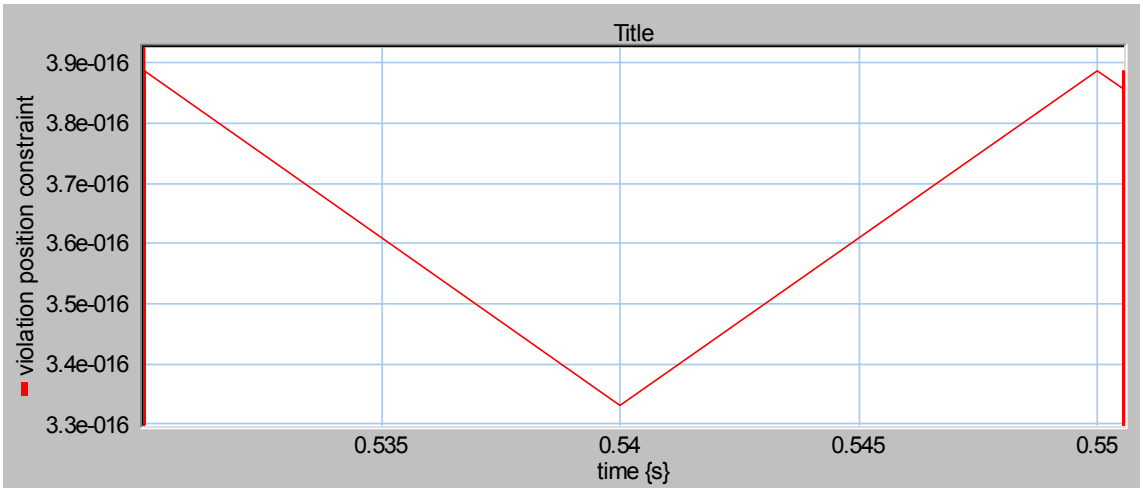
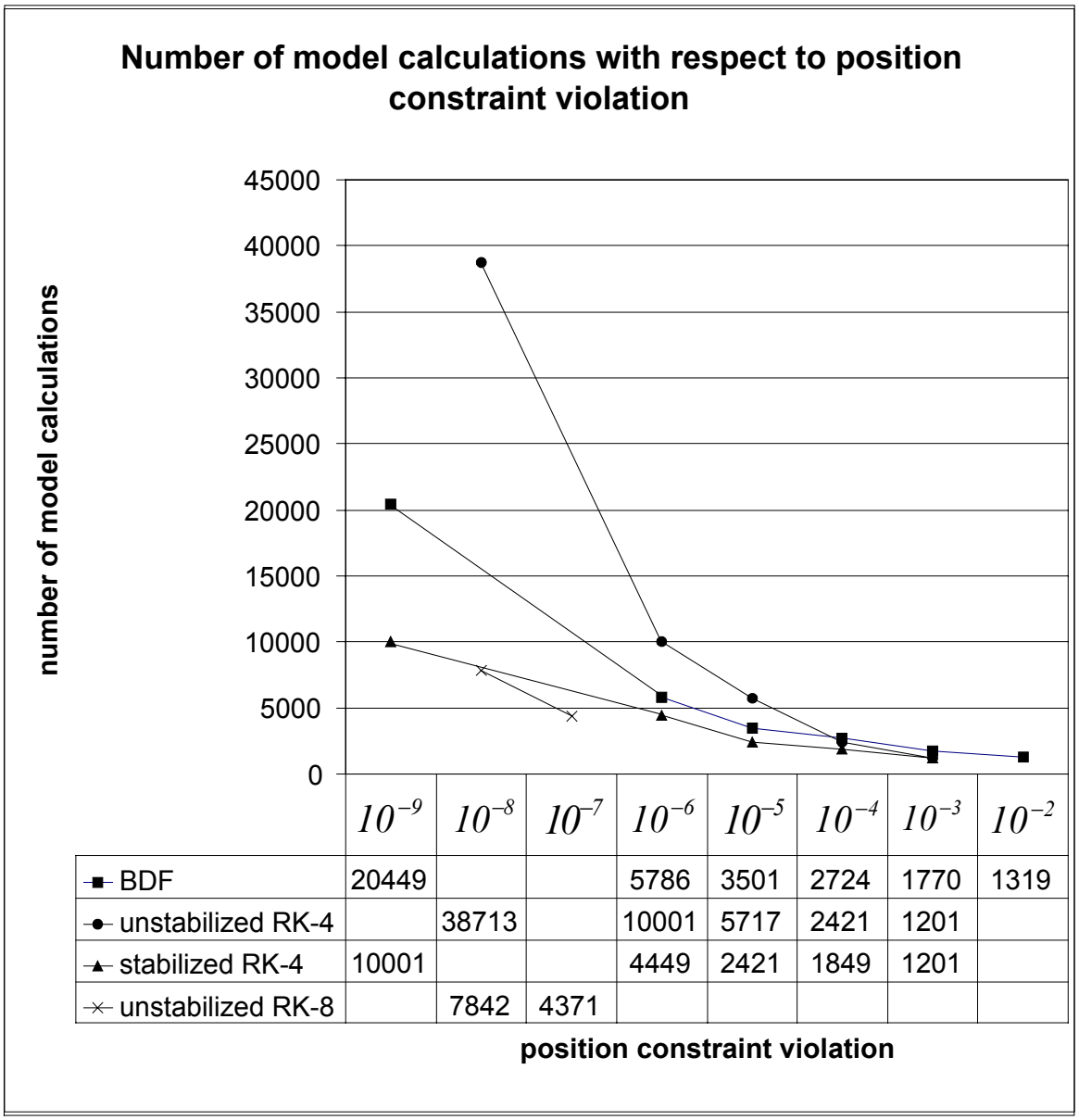


Figure 14 : RK-8 method (stabilized): Violation of position constraint, detail

5.3.3 Summary

Simulation results

Figure 15 shows the number of model calculations and the violation of the position constraint for the BDF method, unstabilized RK-4 method, stabilized RK-4 method and unstabilized RK-8 method.



	10^{-9}	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
BDF	20449			5786	3501	2724	1770	1319
unstabilized RK-4		38713		10001	5717	2421	1201	
stabilized RK-4	10001			4449	2421	1849	1201	
unstabilized RK-8		7842	4371					

Figure 15 : Number of model calculations and the violation of the position constraint

When the position constraint violation is large (10^{-2}-10^{-5}) the number of model calculations of the BDF method, unstabilized RK-4 and stabilized RK-4 are almost equal. When the violation of the position constraint becomes smaller (10^{-6}-10^{-9}) the stabilized RK-4 method and unstabilized RK-8 method requires the least number of model calculations. The BDF method requires more model calculations

and the unstabilized RK-4 method requires the most number of model calculations. When we look at the position constraint violation of 10^{-9} the BDF method needs twice as much model calculations then the stabilized RK-4 method.

The results of this simulation show that the unstabilized RK-8 method and RK-4 methods give drift off the position constraint manifold. This is what we expect, in chapter 4 we have shown that unstabilized index reduction by means of feedback linearization is marginally stable on the constraint manifold. The numerical disturbance will cause drift off the constraint manifold.

Analysis numerical techniques

An important difference between the implicit integration method Dassl and the applied implementation of the feedback linearization with the projection algorithm is the way that $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$ and $\boldsymbol{\lambda}$ (control input) are calculated. The implicit integration method calculates $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ by means of interpolating polynomial and the feedback linearization method calculates $\boldsymbol{\lambda}$ by means of analytically differentiation of the constraint equations.

The stabilization techniques of the implicit integration method Dassl and the applied implementation of the feedback linearization are almost equal. The implicit integration method is stabilized by means of the Newton iteration method and the projection algorithm is used for stabilization of feedback linearization. When the Jacobian of the projection algorithm is square, the projection method is equal to the Newton iteration method.

The main difference between the applied implicit integration method Dassl and feedback linearization with the projection algorithm is the way of implementation. Dassl contains a number of options to make the software more numerically efficient. Optimization options are: step size control, order control and a mechanism that reduce the number of Jacobian calculations. So, in case of future research it is recommendable to use the latter optimizations on the feedback linearization and projection algorithm implementation.

5.3.4 Conclusion

The results of this comparison show that index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm is a suitable alternative for implicit integration methods. So, in case of future research we recommend to develop an automatic method for index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm. This automatic method could be implemented in a mathematical simulation package, for example in 20Sim. The straightforward procedure that is proposed in chapter 4 can be used as a basis for the implementation. In order to improve computer efficiency we propose to use variable step size control, variable order control and mechanism that reduce the number of Jacobian calculations.

6 Conclusions and recommendations

6.1 Conclusions

In this study we propose to use index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm as an alternative for implicit integration methods. In order to test the suitability the numerical basic techniques and the implementation strategies are compared by means of theoretical analysis and the performance is compared by means of computer simulation. This study gives the following conclusions.

1. The results of this study show that index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm is a suitable alternative for implicit integration methods.
2. Feedback linearization and the projection algorithm can be applied straightforwardly and the solution can be found by means of simple explicit solvers. The implementation of the feedback linearization and projection algorithm approach is relatively simple. In chapter 4 we propose a systematic procedure to obtain equations.
3. This study shows that the performance of the BDF method (implicit solver) and the unstabilized RK-4 method (explicit solver) are almost equal for relatively large accuracy (10^{-2}-10^{-5}). When we look at the position constraint violation of high accuracy (10^{-9}) the BDF method needs twice as much model calculations as the stabilized RK-4 method.
4. There are limitations to the complexity of the DAE that implicit solvers can handle. For example, the DAE of the flyball governor cannot be integrated by means of the implicit solver that is implemented in Matlab (Ode 15s, Matlab version 5.3). The implicit solver that is implemented in 20Sim (Dassl) is able to solve the DAE that is obtained from the flyball governor.

6.2 Recommendations

1. For future research it is recommended to develop an automatic method for index reduction by means of feedback linearization and stabilization of the constraint manifold by means of the projection algorithm. This automatic method could be implemented in a mathematical simulation package, for example in 20Sim. The straightforward procedure that is proposed in chapter 4 can be used as a basis for the implementation. In order to improve computer efficiency we propose to use variable step size control, variable order control and mechanism that reduce the number of Jacobian calculations.
2. For future research it is recommended to develop an implementation of the flow chart (Figure 1) of the formulation of nonlinear constrained systems as a Decision Support System (DSS) in a simulation package, for example 20Sim. The DSS can help the modeler to choose a suitable strategy in order to find the solution of a nonlinear constrained system.

Appendix A: Implementation Feedback linearization and projection algorithm

When we want to use the *feedback linearization* to control the constraint manifold and when we want to use the *projection algorithm* to stabilize the constraint manifold we can use the following steps for implementation.

1. First we have to describe the constrained nonlinear mechanical system by means of PCH system (4.43)-(4.46).
2. Then we can obtain the control input by means of one analytical differentiation of the constraint equation (4.47) (*feedback linearization*). The analytical differentiation can for example be done with Maple.
3. We can stabilize the constraint manifold by means of the *projection algorithm* (4.58).
4. Write the code of an integration method (for example Runge Kutta 4) or use an available integration method to integrate the problem.

Example: pendulum presented in section 4.4

Step 1: Equations (4.59) show the PCH system description of the pendulum.

Step 2: The constraint equation is given by (4.44)

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} rSin(\theta)\frac{p_J}{J} + \frac{p_x}{M} \\ -rCos(\theta)\frac{p_J}{J} + \frac{p_y}{M} \end{bmatrix}$$

The control input can be obtained by means of analytical differentiation of the constraint equation (4.47). This analytical differentiation can be done automatically.

$$\boldsymbol{\lambda} = \begin{bmatrix} rCos(\theta)\frac{p_J}{J} & \frac{rSin(\theta)}{J} & \frac{1}{M} & 0 \\ rSin(\theta)\frac{p_J}{J} & -\frac{rCos(\theta)}{J} & 0 & \frac{1}{M} \end{bmatrix}.$$

Step 3: We can stabilize the constraint manifold by means of the *projection algorithm* (4.58).

Step 4: The equation are written in the equation editor of 20 Sim, the SIDOPS+ code is given on the next page.

SIDOPS+ code PCH system of the pendulum controlled by feedback linearization and stabilized by the projection algorithm.

```
parameters
real Gv[4,1]=[0;0; 0; 0];
real mw=1;
real mx=1;
real my=1;
real r=1;
real u=0;
real Rx=1;
real g=9.81;

variables
real theta,pw, px ,py;
real theta_dot,pw_dot, px_dot, py_dot;
real G[4,2],dphi_dx[2,4],J_dH_dx[4,1],dH_dx[4],Labda[2],x_dot[4],phi[2];
real F[4,2],z[4],z1[4];

equations
// PCH system description of the pendulum, matrixes (4.59)
G=[0, 0;
-r*sin(theta), r*cos(theta);
-1,0;
0,-1];

dphi_dx=[-r*cos(theta)*pw/mw, -r*sin(theta)/mw, -1/mx, 0;
-r*sin(theta)*pw/mw, r*cos(theta)/mw, 0, -1/my];

J_dH_dx=[pw/mw;
-my*g*r*cos(theta)-Rx*pw/mw;
0;
0];

dH_dx=[-my*g*r*cos(theta);pw/mw;px/mx;py/my];
```

```
// control input calculated by means of one analytical differentiation of the constraint equation (4.47)
Labda=(-inverse(dphi_dx*G))*((dphi_dx*J_dH_dx)+(dphi_dx*Gv*u));

//differential equation (4.43)
x_dot=(J_dH_dx+(G*Labda)+(Gv*u));
theta_dot=x_dot[1];
pw_dot=x_dot[2];
px_dot=x_dot[3];
py_dot=x_dot[4];

//integration
theta_dot=ddt(theta,0);
pw=int(pw_dot,0);
px=int(px_dot,0);
py=int(py_dot,0);

//stabilization of the constraint manifold by means of the projection algorithm (4.58)
if major then
z[1]=theta;
z[2]=pw;
z[3]=px;
z[4]=py;

F=(transpose(dphi_dx)*(inverse(dphi_dx*transpose(dphi_dx)));
phi=transpose(G)*dH_dx;
z1=z-(F*phi);

theta=z1[1];
pw=z1[2];
px=z1[3];
py=z1[4];
end;
```

References

[1] Ascher U.M., Petzold L.R. (1998), *Computer methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, ISBN 0-89871-412-5.

[2] Ascher U.M.A. (1996). *Stabilization of Invariants of Discretized Differential Systems.* NSERC Canada Grant OGP0004306, pp 14:1-24.

[3] Ascher U.M., Petzold L.R. (1993), *Stability of Computational Methods for Constrained Dynamics and Systems*, SIAM J. Sci Comput, Vol. 14, No. 1, pp. 95-120, January.

[4] Baumgarte,1972, *Stabilization of Constraints and Integrals of Motion in Dynamical Systems,* Computer methods Appl. Mech. Eng., 1, pp 1-16.

[5] Boomgaard M.B., Clemens F.(2001), *Genetische algoritme bij optimalisatievraagstukken in de riolering*, Rioleringswetenschap, jaargang 1, Nr 1 februari.

[6] Bos A.M. (1986). *Modeling multibody systems in terms of multibond graphs*, PhD-Thesis University of Twente, The Netherlands.

[7] Brean K.E., Campbell S.L., Petzold L.R. (1989). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, ISBN 089871-353-6.

[8] Breedveld P.C, (1985). *Multibond graph elements in physical systems theory*, Hournal of the Franklin Institute, 1985.

[9] Broenink J.F. (1990). *Computer-aided physical-systems modeling ans simulation: a bond-graph approach*, PhD-Thesis University of Twente, The Netherlands.

[10] Chiou J.C, Wu S.D. (1998). *Constraint Violation Stabilization Using Input-Output Feedback Linearization in Multibody Dynamic Analysis.* Journal of Guidance Control and Dynamics, Vol. 21 No. 2, March-April 1998, pp. 222-228.

[11] Compere M. D, Longoria R.G. (2000), *Combined DAE and Sliding Mode Control Methods for Simulation of Constrained Mechanical Systems*, Journal of Dynamic Systems, Measurement, and Control, Vol 122, pp 691-698.

[12] Van Dijk J. (1994). *On the role of bond graph causality in modeling mechatronic systems*. PhD-Thesis, University of Twente, The Netherlands.

[13] Dold A., Eckmann B, Takens F. (1989), *The Numerical Solution of Differential Algebraic Systems by Runge-Kutta Methods.* Springer-Verslag, Berlin Heidenberg.

[14] Gear C.W. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations,* Prentice Hall. United States of America.

[15] Golo G, Breedveld P.C, Maschke B.M, van der Schaft A.L., 2000, *Geometric Formulation of Generalized Bond Graph Model. Part I: Generalized Junction Structures*, Journal of the Franklin Institute, ISSN 0169-2690.

[16] Golo G, 2001, *Flyball Governor,* Private communication.

[17] Golo G, van der Schaft A.L., Breedveld P.C, Maschke B.M, 2000, *Implicit Hamiltonian formulation of bond graphs*, Private communication,

[18] Houck C.R., Joines J.A., Kay M.G., (1995), *A Genetic Algorithm for Function Optimization: A Matlab Implementation*, North Carolina State University, DMI-9322834.

[19] Jacksom, K.R, Sacks-Davis R. (1980), *An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODEs*, ACM Transactions on Mathematical Software, Vol. 6. No. 3, pages 295-318.

[20] Jeng Yen, Petzold L.R, 1995, *On Numerical Solutions of Constrained Multibody Dynamic Systems*, ARO DAAL03-89-C-0038.

[21] Van Kan J., (1988), *Numerieke Wiskunde voor Technici,* Delftse Uitgevers Maatschappij b.v.

[22] Löhnberg P. (2000). *System Identification and Adaptive Control.* course code 124118 UT

[23] Ortega R, Spong M.W (12-1-2000). *Stabilization of Underactuated mechanical Systems via Interconnection and Damping Assignment*.

[24] Ortega R, Spong M.W, Gomez-Estern F. (20-11-2000). *Stabilization of Underactuated mechanical Systems via Interconnection and Damping Assignment*.

[25] Paynter H.M, (1961), *Analysis and Design of Engineering Systems,* The M.I.T. Press, Cambridge, Massachusetts.

[26] Petzold L.R. (1982), *A Description of DASSL: A differential/algebraic system solver*, IMACS world congress on system simulation and scientific computing, IMACS.

[27] Van der Schaft (2000). *Implicit port-controlled Hamiltonian systems*. Jounal of the SICE of Japan, Vol. 39, nr. 2, pp. 410-418.

[28] Van der Schaft (2000). *Port-controlled Hamiltonian systems: towards an theory for control design for nonlinear physical systems*. Jounal of the SICE of Japan, Vol. 39, nr. 2, pp. 91-98.

[29] Shampine L.F, Gordon M.K, (1975), *Computer Solutions of Ordinary Differential Equations The Initial Value Problem*, W.H. Freeman and Company.

[30] Shampine L.F, Reichelt M.W, *The Matlab ODE Suite*, 2000, added to the Matlab reference guide.

[31] Slotine J.J.E, Weiping Li (1991). *Applied Nonlinear Control,*Prentice-Hall International, Inc. ISBN 0-13-040049-1

[32] Stramigioli. S, Maschke B, Bidard B. (2000). *Modeling of Mechanical Systems using Screw vectors.*

[33] Velthuis W.J., 2000, *Learning Feed-Forward Control*, PhD-Thesis University of Twente, The Netherlands.

[34] Vladimir Stejskal, Michael Valasek,1996, *Kinematics And Dynamics of Machinery*, Marcel Dekker, New York.

[35] Favre W., Scavarda S., (1998), *Bond Graph Representation of Multibody Systems with Kinematic Loops*, J. Franklin Inst., Vol 335B, No. 4, pp. 643-660.

[36] Xiaoping Yun and Nilanjan Sarkar, 1998, *Unified Formulation of Robotic Systems with Holonomic and Nonholonimic Constraints*, IEEE Transactions on Robotics and Automation, vol, 14, no. 4.