Dimension Reduction of Biochemical Network Models: Implementation and Comparison of Different Algorithms

Marvin Schulz

THESIS

In partial fulfillment of the requirements for the degree Bachelor of Science in Bioinformatics

Freie Universität Berlin
Department of Mathematics and Computer Science

Advisors:
Dr. Wolfram Liebermeister
Dr. Ralf Herwig

Max Planck Institute
for Molecular Genetics
Ihnestr. 73
14195 Berlin

6.6.2006 - 1.8.2006

Abstract

In this thesis different dimension reduction algorithms for linear, time-invariant systems are applied to biochemical network models. A general framework for the linearization of biochemical network models from the literature is shown and a way to export reduced order, linear systems to the format Systems Biology Markup Language is developed. The well known dimension reduction algorithms balanced truncation, singular perturbation approximation and matrix-Padé-via-Lanczos are described and compared for their computational cost and for the behaviour of their output models in the time domain. Finally an open-source program for dimension reduction of biochemical network models in the SBML format is introduced.

Inhalt

In dieser Arbeit werden verschiedene Dimensionsreduktionsalgorithmen für lineare, zeitinvariante System auf mathematische Modell biochemischer Reaktionsnetzwerke angewendet. Es wird eine bekannte, generelle Möglichkeit zur Linearisierung von Reaktionsnetzwerken erläutert und verschiedene Dimensionsreduktionsalgorithmen (Balanced Truncation, Singular Perturbation Approximation und Matrix-Padé-via-Lanczos) werden aus der Literatur aufgegriffen. Außerdem werden verschiedene Wege zum Export von dimensionsreduzierten Systemen in das Format SBML gegeben. Die Reduktionsalgorithmen werden hinsichtlich ihrer Effizienz und Akkuranz bei biochemischen Netzwerken verglichen. Schließlich wird ein quelloffenes Programm zur automatischen Dimensionsreduktion für biochemische Reaktionsnetzwerke diskutiert.

Contents

Abbreviations

Abbreviation	Name
SBML	Systems Biology Markup Language
LTI system	Linear, time-invariant system
SISO	single-input-single-output
MIMO	multiple-input-multiple-output
TFM	Transfer-function matrix
BT	Balanced truncation
SPA	Singular perturbation approximation
AWE	Asymptotic waveform evaluation
PVL	Padé-via-Lanczos
MPVL	Matrix Padé-via-Lanczos
AORGA	Adaptive-order rational global Arnoldi
HSV	Hankel singular value
SVD	Singular value decomposition
flop	Floating point operation
MSE	Mean square error

Symbols

Symbol	Name	Definition
N	Stoichiometric matrix	
c	Vector of metabolite amounts	
v	Vector of reaction velocities	
c_1	Vector of amounts of metabolites from *class 1*	
$\overline{c}$	Vector of amounts of metabolites in steady state	
Δc	Difference from a concentration to its steady state value	$c - \overline{c}$
N_{3-4}^{1-2}	Part of the stoichiometric matrix denoting the influence of reaction from the *classes 3* and *4* on metabolites from the *classes 1* or *2*	
ε	Substrate elasticities	
Π	Parameter elasticities	
e^M	Matrix exponential of the square matrix M	$\sum_{k=0}^{\infty} \frac{M^k}{k!}$
M^*	Conjugate transpose of a matrix M	
M^{-T}	The transpose of the inverse of the matrix M	$(M^{-1})^T$
$\jmath$	Imaginary number	$\sqrt{-1}$

Chapter 1

Introduction

As nowadays more and more kinetic information becomes available, biochemical network models produced by system biologists get larger in number and size. Building these models includes numerically demanding parameter estimation (with tools like SBML-PET [1]) to fit a model to given experimental results. Usually, to lower the effort in parameter estimation, the small model whose parameters are unknown is embedded into an environment where peripheral substances of the model are kept at a fixed concentration. Making this assumption, the parameter estimation procedure becomes fast. Though, certain impacts that the rest of a cell's metabolism might have on the small model are neglected.

Since databases for both qualitative (e.g. KEGG [2], Reactome [3]) and quantitative (e.g. BioModels [4], JWS [5]) network models are being constructed, information given by them can be integrated into model building processes. Models from these databases can be combined with a given model (e.g. with SBMLmerge [6]) to gain a broader description of a cell's metabolism, but for the resulting big model parameter estimation might become unfeasible.

A tradeoff between the needs for computational efficiency and accuracy of the estimated parameters in a built small model is to add an environment of models from databases to this model, reduce the dimension of the state space of this environment, and estimate parameters in the small model afterwards. The resulting model could be both, efficient in the parameter estimation process and accurate in the description of effects the environment can have on the small model.

A framework for the dimension reduction of biochemical network models has been given in [7]. In this approach, a given big model that consists of an inner model and an environment, is first divided into four classes as shown in figure 1.1. Then, a steady state for the whole model is computed, and the differential equations of all substances in the environment (classes 3 and 4) are linearized around this steady state. Afterwards, the linearized

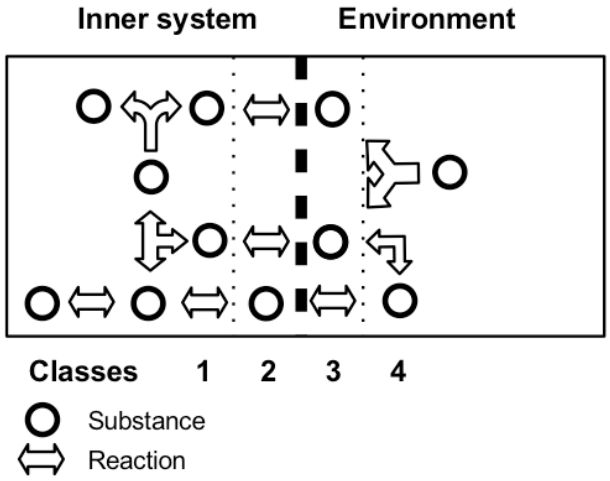

Figure 1.1: An artificial model in which the reaction network is divided into an inner model, which should be preserved during the dimension reduction process, and an environment that should be reduced. The diagram shows a division into so-called *location classes*. Substances or reactions from the inner model which are connected to elements from the environment are sorted into class 2 (the border of the inner model) and analogously elements from the environment connected to elements in the inner model into class 3. This classification is needed for the following treatment of biochemical network models.

environment is reformulated into the standard form for linear, time-invariant systems and reduced by the algorithm balanced truncation [8]. Thereby the reactions and substances from class 3 are preserved as an interface to the inner model.

This thesis uses a similar approach with fewer underlying assumptions. Also further model reduction algorithms (singular perturbation approximation [9], and matrix Padé-via-Lanczos [10, 11]) are tested in this framework on different models and compared for their efficiency and accuracy. One major goal of this thesis is to create an open-source program named *PyLESS* (**Py**thon **MO**dell **RE**duction), which is able to perform model reduction on different types of input models given in SBML format or in matrix form. The implementation has been done in Python [12] in order to be able to interact with a modeling tool developed at the Max Planck Institute for Molecular Genetics (PyBioS [13]).

The algorithm PyLESS proceeds as described in figure 1.2, analogue to the framework presented in [7]. Different methods of how the reduced order models produced by the dimension reduction algorithms can be exported to SBML are given in this thesis. SBML has been chosen as a format for input and output models because it is likely to become one of the main

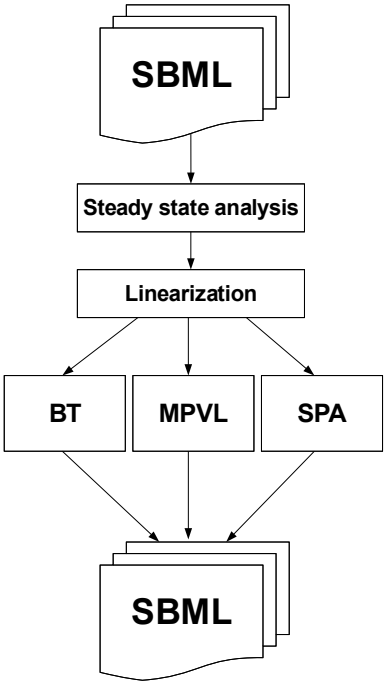

Figure 1.2: Flowchart of PyLESS: Input SBML files are linearized around a steady state, then this system can be reduced by the algorithms balanced truncation, singular perturbation approximation, or matrix Padé-via-Lanczos to the desired order, and finally they are converted back to SBML again.

standards in the description of biochemical network models. In this format, which is useable for both, structural and mathematical descriptions of a biochemical reaction network, the topology and the underlying differential equations are stored in terms of substances (*species*) and reactions. These reactions are used to describe "real chemical reactions" and their kinetics. Furthermore it is also possible to describe parameters and mathematical equations that cannot be expressed in form of reactions (differential and algebraic, via *rate* and *assignment rules*) in SBML. Additional features such as *compartments* for the structuring of a model's topology and conditional expressions (*events*) are less important in this context.

This thesis is organized as follows: In section 2 the linearization of metabolic network models is described. It mainly follows the ideas presented in [7] and describes ways to convert reduced order models into the SBML format. Descriptions of the algorithms used, are summarized from the literature in section 3. A comparison of different results of the model reduction algorithms is given in section 4 and discussed in section 5. Refer-

ences to the software that is used in PyLESS and parts of the Python code are presented in the appendix.

Chapter 2

Linearization of Biochemical System Models

2.1 Linear, Time-invariant Systems

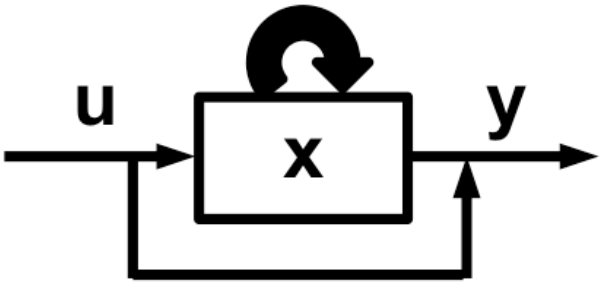

Figure 2.1: Linear, time-invariant system

A linear, time-invariant (LTI) system as illustrated in figure 2.1 can be formulated in the standardized way

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad t > 0, \quad x(0) = 0, \\ y(t) &= Cx(t) + Du(t), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, y \in \mathbb{R}^p, \end{aligned} \tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, and $D \in \mathbb{R}^{p \times m}$ [14]. This represents a system of n internal state variables x, which have an impact on themselves through the matrix A, and are influenced by a vector of m external input-variables u throught the matrix B. The vector y of external output-variables is determined by x via the matrix C and additionally by u via the matrix D. Such a system is called linear, because the rate of change of the internal variables and the values of the external output-variables depend linearly on x and u, which results in a proportionality between the input signal and the

steady state of the output signal. Additionally, it is called time-invariant because the matrices A, B, C, and D do not change in time.

This is often referred to as the *state space representation* in which one interprets the vector x as a point in an n dimensional space. Simulating the model over time can then be seen as observing the vector x moving through that space. That is why this representation is also called the *time domain approach*.

2.2 Metabolic Network Models

Metabolic reaction network models are very different to LTI systems. Generally, there are two ways of describing these models. The first description declares the topology of the network in terms of a directed graph on the chemical substances of a model, which are connected through reactions. An example for such a topology is given in figure 1.1. The second way of describing a model is a mathematical approach. The concentrations or amounts of the substances in a reaction network are represented by a vector c and the system is now described as a vector of initial substance concentrations $c(0)$ and a differential equation that has an impact on the vector c in changing the concentrations of the substances within in the system over time.

These two approaches can be combined into a single description of the system. Again, the concentrations are regarded as a vector c having an initial state $c(0)$. Then, a vector of functions $v(\cdot)$ is introduced which represents the reaction velocities of the reactions described in the topological view on the system. Differential equations for the elements of the vector c can be seen as linear combinations of these reaction velocities. Thereby the coefficients in this linear combinations describe whether a certain substrate participates in a reaction or not. It is negative if it is a reactant to this reaction, positive if it is a product to this reaction, or 0. If this coefficient is non-zero, then its absolute value also denotes in which stoichiometry it participates in this reaction. For example a model with one single reaction $O_2 + 2H_2 \rightarrow 2H_2O$ with reaction velocity v_1 has the system of differential equations

$$\begin{aligned} \dot{O_2} &= -v_1 \\ \dot{H_2} &= -2v_1 \\ \dot{H_2O} &= 2v_1. \end{aligned} \tag{2.2}$$

These coefficients can be aranged into a matrix N, which is from now on called *stoichiometric matrix*, so the system can be formulated in matrix form

$$\begin{pmatrix} \dot{O_2} \\ \dot{H_2} \\ \dot{H_2O} \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ 2 \end{pmatrix} \cdot v_1. \tag{2.3}$$

Usually the reaction velocities of a system are not constant. They depend on parameters p and on current substrate concentration, which themselves might also be affected by parameters. With respect to this, the standard form of biochemical network models is denoted by

$$\dot{c} = N \cdot v(c(p), p). \tag{2.4}$$

For analysis of the system it is usually observed in its *steady state*. The steady state is a set of concentrations $\overline{c}$ under which the rate of change for all concentrations equals zero

$$0 = N \cdot v(\overline{c}(p), p). \tag{2.5}$$

The major difference between LTI systems and metabolic network models is that the reaction velocities are not neccessarily linear. As a matter of fact in most cases they are highly non-linear as for example in the case of a Michaelis-Menten kinetic

$$V = \frac{V_{\max} C}{C + K_m}. \tag{2.6}$$

To make use of the dimension reduction algorithms described in this thesis, the non-linear metabolic network model has to be approximated by a model in linear form. Prior to the presentation of this approximation/transformation method, a few definitions have to be introduced.

2.3 Metabolic Control Analysis in Model Linearization

As already pointed out, the reaction velocities of a metabolic network model may be highly non-linear in terms of parameters and substrate concentrations as shown in picture 2.2. Linearizing a reaction velocity means to replace the original function, by an approximative function that is linear in its input variables. For example a function, which only depends on one substrate concentration and no parameter $v(c)$, would be approximated by the first order function

$$v_{\text{lin}}(c) = v(c(0)) + v'(c(0)) \cdot (c - c(0)), \tag{2.7}$$

where $c(0)$ is the substrate concentration at which the original function should be approximated by a tangent and $v'(c(0))$ is the derivate of v at $c(0)$.

To approximate the reaction velocities of a metabolic network, two important questions have to be asked. (i) At which value of the variables the linearized function should be a tangent to the original function? (ii) How

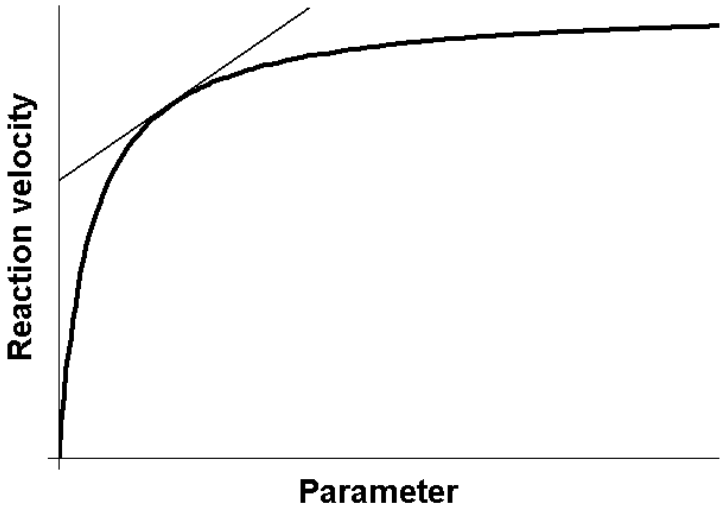

Figure 2.2: Reaction velocity dependent on one parameter that is approximated by a linear function at a certain parameter value. While the linear function estimates the original function well in a region around this parameter value, the estimation gets worse further away from it.

to find the derivate of this non-linear reaction velocities with respect to a certain variable?

The first question can be answered easily. In order to approximate the behaviour of the original system in steady state, the reaction velocities are linearized around the steady state values of the variables. To answer the second question, two important measures from metabolic control analysis [15, 16] have to be explained.

Definition 2.3.1 *Substrate elasticities [15, 16]*
Substrate elasticities at a certain point in concentration state space denote the sensitivity of a certain reaction v_k to a small change in the concentration of a certain substance c_i. So they define a derivate of a reaction velocity with respect to a certain variable.

$$\varepsilon_i^k = \frac{\partial v_k}{\partial c_i} \tag{2.8}$$

The derivates of all reaction velocities with respect to all substrate concentrations can be written in matrix form

$$\varepsilon = \begin{pmatrix} \varepsilon_1^1 & \varepsilon_2^1 & \cdots \\ \varepsilon_1^2 & \varepsilon_2^2 & \\ \vdots & & \ddots \end{pmatrix} \tag{2.9}$$

Definition 2.3.2 *Parameter elasticities [15, 16]*
Parameter elasticities are defined analogously to the substrate elasticities. They denote the derivate of a certain reaction with respect to a certain parameter:

$$\Pi_i^k = \frac{\partial v_k}{\partial p_i} \tag{2.10}$$

The derivates of all reaction velocities with respect to all substrate concentrations can be written in matrix form

$$\Pi = \begin{pmatrix} \Pi_1^1 & \Pi_2^1 & \dots \\ \Pi_1^2 & \Pi_2^2 & \\ \vdots & & \ddots \end{pmatrix} \tag{2.11}$$

Making use of these two matrices, the linearization of a complete network model can be defined as follows:

$$\dot{c} = N \cdot v_{\text{lin}}(c(p), p), \tag{2.12}$$

where

$$v_{\text{lin}}(c(p), p) = \overline{v} + \varepsilon \cdot (c(p) - \overline{c}(p)) + \Pi \cdot (p - \overline{p}). \tag{2.13}$$

2.4 Linearization of Metabolic Networks of Model Reduction

In this section the environment, which denotes the part of the complete model that is to be reduced, is linearized and brought to the form of an LTI system. This is required for all dimension reduction algotihms described in this thesis. To simplify these steps, this is first done for the case in which the reaction velocities do not depend on additional parameters.

For a linearization of only the "to be reduced" part of the model a reordering of the system's equation

$$\dot{c} = Nv(c) \tag{2.14}$$

is done prior to this step (at this point we forget about the parameter dependence). Here c and v are reordered according to their location classes (as seen in the introduction):

$$c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} \qquad v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}. \tag{2.15}$$

Due to this reordering, it is possible two write the stoichiometric matrix as

$$N = \begin{pmatrix} N_1^1 & N_2^1 & 0 & 0 \\ N_1^2 & N_2^2 & N_3^2 & 0 \\ 0 & N_2^3 & N_3^3 & N_4^3 \\ 0 & 0 & N_3^4 & N_4^4 \end{pmatrix}, \tag{2.16}$$

where N_2^1 denotes the stoichiometry in which the substances from location class 1 participate in reactions from class 2.

2.4.1 Parameter Independent Linearization

Using this reordering the differential equations for the concentrations of the substances can be rewritten to

$$\begin{aligned} \dot{c_1} &= N_1^1 v_1(c_1, c_2) + N_2^1 v_2(c_1, c_2, c_3) \\ \dot{c_2} &= N_1^2 v_1(c_1, c_2) + N_2^2 v_2(c_1, c_2, c_3) + N_3^2 v_3(c_2, c_3, c_4) \\ \dot{c_3} &= N_2^3 v_2(c_1, c_2, c_3) + N_3^3 v_3(c_2, c_3, c_4) + N_4^3 v_4(c_3, c_4) \\ \dot{c_4} &= N_3^4 v_3(c_2, c_3, c_4) + N_4^4 v_4(c_3, c_4) \end{aligned} \tag{2.17}$$

and partially linearized to

$$\begin{aligned} v_3 &\approx \overline{v}_3 + \varepsilon^3 \Delta c \\ &= \overline{v}_3 + \varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 \\ v_4 &\approx \overline{v}_4 + \varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4, \end{aligned} \tag{2.18}$$

where $\Delta c = c - \overline{c}$. The restating of these equations to the standard LTI form proceeds now as follows: At first one inserts

$$x = \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix}, \qquad y = \begin{pmatrix} \Delta c_3 \\ \Delta v_3 \end{pmatrix} \qquad \text{and} \qquad u = \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \end{pmatrix} \tag{2.19}$$

into equation (2.1) and then rewrites the system with the equations given in (2.17):

$$
\begin{aligned}
\dot{x}(t) &= \begin{pmatrix} \dot{\Delta c_3} \\ \dot{\Delta c_4} \end{pmatrix} \\
&= \begin{pmatrix} N_2^3 \Delta v_2 + N_3^3 \Delta v_3 + N_4^3 \Delta v_4 \\ N_3^4 \Delta v_3 + N_4^4 \Delta v_4 \end{pmatrix} \\
&\approx \begin{pmatrix} N_2^3 \Delta v_2 + N_3^3 (\varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4) + N_4^3 (\varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4) \\ N_3^4 (\varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4) + N_4^4 (\varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4) \end{pmatrix} \\
&= \begin{pmatrix} N_3^3 \varepsilon_3^3 + N_4^3 \varepsilon_3^4 & N_3^3 \varepsilon_4^3 + N_4^3 \varepsilon_4^4 \\ N_3^4 \varepsilon_3^3 + N_4^4 \varepsilon_3^4 & N_3^4 \varepsilon_4^3 + N_4^4 \varepsilon_4^4 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix} + \begin{pmatrix} N_3^3 \varepsilon_2^3 & N_2^3 \\ N_3^4 \varepsilon_2^3 & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \end{pmatrix} \\
&= \left(N_{3-4}^{3-4} \cdot \varepsilon_{3-4}^{3-4} \right) \cdot x + \begin{pmatrix} N_3^3 \varepsilon_2^3 & N_2^3 \\ N_3^4 \varepsilon_2^3 & 0 \end{pmatrix} \cdot u \\
y(t) &= \begin{pmatrix} \Delta c_3 \\ \Delta v_3 \end{pmatrix} \\
&\approx \begin{pmatrix} \Delta c_3 \\ \varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ \varepsilon_3^3 & \varepsilon_4^3 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \varepsilon_2^3 & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ \varepsilon_3^3 & \varepsilon_4^3 \end{pmatrix} \cdot x + \begin{pmatrix} 0 & 0 \\ \varepsilon_2^3 & 0 \end{pmatrix} \cdot u
\end{aligned}
\tag{2.20}
$$

Restated in such a way, the part of the input model, which is to be reduced, is in the standard form of LTI systems. The gained matrices can now be used in model reduction.

2.4.2 Parameter Dependent Linearization

In case we want the kinetic parameters included into our linear system, we have to rewrite the differential equations to

$$
\begin{aligned}
\dot{c_1} &= N_1^1 v_1(c_1, c_2, p) + N_2^1 v_2(c_1, c_2, c_3, p) \\
\dot{c_2} &= N_1^2 v_1(c_1, c_2, p) + N_2^2 v_2(c_1, c_2, c_3, p) + N_3^2 v_3(c_2, c_3, c_4, p) \\
\dot{c_3} &= N_2^3 v_2(c_1, c_2, c_3, p) + N_3^3 v_3(c_2, c_3, c_4, p) + N_4^3 v_4(c_3, c_4, p) \\
\dot{c_4} &= N_3^4 v_3(c_2, c_3, c_4, p) + N_4^4 v_4(c_3, c_4, p)
\end{aligned}
\tag{2.21}
$$

Linearization:

$$
\begin{aligned}
v_3 &\approx \overline{v}_3 + \varepsilon^3 \Delta c + \Pi^3 \Delta p \\
&= \overline{v}_3 + \varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 + \Pi^3 \Delta p \\
v_4 &\approx \overline{v}_4 + \varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4 + \Pi^4 \Delta p
\end{aligned}
\tag{2.22}
$$

The linearization is similar to the parameter independent linearization. At first one inserts

$$x = \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix}, \qquad y = \begin{pmatrix} \Delta c_3 \\ \Delta v_3 \end{pmatrix} \qquad \text{and} \qquad u = \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \\ \Delta p \end{pmatrix} \tag{2.23}$$

into equation (2.1) and then rewrites the system with the equations given in (2.21):

$$\begin{aligned}
\dot{x}(t) &= \begin{pmatrix} \dot{\Delta c_3} \\ \dot{\Delta c_4} \end{pmatrix} \\
&= \begin{pmatrix} N_2^3 \Delta v_2 + N_3^3 \Delta v_3 + N_4^3 \Delta v_4 \\ N_3^4 \Delta v_3 + N_4^4 \Delta v_4 \end{pmatrix} \\
&\approx \begin{pmatrix} N_2^3 \Delta v_2 + N_3^3 (\varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 + \Pi^3 \Delta p) + N_4^3 (\varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4 + \Pi^4 \Delta p) \\ N_3^4 (\varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 + \Pi^3 \Delta p) + N_4^4 (\varepsilon_3^4 \Delta c_3 + \varepsilon_4^4 \Delta c_4 + \Pi^4 \Delta p) \end{pmatrix} \\
&= \begin{pmatrix} N_3^3 \varepsilon_3^3 + N_4^3 \varepsilon_3^4 & N_3^3 \varepsilon_4^3 + N_4^3 \varepsilon_4^4 \\ N_3^4 \varepsilon_3^3 + N_4^4 \varepsilon_3^4 & N_3^4 \varepsilon_4^3 + N_4^4 \varepsilon_4^4 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix} \\
&+ \begin{pmatrix} N_3^3 \varepsilon_2^3 & N_2^3 & N_3^3 \Pi^3 + N_4^3 \Pi^4 \\ N_3^4 \varepsilon_2^3 & 0 & N_3^4 \Pi^3 + N_4^4 \Pi^4 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \\ \Delta p \end{pmatrix} \\
&= \left(N_{3-4}^{3-4} \cdot \varepsilon_{3-4}^{3-4}\right) \cdot x + \begin{pmatrix} N_3^3 \varepsilon_2^3 & N_2^3 & N_{3-4}^3 \Pi^{3-4} \\ N_3^4 \varepsilon_2^3 & 0 & N_{3-4}^4 \Pi^{3-4} \end{pmatrix} \cdot u \\
y(t) &= \begin{pmatrix} \Delta c_3 \\ \Delta v_3 \end{pmatrix} \\
&\approx \begin{pmatrix} \Delta c_3 \\ \varepsilon_2^3 \Delta c_2 + \varepsilon_3^3 \Delta c_3 + \varepsilon_4^3 \Delta c_4 + \Pi^3 \Delta p \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ \varepsilon_3^3 & \varepsilon_4^3 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_3 \\ \Delta c_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ \varepsilon_2^3 & 0 & \Pi^3 \end{pmatrix} \cdot \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \\ \Delta p \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ \varepsilon_3^3 & \varepsilon_4^3 \end{pmatrix} \cdot x + \begin{pmatrix} 0 & 0 & 0 \\ \varepsilon_2^3 & 0 & \Pi^3 \end{pmatrix} \cdot u
\end{aligned} \tag{2.24}$$

Via these steps the part of the parameter dependent model, which is to be reduced, can be brought to the form of standard LTI systems. In the program PyLESS this model restatement is used to prepare the model for reduction. The complete model with the linearized part (that is to be

reduced in the following steps) is given by

$$
\begin{aligned}
\dot{c_1} &= N_1^1 v_1(c_1, c_2, p) + N_2^1 v_2(c_1, c_2, c_3, p) \\
\dot{c_2} &= N_1^2 v_1(c_1, c_2, p) + N_2^2 v_2(c_1, c_2, c_3, p) + N_3^2 v_3 \\
\dot{x} &= \left(N_{3-4}^{3-4} \cdot \varepsilon_{3-4}^{3-4}\right) \cdot x + \begin{pmatrix} N_3^3 \varepsilon_2^3 & N_2^3 & N_{3-4}^3 \Pi^{3-4} \\ N_3^4 \varepsilon_2^3 & 0 & N_{3-4}^4 \Pi^{3-4} \end{pmatrix} \cdot \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\begin{pmatrix} c_3 \\ v_3 \end{pmatrix} &= \begin{pmatrix} \overline{c}_3 \\ \overline{v}_3 \end{pmatrix} + \begin{pmatrix} I & 0 \\ \varepsilon_3^3 & \varepsilon_4^3 \end{pmatrix} \cdot x + \begin{pmatrix} 0 & 0 & 0 \\ \varepsilon_2^3 & 0 & \Pi^3 \end{pmatrix} \cdot \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix}.
\end{aligned}
\tag{2.25}
$$

Two important facts about this equation system should be denoted. At first, it contains as many differential equations as the original model plus additional algebraic equations. Therefore this model might be computationally harder to simulate. And second, at first glance in the last equation c_3 seems to be dependent of c_3. In the current form of the system this is not a problem since the c_3 vanishes from the formula as it is multiplied by zero. But during model reduction this matrix (later referred to as D) might change and the dependency would arise. This has to be kept in mind in later steps.

2.5 The Reduced Order Model

After a reduction step from the next chapter that produces the matrices T_l and T_d, which define the environment reduction of the matrices (A, B, C and D) from the LTI system (2.1) via a projection into a lower dimensional space

$$
\begin{aligned}
x_r &= T_l x \\
A_r &= T_l A T_d \\
B_r &= T_l B \\
C_r &= C T_d \\
D_r &= D,
\end{aligned}
\tag{2.26}
$$

one obtains a model of the reduced environment via implementation of the following differential and assignment equations

$$\begin{aligned}
\dot{x}_r &= A_r x_r + B_r \begin{pmatrix} \Delta c_2 \\ \Delta v_2 \\ \Delta p \end{pmatrix} = A_r x + B_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\begin{pmatrix} \Delta c_3 \\ \Delta v_3 \end{pmatrix} &= C_r x_r + D_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\begin{pmatrix} c_3 \\ v_3 \end{pmatrix} &= \begin{pmatrix} \overline{c}_3 + \Delta c_3 \\ \overline{v}_3 + \Delta v_3 \end{pmatrix}
\end{aligned} \tag{2.27}$$

with its starting values taken from the input model and determined by

$$x_r(0) = T_l \begin{pmatrix} c_3(0) - \overline{c}_3 \\ c_4(0) - \overline{c}_4 \end{pmatrix} \tag{2.28}$$

As already pointed out, the assigments made in the equation system (2.25) are problematic, because the variables in the vector c_3 are determined by the variables of the vector v_2, which themselves depend on the variable in the c_3 vector.

$$c_3 = \overline{c}_3 + \begin{pmatrix} I & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix}. \tag{2.29}$$

If a model reduction algorithm changes the matrix which is later referred to as D, the zero matrix in this equation changes and a dependence might arise.

A possible solution to this problem is to introduce another vector c_3^* that is set by the assignment

$$\begin{pmatrix} c_3^* \\ v_3 \end{pmatrix} = \begin{pmatrix} \overline{c}_3 \\ \overline{v}_3 \end{pmatrix} + C_r x_r + D_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix}. \tag{2.30}$$

This vector is afterwards used to determine the real c_3 by "pulling" it after itself

$$\dot{c}_3 = \alpha \cdot (c_3^* - c_3), \tag{2.31}$$

where α denotes a positive, real, big value. By this trick one avoids the direct dependency of c_3 on itself. After applying this trick, the final model

reads

$$\begin{aligned}
\dot{x}_r &= A_r x + B_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\begin{pmatrix} c_3^* \\ v_3 \end{pmatrix} &= \begin{pmatrix} \overline{c}_3 \\ \overline{v}_3 \end{pmatrix} + C_r x_r + D_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\dot{c}_3 &= \alpha \cdot (c_3^* - c_3) \\
\dot{c}_1 &= N_1^1 v_1(c_1, c_2, p) + N_2^1 v_2(c_1, c_2, c_3, p) \\
\dot{c}_2 &= N_1^2 v_1(c_1, c_2, p) + N_2^2 v_2(c_1, c_2, c_3, p) + N_3^2 v_3.
\end{aligned} \tag{2.32}$$

A different idea is to solve equation (2.29) explicitly for c_3 by linearizing v_2 around the steady state with the partition

$$\begin{aligned}
C_r &=: \begin{pmatrix} C_r^{(c_3)} \\ C_r^{(v_3)} \end{pmatrix} \\
D_r &=: \left(\begin{array}{ccc} D_r^{(c_3,c_2)} & D_r^{(c_3,v_2)} & D_r^{(c_3,p)} \\ \hline \multicolumn{3}{c}{D_r^{v_3}} \end{array} \right),
\end{aligned} \tag{2.33}$$

($D_r^{(c_3,c_2)}$ is here the part of D_r through which c_2 acts on c_3) in this equation

$$\begin{aligned}
c_3 &=: \overline{c}_3 + C_r^{(c_3)} \cdot x + \begin{pmatrix} D_r^{(c_3,c_2)} & D_r^{(c_3,v_2)} & D_r^{(c_3,p)} \end{pmatrix} \cdot \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
&\approx \overline{c}_3 + C_r^{(c_3)} \cdot x + D_r^{(c_3,c_2)} c_2 - D_r^{(c_3,c_2)} \overline{c}_2 \\
&+ D_r^{(c_3,v_2)} \left[\varepsilon_1^2 c_1 + \varepsilon_2^2 c_2 + \varepsilon_3^2 c_3 + \Pi^2 p \right] - D_r^{(c_3,v_2)} \overline{v}_2 + D_r^{(c_3,p)} p - D_r^{(c_3,p)} \overline{p},
\end{aligned} \tag{2.34}$$

which would yield the following equations

$$\begin{aligned}
c_3 - D_r^{(c_3,v_2)} \varepsilon_3^2 c_3 &= \overline{c}_3 + C_r^{(c_3)} \cdot x + D_r^{(c_3,c_2)} c_2 - D_r^{(c_3,c_2)} \overline{c}_2 \\
&\quad + D_r^{(c_3,v_2)} \left[\varepsilon_1^2 c_1 + \varepsilon_2^2 c_2 + \Pi^2 p \right] - D_r^{(c_3,v_2)} \overline{v}_2 + D_r^{(c_3,p)} p - D_r^{(c_3,p)} \overline{p} \\
\Leftrightarrow c_3 &= (I - D_r^{(c_3,v_2)} \varepsilon_3^2)^{-1} \cdot [\overline{c}_3 + C_r^{(c_3)} \cdot x + D_r^{(c_3,c_2)} c_2 - D_r^{(c_3,c_2)} \overline{c}_2 \\
&\quad + D_r^{(c_3,v_2)} \left[\varepsilon_1^2 c_1 + \varepsilon_2^2 c_2 + \Pi^2 p \right] - D_r^{(c_3,v_2)} \overline{v}_2 + D_r^{(c_3,p)} p - D_r^{(c_3,p)} \overline{p}]
\end{aligned} \tag{2.35}$$

This solution might be more prone to additional errors, because of the linearization of v_2 in equation (2.29). But on the other hand it is not introducing additional differential equations. Since the aim of the whole procedure is to reduce the number of differential equations, this solution should be

favored. Applying this solution the final model reads

$$\begin{aligned}
\dot{x}_r &= A_r x + B_r \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
c_3 &= (I - D_r^{(c_3,v_2)} \varepsilon_3^2)^{-1} \cdot [\overline{c}_3 + C_r^{(c_3)} \cdot x + D_r^{(c_3,c_2)} c_2 - D_r^{(c_3,c_2)} \overline{c}_2 \\
&\quad + D_r^{(c_3,v_2)} \left[\varepsilon_1^2 c_1 + \varepsilon_2^2 c_2 + \Pi^2 p\right] - D_r^{(c_3,v_2)} \overline{v}_2 + D_r^{(c_3,p)} p - D_r^{(c_3,p)} \overline{p}] \\
v_3 &= \overline{v}_3 + C_r^{v_3} x_r + D_r^{v_3} \begin{pmatrix} c_2 - \overline{c}_2 \\ v_2(c_1, c_2, c_3, p) - \overline{v}_2 \\ p - \overline{p} \end{pmatrix} \\
\dot{c_1} &= N_1^1 v_1(c_1, c_2, p) + N_2^1 v_2(c_1, c_2, c_3, p) \\
\dot{c_2} &= N_1^2 v_1(c_1, c_2, p) + N_2^2 v_2(c_1, c_2, c_3, p) + N_3^2 v_3.
\end{aligned} \tag{2.36}$$

2.6 Building a Reduced SBML Model

Converting a reduced model of the form (2.36) into SBML format is fairly simple. At first it should be explained for the idea with the additional c_3^* variables and afterwards for the second idea of how a reduced order model can be constructed.

In the beginning a new, artificial compartment with volume 1 is created. This has to be done since all SBML species, which are the variables that denote the amount of a modeled substance, have to be assigned a physical space. It is assigned this artificial value since compartments from input models are ignored by PyLESS. This is done as they are not part of the matrix-form-input it is also supposed to handle. Then SBML species for all the substances from the classes 1, 2, and 3 are created with the initial amounts given in the input to the algorithm. The last SBML species to be created are the variables given in the vector x. Since these variables represent linear combinations of the variables given in the vectors Δc_3 and Δc_4, their initial amounts have to be derived from the formula (2.28).

As pointed out in section 2.5, the values of the variables in the c_3 vector cannot be set directly by an assignment. That is why additional SBML parameters c_3^* are created for each variable in the vector c_3. Since the values for these parameters are later defined by assignments, their initial values can be set to zero. Also all parameters from the input model are transfered to the output model.

A feature for possibly shortening the size of the resulting SBML model is the declaration of additional parameters v_3^*. They are supposed to denote the reaction velocities of the reactions from class 3. The values for these variables can easily be set by equation (2.30). Storing the reaction velocities for the reactions v_3 in a vector allows us the definition of SBML reactions for the reactions v_3 with a value from the v_3^* vector as reaction velocity. Otherwise

additional declarations of *modifiers*, which are SBML species that appear in the reaction velocity term would have to be made. Applying this trick, one does not have to extract SBML modifiers from v_3 reaction velocities. This saves computational costs in building the model and storage space for left-out modifier declarations.

In a third step SBML rules are created. The SBML language provides different types of rules, from which *assignment rules* (determining the value of a variable) and *rate rules* (determining the derivate of a variable) are used in this context. First, assignment rules for the values of the parameters in the vectors c_3^* and v_3^* are created. The assigned values are lines from the matrix equations for the vectors from (2.36). Second, rate rules for the change of the amounts of the substances in the vector x are created. For each amount the rate of change is again a line from the matrix equation for the vector x from (2.36). In a third step, equation (2.31) has to be realized by a set of rate rules for the substance amounts from the vector c_3. In doing so one has to set a value for the parameter α explicitly.

A heuristically determined order of magnitude for α is the time step size of the plotting that is later used to compare resulting time courses. If the solver would use the explicit Euler algorithm [17] with a fixed step size, this would lead to exact results for the values in the c_3 vector. But since the solver should use neither explicit Euler nor fixed step sizes, this value is only an approximation.

In the final step, the differential equations for the variables in the vectors c_1 and c_2 are realized by SBML reactions. SBML reactions represent chemical reactions and contain a list of all reactants and products with their stoichiometry, an equation denoting the velocity of this reaction, a list of local parameters, and a list of modifiers, which are substances that influence the reaction's velocity. Implementing the differential equations in SBML reactions has the advantages that the inner model of the resulting SBML file can be visualized and can be simulated easier. The assembling of the SBML reactions can be done as follows: For all reaction velocities from the vectors v_1, v_2, and v_3 create a new SBML reaction with this velocity. Then take the corresponding column from the stoichiometric matrix and extract all non-zero entries. For each negative entry of the column create a new reactant for the current reaction and set its stoichiometry to the absolute value of the entry. For each positive entry of the column create a product the same way.

To this simple scheme two exceptions have to be made. In case the algorithm creates a reaction from the location class 3 , its velocity has to be set to the corresponding parameter from the vector v_3^* (since the reaction velocity has already been computed). In case the algorithm adds a substance from class 3 as a reactant or product to a reaction, it has to add it as a modifier instead. The reactions created must not modify substances from the c_3 vector, since these are already modified by SBML rate rules.

Implementing the conversion to SBML for the second idea of how a reduced order model can be created is a little more simple. No parameters for the c_3^* and no differential equations for the lagging c_3 have to be created. Instead of assigning a value to the c_3^* parameters, the c_3 are assigned values by the equation given in (2.35).

With this simple scheme a reduced order SBML model can be created.

Chapter 3

Dimension Reduction Algorithms

3.1 General Model Reduction

In every model reduction algorithm following we try to approximate the behaviour of an LTI system of order n (2.1) by a system

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r u, \qquad x_r \in \mathbb{R}^m \\ \tilde{y} &= C_r x_r + D u \end{aligned} \tag{3.1}$$

of order $m \ll n$. This approximation is supposed to minimize an error between the *transfer-function matrices* (TFM) of the original and the reduced-order model, where the TFM G of an LTI system is defined as follows [18, 19]: By applying the Laplace transformation to (2.1) one obtains the system

$$\begin{aligned} sx(s) &= Ax(s) + Bu(s) \\ y(s) &= Cx(s) + Du(s), \qquad s \in \mathbb{C}^+, \qquad s = \sigma + \omega\jmath \end{aligned} \tag{3.2}$$

which can be rewritten to

$$y(s) = (C(sI - A)^{-1}B + D)u(s) =: G(s)\ u(s). \tag{3.3}$$

Via this operation the LTI system is transferred into the frequency domain. This new function is interesting for input values $\omega\jmath$, where $\omega > 0$ denotes the frequency of an input signal in the time domain under which the system is observed.

The way the error between G and G_r (the TFM of the reduced order model) is defined, is different among model reduction algorithms. *Absolute error* model reduction algorithms (e.g. balanced truncation, adaptive-order rational global Arnoldi) try to minimize

$$||G - G_r||_\infty , \tag{3.4}$$

where

$$||G||_\infty := \sup_{\omega \in \mathbb{R}} \overline{\sigma}_{\max}(G(\omega \jmath)), \tag{3.5}$$

with $\overline{\sigma}_{\max}$ denoting the maximum singular value of $G(\omega\jmath)$. This leads to a minimization of the total output error

$$||y - y_r||_2 \leq ||G - G_r||_\infty \, ||u||_2 \,, \tag{3.6}$$

where

$$||y||_2 := \left(\int_{-\infty}^{\infty} y^*(t) y(t) dt \right)^{\frac{1}{2}} \tag{3.7}$$

[20]. *Relative error* (e.g. balanced stochastic truncation [21]) and *frequency-weighted* methods (e.g. frequency-weighted balanced truncation [22]) minimize a weighted error

$$||W_o(G - G_r)W_i||_\infty \,, \tag{3.8}$$

where W_o and W_i are weighting TFMs. The latter methods are advantageous if one needs a good approximation of the model over the complete frequency range, because absolute error methods "spend" much approximation accuracy on error peaks [23].

All the used error measures are defined in the frequency domain, but the error bounds also hold for the time domain due to the *Paley-Wiener theorem*, which is an analogue of the Parseval's formula for the Laplace transformation. For further information on this topic see [24].

An interesting property of the TFM representation of an LTI system is, that it has an infinite number of representations in the time domain. For example further dimensions can be added to an existing state space representation of a system by expanding the matrices A, B, and C. As long as the matrix C is only added zero vectors, these additional dimensions do not affect the output y at all, which means that G remains unchanged. Instead of using the state space vector x in the LTI system one can also use a vector of linear-independent linear combinations from the variables in x as a new vector. If one now changes the matrices A, B, and C with respect to this linear combination, the input-output-behaviour of the system is not altered. Because of this property the state space can be transformed with a square matrix T of full rank

$$\begin{aligned} x &\to Tx \\ A &\to TAT^{-1} \\ B &\to TB \\ C &\to CT^{-1} \\ D &\to D \end{aligned} \tag{3.9}$$

without affecting the TFM as proved by the following equation.

$$\begin{aligned} G_{\text{transformed}} &= CT^{-1}T(Ts - TAT^{-1}T)^{-1}TB + D \\ &= C(T(sI - A))^{-1}TB + D \\ &= C(sI - A)B + D \end{aligned} \tag{3.10}$$

This invariance of the TFM will be used in all model reduction algorithms.

A further property of a TFM is the unique minimal number of dimensions the state space representation has to have to produce the matrix G. This number of dimensions is called the *McMillan degree* of a system. Reducing the state space representation to this McMillan degree can be regarded as removing redundant states and should always be the first step in model reduction. As this reduction is part of balanced truncation, using BT as a prior step before other reduction algorithms is favorable.

Model reduction can now be regarded as finding a state space transformation into a **lower** dimensional space. So the aim of model reduction algorithms is to find such an appropriate transformation. It should be noted, that these transformations cannot use a matrix T^{-1} since T is no square matrix. That is why in the following two matrices have to be determined for the transformation as already seen in equation (2.26).

3.2 Balanced Truncation

Balanced truncation (BT) is one of the most prominent model reduction methods. The idea behind it is to find a transformation for the state space so that states, which are less affected by the input u do also contribute less to the output y. These states are then removed from the system.
For a mathematical definition of how the transformation matrix T of equation (3.9) for BT can be constructed, at first a few definitions have to be made.

Definition 3.2.1 (Controllable) *A pair of (time and state space) starting points (t_0, x_0) for an LTI system 2.1 is called controllable if a stepwise continuous input function can be given, which moves the system into the state x_1 at time t_1. A system is called controllable if for every starting point (t_0, x_0) the system can be moved to every end point (t_1, x_1) where $t_1 > t_0$ [14].*

Definition 3.2.2 (Observable) *An LTI system 2.1 is called observable if for every time point t_1 after the starting time point t_0 the system's initial state x_0 can be deduced from the system's state at that time point (x_1) and the input and the output function on the interval $[t_0, t_1]$ [18].*

Definition 3.2.3 (Controllability and observability gramians) *By observing an LTI system with Dirac's delta-function*

$$\delta(t-t_0) = \lim_{T\to 0} \frac{1}{T}[u(t-t_0) - u(t-t_0-T)]$$
$$\text{with} \qquad u(t-t_0) = \begin{cases} 1 & for \quad t > t_0 \\ \frac{1}{2} & for \quad t = t_0 \\ 0 & for \quad t < t_0 \end{cases} \tag{3.11}$$

[25] as an input for every component of the vector u, one gets the output $h(t) = Ce^{At}B$*. This output can be factorized into a function for the state variables* $x(t) = e^{At}B$ *(for* $x(t_0) = 0$*) and a function for the state to output conversion* $\eta(t) = Ce^{At}$*. The latter function would determine the output of a system completely for the case of a zero-input. In this case the system output is* $y(t) = \eta(t)x_0$*. The controllability and the observability gramians of the system are given by*

$$W_c \quad = \textstyle\sum_t x(t)x(t)^* \quad = \displaystyle\int_0^\infty e^{At}BB^*e^{A^*t}dt, \tag{3.12}$$

$$W_o \quad = \textstyle\sum_t \eta^*(t)\eta(t) \quad = \displaystyle\int_0^\infty e^{A^*t}C^*Ce^{At}dt. \tag{3.13}$$

[26] It should be outlined that the Laplace domain response of the LTI system under the delta-function is exactly the TFM of the system.

If one defines a controllability measure

$$E^c(p) = \max\left\{|pXf|^2\right\} \tag{3.14}$$

where f denotes an input of limited energy ($||f||_2 \leq 1$) that moves the system's states from $x(-\infty) = 0$ to $x(0) = Xf$, X is a projection from the function space to the state space (also called *analysis operator* [27]) and p denotes a row vector that produced a linear combination of the state variables, this formula can be rewritten to

$$E^c(p) = \int_0^\infty \left|pe^{At}B\right|^2 dt = pW_cp^T. \tag{3.15}$$

So W_c is a measure of control the input function has on the state of a system. If one defines the output energy of a system by

$$E_o = \int_0^\infty |y(t)|^2 \, dt, \tag{3.16}$$

the observability gramian defines the relation of state starting point x_0 to the output energy for a system without an additional input u

$$E_o(x_0) = x_0^T W_o x_0. \tag{3.17}$$

So W_o denotes a measure of observability for each dimension in the state space. [28]

When a system is controllable and observable, it is also minimal (according to the McMillan degree) and all eigenvalues of the product $W_c W_o$ are positive real numbers. The square roots of these eigenvalues are called *Hankel singular values* (HSV) of a system. Since the HSV are invariants of a system, they do not change under state space transformations. The idea is now to find a transformation under which both gramians become equal and diagonal and therefore have the HSVs on the diagonal. Such a realization of the system is called *balanced*.

Theorem 3.2.1 *The transfer matrix T for balancing of a stable, minimal system can be gained from the equation*

$$T = \Sigma^{\frac{1}{2}} U^T R^{-T} \tag{3.18}$$

where $R^T R = W_c$ is a Cholesky factorization of the gramian and $U\Sigma^2 U^T = RW_oR^T$ is a singular value decomposition of a symmetric matrix. Σ is here equal to the balanced gramians.

Proof 3.2.1 *[29]*

$$TW_cT^T = (\Sigma^{\frac{1}{2}} U^T R^{-T})(R^T R)(R^{-1} U \Sigma^{\frac{1}{2}}) = \Sigma \tag{3.19}$$

$$T^{-T} W_o T^{-1} = ((\Sigma^{-\frac{1}{2}})^T U^T R) W_o (R^T U \Sigma^{-\frac{1}{2}}) = \Sigma \tag{3.20}$$

□

It has been shown that also non-minimal systems can be reduced by balanced truncation via the following reduction matrices [30, 31]. Instead of using a Cholesky factorisation of the gramian, also full rank factors can be used, which significantly reduces the dimensions of the matrices the algorithm works with, since in most cases the gramians have a very low rank [32]. In this case the projection matrices from equation (2.26) are denoted by

$$\begin{aligned} T_l &= \Sigma^{\frac{1}{2}} V^T R \\ T_d &= S^T U \Sigma^{\frac{1}{2}} \end{aligned} \tag{3.21}$$

If one now uses only the first r rows of the matrix T_l and the first r columns of T_d as a projection matrix for the model reduction from equation (2.26), one removes states from the balanced system, which are "difficult to reach", because of their small corresponding HSV in the controllability gramian, and "difficult to observe", because of their small corresponding HSV in the observability gramian. The system is then reduced to an order

of r. For an implementation of BT making use of the sign function method for solving both Lyapunov equations at once, see the appendix.

The advantages of BT are the preservation of stability [33] and the existence of the global error bound

$$||G - G_r||_\infty \leq 2 \sum_{k=r+1}^{n_m} \sigma_k, \tag{3.22}$$

where n_m is the McMillan degree of the system and σ_k is the k^{th} Hankel singular value in descending order [34]. For a proof on the error bound see [28].

3.2.1 Computation of the Gramians via the Sign Function Method

The computationally demanding part in BT is the computation of the gramians, which can be derived from the Lyapunov equations

$$AW_c + W_c A^T + BB^T = 0 \tag{3.23}$$
$$A^T W_o + W_o A + CC^T = 0. \tag{3.24}$$

A fast method for the computation of the solutions to this equations is the *sign function method* [35].

Definition 3.2.4 *Sign function*
Let

$$Z = S^{-1} \begin{pmatrix} J_l^- & 0 \\ 0 & J_{n-l}^+ \end{pmatrix} S \tag{3.25}$$

denote the Jordan decomposition of a square real matrix Z, where J_l^- has only eigenvalues with a negative real part and J_{n-l}^+ having all eigenvalues in the right open complex half-plane. The sign function is defined as

$$sign(Z) := S^{-1} \begin{pmatrix} -I_l & 0 \\ 0 & I_{n-l} \end{pmatrix} S \tag{3.26}$$

[36]

The sign function has two interesting properties

- $(\text{sign}(Z))^2 = I$, which is why the Newton iteration can be applied to $Z^2 = I$ and
- $\text{sign}(T^{-1}ZT) = T^{-1}\text{sign}(Z)T$ for a square, regular, real matrix T.

Theorem 3.2.2 *Defining*

$$\begin{aligned} Z &:= \begin{pmatrix} A & W \\ 0 & -A^T \end{pmatrix}, \\ T &:= \begin{pmatrix} I & X \\ 0 & I \end{pmatrix}, \\ T^{-1}ZT &= \begin{pmatrix} A & AX + XA^T + W \\ 0 & -A^T \end{pmatrix} \end{aligned} \tag{3.27}$$

one can find the unique solution to the equation $AX + XA^T + W = 0$ *for a stable matrix* A *via the solution of the sign function*

$$sign(Z) = \begin{pmatrix} -I & 2X \\ 0 & I \end{pmatrix}. \tag{3.28}$$

Proof 3.2.2 *By the definition of* T *and* Z *it is clear that*

$$\begin{aligned} sign(Z) &= sign\left(T \begin{pmatrix} A & 0 \\ 0 & -A^T \end{pmatrix} T^{-1}\right) \\ &= T\, sign\left(\begin{pmatrix} A & 0 \\ 0 & -A^T \end{pmatrix}\right) T^{-1}. \end{aligned} \tag{3.29}$$

Since A *is stable, all of its eigenvalues have a negative real part. Therefore,* $-A^T$ *has all of its eigenvalues in the right open complex half-plane. That is why the following equation is a Jordan decomposition*

$$\begin{pmatrix} A & 0 \\ 0 & -A^T \end{pmatrix} = I \begin{pmatrix} A & 0 \\ 0 & -A^T \end{pmatrix} I, \tag{3.30}$$

and

$$sign\left(\begin{pmatrix} A & 0 \\ 0 & -A^T \end{pmatrix}\right) = \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix}. \tag{3.31}$$

So

$$\begin{aligned} sign(Z) &= T \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix} T^{-1} \\ &= \begin{pmatrix} -I & 2X \\ 0 & I \end{pmatrix} \end{aligned} \tag{3.32}$$

[36] □

Computationally this is done via the Newton iteration

$$Z_{k+1} \leftarrow \frac{1}{2}(Z_k + Z_k^{-1}), \tag{3.33}$$

which converges against the solution to the sign function. [24]
A faster convergence can be achieved by scaling Z_k in each step. Various factors for this scaling have been suggested, but the most frequently used are *norm scaling*, which generally results in the fastest convergence and *approximate norm scaling* whose computation can be parallelized [37].

This computation via the sign function method is computationally expensive, because it requires the computation of the inverse of A in every iteration step. Luckily the number of iteration steps is independent of the dimension (n) of A, but still the required time of this algorithm is $O(n^3)$ if A has no special structure, which can be exploited in the inversion step. The other steps in the BT algorithm do not contribute that much to the computational cost. Singular value decomposition also requires $O(n^3)$ flops (floating point operations), but is only computed once for the product of the full rank gramian factors.

3.3 Singular Perturbation Approximation

A further popular model reduction algorithm is *singular perturbation approximation* (SPA) [9]. It is very similar to BT, but has different interesting properties. While BT produces a reduced order model, which approximates the behaviour of the original model under an input function of a high frequency ($G(\infty) - G_r(\infty) = 0$), SPA approximates the model at frequency 0 ($G(0) - G_r(0) = 0$). This means for our system that the steady state error of the state space representation is minimized. Thereby the error bound for BT (3.22) remains also valid for a model reduced by SPA to the identical order [38].

SPA works as follows: At first, the system is balanced via BT without any truncation but the reduction to its McMillan degree (its minimal realization). Secondly, the resulting balanced state space matrices $(A^{\text{bal}}, B^{\text{bal}}, C^{\text{bal}})$ are partitioned

$$\begin{aligned} A^{\text{bal}} &= \begin{pmatrix} A_{11}^{\text{bal}} & A_{12}^{\text{bal}} \\ A_{21}^{\text{bal}} & A_{22}^{\text{bal}} \end{pmatrix} \\ B^{\text{bal}} &= \begin{pmatrix} B_1^{\text{bal}} \\ B_2^{\text{bal}} \end{pmatrix} \\ C^{\text{bal}} &= \begin{pmatrix} C_1^{\text{bal}} & C_2^{\text{bal}} \end{pmatrix} \end{aligned} \tag{3.34}$$

and joined to the reduced order model

$$\begin{aligned} A_r &= A_{11}^{\text{bal}} + A_{12}^{\text{bal}}(A_{22}^{\text{bal}})^{-1} A_{21}^{\text{bal}} \\ B_r &= B_1^{\text{bal}} + A_{12}^{\text{bal}}(A_{22}^{\text{bal}})^{-1} B_2^{\text{bal}} \\ C_r &= C_1^{\text{bal}} + C_2^{\text{bal}}(A_{22}^{\text{bal}})^{-1} A_{21}^{\text{bal}} \\ D_r &= D + C_2^{\text{bal}}(A_{22}^{\text{bal}})^{-1} B_2^{\text{bal}} \end{aligned} \tag{3.35}$$

[39].

3.4 Padé via Lanczos

During this section the idea behind the *implicit moment matching* via the *Lanczos* algorithm is pointed out. For simplicity the idea is first described for the single-input-single-output (SISO) case (an LTI system with one input and one output variable) and afterwards extended to multiple-input-multiple-output (MIMO) cases.

A SISO LTI system

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= cx + du, \end{aligned} \tag{3.36}$$

where A is an $n \times n$ matrix, b is a column vector, c is a row vector and d, u, and y are numbers, has the transfer function

$$h(s) = c(sI - A)^{-1}b + d. \tag{3.37}$$

This transfer function has a Taylor expansion

$$\begin{aligned} h(s) &= d + cb(s - s_0) + c(sI - A)^{-1}b(s - s_0)^2 + c(sI - A)^{-2}b(s - s_0)^3 + ... \\ &:= d + m_0(s - s_0) + m_1(s - s_0)^2 + m_2(s - s_0)^3 + ... \end{aligned} \tag{3.38}$$

around s_0, where the $m_z(s_0)$ are called *moments*. Hankel has shown, that from the first $2n$ moments of a minimal system, A, b, and c can uniquely be derived [40]. (On non-minimal systems a minimal realization of degree n_{mm} could be derived from the first $2n_{mm}$ moments, where n_{mm} denotes the McMillan degree of the system.) The idea of the Padé approximation is now to produce a reduced order transfer function h_r of dimension $n_r \ll n$ only from the first $2n_r$ moments at a given frequency s_0.

$$h_r(s) = \frac{\alpha_{n_r-1}s^{n_r-1} + ... + \alpha_1 s + \alpha_0}{\beta_{n_r}s^{n_r} + ... + \beta_1 s + 1} \tag{3.39}$$

where the α and the β can be computed via

$$\begin{pmatrix} m_0 & m_1 & \cdots & m_{n_r-1} \\ m_1 & m_2 & \cdots & m_{n_r} \\ \vdots & \vdots & & \vdots \\ m_{n_r-1} & m_{n_r} & \cdots & m_{2n_r-2} \end{pmatrix} \begin{pmatrix} \beta_{n_r} \\ \beta_{n_r-1} \\ \vdots \\ \beta_1 \end{pmatrix} = \begin{pmatrix} m_{n_r} \\ m_{n_r+1} \\ \vdots \\ m_{2n_r-1} \end{pmatrix} \tag{3.40}$$

and

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n_r-1} \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 0 & m_0 \\ \vdots & & m_0 & m_1 \\ 0 & & & \vdots \\ m_0 & m_1 & \vdots & m_{n_r-1} \end{pmatrix} \begin{pmatrix} \beta_{n_r-1} \\ \vdots \\ \beta_1 \\ 1 \end{pmatrix}. \tag{3.41}$$

This explicit moment matching is done in the *asymptotic waveform evaluation* (AWE) algorithm [41], which usually suffers from the numerically ill-conditioned moment-matrix from equation (3.40), which is also often referred to as *Hankel matrix*. AWE is not explained in this context in detail, but it should be mentioned that the resulting reduced order LTI system (A_r, b_r, c_r) in the form $\begin{bmatrix} 0 & c_r \\ b_r & A_r \end{bmatrix}$ has a tridiagonal structure and is therefore cheap in computation.

3.4.1 Implicit Moment Matching

A solution to this problem is given by the Padé-via-Lanczos (PVL) algorithm, which computes a Padé approximation of a given system, without computing the Hankel matrix. Therefore it (and other similar algorithms) are often referred to as *implicit moment matching* algorithms.

Definition 3.4.1 *Krylov space*
A right Krylov space of order n defined by a matrix A and a given column vector r is the space spanned by the vectors $r, Ar, A^2r, ..., A^{n-1}r$, *which are called* Krylov sequence.

$$\mathcal{K}_n(A, r) = \text{span}\left\{r, Ar, A^2r, ..., A^{n-1}\right\} \tag{3.42}$$

A left Krylov space is defined analogously as the space spanned by the vectors $l, lA, lA^2, ..., lA^{n-1}$, where l is a row vector.

Theorem 3.4.1 *The vector spanning the Krylov space and the systems moments are connected.*

Proof 3.4.1

$$\begin{aligned} m_{2z} &= c(A - s_0I)^{-2z}b \\ &= c(A - s_0I)^{-z}(A - s_0I)^{-(z-1)}(A - s_0I)^{-1}b \\ &= \left[((A - s_0I)^{-z})^T \cdot c^T\right]^T \cdot \left[(A - s_0I)^{-(z-1)} \cdot ((A - s_0I)^{-1}b)\right] \\ m_{2z+1} &= c(A - s_0I)^{-(2z+1)}b \\ &= c(A - s_0I)^{-z}(A - s_0I)^{-z}(A - s_0I)^{-1}b \\ &= \left[((A - s_0I)^{-z})^T \cdot c^T\right]^T \cdot \left[(A - s_0I)^{-z} \cdot ((A - s_0I)^{-1}b)\right]. \end{aligned} \tag{3.43}$$

The right hand sides of these equations are nothing else but a product of the z^{th} *vector spanning the left Krylov space induced by* $((A - s_0I)^{-T}, c^T)$ *and the* z^{th} $((z-1)^{th})$ *vector spanning the right Krylov space induced by* $((A - s_0I)^{-1}, ((A - s_0I)^{-1}b))$ *[42].*

□

The idea is now to use a Krylov space projector $\Pi_{n_r} = \Pi_{n_r}^2 = V_{n_r} W_{n_r}^T$ to produce a reduced order model, where the columns of the matrix V have to span the n_r^{th} order right Krylov space of $(A - s_0 I)^{-1}$ and $(A - s_0 I)^{-1} b$ and the columns of W have to span the n_r^{th} order left Krylov space of $(A - s_0 I)^{-1}$ and c^T. To obtain the formulas for the reduced order model, one starts with the LTI system

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= cx + du. \end{aligned} \tag{3.44}$$

Now the state space vector is approximated by a lower order vector $x \approx V_{n_r} x_r$ and the first equation is left multiplied with the matrix $W_{n_r}^T$, which results in a reduced order model

$$\begin{aligned} W_{n_r}^T V_{n_r} \dot{x_r} &= W_{n_r}^T A V_{n_r} x_r + W_{n_r}^T b u \\ y &= c V_{n_r} x_r + du. \end{aligned} \tag{3.45}$$

So the transformation

$$\begin{aligned} x_r &:= W^T x \\ A_r &:= (W_{n_r}^T V_{n_r})^{-1} W_{n_r}^T A V_{n_r} \\ b_r &:= (W_{n_r}^T V_{n_r})^{-1} W_{n_r}^T b \\ c_r &:= c V_{n_r} \\ d_r &:= d \end{aligned} \tag{3.46}$$

produces the reduced order approximation of the original LTI system.

Via this operation the matrix A is projected into the right Krylov space and orthogonally into the left Krylov space. The original and such an reduced order system will have common first $2n_r$ moments [43, 44]. That is why this algorithm produces a Padé approximation of the original model. It should also be denoted that the matrix $W_{n_r}^T A V_{n_r}$ has the same tridiagonal structure, which is achieved by explicit moment matching.

For the PVL algorithm an inaccurate error bound exists [45], but since this cannot easily be extended to the MIMO case, it is not shown here.

3.4.2 The Lanczos Algorithm

In the normal case constructing a Krylov space via a Krylov sequence does not reveal an orthogonal or even near-orthogonal basis. Since such a basis of the Krylov space is preferred for numerical reasons in further computations, it is constructed in a different way.

Given a matrix $A \in \mathbb{R}^{n \times n}$, two starting column vectors $r, l \in \mathbb{R}^n$, and a desired order n_r, the non-symmetric Lanczos algorithm constructs two

biorthogonal sequences of vectors $V_{n_r} = [v_1, ..., v_{n_r}] \in \mathbb{R}^{n_r \times k}$ and $W_{n_r} = [w_1, ..., w_{n_r}] \in \mathbb{R}^{n_r \times k}$ ($V_{n_r}^T W_{n_r} = 0$), spanning the left and right Krylov spaces of (A^T, l) and (A, r) [46].

Algorithm 1 *Non-symmetric Lanczos*
$v_1 = r/\beta_1$ *and* $w_1 = l/\gamma_1$ *where* $\beta_1 = \pm\gamma_1$ *and* $w_1^T v_1 = 1$
For j in 1..k:
. $\alpha_j = w_j^T A v_j$
. $r_j = A v_j - \alpha_j v_j - \gamma_j v_{j-1}$
. $q_j = A^T w_j - \alpha_j w_j - \beta_j w_{j-1}$
. $\beta_{j+1} = \sqrt{|r_j^T q_j|}$
. $\gamma_{j+1} = signum(r_j^T q_j)\beta_{j+1}$
. $v_{j+1} = r_j/\beta_{j+1}$
. $w_{j+1} = q_j/\gamma_{j+1}$
[40]

Using the matrices derived via this algorithm results in a numerically better conditioned model reduction.

3.4.3 Matrix Padé via Lanczos

Matrix Padé via Lanczos (MPVL) is an extension of the PVL algorithm to MIMO LTI systems [11]. The idea behind it is to construct slightly different Krylov spaces with respect to the matrices $(A - s_0 I)^{-1}, B$ and C, which also contain the first moments of the Taylor approximation of the TFM, that are denoted by

$$m_0 = D, \qquad m_z = C(A - s_0 I)^{-z} B \qquad \text{for } z \geq 1 \tag{3.47}$$

in the MIMO case. Therefore another definition has to be made.

Definition 3.4.2 *Block Krylov space*
A right block Krylov space of n_r^{th} *order induced by the matrices A and R is the space spanned by the columns of the matrix* $\begin{bmatrix} R & AR & \cdots & A^{n_r-1}R \end{bmatrix}$. *The left block Krylov space is defined analogously.*

Analogously to (3.43), one can see that the first $2n_r$ moments of an n_r^{th} order MIMO system again match the moments of the original system, since these moments lay in space of the left and the right block Krylov space induced by $((A - s_0 I)^{-1}, (A - s_0 I)^{-1} B)$ and $((A - s_0 I)^{-T}, C^T)$ and are therefore projected on themselves during reduction.

For the computation of the block Krylov subspace, the Lanczos algorithm has to be extended. To compute a basis for the matrix that defines the block Krylov space one can successively add column vectors from this matrix to the basis and omit those, which are linearly dependent of the current basis.

This process is from now on referred to as *deflation* or *inexact deflation* if not only linearly dependent but also almost linearly dependent columns are not taken into the basis. Another problem in the MIMO case is that the starting matrices for the left and the right block Krylov space can be of different dimensions. The last problem is a numerically one. During the iteration two vectors are normalized (divided by their norm), which could result in a division by zero, if the vectors are (almost) the zero vector. This can be dealt with by incorporating a method called *look-ahead* [47], that continues the Krylov space building process in such a case by constructing different Krylov spaces, which are not completely biorthogonal anymore. An implementation of this generalized Lanczos algorithm with deflation but without look-ahead techniques can be found in the appendix. Due to the time limitation of this thesis, the better algorithm could not be implemented. The algorithm without look-ahead might have problems in constructing Krylov spaces of relative high order, which results in problems when MPVL is supposed to construct reduced models of a dimension near the McMillan degree.

Applying this algorithm to produce the block Krylov spaces V_{n_r} and W_{n_r}, one obtains a reduced order approximation of a MIMO LTI system via the transformation given in (3.46). Although the computation of the Krylov spaces is quiet cheap ($O(n^2 n_r)$) as it only includes $O(n_r)$ matrix-vector-multiplications ($O(n^2)$), the complete reduction takes $O(n^3)$ flops as the inverse of the matrix $(A - s_0 I)$ has to be computed explicitly. But since this matrix is sparse, the $O(n^3)$ bound can in general be broken with appropriate inversion algorithms. Currently no Python implementations of such algorithms are known to the author. That is why the implemented algorithm in the appendix still needs $O(n^3)$ flops to compute the reduction matrices V and W.

3.4.4 Multi Point Padé Approximations

The Padé approximation can also be generalized to produce an reduced order system, which tries to match moments for different expansion frequencies. A method for producing such an approximation via a modified Arnoldi algorithm has been developed in [48] and it has been transferred to the Lanczos algorithm [49]. These methods are referred to as "rational" moment matching algorithms.

A newer moment matching algorithm called Adaptive-order rational global Arnoldi (AORGA) [50] makes use of all aforementioned techniques and also includes a global error bound, which makes it possible to adapt the order of the reduced model to match with a given maximum error. It should be noted that the *rational algorithms* cannot compute the reduction matrices in $O(n^2)$ and therefore are not assumed to be much faster than singular value decomposition (SVD) methods.

3.5 Decompositions of the A Matrix

3.5.1 Decomposition of A into a Regular and a Singular Part

Since for the model reduction algorithms based on SVD (BT and SPA) the matrix A is required to be regular, a model with a singular matrix A has to be decomposed before reduction. If one looks at the construction of the matrix A in section 2.4, it is easy to see that it can get rank-deficient either because of a row-rank-deficient N_{3-4}^{3-4} or a column-rank-deficient ε_{3-4}^{3-4}. N can get rank-deficient because of conservation relations inside the environment (these do not need to be conservation relations of the complete input model) and ε can contain columns (and also rows) of zeros if a substance in the environment does not influence the velocity of a reaction of the environment. Also other reasons for A becoming singular are possible, but these are the most prominent ones.

To overcome this problem, a new matrix S is introduced, which decomposes the state space representation of a singular LTI system as follows:

$$\begin{aligned} A' &= S^{-1}AS \qquad \text{where} \qquad A' = \begin{pmatrix} A'_{reg} & 0 \\ 0 & 0 \end{pmatrix} \\ B' &= \begin{pmatrix} B'_{reg} \\ B'_{sin} \end{pmatrix} = S^{-1}B \\ C' &= \begin{pmatrix} C'_{reg} & C'_{sin} \end{pmatrix} = CS \\ x' &= \begin{pmatrix} x_{reg} \\ x_{sin} \end{pmatrix} = S^{-1}x. \end{aligned} \tag{3.48}$$

Afterwards the regular part of the system $(A'_{reg}, B'_{reg}, C'_{reg}, D, x_{reg})$ can be reduced to $(A'_r, B'_r, C'_r, D'_r, x_r)$ and combined again with the singular part, which results in a reduced order model

$$\begin{aligned} \begin{pmatrix} \dot{x_r} \\ \dot{x_{sin}} \end{pmatrix} &= \begin{pmatrix} A'_r \\ 0 \end{pmatrix} x_r + \begin{pmatrix} B'_r \\ B'_{sin} \end{pmatrix} u \\ y &= \begin{pmatrix} C'_r & C_{sin} \end{pmatrix} \begin{pmatrix} x_r \\ x_{sin} \end{pmatrix} + D'_r u. \end{aligned} \tag{3.49}$$

The needed matrix S can be computed via the following procedure [51]:

- Compute a "swapped" Schur decomposition [52]

$$A = Q \begin{pmatrix} A_1 & A_2 \\ 0 & A_3 \end{pmatrix} Q^*, \tag{3.50}$$

  where A_3 contains all zero diagonal entries.

- Solve the Sylvester equation $A_1 X - X A_3 + A_2 = 0$

- Compute

$$S = Q \begin{pmatrix} I & X \\ 0 & I \end{pmatrix} \tag{3.51}$$

PROVE ME!

3.5.2 Decomposition of A into a Stable and an Unstable Part

The algorithms BT and SPA cannot be applied to unstable systems. Therefore when trying to reduce an unstable system with these algorithms, one has to employ further methods like modal separation [53, 54], which decomposes the systems TFM into a stable and an unstable part $G = G^- + G^+$, reduce the stable system G^- to G_r^-, and combines the reduced stable and the unstable part again $G_r = G_r^- + G^+$, or coprime factorisation of the TFM [39, 55]. But since in this context no unstable system shall be treated none of these methods have been implemented.

3.5.3 Shifting

A more simple idea to make A stable is to construct a matrix $A_{shift} = A - sI$, where s denotes a small, positive, real number, and use this matrix for model reduction instead of A. By choosing a suitable value for s the matrix A_{shift} can be constructed to be regular (but numerically ill-conditioned) and stable. The value s should be sufficiently large to make A invertible, but it should be as small as possible since it significantly changes the behaviour of the "shifted" LTI system.

For the implementation of the model reduction program this simple method to make A regular has been chosen. A numerically stable implementation of the algorithm mentioned in 3.5.1 would have been too time consuming since the needed "swapping Schur decomposition" is not available in Python. The ideas mentioned in 3.5.2 also have not been implemented in PyLESS.

Chapter 4

Experimental Results

4.1 Different Error Measures

Before the different model reduction algorithms can be compared, a measure of approximation quality has to be developed. In the introduction of the ideas behind model reduction algorithms, it has already been pointed out that a minimization of the error between the original and the reduced order TFM (via an ∞-norm) leads to a minimization of the error on the output of the reduced order model (via the 2-norm) (see equation 3.6). Since this error depends on the input to the reduced order model, the first idea of an objective approximation error could be

$$E := ||y - y_r||_2 \,/\, ||u||_2 \,. \tag{4.1}$$

Defining an error like this would be a good idea if only the reduced order model would be observed. But if the complete system including the inner model and the reduced order environment is observed, feedbacks could arise. The reduced order model's output might feed back on its input through the inner model, which could result in a partial cancellation of the error. In general we would expect the model with the reduced order environment to be a good estimation if the time courses of the concentrations are similar to those of the original model.

A different measure of the approximation error, which also takes the behaviour of the inner model into account, is the mean square error (MSE) of the concentration time courses in the original and the complete reduced order model, which is defined as follows

$$E := \frac{1}{|c| \cdot |t|} \sum_{c_i \in c} \sum_{t_i=0}^{t_{\max}} (c_{i,\mathrm{orig}}(t_i) - c_{i,\mathrm{red}}(t_i))^2, \tag{4.2}$$

where $|c|$ is the number of compared substances and $|t|$ is the number of simulated time points, and is also used in linear regression. Then again, this

measure is very sensitive to high errors in only a single variable. A more robust measure with respect to possible outliers is a mean square relative error (MSRE)

$$E := \frac{1}{|c| \cdot |t|} \sum_{c_i \in c} \sum_{t_i=0}^{t_{\max}} \left(\frac{|c_{i,\text{orig}}(t_i) - c_{i,\text{red}}(t_i)|}{\max\left(|c_{i,\text{orig}}(t_i)|, |c_{i,\text{red}}(t_i)|\right)} \right)^2. \tag{4.3}$$

On the one hand, this measure would better qualify the similarity in the time courses also for low concentrated substances, but on the other hand it does not quantify the difference in the same way as the MSE error. Especially for models where the concentrations do not differ by orders of magnitude this could be preferred. In the model comparison both error measures are used.

4.2 Example Models

Rohwer model of Sucrose Accumulation in Sugar Cane Culm (Taken from Biomodels) [56]

The model of Rohwer (the topologies of this and the other models mentioned is shown in the appendix) has been chosen as a first application of the model reduction algorithms to biochemical network models, because many subsystems of this model are stable. As mentioned before, unstable systems are not regarded in the context of this thesis as they are harder to handle. But since most of the models available in databases do not have that many interesting stable subsystems, they have to be modified before this algorithm can be applied to it. Artificially stabilizing an unstable system can be done by adding a *dilution* to the system. So for all non-constant substances in the model a new degrading reaction with the linear kinetic ($v_{c_i} = kc_i$) is introduced. Therefore all substances have more *control* over their degradation, as a higher concentration leads to a faster degradation. So the values on the diagonal of the system's Jacobian, which are a measure for the control of a substance on itself, get more negative. Due to this fact, the eigenvalues become more negative. Since a system is stable if all eigenvalues have negative real parts, the system gains stability through this procedure. By introducing these dilution reactions with an appopriate k, every model can be converted to a different, but stable model. This has been done for the rest of the model with $k = 0.5$.

The model describes the sucrose accumulation in a developing sugar cane (*Saccharum officinarum*) culm tissue and the main anabolic and catabolic processes sucrose is involved in. Since this model is very small, it has been investigated for its behaviour if the system is completely reduced. This is done by defining one single substance as the inner model and the rest as environment. The behaviour has been observed for different substances as the inner model, for different algorithms, for different desired orders of

the reduced environment, and in the case of the algorithm MPVL also for different expansion frequencies.

Hynne Model of Glycolysis (Taken from JWS) [57]

The Hynne model of the glycolysis has also been observed under dimension reduction for a small inner model and a large environment. It describes the glucose uptake, the glycolysis, and the ethanol fermentation, which altogether is known as anaerobic respiration, in *Saccharomyces cerevisiae*. This represents the normal way of glucose catabolism in yeast when not enough oxigen is available for aerobic respiration. As an inner model the two variable glucose concentrations in two different compartments and the transport connecting them are declared. The rest of the model serves as an environment. The uptake of glucose in the model has been lowered by a factor of 10 in order to prevent the system from oscillating.

Hoefnagel Model of Glycolysis and Pyruvate Branch (taken from JWS) [58]

On the model of Hoefnagel the principal idea of reducing an environment built from *modules* is tested. The model contains a module for the glycolysis and the pyruvate branch in *Lactococcus lactis*, where the glycolysis is treated as the inner model and the rest as the environment. In fact, this model has a very similar topology to the Hynne model, but it contains more reactions downstream of pyruvate. This is also the only connection between the glycolysis and the pyruvate branch, which makes the *interface* between the inner model and the environment low-dimensional.

4.3 Comparison of Time Courses

In this section the simulation results of an original models are compared to its linearized and its reduced order model. The different SBML files produced by PyLESS have been compared by simulating them over time using the SBMLodeSolver [59]. Then the results are plotted with GnuPlot [75] and the mean square errors, that are introduced by the linearization and the reduction, are computed.

At first it should be shown along time courses, that a computed reduced order model is able to approximate the behaviour of the original model. In the figures 4.1, 4.2, and 4.3 three simulations are performed. The pictures show that the time courses produced by the Hoefnagel model reduced by SPA to dimension 4 catch the major dynamics of the system. A closer look at the time courses reveals, that minor movements in the state space are only approximated.

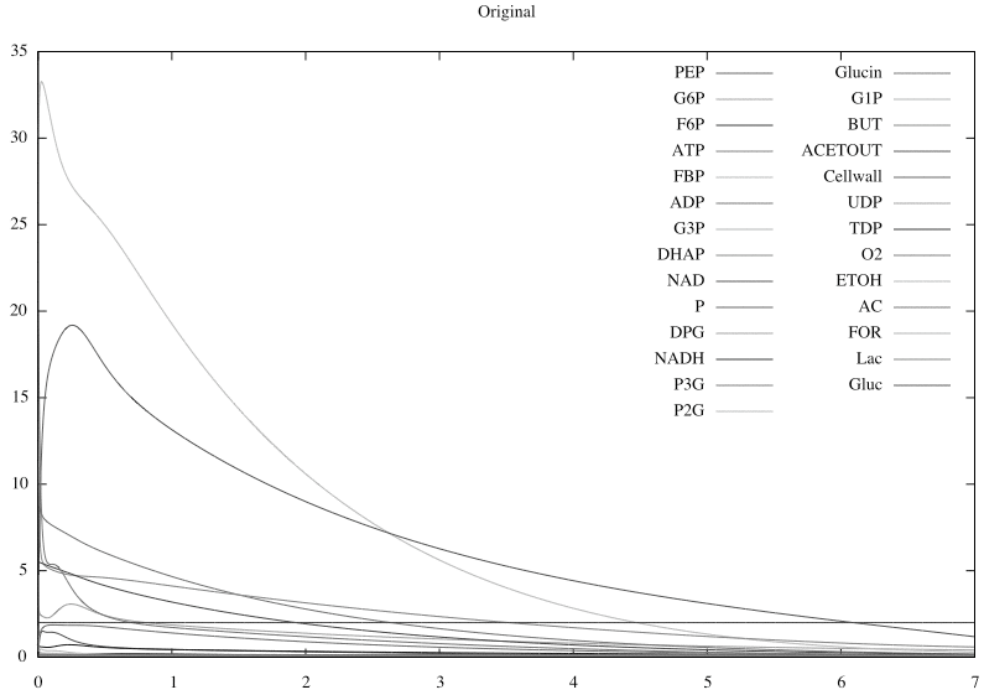

Figure 4.1: Original stabilized model of Hoefnagel simulated over time until it reaches its steady state.

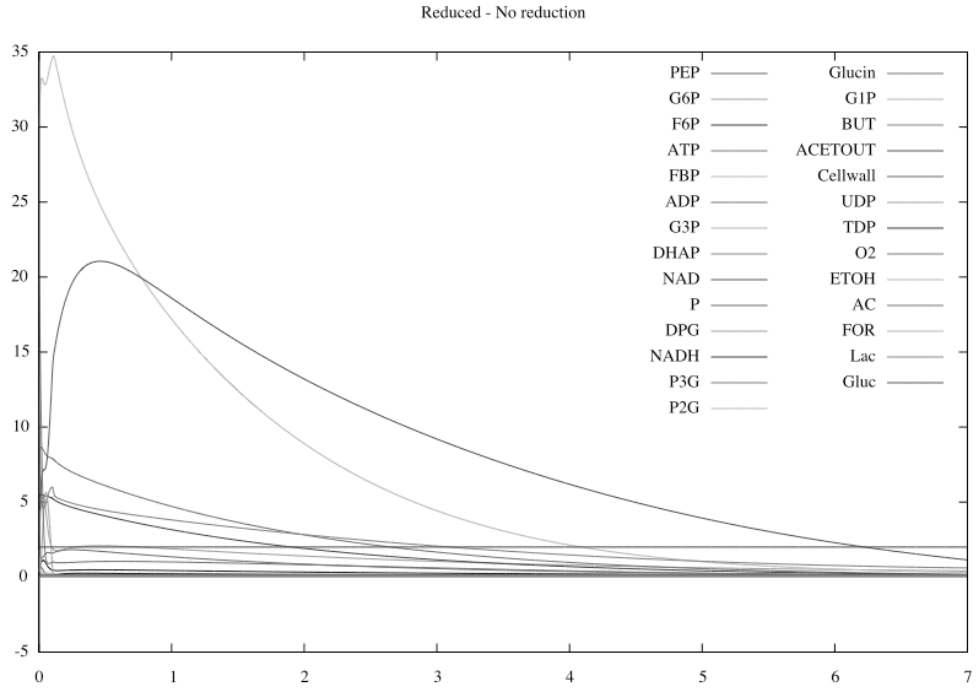

Figure 4.2: Linearized model simulated over the same period of time.

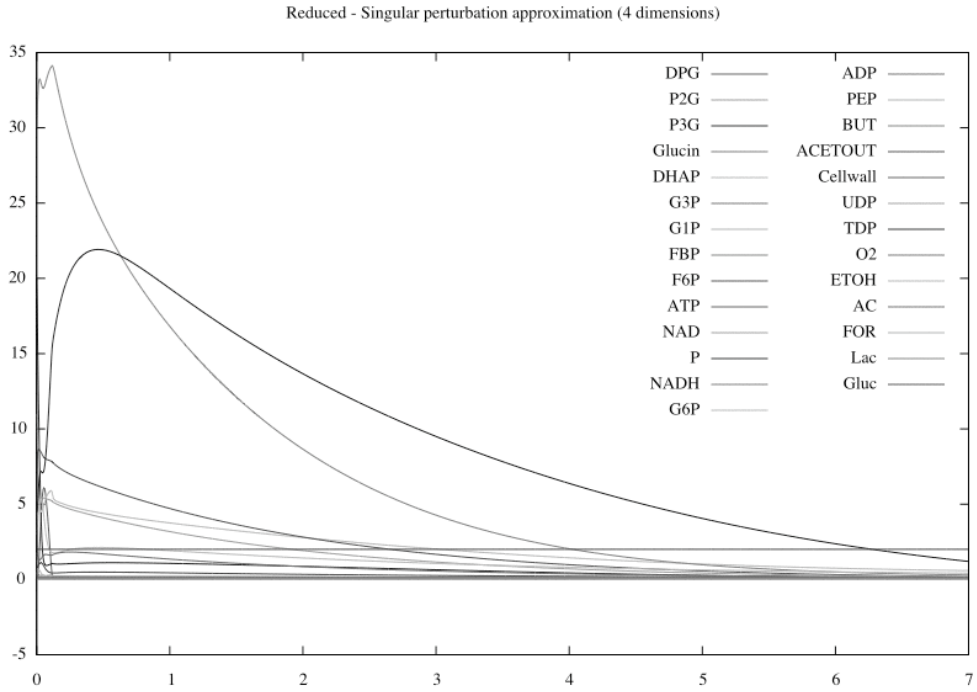

Figure 4.3: Model reduced by SPA to dimension 4.

Most of the difference between this reduced order model and the original one results from the linearization. The plot of the linearized model differs more from the original model than from the reduced order one. This difference can also be quantified by observing the MSRE (MSE) which is 0.0035 (0.74) compared to the original model and 0.0018 (0.014) compared to the model reduced by SPA. Having a closer look at the error measures reveals that this difference is even more clear in the MSE values. This results from the fact that the curves of the two substances with the highest concentration differ clearly between the linearized and the original model. In the following observations the linearization and the reduction error are treated separately as the linearization error does not change for a certain model.

These figures are thought to convince the reader that reduced order models perform well in simulation. Such time courses cannot be shown for the following comparison of the algorithms and the discussion of the appropriate order and frequency selection. Instead of that MSRE and MSE are from now on used to quantify the quality of the approximation.

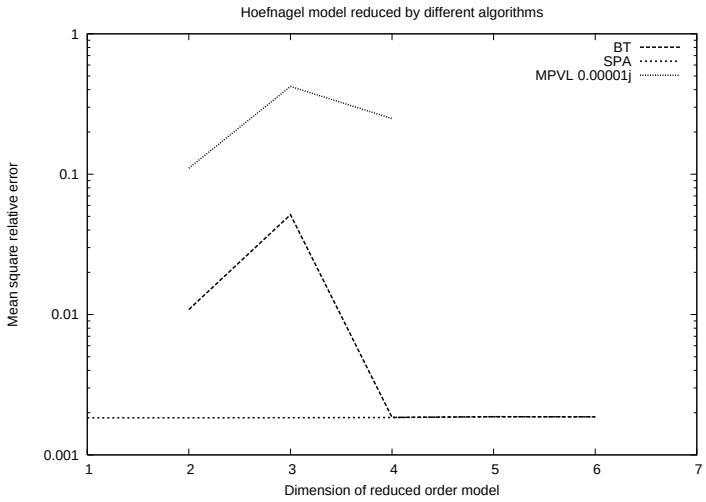

Figure 4.4: MSRE for the Hoefnagel model reduced with different reduction algorithms (BT, SPA, and MPVL with expansion frequency $0.00001j$) to different dimensions. The additional relative linearization error for this model is 0.0035.

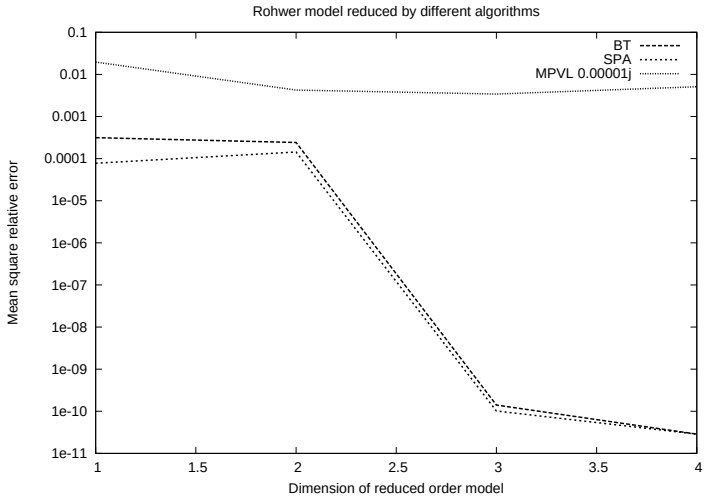

Figure 4.5: MSRE for the Rohwer model reduced around fructose with different reduction algorithms to different dimensions. The additional relative linearization error for this model is 0.0036.

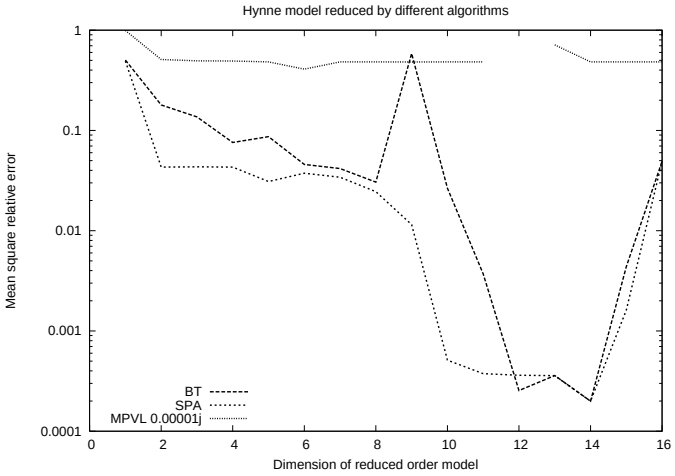

Figure 4.6: MSRE for the Hynne model reduced by different reduction algorithms. The additional relative linearization error is here 0.46 and the absolute one is 1.44.

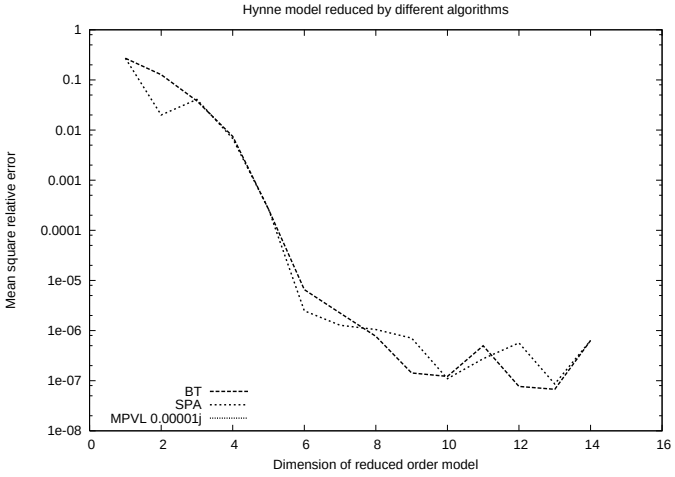

Figure 4.7: MSRE for the Hynne model with altered initial conditions reduced by different reduction algorithms. The initial concentrations for the glucose (2 variables in two different compartments) has been increased by two orders of magnitude. This leads to a significantly lower relative linearization error of 0.18, but a higher absolute one (16.34).

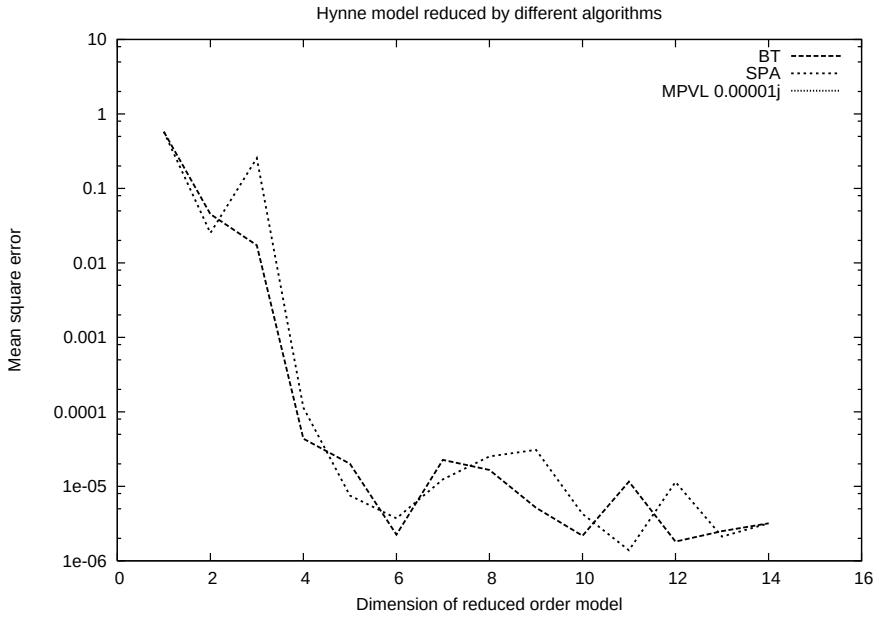

Figure 4.8: MSE for the Hynne model with altered initial conditions reduced by different reduction algorithms. The initial concentrations for the glucose (2 variables in two different compartments) has been increased by two orders of magnitude.

4.4 Order Selection

In this section the MSRE and the MSE for different models reduced by different algorithms is observed. In the figures 4.4, 4.5, and 4.6 the approximation error between the linearized and the reduced order model for three different original models are compared. A trivial observation is that the reduction error decreases as more dimensions are retained in the environment. In all of these figures and in almost every test case the author came up with SPA clearly performed best, while MPVL produced quite bad approximations. The Rohwer model reduced around sucrose (not shown) can be taken as an example on which BT produces very good results. Another observation is that the algorithms did not produce a result in every case. The reason for that is the SBMLodeSolver encountered problems during the simulation of these models as probably contained too stiff differential equations. More often than the other algorithms MPVL failed to produce a result. The reason for this is that MPVL automatically reduces the desired order of the environment in case it encounters problems in constructing the Krylov spaces.

If such a problem occured, the resulting approximation error is not shown in the figures.

Comparing the relative linearization error with the MSRE values from the figure leads to the observation that it has the same order of magnitude as the reduction error of SPA for very low dimensions. So in most of the cases the environment can be reduced to a very low dimension without additionally increasing the approximation error due to the linearization. Having said that it should also be clear that for most non-linear systems the linearization error determines the quality of the approximation.

In the figures 4.6 and 4.7 the approximation error for a model with two different initial conditions is observed. The second condition starts further away from its steady state than the first one. As already mentioned, the linearization error gets larger the further the system is draw away from its steady state. This can be observed by comparing the absolute errors introduced by the linearization which gets larger for the second conditions. Surprisingly, the relative linearization error behaves controversially. The reason for that might be that the linearized model is still able to approximate the relative fast convergence towards the steady state in the beginning of the time course. This fast convergence dominates the overall behaviour of the system and so the relative error under the second conditions gets better.

Keeping this fast convergence towards the steady state in mind one can explain the difference between the absolute and the relative error shown in the figures 4.7 and 4.8. The absolute error decreases very fast in the beginning for increasing dimensions of the reduced order model. It is already very low for dimension 6. The relative error shows a slower decrease up to dimension 10. This fact can be explained by the way the *absolute error reduction* methods are supposed to work. As they try to reduce this kind of error, for low dimensions the reduced order model approximates the behaviour of higher concentrated substances better. For these substances the absolute error increase significantly already for a small relative error. As the relative error does not decay as fast as the absolute error in the figures, this fact is underlined. In case such a behaviour is unwanted, *weighted error* methods should be employed for the model reduction.

4.5 Comparison of Different Frequencies as an Expansion Point for MPVL

For the model reduction algorithm MPVL the expansion frequency is important for the quality of the approximation. As shown in figure 4.9 the resulting approximation error differs for a model reduced to identical order at different expansion frequencies. A way of how to determine a good expansion frequency is not known in the literature. So, when using MPVL for model reduction the user has to test the quality of the approximation for

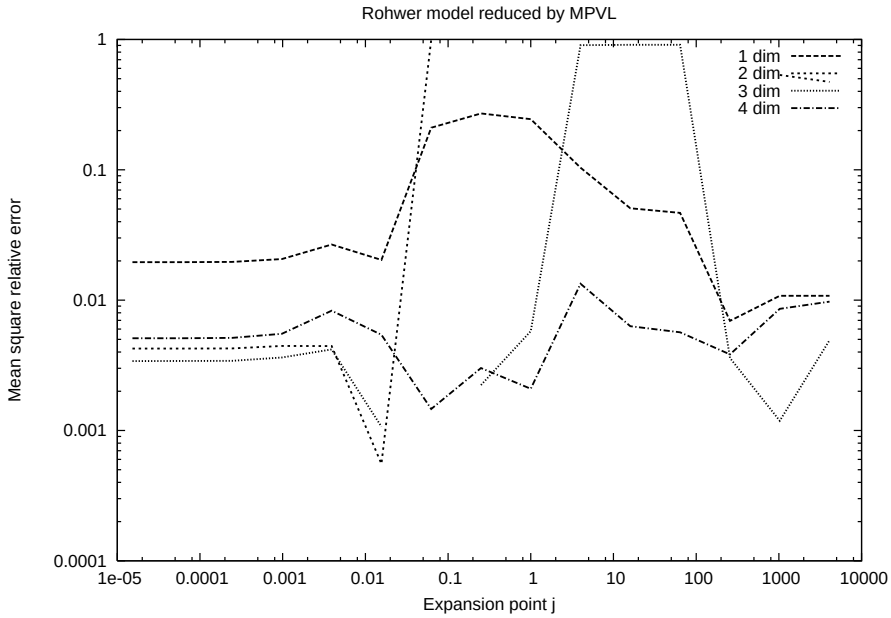

Figure 4.9: MSRE for the Rohwer model reduced by MPVL around fructose to different dimensions at various expansion frequencies.

different expansion frequencies by hand.

4.6 Comparison of Computational Efficiency

The following computations were performed on an *Apple G4 1.2GHz 512 MB* using Python 2.4.2 and SciPy 0.4.9.

As already mentioned before, the computational costs of the algorithms

	Reduction algorithms		
Model	MPVL	BT	SPA
Rohwer (4 dim)	0.33	0.19	0.21
Hoefnagel (7 dim)	0.49	0.34	0.35
Hynne (20 dim)	0.34	0.61	0.60
Artificial (32 dim)	0.70	3.08	3.09

Table 4.1: Time needed for the reduction of different models to dimension 1 in seconds.

differ. While the costs of BT and SPA scale cubically in the size of the state space of the environment model, the costs of MPVL only scale quadratically (excluding the computation of one inverse which costs again $O(n^3)$). This fact shows up when comparing the time needed for reduction of different models by different algorithms (table 4.1). On the one hand, for models with a high dimensional state space BT and SPA might not be feasible, while the costs of MPVL could still be acceptable. On the other hand, with respect to the number and the size of the models currently available in public databases, the size of the models to be reduced will probably not exceed a few thousand substances. That is why for relatively huge models the costs of BT and SPA might still be acceptable.

By the way, in all tests performed by the author the computation of the steady state took much longer than the reduction procedure, so this might rather be the computational bottleneck of the whole procedure.

4.7 Comparison of Efficiency in Simulation of Different SBML Exports

	Simulation time			MSE	
Reduced model	Orig	Exp1	Exp2	Exp1	Exp2
Rohwer (4 dim)	0.78	2.02	1.30	1.90	1.90
Hoefnagel (7 dim)	4.19	301.96	11.96	0.92	0.92
Hynne (20 dim)	3.71	3.64	3.09	3.62	3.65

Table 4.2: Comparison of the simulation time in seconds and the approximation accuracy of SBML models exported via different methods. Exp1 denoted the export with the *lagging* c_3 and Exp2 denotes the export including the additional linearization of the class 2 reactions. In both cases the original model has been reduced by SPA to dimension 1 and the simulation time until steady state in the SBMLodeSolver [59] has been compared. For both models the accuracy of the approximation is compared by the mean square error of their simulated time courses.

On the one hand, during the various simulations of reduced order models the second method of exporting SBML files has been significantly faster in simulation of even small models. The reason for that might be, that the first method introduces stiff differential equations by the rules for the *lagging* c_3. On the other hand, it looses accuracy by linearizing additional reaction velocities. But due to the heavy performance gain and the observation that the additional error for the linearization of class 2 reactions is insignificant compared to the general linearization error, the second method is preferred by the author. A comparison of the export methods in efficiency and accuracy

is given in table 4.2. Without the use of this method the high-throughput testing shown in this chapter would probably have been impossible as the difference in the simulation time is even more remarkable on models reduced by MPVL.

The reason for this performance gain might be, that the *lagging* c_3 rules often become quite stiff. This makes the simulation of such a model very slow as the solver has to use a very small internal step size.

Chapter 5

Discussion

5.1 Framework

Dimension reduction of metabolic network models is a good way of reducing the computational costs for its simulation. When working on big network models with computationally demanding algorithms like parameter estimation, it might be crucial for the procedure to reduce the order of the model's state space beforehand. This can be done by making use of the algorithms and the framework concluded in this thesis and implemented in the open-source program PyLESS.

The framework developed in [7], which is employed in this thesis, has one central issue: it needs to linearize the part of the model that is to be reduced. Making this approximation, one remarkably changes the reaction velocities of a system. While this approximation is good near the steady state, it gets worse the further the system is drawn away from it. This issue has also apppeared during simulation runs under different starting conditions. The bigger the *distance* between starting state and steady state is, the bigger gets the error of the linearization. Because of that potential users of this model reduction framework should keep in mind that the reduced order model is only a good approximation for states *near* the steady state.

Interesting about the framework is also the export to SBML. It has been shown that both alternatives either suffered from losing accuracy or efficiency in simulation. All simulations have been performed by the SBML-odeSolver [59], which is nowadays used in many modeling tools such as the popular CellDesigner [60]. That is why this solver has been chosen for comparing the efficiency of the reduced order model in simulation. Though this solver is optimized for structuring of differential equations in *reactions*, and therefore is slower on SBML files stated in terms of *rules*, the simulation time for some models could be reduced by dimension reduction (when using the SBML output optimized for efficiency).

One drawback of the proposed method is that it is only applicable to

stable metabolic network models. Most of the models in the Biomodels or the JWS database are not stable and therefore the method cannot be directly applied to them. Furthermore PyLESS does not support the complete SBML language. For example *events* in the input mode are simply ignored. Though this might make this method inapplicable to further models, it has not been implemented, because full support of SBML is very complex. Currently only *species* and *reactions* (which are the most common elements) are taken care of.

5.2 Model Reduction Algorithms

During the comparison of simulation results of reduced artificial models, all model reduction algorithms had examples on which they performed well. Though, during the simulation of real biochemical network models, SPA and BT yielded considerably better results. In most cases SPA performed a little better than BT on the chosen examples. This is why SPA is in general preferred by the author. In his opinion, methods based on SVD are in general more appropriate for the reduction of metabolic network models than Krylov space methods.

The failure of MPVL to yield a reduced order model that approximates the concentration time courses is due to its structure. The algorithm tries to construct a model, which approximates the response of the original system under a given input function of a certain frequency. In metabolic network models, the frequencies of the rate of change of either substance concentrations or reaction velocities can in general not be assumed to be constant. While for small, artificial reaction networks a frequency could be found that leads to good approximation of the original model by the model reduced by MPVL, for a bigger network such a frequency cannot be found. This can be explained by the simple fact that the reactions in a big network usually have very different reaction velocities and therefore behave very differently. On the other hand, the implementation of the MPVL algorithm is not the best known in literature. Various improvements of this algorithm have been supposed in order to overcome certain problems like bad starting vectors (look-ahead or implicit restarting [61, 40]). Due to the limited time available for this thesis not all of them could be implemented. This might also contribute to the bad approximations produced by the MPVL algorithm in this context.

Since the Krylov space based methods approximating only a single frequency cannot be employed on metabolic network models it might be interesting to observe the accuracy of models constructed by algorithms like AORGA. But also these multi-point Padé methods suffer from the problem of finding good approximation frequencies, which is nowadays in most cases solved heuristically. So the employment of multi-point methods would def-

initely require the development of methods to derive good approximation frequencies for a given model. Another problem of multi-point approximations is that they are computationally harder to derive. While the costs of single-point approximations scale quadratically in the order of the original model's state space, the multi-point approximations scale cubically like the SVD based methods. So these methods do not have the advantage that they are more easy to compute.

The difference in the approximations made by SPA and BT is less remarkable than the difference to MPVL. Nevertheless, in most cases SPA produced better results. The reason for that lays in the idea behind SPA because it tries to approximate the behaviour of the original model at the zero input-frequency (steady state). For linearized metabolic network models this seems to fit the dynamic behaviour better than an approximation made at ∞ frequency. One reason for this might be that regardless of how far away the starting point is from the steady state, after a short period of relatively fast movement, the system converges only slowly towards the steady state. Hence, most of the time under which the system is observed, it moves relatively slow. This supports the idea of approximating the system for small frequencies. As already pointed out, due to the linearization, the model should be observed near the steady state. The further the system is drawn away from this steady state, the worse gets the linearization approximation. So the approximation of the linearized system under high frequencies is useless anyways because such frequencies can only be observed far away from the steady state, a state in which the linearization approximation is already bad. But despite of the details, the approximations made by SPA and BT are still quite similar. In fact the difference between them might in most cases be neglectable compared to the error introduced by the linearization.

The SVD based methods do also contain a technique to choose the order of the reduced system automatically, given a maximum relative error. This has not been made use of, since an appropriate relative error should depend on the already present linearization error. If the linearization error would be relatively small, the reduction largely contributes to the final error. So a higher dimension of the reduced model should be chosen. In case the error introduced by the linearization is already large, the model could be reduced to a much smaller dimension. The selection of the order of the reduced model should always depend on the linearization error and therefore depend on the *degree of linearity* of a given input model. Since no further suggestion for the desired order can be given, it is up to the user to select it carefully.

One drawback of the algorithms used is that none of them is able to preserve the passivity of a system during reduction. Passivity in electrical terms means that the energy which comes out of the system, when observed for a longer time, has to be lower than the input energy. The definition of passivity, e.g. in [62], cannot so easily be extended to reduced metabolic

network models. But it should intuitively be clear that a violation of **any** property of the original system, even if its meaning in our case cannot be outlined, could lead to a problematic behaviour of the reduced order model. An Krylov space based method which preserves this property is PRIMA (Passive Reduced-order Interconnect Macromodeling Algorithm) [63]. But in general Krylov based methods have the problem that they trade accuracy or computational complexity for the preservation of a certain property.

The number of algorithms compared in this thesis is very limited. However the application of further algorithms in this context might be interesting. For example the already mentioned PRIMA method could be applied. As other Arnoldi-based algorithms, which make only use of one Krylov space as opposite to the Lanczos method, it is usually less accurate because only n_r moments are matched instead of $2n$. On the other hand, these type of algorithms is usually numerically more stable and is also able to preserve stability [64]. One further interesting type of algorithms is the multi-point approximation type (e.g. AORGA). Algorithms of this type have the advantage that they are able to approximate a model over a broader frequency range and that they have a global error bound. From the SVD type algorithms, Optimal Hankel Norm Approximation [65] could be applied to metabolic network reduction, which usually yields very good approximations but trades the stability preservation for that. The application of an SVD type algorithm which preserves the passivity of a system like Positive Real Balancing [21] could also be investigated, since the other main important features (stability preservation, existance of a global error bound) are already included in most SVD type methods.

Since the *absolute error* methods employed in this thesis tend to approximate the behaviour of the high concentrated substances better, *weighted error* methods should also be taken into account. With such methods the relative error on all substances, independent of their concentration, could be reduced. One example of such a method is frequency-weighted balanced truncation [22].

5.3 Conclusion

Altogether, model reduction has proven to be a good method for reducing the costs of the simulation of metabolic network models. As shown, the computational costs of simulating these models are drastically reduced when using the second method for the SBML export. Although the SVD methods are relatively time consuming, their computational costs are minimal compared to the costs of estimating parameters in an unreduced model. Hence, this method makes the use of non-fixed environments in model construction highly practicable and therefore may, together with the growing databases for biochemical network models, revolutionize the process of model construc-

tion.

Appendix A

Implementation / used Software

PyLESS heavily makes use of other open-source tools.

- General computation: Python [12], SciPy [66] (Numeric, NumArray, Numpy), GSL [67], PyGSL [68]
- SBML reading and writing: libSBML [69]
- Conservation relations finding: PySCeS [70]
- Steady state computation: lsode (odepack) [71] (integrator), hybrd [72] (solver)
- Elasticity computation: DerivVar (ScientificPython) [73]
- Model reduction: Python implementations of different algorithms from different papers or from a MatLab script from Peter Benner, which has been published together with an article
- Model comparison: SOSlib [59] using Sundials CVODE [74]
- Visualisation: GnuPlot [75], gnuplot.py [76] , GraphViz [77] , SBMLmerge [6]

Appendix B

Topologies of Models from the Results Section

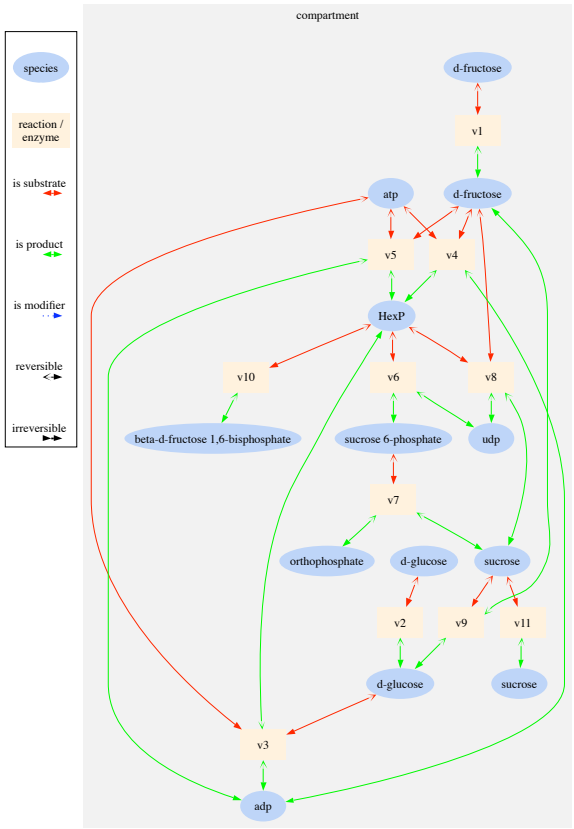

Figure B.1: Topology of the Rohwer model. Various substances are chosen as the inner model. This figure and the following topologies have been drawn with SBMLmerge.

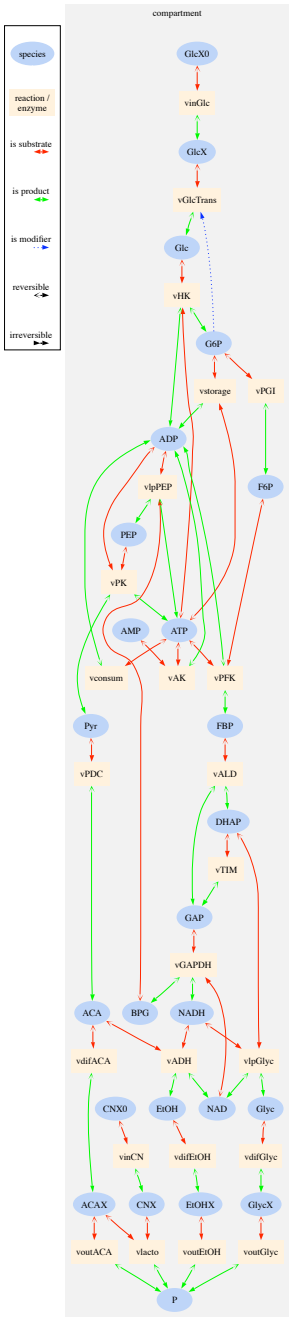

Figure B.2: Topology of the Hynne model. Glc and GlcX and the reaction connecting them are chosen as the inner model.

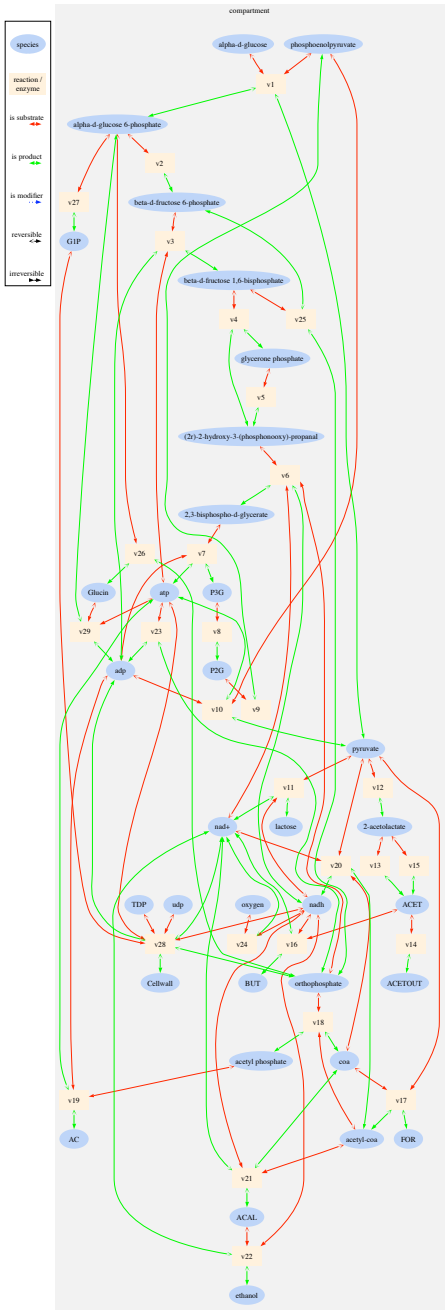

Figure B.3: Topology of the Hoefnagel model. The glycolysis is chosen as the inner model.

Appendix C

PyLESS code

The complete code of the program including a documentation can be downloaded from `http://sysbio.molgen.mpg.de/pyless`.

Balanced truncation

Taken from [24].

```
def duallyap(self,A,B,C):
    """Solve the stable dual Lyapunov equations.

    Output: [S,R] (full rank factors of the solutions)."""
    # initialize
    n = len(A)
    iter = 0
    tol = 10 * scipy.sqrt( n * self.threshold )
    maxstep = 50
    S = copy.deepcopy(B.getT())
    R = copy.deepcopy( C )
    E = scipy.eye(n)
    Ac = copy.deepcopy(A)
    Err = scipy.linalg.norm(Ac+E)
    onemore = 0
    convergence = (Err <= tol)
    # repeat until convergence
    while iter<maxstep and ((not convergence)
        or (convergence and (onemore < 3))):
        # invert A
        [L,U,P] = lup(copy.deepcopy(Ac))
        Y = L.getI() * P
        Y = U.getI() * Y
        # scaling of A for a faster convergence
```

```
if Err > 0.1:
    # frobenius norm scaling
    d = scipy.sqrt( scipy.linalg.norm(Ac,ord="fro")
        / scipy.linalg.norm(Y,ord="fro") )
else:
    d = 1
# Newton iteration step on A
Ac = (Ac/d + d*Y)/scipy.double(2)

# Iteration on the controllability gramian
S = self.__super_matrix__([[scipy.asmatrix(S)],
    [scipy.asmatrix(d*S*(Y.getT()))]])/scipy.sqrt(2*d)
U,T,P = qrpt(copy.deepcopy(S))
U = scipy.matrix(U)
T = scipy.matrix(T)
P = scipy.matrix(P)
p = [line.index(1) for line in P.tolist()]
# computing the rank of T
[svd_u, svd_s, svd_v] = scipy.linalg.svd(T)
r = [scipy.absolute(x) > 1e-13 for x in
    scipy.asmatrix(svd_s).flatten().tolist()[0]].count(True)
S = T[:r,p]

# Iteration on the observability gramian
R = self.__super_matrix__([[scipy.asmatrix(R)],
    [scipy.asmatrix(d*R*Y)]])/scipy.sqrt(2*d)
U,T,P = qrpt(copy.deepcopy(R))
U = scipy.matrix(U)
T = scipy.matrix(T)
P = scipy.matrix(P)
p = [line.index(1) for line in P.tolist()]
# compute rank of T
[svd_u, svd_s, svd_v] = scipy.linalg.svd(T)
r = [scipy.absolute(x) > 1e-13 for x in
    scipy.asmatrix(svd_s).flatten().tolist()[0]].count(True)
R = T[:r,p]

# Error computation
Err = scipy.linalg.norm(Ac+E,ord="fro")
iter = iter + 1
convergence = Err <= tol
if convergence:
    onemore = onemore + 1
```

```
    # return the full rank factors
    S = S.getT() / scipy.sqrt( scipy.double(2) )
    R = R / scipy.sqrt( scipy.double(2) )
    return [S,R]

def balanced_truncation(self,A,B,C,D,desired_order,tolerance=0):
    """Balanced truncation of a linear, continuous-time system using the
    SR implementation based on spectral projection.

    Original Matlab script from Peter Benner.
    Input: A,B,C are matrices, desired_order is the desired order of the
    output system, if desired_order < 0, tolerance is absolute error
    tolerance for the output model.

    Output: Ar,Br,Cr, abserr ()"""
    n = len(A)
    # solve Lyapunov equation
    [S,R] = self.duallyap(A,B,C)

    # compute the error on truncation
    [U,hksv,V] = scipy.linalg.svd(S.getT()*R.getT())
    V = scipy.asmatrix(V).getT()
    s = scipy.minimum(len(S.getT()),len(R))
    if desired_order<=0:
        abserr = 0
        desired_order = s+1
        while abserr < (tolerance / scipy.double(2)) and desired_order > 1:
            desired_order = desired_order-1
            abserr = abserr + hksv[(desired_order-1)]
    if desired_order > len(hksv):
        desired_order = len(hksv)

    # truncate the matrices
    S1 = scipy.asmatrix( scipy.diag( scipy.sqrt( hksv[:desired_order] ) ) )
    U1 = scipy.asmatrix(U[:,:desired_order])
    V1 = scipy.asmatrix(V[:,:desired_order])

    # compute the transfer matrices
    if hksv[0] > 0:
        self.Tl = S1.getI() * V1.getT() * R
        self.Tr = S*U1 * S1.getI()
    else:
        # the part of the model has no influence on the rest!
        self.Tl = scipy.matrix([]).reshape(0,n)
```

```
        self.Tr = scipy.matrix([]).reshape(n,0)

    # compute the reduced order model
    self.Ar = self.Tl * A * self.Tr
    self.Br = self.Tl * B
    self.Cr = C * self.Tr
    self.Dr = copy.deepcopy( D )
```

MPVL algorithm with deflation but without lookahead

Taken from [11].

```
def deorthogonalize(R,L):
    """Resorts R and L so that the first min(len(R),len(L)) vectors are
    not orthogonal."""
    # check the colvectors to be orthogonal
    threshold = scipy.MachAr().eps
    Rt = copy.deepcopy(R.getT())
    Lt = copy.deepcopy(L.getT())
    r_length = len(Rt)
    l_length = len(Lt)
    r_return = []
    l_return = []
    # remove zero vectors
    for i in range(r_length-1,-1,-1):
        if Rt[i,:]*Rt[i,:].getT() < threshold:
            Rt = __super_matrix__([[Rt[:i,:]],[Rt[i+1:,:]]])
            r_length = r_length - 1
    for i in range(l_length-1,-1,-1):
        if Lt[i,:]*Lt[i,:].getT() < threshold:
            Lt = __super_matrix__([[Lt[:i,:]],[Lt[i+1:,:]]])
            l_length = l_length - 1
    if r_length == 0 or l_length == 0:
        print "Krylov space cannot be build due to empty input matrices."
        sys.exit()
    # determine which is the larger matrix
    if l_length < r_length:
        smaller = Lt
        smaller_return = l_return
        bigger = Rt
        bigger_return = r_return
    else:
```

```
        smaller = Rt
        smaller_return = r_return
        bigger = Lt
        bigger_return = l_return
    # the resorting begins
    s_remove_indices = []
    for s_index in range(len(smaller)):
        # check all smallers against the bigger ones
        for b_index in range(len(bigger)):
            if smaller[s_index,:] * bigger[b_index,:].getT() > threshold:
                # append the orthogonal vectors to the return matrices
                smaller_return.append( smaller[s_index,:].tolist()[0] )
                bigger_return.append( bigger[b_index,:].tolist()[0] )
                # remove them from the working matrices
                bigger = __super_matrix__([[bigger[:b_index,:]],
                    [bigger[(b_index+1):,:]]])
                s_remove_indices.append(s_index)
                # removing the vectors from the smaller matrix has to be
                # shoved to a different iteration
                break
    # remove also from the smaller matrix
    s_remove_indices.reverse()
    for s_index in s_remove_indices:
        smaller = __super_matrix__([[smaller[:s_index,:]],
            [smaller[(s_index+1):,:]]])
    # put the remaining vectors from the working matrices
    # into the return matrices
    for s_index in range(len(smaller)):
        if len(smaller[s_index,:].tolist()[0]) == 0: continue
        smaller_return.append( smaller[s_index,:].tolist()[0] )
    for b_index in range(len(bigger)):
        if len(bigger[b_index].tolist()[0]) == 0: continue
        bigger_return.append( bigger[b_index,:].tolist()[0] )
    # make them matrices again
    l_return = scipy.matrix(l_return).getT()
    r_return = scipy.matrix(r_return).getT()
    return r_return, l_return

def matrix_lanczos(A,n,R,L):
    """ MPVL algorithm.

    Input: A <- the matrix from which both
        Krylov spaces are contructed
        R,L <- matrices of column-vectors
```

```
        which induce the right and left
        block Krylov space
    n <- the desired order of the Krylov
        spaces
"""
threshold = scipy.MachAr().eps * len(A)
# for convenience
A = scipy.asmatrix(scipy.double(A))
R = scipy.asmatrix(scipy.double(R))
L = scipy.asmatrix(scipy.double(L))
# preperation against breakdown
R,L = deorthogonalize(R,L)
# initialize
t = {}
ttilde = {}
m = len(R.getT())
p = len(L.getT())
Delta = []
Vdl = {}
Wdl = {}
V = {}
W = {}
mu = -m
sigma = -p
Mu = []
Sigma = []
Iv = []
Iw = []
# iteration
for index in range(n):
    while True: # produce the right block Krylov space
        mu = mu + 1
        if mu == index+1: break
        if mu <= 0:
            # take from the starting vectors
            v = R[:,(mu+m-1)]
            iv = 0
        else:
            # generate new vectors
            v = A*V[mu-1]
            iv = scipy.maximum(0,Sigma[mu-1])
        # biorthogonalize
        I = range(iv,index) + [i for i in Iv if i<iv]
        for i in I:
```

```
        t[i][mu] = W[i].getT()*v / Delta[i]
        v = v - (V[i] * t[i][mu])
    # deflation
    if scipy.linalg.norm(v)<threshold:
        if mu > 0 and True in [scipy.absolute(x) > threshold
            for x in v.flatten().tolist()[0]]:
            Iw = Iw + [mu]
            Vdl[mu] = copy.deepcopy(v)
            Wdl[mu] = copy.deepcopy(w)
    else:
        break
while True: # same for the left Krylov space
    sigma = sigma + 1
    if sigma == index+1: break
    if sigma <= 0:
        w = L[:,(sigma+p-1)]
        iw = 0
    else:
        w = A.getT()*W[sigma-1]
        iw = scipy.maximum(0,Mu[sigma-1])
    I = range(iw,index) + [i for i in Iw if i<iw]
    for i in I:
        ttilde[i][sigma] = (V[i].getT()*w) / Delta[i]

        w = w - (W[i] * ttilde[i][sigma])
    if scipy.linalg.norm(w)<threshold:
        if sigma > 0 and True in [scipy.absolute(x) > threshold
            for x in w.flatten().tolist()[0]]:
            Iv = Iv + [sigma]
            Vdl[sigma] = copy.deepcopy(v)
            Wdl[sigma] = copy.deepcopy(w)
    else:
        break
# update t
if index not in t: t[index] = {}
t[index][mu] = scipy.linalg.norm(v)
if index not in ttilde: ttilde[index] = {}
ttilde[index][sigma] = scipy.linalg.norm(w)

# in case of a breakdown
if t[index][mu] == 0: t[index][mu] = 1
if ttilde[index][sigma] == 0: ttilde[index][sigma] = 1

# generate new vectors
```

```
    V[index] = (v/t[index][mu])
    W[index] = (w/ttilde[index][sigma])
    # add new coefficients
    Mu.append(mu)
    Sigma.append(sigma)
    Delta.append(W[index].getT()*V[index])
    if scipy.absolute(Delta[index]) < threshold:
        break
# build matrices V and W
keys = [x for x in V.keys() if x >= 0]
keys.sort()
V = scipy.asmatrix(scipy.double(__super_matrix__([[V[x] for x in keys]])))
keys = [x for x in W.keys() if x >= 0]
keys.sort()
W = scipy.asmatrix(scipy.double(__super_matrix__([[W[x] for x in keys]])))
return [V,W]
```

Bibliography

[1] Z. Zi and E. Klipp, "SBML-PET: A Systems Biology Markup Language based Parameter Estimation Tool," *Submitted*, 2006.

[2] M. Kanehisa, S. Goto, M. Hattori, K. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa, "From genomics to chemical genomics: new developments in KEGG," *Nucleic Acids Research*, vol. 34, pp. 354–357, 2006.

[3] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. Gopinath, G. Wu, L. Matthews, *et al.*, "Reactome: a knowledgebase of biological pathways," *Nucleic Acids Res*, vol. 33, pp. 151–2, 2005.

[4] N. Le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, *et al.*, "BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems," *Nucleic Acids Research*, 2006.

[5] B. Olivier and J. Snoep, "Web-based kinetic modelling using JWS Online," *Bioinformatics*, vol. 20, no. 13, pp. 2143–2144, 2004.

[6] M. Schulz, J. Uhlendorf, E. Klipp, and W. Liebermeister, "SBMLmerge, a system for combining biochemical network models," *Genome Informatics (in press)*, vol. 17, no. 1, 2006.

[7] W. Liebermeister, U. Baur, and E. Klipp, "Biochemical network models simplified by balanced truncation," *FEBS Journal*, vol. 272, no. 16, p. 4034, 2005.

[8] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *Automatic Control, IEEE Transactions on*, vol. 26, no. 1, pp. 17–32, 1981.

[9] Y. Liu and B. D. O. Anderson, "Singular Perturbation Approximation of Balanced Systems," *Int. J. Contr*, vol. 50, no. 4, p. 13791405, 1989.

[10] P. Feldmann and R. Freund, "Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," *Proceedings of the 32nd ACM/IEEE conference on Design automation*, pp. 474–479, 1995.

[11] J. Aliaga, D. Boley, R. Freund, and V. Hernandez, "A Lanczos-type method for multiple starting vectors," *Math. Comp*, vol. 69, no. 232, pp. 1577–1601, 2000.

[12] G. Van Rossum *et al.*, "Python." http://www.python.org.

[13] C. Wierling, E. Maschke-Dutz, R. Herwig, and H. Lehrach, "PyBioS - a tool for modeling and simulation of cellular systems." http://pybios.molgen.mpg.de/.

[14] H. W. Knobloch and H. Kwakernaak, *Lineare Kontrolltheorie*. Springer-Verlag, 1985.

[15] R. Heinrich and T. Rapoport, "A linear steady-state treatment of enzymatic chains. General properties, control and effector strengh," *Eur. J. Biochem.*, vol. 42, pp. 89–95, 1974.

[16] H. Kacser and J. Burns, "The control of flux," *Symp. Soc. Exp. Biol*, vol. 27, no. 65, p. 104, 1973.

[17] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 1996.

[18] C. Marahrens, "Balancierungstechniken: Balanciertes Abschneiden," 2004. www-user.tu-chemnitz.de/~benner/Lehre/Modellreduktion.html.

[19] G. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach*. Springer, 2005.

[20] H. Jin Kim, "Script: Systems norms (Topics in Aerospace Systems)," 2005. http://plaza.snu.ac.kr/~hjinkim/aerotopics_sp05/topic4-norm.pdf.

[21] U. Desai and D. Pal, "A transformation approach to stochastic model reduction," *Automatic Control, IEEE Transactions on*, vol. 29, no. 12, pp. 1097–1100, 1984.

[22] D. Enns, *Model reduction for control system design*. PhD thesis, Stanford University, 1984.

[23] A. Varga, "New Numerical Software for Model and Controller Reduction," *SLICOT Working Note SLWN2002*, vol. 5, 2002.

[24] P. Benner and E. Quintana-Ortı, "Model Reduction Based on Spectral Projection Methods," *Dimension Reduction of Large-Scale Systems*, vol. 45, pp. 5–45, 2005.

[25] I. Bronstein, K. Semendjajew, G. Musiol, and H. Muehlig, *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2001.

[26] H. Fassbender and P. Benner, "Numerische Methoden zur passivittserhaltenden Modellreduktion," 2005.

[27] PlanetMath, "Grammian." http://planetmath.org/encyclopedia/Grammian.html.

[28] A. Megretski, "Model reduction script: Balanced truncation," 2004. http://web.mit.edu/6.242/www/images/lec5_6242_2004.pdf.

[29] M. Fellner, R. Hoeldrich, and W. Werth, "Modellierung von HRTF - Kurven," *IEM-Report 07/98*, 1998.

[30] A. Laub, M. Heath, C. Paige, and R. Ward, "Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms," *Automatic Control, IEEE Transactions on*, vol. 32, no. 2, pp. 115–122, 1987.

[31] M. Tombs and I. Postlethwaite, "Truncated balanced realization of a stable non-minimal state-space system," *International Journal of Control*, vol. 46, no. 4, pp. 1319–1330, 1987.

[32] P. Benner, E. Quintana-Ortı, and G. Quintana-Ortı, "Balanced truncation model reduction of large-scale dense systems on parallel computers," *Math. Comput. Model. Dyn. Syst.*, vol. 6, no. 4, pp. 383–405, 2000.

[33] L. Pernebo and L. M. Silverman, "Model reduction vie balanced state space representation," *IEEE Transactions on Automatic Control*, vol. 27, pp. 382–382, 1982.

[34] D. Enns, "Model reduction with balanced realizations: An error bound and a frequency weighted generalization," *Proceedings of the 23rd IEEE Conference on Decision and Control*, 1984.

[35] J. D. Roberts, "Linear model reduction and solution of the algebraic Riccati equation by use of the sign function (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department (1971))," *Internat. J. Control*, vol. 32, pp. 677–687, 1980.

[36] U. Baur, "Low Rank Solution of Data-Sparse Sylvester Equations," 2005.

[37] N. J. Higham, "Computing the polar decomposition – with applications," *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 1160–1174, 1986.

[38] P. Benner, E. Quintana-Orti, and G. Quintana-Orti, "Singular perturbation approximation of large, dense linear systems," *Computer-Aided Control System Design, 2000. CACSD 2000. IEEE International Symposium on*, pp. 255–260, 2000.

[39] Y. Liu and D. Anderson, "Controller reduction via stable factorization and balancing.," *International Journal of Control*, vol. 44, no. 2, pp. 507–531, 1986.

[40] K. Gallivan, E. Grimme, D. Sorensen, and P. Van Dooren, "On some modifications of the Lanczos algorithm and the relation with Padé approximations,"

[41] L. T. Pillage and R. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 352–366, 1990.

[42] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied Numerical Mathematics*, vol. 43, pp. 9–44, 2002.

[43] W. B. Gragg and A. Lindquist, "On the partial realization problem," *Linear Algebra and Appl.*, vol. 50, pp. 277–319, 1983.

[44] C. D. Villemagne and R. E. Skelton, "Model reduction using a projection formulation," *Internat. J. Control*, vol. 46, pp. 2141–2169, 1987.

[45] Z. Bai, R. D. Slone, W. T. Smith, and C. Ye, "Error bounds for reduced system model by Padé approximation via the Lanczos process," *IEEE Trans. Comput. Aided Design*, vol. 18, pp. 133–141, 1999.

[46] C. Lanczos, "An iteration method for the solution of the eigenvalue problem problem of linear differential and integral operators," *J. Res. Nat. Bur. Standards*, vol. 45, pp. 255–282, 1950.

[47] D. R. Taylor, *Analysis of the look ahead Lanczos algorithm.* PhD thesis, Department of Mathematics, University of California, Berkeley, 1982.

[48] A. Ruhe, "Rational Krylov algorithms for nonsymmetric eigenvalue problems, II: Matrix pairs," *Linear Algebra and Appl.*, vol. 197, pp. 283–295, 1994.

[49] K. Gallivan, G. Grimme, and P. Van Dooren, "A rational Lanczos algorithm for model reduction," *Numerical Algorithms*, vol. 12, no. 1, pp. 33–63, 1996.

[50] C. C. Chu, M. H. Lai, and W. S. Feng, "MIMO interconnects order reductions by using the multiple point adaptive-order rational global

Arnoldi algorithm," *IEICE trans. elec.*, vol. E89-C, no. 6, pp. 792–802, 2006.

[51] T. Stykel and U. Baur, "Personal communication with," 2006.

[52] D. Kressner, "Block Algorithms for Reordering Standard and Generalized Schur Forms," *LAPACK Working Note 171*, 2006.

[53] A. Varga, "Enhanced modal approach for model reduction," *Mathematical Modelling of Systems*, vol. 1, pp. 91–105, 1995.

[54] S. Barrachina, P. Benner, E. Quintana-Orti, and G. Quintana-Orti, "Parallel Algorithms for Balanced Truncation of Large-Scale Unstable Systems," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 2248–2253, 2005.

[55] A. Varga, "Coprime factors model reduction based on square-root balancing-free techniques," *Computational System Analysis*, pp. 91–96, 1992.

[56] J. Rohwer and F. Botha, "Analysis of sucrose accumulation in the sugar cane culm on the basis of in vitro kinetic data," *Biochem. J*, vol. 358, pp. 437–445, 2001.

[57] F. Hynne, S. Dano, *et al.*, "Full-scale model of glycolysis in Saccharomyces cerevisiae," *Biophysical Chemistry*, vol. 94, no. 1, pp. 121–163, 2001.

[58] M. Hoefnagel, A. van der Burgt, D. Martens, J. Hugenholtz, and J. Snoep, "Time Dependent Responses of Glycolytic Intermediates in a Detailed Glycolytic Model of Lactococcus Lactis During Glucose Run-Out Experiments," *Molecular Biology Reports*, vol. 29, no. 1, pp. 157–161, 2002.

[59] R. Machne, A. Finney, S. Muller, J. Lu, S. Widder, and C. Flamm, "The SBML ODE Solver Library: a native API for symbolic and fast numerical analysis of reaction networks," *Bioinformatics*, vol. 22, no. 11, pp. 1406–1407, 2006.

[60] A. Funahashi and H. Kitano, "CellDesigner: a process diagram editor for gene-regulatory and biochemical networks," *BioSilico*, vol. 1, pp. 159–162, 2003.

[61] D. C. Sorensen, "Implicit application of polynomial filters in a K-step Arnoldi method," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 357–385, 1992.

[62] A. Megretski, "Model reduction script: Model reduction by projection: general properties," 2004. http://web.mit.edu/6.242/www/images/lec4_6242_2004.pdf.

[63] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm," *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pp. 58–65, 1997.

[64] E. Rudnyi, "Automatic Compact Modelling for MEMS: Applications, Methods and Tools," 2006. http://modelreduction.com/Teaching.html.

[65] K. Glover, "All optimal Hankel-norm approximations of linear multivariable systems and their L_∞-error bounds," *International Journal of Control*, vol. 39, no. 6, pp. 1115–1193, 1984.

[66] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. http://www.scipy.org.

[67] G. Galassi *et al.*, "GNU Scientific Library Reference Manual." http://www.gnu.org/software/gsl.

[68] A. Gaedke, J. Kuepper, S. Maret, and P. Schnizer, "PyGSL Reference Manual." http://pygsl.sourceforge.net.

[69] B. Bornstein, S. Keating, M. Hucka, A. Finney, B. Shapiro, A. Funahashi, and R. Gauges, "LibSBML." http://sbml.org/software/libsbml/.

[70] B. Olivier, J. Rohwer, and J. Hofmeyr, "Modelling cellular systems with PySCeS," *Bioinformatics*, vol. 21, no. 4, pp. 560–561, 2005.

[71] A. Hindmarsh *et al.*, "ODEPACK, a systematized collection of ODE solvers," *Scientific Computing*, vol. 1, pp. 55–64, 1983.

[72] B. Garbow, K. Hillstrom, and J. More, "Program hybrd, 1980," *Part of the Argonne National Laboratory MINPACK project, publicly available via the netlib server.*

[73] K. Hinsen and R. Sadron, "ScientificPython Users Guide," *Centre National de la Recherche Scientifique dOrléans, Orléans, France*, 2002.

[74] S. Cohen and A. Hindmarsh, "CVODE, a stiff/nonstiff ODE solver in C," *Computers in Physics*, vol. 10, no. 2, pp. 138–143, 1996.

[75] T. Williams and C. Kelley, "gnuplot: An Interactive Plotting Program," *Manual, version*, vol. 3, 1998.

[76] M. Haggerty, T. Ingraldi, and K. Hinsen, "GnuPlot.py." http://gnuplot-py.sourceforge.net/.

[77] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull, "Graphviz-open source graph drawing tools," *Graph Drawing*, pp. 483–485, 2001.

Acknowledgments

The author would like to thank his advisors Wolfram Liebermeister and Ralf Herwig for having supported this work with great ideas and comments of infinite number on the code and the thesis. Additional thanks goes to Sabine Pilari (also for help with some figures), Jannis Uhlendorf, Jörg Decker, and Ann-Christin Schulz for carefully proof-reading the script and to the Kinetic Modelling and the Bioinformatics group (especially to Elisabeth Maschke-Dutz and Christoph Wierling) at the Max Planck Institute for Molecular Genetics. Also the author would like to thank Peter Benner for allowing him to include his code for balanced truncation by spectral projection into PyLESS, which has brought a significant performance gain, and Ulrike Baur and Tatjana Stykel for insightful discussions.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt bernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet.

Weitere Personen mit Ausnahme der Betreuer und der zitierten Personen waren an der inhaltlich-materiellen Herstellung der vorliegenden Arbeit nicht beteiligt.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.