TAMPERE UNIVERSITY OF TECHNOLOGY

Mikko Lehtimäki

Dimensionality reduction for mathematical models in neuroscience

Master of Science Thesis

Examiners: Group Leader, Adjunct Professor Marja-Leena Linne, Emeritus Professor Seppo Pohjolainen, Assistant Professor Lassi Paunonen
Examiners and topic approved in the Faculty of Natural Sciences council meeting on 4 May 2016

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Biotekniikan koulutusohjelma
Lehtimäki, Mikko: Matemaattisten mallien dimension redusointi neurotieteessä
Diplomityö, 67 sivua, 23 liitesivua
Elokuu 2016
Pääaine: Informaatioteknologia terveys- ja biotieteissä
Tarkastajat: Marja-Leena Linne, Seppo Pohjolainen, Lassi Paunonen
Avainsanat: Redusointi, dimension pienentäminen, laskennallinen mallintaminen, neurotiede, hermosolu

Systeemin dimension redusointi eli koon pienentäminen on menetelmä, jota käytetään esimerkiksi säätöteorian alalla parantamaan epälineaaristen matemaattisten mallien laskentatehokkuutta. Lisäksi redusointimenetelmien avulla saadaan esiin mallinnettavan systeemin dynamiikkaan merkittävimmin vaikuttavia tekijöitä. Laskennallisessa neurotieteessä aivojen toimintaa kuvaaviin laajoihin matemaattisiin malleihin täytyy sisällyttää myös solu- ja molekyylitason ilmiöiden kuvauksia, jotta oppimista ja muita aivojen monimutkaisia ilmiöitä voitaisiin ymmärtää. Tämä ei ole mahdollista nykyisillä menetelmillä, sillä malleista tulisi laskennallisesti liian raskaita pientenkin soluverkostojen toiminnan simuloimiseksi.

Työn teoriaosuudessa esitellään ja tutkitaan matemaattisten mallien redusointimenetelmien käyttöä neuro- ja biotieteissä. Biotieteissä mallien yksinkertaistamista lähestytään tyypillisesti eliminoimalla muuttujia ja yhtälöitä mallinnettavasta systeemistä erilaisten oletusten perusteella. Tämä lähestymistapa ei ole aina suositeltava, sillä se kadottaa mallista informaatiota. Matemaattisilla redusointimenetelmillä ei ole vastaavaa ongelmaa, sillä ne approksimoivat kaikkia redusoidun systeemin tekijöitä ja mallin osia hyödyntämällä pieniulotteisia aliavaruuksia.

Tässä tutkimuksessa redusoidaan aivojen muovautuvuudessa ja siten oppimisessa tärkeän solunsisäisen signalointiverkoston tutkimusdatalla verifioitu matemaattinen malli. Malli on yksi laajimmista molekyylitason malleista, joka pystyy kuvaamaan muovautuvuutta biokemiallisten reaktioiden ja massavaikutuslain mukaisesti. Redusointi tehtiin yhdistämällä Proper Orthogonal Decomposition ja Discrete Empirical Interpolation Method (POD+DEIM) -redusointimenetelmät. Diplomityön tulokset osoittavat, että mallia on mahdollista simuloida huomattavasti lyhyemmässä ajassa pienillä virhemarginaaleilla verrattuna alkuperäisen mallin simulointituloksiin. Vaadittava simulaatiotarkkuus vaihtelee mallin sovelluskohteen mukaan ja tarkkuuden huomattiin riippuvan simulointiajasta. Työn tuloksena suositellaan mallien redusointia matemaattisin menetelmin laskennallisessa neurotieteessä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master of Science Degree Programme in Bioengineering
Lehtimäki, Mikko: Dimensionality reduction for mathematical models in neuroscience
Master of Science Thesis, 67 pages, 23 Appendix pages
August 2016
Major: Information Technology for Health and Biology
Examiners: Marja-Leena Linne, Seppo Pohjolainen, Lassi Paunonen
Keywords: dimensionality reduction, order reduction, computational modeling, neuroscience, neuron

Dimensionality reduction is a commonly used method in engineering sciences, such as control theory, for improving computational efficiency of simulations of complex nonlinear mathematical models. Additionally, it is a way of surfacing the most important factors that drive the dynamics of the system. In the field of neuroscience, there is a great demand to incorporate molecular and cellular level detail in large-scale models of the brain in order to produce phenomena such as learning and behavior. This cannot be achieved with the computing power available today, since the detailed models are unsuitable for large-scale network or system level simulations.

In this thesis, methods for mathematical model reduction are reviewed. In the field of systems biology, models are typically simplified by completely eliminating variables, such as molecules, from the system, and making assumptions of the system behavior, for example regarding the steady state of the chemical reactions. However, this approach is not meaningful in neuroscience since comprehensive models are needed in order to increase understanding of the target systems. This information loss problem is solved by mathematical reduction methods that strive to approximate the entire system with a smaller number of dimensions compared to the original system.

In this study, mathematical model reduction is applied in the context of an experimentally verified signaling pathway model of plasticity. The chosen biophysical model is one of the most comprehensive models out of those that are currently able to explain aspects of plasticity on the molecular level with chemical interactions and the law of mass action. The employed reduction method is Proper Orthogonal Decomposition with Discrete Empirical Interpolation Method (POD+DEIM), a subspace projection method for reducing the dimensionality of nonlinear systems. By applying these methods, the simulation time of the plasticity model was radically shortened although approximation errors are present if the model is reviewed on large time scales. It is up to the final application of the model whether some error or none at all is tolerated. Based on these promising results, subspace projection methods are recommended for dimensionality reduction in computational neuroscience.

PREFACE

The work has been carried out in the Computational Neuroscience Research Group at the Department of Signal Processing, in collaboration with the Systems Theory Group of the Department of Mathematics. The research leading to these results has received partial funding from the European Union Seventh Framework Programme (FP7) under grant agreement no 604102 Human Brain Project (HBP).

I am thankful to my supervisor Marja-Leena for introducing me to the fascinating world of neuroscience, making this work possible and guiding me through the process of writing this thesis. I wish to express my gratitude to my supervisors Seppo Pohjolainen and Lassi Paunonen for their valuable advice and effort for teaching me mathematics and making me a better engineer. I also thank Riikka Havela and Tiina Manninen for wisdom and proofreading, Professor Danny Sorensen, Professor Karen Willcox and Doctor Benjamin Peherstorfer for answering my questions regarding the DEIM algorithm and Professor Elena Kutumova for answering questions about model simplification in systems biology.

My sincerest thanks goes to all my friends for making the past years forever memorable. A special shoutout is in order for extremely insightful and stimulating lunch sessions during the past year. Finally I wish to thank my family and Anni for their endless support.

Mikko Lehtimäki
25.7.2016

CONTENTS

TERMS AND SYMBOLS

LTP	Long term potentiation
LTD	Long term depression
HH	Hodgkin-Huxley model
MIMO	Multi-Input Multi-Output
LTV	Linear time-varying
LTV	Linear time-invariant
ODE	Ordinary Differential Equation
SISO	Single-Input Single-Output
QSSA	Quasi-Steady-State-Approximation
MiMe	Michaelis-Menten
SBML	Systems Biology Markup Language
BT	Balanced Truncation
SVD	Singular Value Decomposition
MM	Moment Matching
PVL	Padé via Lanczos
POD	Proper Orthogonal Decomposition
DEIM	Discrete Empirical Interpolation Method
PCA	Principal Component Analysis
KLT	Karhunen-Loeve Transformation
2Ag	2-arachidonoylglycerol
Ca	Calcium
DAG	Diacylglycerol
Gabg	G protein with α, $beta$ and γ subunits
PLC	Phospholipase C

Glu	Glutamate
IP_3	Inositol trisphosphate
PIP	Phosphatidylinositol
DAGK	Diacylglycerol kinase complex
A	Matrix (capital), state matrix
B	Input matrix
C	Output matrix
D	Feedthrough matrix
I_n	$n \times n$ identity matrix
A^*	Conjugate transpose of A
A^T	Transpose of A
a	Scalar
a_{ij}	Element of matrix A at row i and column j
$\boldsymbol{x}$	Vector (bold), state vector
$\boldsymbol{x}(t)$	Time dependent vector
$\boldsymbol{u}(t)$	Input vector
$\dot{x}$	Differentiation (Newton's dot notation)
$f()$	Function
$[S]$	Concentration of S
W_c	Controllability grammian matrix
W_o	Observability grammian matrix
Σ	Matrix of singular values
$H(s)$	Transfer function

1. INTRODUCTION

Mathematical modeling of dynamical systems is a way to enhance understanding of the driving processes of the systems. In biosciences, models provide a safe and low-cost method of designing laboratory experiments, planning hardware architecture and predicting drug effectiveness, among others. With the availability of experimental data in increasing quality and quantity, models become all the more sophisticated. Mathematical theory provides an analytical method for studying model behavior. However, analytical solutions are limited to elementary, conceptual models. Computational neuroscience utilizes numerical methods, such as simulations, for predicting and validating parameter spaces of large scale models. These two approaches, theoretical and computational, benefit from each other by providing information any one method alone cannot uncover [1].

The purpose of neuroscience is to provide a realistic model of the human brain, which consists of millions of neurons, their connections and other cells, such as glial cells that have recently received considerable attention from computational neuroscientists [2]. In neuroscience, mathematical models are constructed to describe different phenomena from the molecular level up to networks of cells and brain regions, with the ultimate goal to explain learning, behavior and diseases in which plasticity has a central role. Plasticity means adaptation to environment. Moreover, learning can be described by plasticity, which generally in the context of the brain is a phenomenon where the function of the synapse, a connective gap between neurons, is modified depending on activity in the synapse. In other words, the brain is plastic because it is constantly evolving and the structure is changing. In a recent model developed in [3] plasticity is explained with long term potentiation (LTP) and long term depression (LTD) that represent strengthening or weakening of the synapse in information transfer between neurons [3, 4].

Typical models in neuroscience include different types of neurons and other cells, models of signalling pathways and associated molecular dynamics, models of networks of cells and cognitive models. Modeling single neuron dynamics is commonly approached by the Hodgkin-Huxley (HH) model [5] while signalling pathways are traditionally modeled using laws of biochemistry, including the law of mass action and Michaelis-Menten kinetics [3]. Furthermore, tissue level phenomena are modeled using neural mass theory. Additionally, many modeling studies combine different approaches and aim to describe the neural system using a multiscale approach, where a certain behavior is explained *bottom up* starting from molecular kinetics and chemical equations (further information can

be obtained e.g. from [6, 7]).

Dynamical systems found in nature tend to have nonlinear characteristics so that their mathematical descriptions become rather complex, making them difficult for humans to comprehend. In other words, this means that straightforward linear system models (consider for example lines and their equations), with established mathematical properties, are insufficient for capturing advanced kinetics. Additionally, the lack of linearity results in difficulty in studying these models analytically, making numerical simulations especially insightful in understanding their behavior. Moreover, nonlinear systems require more computing power to solve and simulate compared to linear ones. For instance, chemical stoichiometric reactions that are the basis of all natural phenomena, when expressed as differential equations, result in nonlinear equations.

The more detail and explanative capability the model has, the bigger the computational burden in the simulation and analysis phase. The current state of the art supercomputers are able to simulate brain activity in one brain region, leaving out molecular mechanics on cellular level, with animal models that are less complex than human brain models, and the computations consume weeks of time [8]. Accordingly, in practice constructing a neural model is always a compromise between fine detail and resolution.

In the recent years the goal of achieving larger and more realistic models has been brought forward mainly by advances in computing hardware and parallel calculations, for example with supercomputers. However, the increase of processing power cannot compete with the rate modern methods collect data, resulting in models and databases of tremendous size beyond processing capability. Although more powerful computational machinery is required to further the field, it is not the most scalable and accessible approach. For this end model reduction serves an important purpose. By providing simpler yet accurate descriptions of cells and networks, more advanced models can be built, simulated and analyzed with existing hardware and more detail can be included in the models.

Model reduction methods have arisen from the need to diminish the complexity of heavy models found for example in circuit simulation and structural and flow dynamics. The mathematical nature of these fields of schience has laid the groundwork for developing good quality reduced order models. In addition to increasing resolution and simulation speed for models, reduction methods provide both a basis and a way to analyze large scale systems more effectively. Examples include equilibrium point analysis ($x(t) = 0$), frequency response analysis, transient analysis to compute the relation of output to stimulus and sensitivity analysis to determine reactions to changes in system parameters [9].

Model reduction is not a new concept in neuroscience. The HH model was first simplified from four to only two equations by FitzHugh and Nagumo [10, 11] and later to three equations by Hindmarsh and Rose [12]. Recently, Diekman introduced a reduced model of astrocyte metabolism [13]. The common factor in these simplifications has been the careful elimination and subsequent modification of variables in the models, and in

systems biology this type of simplification by pruning is common practice. Algorithms to achieve the same especially for nonlinear neuronal models have been proposed for instance by Kepler [14], Woo [15], DeWeerth and Sorensen [16], and later by Shin [17]. These methods are effective in certain specific scenarios, but the perfect model reduction method would be generally applicably to different types of systems while providing an approximation of all the variables and preserving the original inputs and outputs of the system.

In this thesis, general tools for model reduction that are mathematically rigorous and provide estimates of all system variables will be introduced and their applicability to models of neuroscience examined. An additional goal is to present the theory behind the methods so that is approachable with little mathematical background. Furthermore, model reduction will be performed for a nonlinear chemical equation based data driven model published in [3], that describes learning in the brain via plasticity and calcium currents. The original model is too detailed for utilization in network simulations, greatly reducing the usability of the model but also serving as motivation for the present study. In addition to nonlinearity, the model includes time-dependent terms, which pose an additional challenge both computational efficiency and reduction wise. With this case study the aim is to demonstrate that the behavior of the model can be analysed faster yet with satisfactory accuracy by using a reduced order model.

Chapter 2 will briefly introduce necessary mathematical concepts for understanding model order reduction. The concepts are followed by model order reduction algorithms and insights into applicability of the methods for different types of systems. In chapter 3 a model for which dimensionality reduction was applied is explained in detail along with the implementation and reduction method used in this thesis. Chapter 4 presents simulation results of the reduced order model. Finally in chapter 5 the model order reduction methods as well as the results obtained in this thesis are discussed further and conclusions are given in chapter 6.

2. MODEL REDUCTION THEORY AND ALGORITHMS

In this chapter a literature survey of the current model reduction methods and their applicability to nonlinear models is given.

Model order reduction, model rank reduction or model reduction has been an active research topic in the field of control theory. There, it has been motivated by large scale models rising in circuit simulation and structural dynamics among other computationally demanding models. Using reduced order models, control theory studies the behavior of the system and how it can be directed by modification or excitation. In control theory, a discrete multi-input multi-output (MIMO) linear time-variant (LTV) first order differential equation system is commonly written as

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= C(t)\boldsymbol{x}(t) + D(t)\boldsymbol{u}(t), \end{aligned} \tag{1}$$

which is also called the state-space representation. In Equation (1) the change of the state of the system $\dot{\boldsymbol{x}}$ is modeled as a linear combination of the current state of the system $A(t)\boldsymbol{x}(t)$ and inputs to the system $B(t)\boldsymbol{u}(t)$ by the *state equation*, where $A(t)$ is a state matrix and $B(t)$ is an input matrix. The output equation $\boldsymbol{y}(t)$ models the outputs of the system as a combination of the output $C(t)$ and feedthrough $D(t)$ matrices. Moreover, the general nonlinear form of the state-space system is

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= g(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \end{aligned} \tag{2}$$

where $\dot{x}(t)$ is again the state of the system and $y(t)$ the output of the system. [18] In this thesis, the equations in the above systems are ordinary differential equations (ODEs), although the literature also features models with other types of differential equations.

The dimension of the system is determined by the number of state equations. It is equal to the number n of state variables in $x(t) \in \mathbb{R}^n$, so that the size of vector $x(t)$ is $n \times 1$. Additionally, the size of the input vector u is determined by the number of inputs r as $r \times 1$ and the number of outputs is $q \leq n$ so that $y \in \mathbb{R}^q$. Accordingly, the sizes of the matrices in the linear representation can now be determined so that all dimensions agree to matrix operations such as addition and multiplication. Matrix $A(t)$ has to be $n \times n$ and

$B(t)$ is $n \times r$. The number of outputs determines the size of $C(t)$ to be $q \times n$ and $D(t)$ has to be $n \times r$. In the nonlinear case, there are n state functions and q output functions.

Linear time-invariant (LTI) systems, where the coefficient matrices are constant in time, are the most elementary form of control systems. For an LTI system, the state equation is written as $\dot{x}(t) = Ax(t) + Bu(t)$ and the output as $y(t) = Cx(t) + Du(t)$. Given the transparent nature of LTI systems, they have been a subject of mathematical studies since their discover and thus additionally the first targets of model reduction methods.

With each model reduction method, a look into modifications for applicability to LTV systems is also given. While this is reasonable due to the array of time varying systems found in nature and biology, LTV systems can also be employed in approximating some nonlinear systems, which as established earlier allow the mathematical presentation of more complex and diverse phenomena.

2.1 Important Concepts

Here some important concepts for understanding model reduction will be introduced.

2.1.1 Linearity and Linearizations

Linearity is a mathematical property that has been successfully utilized in building models with analytically analyzable properties. Methods such as stability analysis, phase plane analysis and optimization techniques are well established for linear systems. For a model to be linear the principle of superposition has to apply, in other words the equations it consists of have to satisfy two properties: additivity and homogeneity of degree one. Additivity is defined as

$$f(a + b) = f(a) + f(b) \tag{3}$$

and homogeneity is defined as

$$f(\alpha v) = \alpha^k f(v), \tag{4}$$

where k symbolizes the degree of homogeneity, and a function can only be linear if $k = 1$. According to these properties, or the principle of superposition, the output of a linear function is the sum of the individual inputs [18]. An example of a linear function is $f(xy) = x + y$, where as a nonlinear function is for example a parabola $f(x) = x^2$.

Linearization is a process in which the value of a nonlinear function at a certain point is estimated with a linear approximation. This usually leads to a simpler representation of the function. The need for linearization stemmed for the difficulty of calculating the values of a function *near* a point, rather than the point itself. Consider for example a quadratic function. Evaluating it around a given point x is more tedious than approximating the values with the tangent line of the function at point x, which is the essential motivation for linearization. This is illustrated in Figure 2.1. [19]

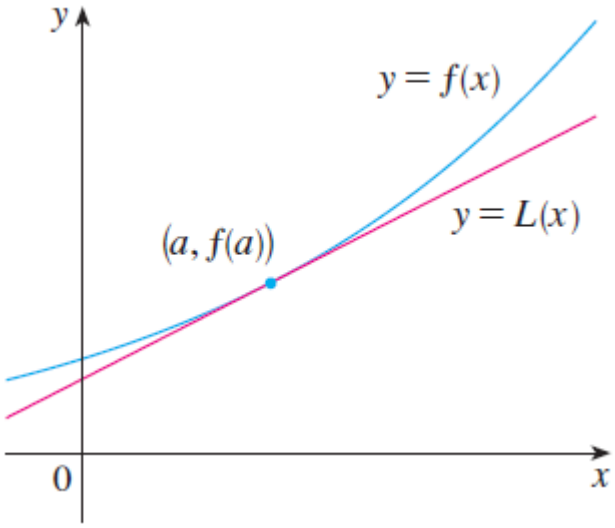

Figure 2.1: The nonlinear function $y = f(x)$ is linearized at point a; the new linear approximation $y = L(x)$ is a straight line. Image from [19].

One potential downside of linearization is that depending on the approximated function, the approximation is valid for a very short range. For example the tangent line linearization seen in Figure 2.1 is accurate only for evaluating values in close proximity to point a. As seen in Figure 2.1, the gap between $f(x)$ and $L(x)$ increases as the point of evaluation moves further from the linearization point a. Although the two functions might intersect again at unknown points, commonly this cannot be assumed and the error at points not in adjacency to a is unacceptably large. However, linearization at an equilibrium point does not have this problem if the system stays constant at all times, making it a very useful tool for mathematical analysis in these cases [20].

Multiple linearization techniques exist and only a brief introduction into linearization methods for differential equations is given here, as non-differentiable equations are a field of study on their own. The simplest form of linearization is the previously mentioned tangent line approximation. Another often employed solution is approximation with the Taylor series. The Taylor expansion uses n terms to arrive at an nth degree approximation at point a. Each term makes the approximation more accurate but also increases the complexity. Linearization with the Taylor series is performed by simply using the first two terms of the series, since they are still linear, making the approach essentially nothing more than obtaining the tangent line through derivatives.

Increasing the accuracy of linearized models is possible by performing linearization in a piece-wise manner [21]. With this approach the nonlinear function is linearized at several points. The approximation then becomes a group of linear functions so that each one is employed in a range where it has been determined to be accurate enough. In other words, at a given proximity of a linearization point, only one linear function is chosen for the approximation. Alternatively a weighing system can be used, where each linearization

is given a weight that changes with time. The desired accuracy and complexity of the nonlinear function determine the number of linearization points.

In addition to piece-wise linearization discussed above, it is also possible bilinearize a nonlinear equation for example by Carleman bilinearization [22]. A brief look at bilinearity is presented here, since in biochemistry stoichiometric equations have bilinear characteristics. Additionally, bilinearization is not the only form of pseudo-linearization. The bilinear form is more complex than the linear form, but is however not as complex as nonlinear forms generally. It allows multiplication and addition of the variables, which is sufficent for representing a wider variety of dynamical systems than the linear form [23]. Mathematically, a bilinear function satisfies the following conditions, first linearity on the left

$$f(u+v,w) = f(u,w) + f(v,w), \tag{5}$$

linearity on the right

$$f(u,v+w) = f(u,v) + f(u,w) \tag{6}$$

and homogeneity in all variables

$$f(\lambda u, v) = f(u, \lambda v) = \lambda f(u,v) \tag{7}$$

which essentially leads to the bilinear state equation

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A(t)\boldsymbol{x}(t) + \sum_{i=0}^{m} N_i \boldsymbol{x}(t) u_i(t) + B(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= C(t)\boldsymbol{x}(t) + D(t)\boldsymbol{u}(t), \end{aligned} \tag{8}$$

where in the MIMO case multiple inputs are multiplied individually with the state vector and then summed together [22, 24].

2.1.2 Stability

Stability of a linear time-invariant system, modeled for example with ordinary differential equations, is a phenomenon that is related to the equilibrium state of the system. A stable system is one that in absence of any disturbance, such as input, the output of the system remains constant. If a stable system is stimulated, it returns to the equilibrium state after the stimulus ends. However, stability exhibits many forms. The one discussed above is referred to as asymptotical stability. Mathematically, matrix A is stable if the real parts of its eigenvalues are negative and in system theory, such a state matrix of an LTI system is also called *Hurwitz*.

An unstable system does not satisfy the Hurwitz criterion and exhibits different dynamical behavior than a stable system. For example unbounded oscillations that continuously

increase in magnitude are a sign of an unstable system and imply the system contains positive feedback loops. A system that, in absence of input, maintains oscillations around an equilibrium point forever is called marginally or critically stable. Although not unstable, such a system does not satisfy the Hurwitz criterion. [18]

2.1.3 Subspace Projection

At the heart of many mathematical model reduction methods is subspace projection. Given that the goal is approximating the output of an original model with less equations, subspace projection gives the mathematical framework for achieving it. A model of a system generates values that form a space, for example an n dimensional model generates n dimensional vectors at every time step t. The space generated by the model can always be expressed with a linear combination of linearly independent orthonormal vectors that form the basis of the space. This space is then spanned by basis vectors $\{\boldsymbol{v}_1, ...\boldsymbol{v}_m\}$ of which there are m. Furthermore, if a subset of those vectors is chosen, say $k < m$ vectors, they span a k dimensional subspace of the original space where the values generated by the model exist. This subset of basis vectors can via linear combination express some of the values in the original space. In fact, mathematical model reduction methods find these basis vectors and subspaces so that as many as possible of the original values can be expressed with as few basis vectors as possible.

Assuming the orthonormal basis vectors $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_k\}$ have been found and they span a subspace, a matrix $V_k = [\boldsymbol{v}_1, ..., \boldsymbol{v}_k]$ is introduced and the system that is to be reduced can then be projected onto the subspace spanned by columns of matrix V_k. The projection happens via matrix multiplication so that the set of basis vectors is treated as a matrix. Consider the n dimensional (n equations) LTI system in Equation (1). A change of basis is performed by projecting the n dimensional state vector $\boldsymbol{x}$ to $k < n$ dimensional space by $\boldsymbol{x}_k = V_k^T \boldsymbol{x}$. Then, the rest of the system is projected to the same subspace by

$$\begin{aligned} \dot{\boldsymbol{x}}_k(t) &= V_k^T A V_k \boldsymbol{x}_k(t) + V_k^T B \boldsymbol{u}(t) \\ \boldsymbol{y} &= C V_k \boldsymbol{x}_k(t) + D \boldsymbol{u}(t), \end{aligned} \tag{9}$$

that defines the reduced order model. Equation (9) is called Galerkin projection. The resulting matrices $A_k = V_k^T A V_k$, $B_k = V_k^T B$ and $C_k = C V_k$ have sizes $k \times k$, $k \times r$ and $q \times k$, respectively, from where it is seen that the system no longer depends on the original dimension n. In other words, there are less equations to process. Moreover, since the matrix multiplications do not depend on time, they can be computed in the so called offline stage before the system is solved or simulated (online stage). It is worth noting that the projected matrices can no longer be interpreted with the original meaning of the variables they consisted of. However, if for example each $\dot{\boldsymbol{x}}_k(t)$ is saved to matrix X_k, an approximation of the original time series of the state variables can be retrieved by

multiplication with the projection matrix $X = V_k X_k$. Additionally, the approximations of the original outputs $\boldsymbol{y}(t)$ are available at all times and the inputs $\boldsymbol{u}(t)$ do not need any processing.

The same subspace projection process can be applied for LTV and nonlinear systems as well. However, the efficiency of the reduced model suffers as precomputation is not as widely possible. Consider for example the nonlinear function $f(\boldsymbol{x}(t))$, which only accepts $\boldsymbol{x}(t)$ as input, and not a reduced $\boldsymbol{x}_k(t)$. In effect, to evaluate this function $\boldsymbol{x}_k(t)$ has to be transformed back to the original space at every time step which results in calculating $f(V_k \boldsymbol{x}_k(t))$ at every time step of the simulation, greatly increasing computational expenses. Typically, the nonlinear reduction problem is solved with a linearization approach, although it introduces error to the system, or alternatively a very efficient storage and lookup system has to available for solving $f(V_k \boldsymbol{x}_k(t))$ [9]. The same problems stand true for LTV systems, since a projection such as $A_k(t) = V_k^T A(t) V_k$ is only valid as long as $A(t)$ remains unchanged. If the issue is unhandled and the basis is not changed, approximation error will increase [25]. Modern approaches for solving these problems, especially for nonlinear model reduction, will be presented later in the thesis.

2.2 Simplification by Pruning

In this section a collection of methods commonly used to reduce mathematical models of dynamical system will be introduced. The steps that will be outlined create an approach that is within biological modelers referred to as model simplification. It differs from more mathematical model reduction methods, such as subspace projection, in that the equations are always processed in their original format and less rigorous mathematics are employed. Simplification done this way is largely heuristic and always slightly *ad hoc*.

The goal of simplification is to gain understanding of the most significant generators of model dynamics, while also improving simulation speed and making analysis easier. The simplification approaches achieve this by identifying unnecessary or low-impact system variables and equations and consequently removing or combining them, which is well described by pruning. The end results consists of a smaller number of variables and equations that are still biologically interpretable. One of the most comprehensive and coherent descriptions of the procedure for models based on chemical equations is given by Kutumova et al. [26]. There, a six-step diagram is presented to guide the simplification workflow, shown also in Figure 2.2.

The process incorporates simplification methods that each address different parts or properties of the model. The first operation is removal of slow reactions, followed by quasi-steady-state approximation, lumping analysis and removal of linearly dependent variables, after which simplification of Michaelis-Menten kinetics and simplification of equations based on the law of mass action can be done in any order. Once an operation is performed to an equation or variable, it is suggested to confirm the preservation of dynam-

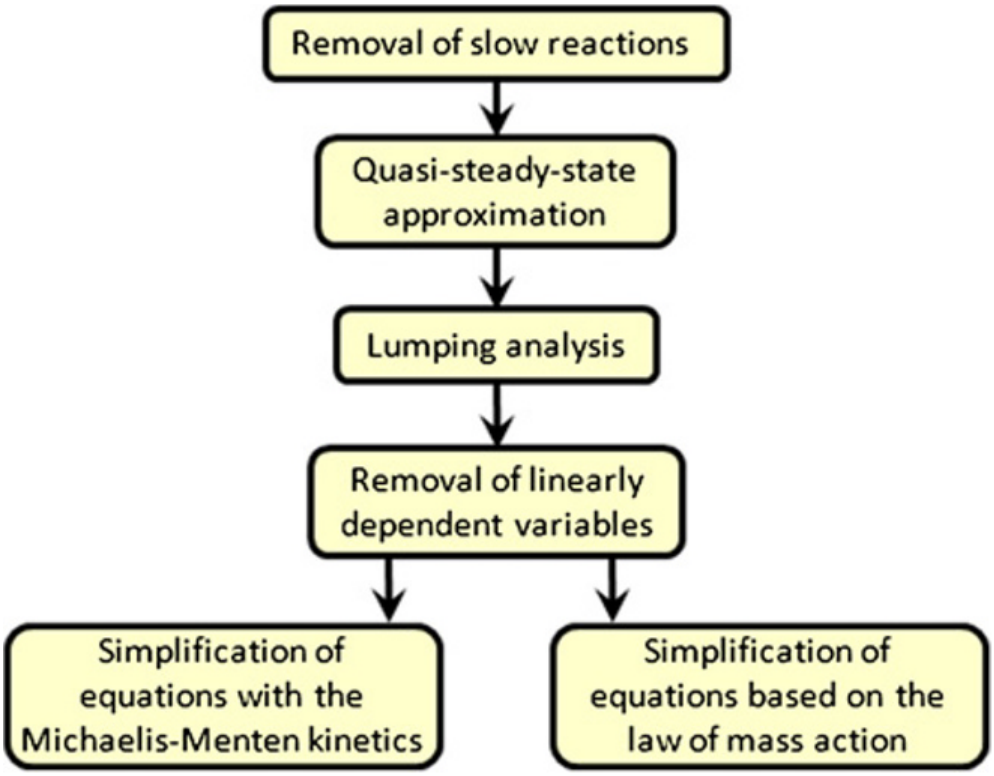

Figure 2.2: The process of simplifying a biochemical model that is based on chemical reaction equations. The figure is reproduced from [26]

.

ics. This can be done by comparing time series or using a mathematical criterion such as normalized sum of squared differences, although the problem of defining a suitable error bound has to then be concerned [26, 27]. Here each of these steps will be explained in more detail.

Removal of slow reactions is justified by the assumption that fast reactions are mostly responsible for producing the dynamics of the model. The speed or reaction rate v of a chemical process is defined as

$$v = k[\Xi]^n[\Theta]^m, \tag{10}$$

where $[\Xi]$ and $[\Theta]$ denote concentrations of species Ξ and Θ, n and m are reaction orders that depend on the underlying chemical equation and k is a rate constant that includes parameters affecting the reaction rate except the concentration. From Equation (10) it is seen that a reaction can happen fast even with a small rate constant given that the reactant concentrations are high and vice versa. In [26], a reaction r_1 is labelled as slow compared to reaction r_2 if

$$\max_t |v_{r_1}(t)| < k \max_t |v_{r_2}(t)|,$$

where $v_r(t)$ is the reaction speed at time t and $k = 10^{-2}$. Basically the maximum speed of a reaction during the simulation is measured, they are ordered based on this value and a threshold is used to neglect the slowest reactions. There is a consensus that splitting the

fast and slow reactions is effective [28, 29]. However, some sources actually suggest that the slow reactions dominate the fast ones, thus being mostly responsible for the dynamics in longer time scales [30, 31]. Additionally, in some situations it could be useful to actually eliminate equations that reach zero or equilibrium *fast* [32] or alternatively make groups of slow and fast equations and approximate them with their own equations [33]. In summary, if equations are going to be removed based on reaction speed as defined here the result has to carefully validated, since there is no unambiguous conclusion on when and how the method should be applied.

Quasi-Steady-State-Approximation (QSSA) is another established method for reducing the number of differential equations resulting from chemical stoichiometry. Consider for example an irreversible chemical equation

$$\mathrm{S} \longrightarrow \mathrm{I} \longrightarrow \mathrm{P}, \tag{11}$$

where substrate S forms an intermediate complex I before turning into product P. QSSA is used to obtain new approximated rate constants, maximum reaction speeds and initial concentrations for the remaining substrates and constant concentrations for the intermediates, for modeling the reaction with

$$\mathrm{S} \longrightarrow \mathrm{P} \tag{12}$$

[26]. The benefit is that this latter reaction requires less differential equations to model than the former and moreover, reduces the stiffness of the system by eliminating very rapid time scales. Originally formulated on the hypothesis that enzyme-substrate intermediate species in chemical reactions have small and stable concentrations or exist only briefly, QSSA has later been developed further to also extend to reactions where all concentrations are comparable, which additionally is more realistic *in vivo* [34]. However, the QSSA approach has drawn critique for being used based on insufficient conditions as for example in [35], where the dynamics of the system were found to considerably change after applying QSSA. Proper conditions for applying QSSA are given in [36, 37].

In lumping analysis, several species are combined into a single component. Lumping the selected equations is possible if a species is present in several equations with similar reaction rates and other reactants have similar concentrations. Consider for example two reactions

$$r_1 : \Xi + \mathrm{S}_1 \longrightarrow \mathrm{P}_1, r_2 : \Xi + \mathrm{S}_2 \longrightarrow \mathrm{P}_2, \tag{13}$$

where the reaction rates $k_1 = k_2$ and concentrations $[S_1](t) = [S_2](t)$, $[P_1](t) = [P_2](t)$ at all times t are (near) identical. Then new species S and P can be introduced and reactions r_1 and r_2 lumped to arrive at

$$2\,\Xi + \mathrm{S} \longrightarrow \mathrm{P}, \tag{14}$$

which replaces the previous two equations while the reaction rate $k = k_1 = k_2$ and concentrations $[S](t) = [S_1](t) = [S_2](t)$, $[P](t) = [P_1](t) = [P_2](t)$ stay the same. Naturally, this leads to smaller number of differential equations needed to model the system.

Removal of linearly dependent variables is a method for reducing the number of independent variables in the system. If the concentration of Ξ is a constant multiple of Θ

$$\Xi(t) \approx k\Theta(t) \tag{15}$$

at all times t, all occurences of Ξ can be replaced with $k\Theta$ [26]. Thus there will be no need to separately calculate the concentration of Ξ.

The last two steps of the procedure can be completed in any order. If Michaelis-Menten (MiMe) enzyme kinetics [38] are present, they can under certain circumstances be simplified. It is worth noting that (MiMe) kinetics is an imperfect model itself, the purpose of which is to calculate the reaction rate of an enzyme catalysed reaction. Based on the reaction rate and concentration of the substrate and enzyme, the (MiMe) equation is

$$v(t) = \frac{k[E]_0(t)[S](t)}{K_M + [S](t)}, \tag{16}$$

where K_M is the Michaelis constant, $[S]$ is the substrate concentration and $[E]_0$ is the initial concentration of the enzyme that catalyses the reaction. In the case $K_M >> [S](t)$ for $t = [0, T]$, the term $[S](t)$ from the denominator can be ignored, reducing the complexity of the calculation. Additionally, if the opposite is true $K_M << [S](t)$ for all t, the Michaelis constant is ignored and the entire equation simplifies to

$$v(t) = \frac{k[E]_0(t)\cancel{[S](t)}}{\cancel{[S](t)}} = k[E]_0(t), \tag{17}$$

which again reduces the computational complexity [26]. The value of the threshold for making the approximation is case specific and no general guidelines exist. It is of extreme importance to make sure the threshold is such that the dynamics of the system do not change.

As a final method is simplification if one reactant dominates others in an equation based on the law of mass action. Law of mass action is a model for calculating the reaction rate from the masses of the reactants. It is the basis for the reaction rate formula of reaction $aS_1 + bS_2 \longrightarrow P$ and is written as

$$v(t) = k[S_1]^a(t)[S_2]^b(t). \tag{18}$$

Given the conditions $[S_1]^a(t) >> [S_2]^b(t)$ and $\int_0^T [S_1]^a(t)\,\mathrm{d}t < \epsilon$ for $t = [0, T]$, an ap-

proximation that is linear in time can be used so that

$$v(t) = k[S_1]^a(0)[S_2]^b(t). \tag{19}$$

In other words, if the concentration of S_1 is much larger than the concentration of S_2, while the relative change of the concentration of S_1 is also smaller than threshold ϵ during a given time frame, the reaction rate is approximated by replacing the time-dependent concentration of S_1 with the initial value of S_1 at time $t = 0$ [26][Personal communication with Dr. E. Kutumova]. No definitive rules for the required thresholds exist.

Simplification with elimination type of model reduction can be performed with several software packages used for modeling and simulation in systems biology. These tools have been extensively revivewed in [39, 40]. A commonly used input format for these programs is the Systems Biology Markup Language (SBML) [41]. Examples of such software include BIOCHAM [42], COPASI [43] and GINsim [44].

Compared to subspace projection methods this approach has the advantage that the simplified model can be interpreted directly and the remaining variables and equations are immediately meaningful to the modeler and the user. In some sense, this gives great insight into the source of the dynamics, assuming that the behavior of the system is unaltered. However, ultimately ignoring the effects of ions and molecules is not biologically purposeful. In the following, methods that are able to approximate all the system variables with reduced systems are introduced.

2.3 Balanced Truncation

Balanced truncation (BT) is a very established model reduction method that is based on subspace projection. In essence, mathematical systems have defined inputs, outputs and state variables. With balancing, the states that are difficult to reach become difficult to observe thus reducing their significance. These states are then truncated, which effectively reduces the dimensions of the model. In such a way the input-output behavior of the system is approximated with a reduced representation. The general BT method is only applicable to linear systems. Moreover, the process requires that the state matrix of the system is stable, or Hurwitz, and this is also used as a condition for whether BT can be applied or not in the first place. BT approach to model reduction was developed by Moore [45] in 1981, while the modern version with error bounds was published by Glover [46] in 1984.

The balancing approach is unique in that the reduction is performed first making the system equally *controllable* (in some sources reachable) and *observable*, which are properties of control systems. If a system is controllable and observable, it is also called a minimal realization [47]. BT also operates directly on the system matrices and there is no need for simulating the large original system before model reduction.

Controllability, or controlling a system, signifies to what extent the outputs or the states of the system can be directed by adjusting the input signal of the system, in a given time interval [47]. A system is said to be *state* controllable at time t_0 only if it is possible to transfer it to any other possible state, such as the origo, from the initial state $\boldsymbol{x}(t_0)$. The system is not required to maintain that state. Moreover, the same condition can be established for *output* controllability with regards to the output $\boldsymbol{y}(t_0)$, although one form of controllability does not imply the other [18]. Generally controllability is used as a measure of state controllability, and in this sense a controllable system may not be output controllable, unless explicitly proved so. Observability on the other hand determines whether the state of the system at time t_0 can be inferred from the output $\boldsymbol{y}(t)$ of the system during a finite time interval $t_0 \leq t \leq t_1$ [47]. If a state is difficult to observe, in practice the output cannot be used to infer knowledge of the state of the system. It is worth remarking that while controllability and observability are properties of any system, a model of the same system might not possess these characteristics [18]. Moreover, for linear time invariant systems, the time interval during which controllability or observability are studied is irrelevant.

Mathematically controllability and observability can be calculated in many ways. The following methods can be employed for testing controllability and observability as well as how achievable the controllable and observable states are. For a linear time-invariant system $\boldsymbol{x}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}$, $\boldsymbol{y}(t) = C\boldsymbol{x}(t) + D\boldsymbol{u}(t)$, where A is an $n \times n$ square matrix, state controllability can be determined directly from the so called controllability matrix

$$P = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix} \tag{20}$$

and output controllability from

$$P_O = \begin{bmatrix} CB & CAB & CA^2B & ... & CA^{n-1}B & D \end{bmatrix}, \tag{21}$$

called the output controllability matrix. The process for finding the observability matrix Q is similar since

$$Q = \begin{bmatrix} C^* & A^*C^* & (A^*)^2C^* & ... & (A^*)^{n-1}C^*. \end{bmatrix} \tag{22}$$

With the controllability and observability matrices defined, controllability is achieved if P has full row rank, meaning that the rows of the matrix are linearly independent. Due to the matrix sizes of the state space format, the size of P is $n \times nr$ and the full row rank is thus n. The size of P_O is $m \times (n+1)r$, so full row rank is m. For the observability matrix Q, $n \times nm$, full column rank n is required, so that all columns of Q are linearly independent. If the full rank condition is not fulfilled, there are analytical methods for finding which individual states are controllable or observable. [18]

One approach often taken in the literature for determining controllability or observability is to calculate the controllability and observability grammian matrices, W_C and W_O. The grammians are obtainable as solutions to the continuous Lyapunov equations [47]. For controllability grammian the equation is

$$AW_C + W_C A^* + BB^* = 0 \tag{23}$$

and observability grammian

$$A^* W_O + W_O A + C^* C = 0, \tag{24}$$

and since both equations are linear their solutions can be calculated analytically. The solution for the controllability grammian then becomes

$$W_C = \int_{t_0}^{t_1} e^{At} B B^T e^{A^T t}\, dt \tag{25}$$

and observability grammian

$$W_O = \int_{t_0}^{t_1} e^{A^T t} C^T C e^{At}\, dt, \tag{26}$$

in the time span $[t_0, t_1]$ of interest, although in practice the integrals need not to be evaluated in order to solve the Lyapunov equations [48]. The matrices W_C and W_O are Hermitian square $n \times n$ positive semidefinite matrices [45]. In accordance to the requirement that the state matrix A has to be stable for BT to be applicable, the Lyapunov equations do not have unique solutions if this condition is not satisfied. Simply put, the controllability and observability grammians are not uniquely defined for unstable systems and thus there is no guarantee that they can be calculated [49].

A system is called controllable or observable if the respective grammian matrix is invertible. In other words this means that for example W_C is invertible if there exists any matrix X so that $XW_C = W_C X = I$, where I is the identity matrix. Otherwise the matrix W_C would be singular and the system could not be said to controllable in this time window. Moreover, the singularity can also be determined from the eigenvalues of the matrix among many other properties. If any of the eigenvalues is zero, the matrix is singular and thus not invertible. The same stands true for the observability grammian.

BT uses the controllability and observability grammians to calculate a reduced order model. The objective is to remove those states that are difficult to control and observe. That specific goal is effectively reached when the system is *balanced*, which means that the difficult or impossible to control states also become difficult to observe [50]. Achieving a balanced realization is possible by scaling the system via matrix multiplications

using a carefully chosen transformation matrix T. This is also seen as a change of basis for the state vector $\hat{\boldsymbol{x}} = T\boldsymbol{x}$. Assuming that T is available, using a similarity transformation the state space matrices become

$$(\hat{A}, \hat{B}, \hat{C}, D) = (TAT^{-1}, TB, CT^{-1}, D) \tag{27}$$

where, if the system is balanced, the system matrices are arranged descendingly based on their contribution to the controllability and observability of the system. Furthermore, this allows for partitioning the matrices into block matrices as

$$(\hat{A}, \hat{B}, \hat{C}, D) = (\begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix}, \begin{bmatrix} \hat{B}_1 \\ \hat{B}_2 \end{bmatrix}, \begin{bmatrix} \hat{C}_1 & \hat{C}_2 \end{bmatrix}, D). \tag{28}$$

Since the system matrices are partitioned and arranged, a reduced system is obtained by keeping only the the most important states $(\hat{A}_{11}, \hat{B}_1, \hat{C}_1, D)$ that are required to approximate the impulse response of the system [45]. In practice, the sizes of the matrices will be determined by how much approximation error is tolerated and how greatly the dimension of the system has to be reduced.

The naturally arising question is then finding the transformation matrix T that balances the system. First, it is useful to define what a balanced system means in the sense of grammians: a balanced system is one where the controllability and observability grammians are equal diagonal matrices $W_C = W_O = \Sigma_H$ that additionally have only positive values on the diagonal. These diagonal values are called Hankel singular values and their magnitude determines the importance of corresponding (balanced) controllable and observable state [45]. In terms of model reduction, once this condition is achieved, the "low energy" states that are rarely reached and observed can be removed, thus Σ_H provides an error estimate for model reduction. [50].

Laub et al. [51] propose a numerically stable method for determining T by applying the Cholesky factorization to the grammians and then the singular value decomposition (SVD) to the product of the factorizations [51]. The algorithm takes advantage of the balancing transformation as it establishes a relationship between the grammians, such that if $\hat{W_C} = TW_CT^*$ and $\hat{W_O} = T^{-*}W_OT^{-1}$ then $\hat{W_C} = \hat{W_O} = \Sigma_H$. More exactly, this stems from the fact that Σ_H can also be obtained as the square roots of the eigenvalues of the product W_CW_O of the grammians [50, 52].

Proceed with calculating the Cholesky decomposition

$$W_C = ZZ^* \tag{29}$$

and

$$W_O = LL^*, \tag{30}$$

where the grammians are factorized into the product of an uncorrelated lower triangular matrix and its conjugate transpose. Strictly speaking the decomposition can only be calculated if the source matrix is positive definite. As the grammians are only guaranteed to be semidefinite, this step might require the use of an approximate factorization, such as finding the nearest factorizable matrix [53]. Moreover, there are numerical methods for solving the Lyapunov equations directly for U and L so that the grammians and their factorizations are not explicitly computed [51]. Next, singular value decomposition (SVD) is applied as

$$Z^*L = U\Sigma V^* \tag{31}$$

to obtain the left singular vectors in U and right singular vectors in V. Then, the left balancing transformation is obtained as

$$T = V^*L^* \tag{32}$$

and the right balancing transformation, which is the inverse of T, is obtained from

$$T^{-1} = ZU \tag{33}$$

in order to avoid computing actual matrix inversions. The singular values Σ obtained with SVD equal exactly the Hankel singular values in Σ_H [51].

Recall that the singular values Σ can be used as a metric for the importance of states. If the result of SVD is partitioned as the system earlier

$$Z^*L = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} {V^*}_1 \\ {V^*}_2 \end{bmatrix}, \tag{34}$$

the truncation step is performed by discarding U_2, Σ_2 and V_2^*. Σ_1 should include as many singular values as deemed important for the approximation. Typically the singular values decay exponentially, so that a very good approximation is obtained with relatively few values. If the original dimension of the system is n (A is an $n \times n$ matrix), k singular values are chosen so that $k < n$ and Σ_1 becomes a $k \times k$ matrix, while U_1 is $n \times k$ and V_1^* is $k \times n$. In other words, this is extracting k first columns from U and rows from V^* (columns from V), effectively defining the number of states that the reduced model will have. In practice, the truncation step is built into the balancing transformation so that both steps are applied concurrently. A left truncating and balancing T_1 that reduces the dimension of the original system is then calculated from

$$T_1 = V_1^*L^*. \tag{35}$$

Additionally, the right truncating transformation matrix

$$\tilde{T_1} = ZU_1, \tag{36}$$

that is equal to truncating the result of Equation (33). The dimension of T_1 is $k \times n$ and $\tilde{T_1}$ is $n \times k$ as per the matrix multiplications.

Performing the Galerkin projection of Equation (27) with the truncating transformation matrices T_1 and $\tilde{T_1}$ results in exactly the reduced state matrices

$$(T_1 A\tilde{T_1}, T_1 B, C\tilde{T_1}, D) = (\hat{A}_{11}, \hat{B}_1, \hat{C}_1, D) \tag{37}$$

that are the most controllable and observable blocks of Equation (28). Moreover, the (initial) state vector is also transformed by reduced balancing basis T_1 so that $\hat{\boldsymbol{x}} = T_1\boldsymbol{x}$ with dimensions of $\hat{\boldsymbol{x}}$ being $k \times 1$, which completes the model reduction with BT. The final system is

$$\begin{aligned} \frac{d\hat{\boldsymbol{x}}(t)}{dt} &= \hat{A}_{11}\hat{\boldsymbol{x}}(t) + \hat{B}_1\boldsymbol{u}(t) \\ \hat{\boldsymbol{y}}(t) &= \hat{C}_1\hat{\boldsymbol{x}}(t) + D\boldsymbol{u}(t). \end{aligned} \tag{38}$$

Here, the input vector $\boldsymbol{u}(t)$ remains at original dimensions $r \times 1$, $\hat{A}_{11}$ is a $k \times k$ matrix, $\hat{B}_1$ is $k \times r$ and $\hat{C}_1$ is $n \times k$, while D stays $n \times r$. It is noteworthy that the reduction process is computed in the so called offline stage of simulating the system, meaning that the calculations are only performed once. Simulating the system in this form then grants a computational advantage due to a smaller number of differential equations that have to be solved. The outputs of the system in $\hat{\boldsymbol{y}}(t)$ are direct approximations of the outputs of the original system.

Error bounds for the BT approximation can be calculated for example with the (Hankel) singular values σ of the system. The upper bound for the time domain error between the original and reduced outputs $\|\boldsymbol{y}(t) - \boldsymbol{y}_r(t)\|$ at time t is

$$\|\boldsymbol{y}(t) - \boldsymbol{y}_r(t)\| \leq 2 \sum_{i=k+1}^{n} \sigma_i \|\boldsymbol{u}(t)\| \tag{39}$$

where $\|\boldsymbol{u}(t)\|)$ is the Euclidian norm of the input vector and k and n the dimensions as number of singular values of the reduced and original system, respectively [46]. Moreover, the calculation is feasible due to Hankel singular values being *invariants* of the system, meaning they are uneffected by transformations such as the one used for balancing and truncating the original system. It is seen that increasing the number of singular values used in the approximation reduces the error, but the magnitude of σ_i also has an effect. If the singular values decay exponentially, there are diminishing returns in increasing their number in the reduced model, with regards to the output error. Finally, it is notewor-

thy that the presented error definition only measures the distance between original and reduced outputs and does not account for actual behavior of the models.

Preservation of stability of the original system in the reduced system is an important aspect of model reduction. BT almost always preserves the stability of the original system [45], which is partly due to the original requirement that A is Hurwitz. However, there are other forms of balancing approaches developed for specific situations, such as LQG balancing [54] for unstable systems, where stability of the reduced model cannot be guaranteed. Generally, the reduced model will be stable if the Hankel singular values do not contain any duplicates [55].

Although the rigorous mathematical proof that exists for calculating and proving controllability and observability makes BT an order reduction method with very solid foundations, the additional required algorithms are also the biggest bottleneck of the process. Matrix computations, such as those required for establishing controllability and observability through Lyapunov equations, are very memory intensive operations on the hardware of the computation system. With large equation systems memory demands for the computational process become intolerable. Solving this problem is related to approximating the solutions to the Lyapunov equations, in other words the grammians, instead of calculating them explicitly [56], after which BT proceeds normally.

BT has extensions for LTV systems [57]. The original procedure is generally not valid since the transformation matrices should now change with time. Time-dependence also causes the choice of suitable reduced block matrices to be very problematic and additionally might affect the stability of the system if the state matrix A changes. However, if in the interval $[t_0, t_1]$ stability can be verified and grammians computed, the traditional BT method can be applied, although similar performance to LTI systems cannot be expected. In principle, if a matrix varies $A(t)$ with time, the transformation $A_t(t) = TA(t)T^{-1}$ is only valid until $A(t)$ changes again, requiring the Galerkin projection to be computed online at every t where $A(t)$ changes, thus negating much of the performance improvement gained through model reduction. Furthermore, consider a time-varying system,

$$\hat{A}(t) = TA_cT^{-1} + TA_t(t)T^{-1} \tag{40}$$

where from the time-varying matrix terms are separated into $A_t(t)$ and a time-invariant A_c. If such a reformation is feasible and $A_t(t)$ is sparse or changes rarely, reduction could be beneficial for computational speed. Shokoohi et al. propose a reduction method for LTV systems through uniformly balanced realizations [57] that results in a periodic subspace projection, which is the simplest form of time-varying, and Ma et al. use the method of snapshots [58] for the same purpose [59]. Moreover, Stykel and Vasilyev present a method by which a time-varying system matrices $B(t)$ and $C(t)$ are made (almost) time-invariant by shifting time-variability to the input vector $\boldsymbol{u}(t)$ [60]. Additionally, although

a reduced time-varying model would not achieve computational savings, reduction can still be feasible for analyzing the system mathematically.

Balancing of time-invariant nonlinear systems has to deal with the same problems as LTV systems: how to define the grammians, since for nonlinear systems, the grammians do not necessarily exist or are not unique. Moreover, there are no a priori guarantees of good approximations. If the grammians can be defined, the nonlinear system can be balanced, with some restricting assumptions [61]. Assuming that the grammians and thus a balancing transformation are found, the next problem is that a nonlinear function cannot be reduced to a subspace by linear projection in the offline stage, since it is only defined in the original space. Consider the case of a reduced model with a nonlinear function $f(\boldsymbol{x})$ and a change of basis by T, so that $\hat{\boldsymbol{x}} = Tx$. The reduced nonlinear function is then

$$G_r(T^{-1}\hat{\boldsymbol{x}}(t)) = Tf(T^{-1}\hat{\boldsymbol{x}}(t)), \tag{41}$$

where $G_r(\cdot)$ is a reduced realization and $f(\cdot)$ is a nonlinear function that produces an output vector from input vector $x = T^{-1}\hat{\boldsymbol{x}}(t)$. It becomes imminent that the projection is possible only after the function has been evaluated. This is a process of first projecting $\hat{\boldsymbol{x}}$ to the original space, evaluating the function in the original (high) dimension and finally left multiplication with T. Furthermore, this would mean the projection has to be calculated at every time step t, which is expensive and completely negates the benefits of model reduction. Moreover, nonlinear BT often relies on linearization of the system [61–64].

2.4 Moment Matching

Model reduction by moment matching (MM) encompasses a wide array of methods that are based on approximating the transfer function, which is a compact representation of the input-output behavior of a system in the frequency plane. A filter-function approach to low rank approximation was first suggested in [65] and later revised in [66]. In some sources MM methods are referred to as rational interpolation methods, since the transfer function is interpolated at specific *moments*. Typically the algorithms in this category are efficient to calculate but struggle to provide error bounds or stability of the reduced model. Additionally, the transfer function is challenging to define for a nonlinear system, making MM a method primarily for linear systems.

Although the transfer function is a very complex topic, a short introduction is given here. Let us define the transfer function $H(s)$ for a MIMO LTI system such as the one in Equation (1). Applying the Laplace transform to the state-space matrices of the system leads to a definition of the transfer function as

$$H(s) = C(sI - A)^{-1}B + D \tag{42}$$

where $s = \sigma + j\omega$ is a complex number on the s-plane on which the Laplace transform is defined [18]. With a discrete time system, the z-transform is used to arrive at similar representation. Using the transfer function, the Laplace transform of the output $pmby(t)$ of the system in the s-plane can be calculated by

$$Y(s) = H(s)U(s) \tag{43}$$

where the transfer function is multiplied with $U(s)$, the Laplace transform of the inputs $\boldsymbol{u}(t)$ of the system. Furthermore, $H(s)$ is a matrix of size $q \times r$, where q is the number of outputs and r is the number of inputs to the system. Basically, each input in r is mapped to q outputs, which creates the matrix structure. In the special SISO case, $H(s)$ is reduced to one function, and due to this simpler structure many MM methods have been, at least initially, developed for SISO systems. Most importantly, it is seen from Equation (42) that the complexity of the transfer function depends on the original dimension N of the system due to operations with state matrix A and accordingly.

MM model reduction finds a suitable approximation $H_k(s)$ of $H(s)$ so that some moments of $H_k(s)$ are matched to the original moments of $H(s)$. The moments of a function are found through power series expansions. Power series, such as the Taylor expansion detailed in the linearization section, represent the target function at a given point s_0 as a sum of terms that form an infinite polynomial. Summing the terms then gives an approximation of the original function at the expansion point based on how many terms are included in the addition [19]. The moments are another characterization of the system dynamics and different expansions are possible for calculating them and while such a representation has many benefits, here it will be employed in model reduction. Several algorithms have been developed to perform MM at different expansion points s_0. For example, if $s_0 = \infty$ the Padé via Lanczos [67] or the Arnoldi procedure [68] are preferred for solving the problem, and in the general case of arbitrary s_0 the rational interpolation methods are suggested [50]. Different s_0 can be related to different properties of the reduced model; $s_0 = 0$ achieves steady state accuracy, small values achieve accuracy at slow dynamics and the larger the value of s_0, the better high frequencies of the transfer function are approximated [69]. Additionally, imaginary expansion points can be related to good local accuracy near s_0, but less generalization capability of the reduced model [69]. Additionally, due to the locality of the expansion point based approach, the choice of the expansion point s_0 is very application dependent and often requires manual selection [50].

Calculating the moments of the transfer function for model reduction purposes is best illustrated under certain assumptions and preparations. First, the target system is LTI and SISO so that matrices C and B are vectors $\boldsymbol{c}, \boldsymbol{b}$, and additionally the feedthrough matrix D is assumed to be zero. Let us introduce matrix G and vector $\boldsymbol{r}$ according to Equation (42)

of the transfer function

$$G = -(sI - A)^{-1} \tag{44}$$

and

$$\boldsymbol{r} = (s_0 I - A)^{-1}\boldsymbol{b} \tag{45}$$

where in both equations A, $\boldsymbol{b}$ are state space matrices. Furthermore, $H(s)$ of a SISO systems then becomes a scalar valued function

$$h(s) = \boldsymbol{c}^T(I - (s - s_0)G)^{-1}\boldsymbol{r} \tag{46}$$

that now only depends on one matrix G. For very small systems G can be diagonalized and reduced, thus reducing $h(s)$ to $k < N$ dimensions. However, the process is numerically unstable so the general case requires extra steps. Next $h(s)$ is expanded for example with the Taylor series to obtain

$$\begin{aligned} h(s) &= \sum_{j=0}^{\infty}(\boldsymbol{c}^T G^j \boldsymbol{r})(s - s_0)^j \\ &= \boldsymbol{c}^T\boldsymbol{r} + \boldsymbol{c}^T G\boldsymbol{r}(s - s_0) + \boldsymbol{c}^T G^2\boldsymbol{r}(s - s_0)^2 + ... \\ &= m_0 + m_1(s - s_0) + m_2(s - s_0)^2 + ..., \end{aligned} \tag{47}$$

where $m_j = \boldsymbol{c}^T G^j \boldsymbol{r}$ is the jth moment of $h(s)$ at the expansion point s_0. [9] Moreover, each moment m_j can be seen as the j:th derivative of the transfer function.

A function that matches as many moments of another function at expansion point s_0 is called a Padé approximant. In other words, the Padé approximant of a function has the same power series as the approximated function, at a given expansion point. The Padé approximant has an order of two values, marked with $[m/n]$ and it uses two polynomial functions $P_m(x)$ and $Q_n(x)$ to describe any other function $f(x)$ with the notation

$$[m/n]_f = \frac{P_m(x)}{Q_n(x)},\ Q_n(x) \neq 0 \tag{48}$$

and for the moments to match $f(x)$ it satisfies the equation

$$f(x) = \frac{P_m(x)}{Q_n(x)} + \mathcal{O}(x^{m+n+1}) \tag{49}$$

up to order $\mathcal{O}(x^{m+n+1})$. If $h_k(s)$ is the Padé approximant of the transfer function $h(s)$, it is written as

$$h(s) = h_k(s) + \mathcal{O}((s - s_0)^{2n}). \tag{50}$$

The k moments of $h(s)$ are already defined from the Taylor series, so the next step is

defining $h_k(s)$. Choosing the order of the Padé approximation as $m = n - 1$ gives

$$h_k(s) = \frac{P_{n-1}(s)}{Q_n(s)} = \frac{a_{n-1}s^{n-1} + ... + a_1 s + a_0}{b_n s^n + b_{n-1}s^{n-1} + ... + b_1 s + 1} \tag{51}$$

where b_0 is set to 1. The coefficients a_j and b_j have to be solved in order to obtain the approximant. Since Equation (50) contains the condition for the moments to match, multiply it by $Q_n(s)$ on both sides to obtain

$$h(s)Q_n(s) = P_{n-1}(s) + \mathcal{O}((s - s_0)^{2n}). \tag{52}$$

Moreover, inserting the moments of $h_k(s)$ to the above equation ultimately leads to the equation system for the coefficients b_j of $Q_n(s)$

$$\begin{bmatrix} m_0 & m_1 & \cdots & m_{n-1} \\ m_1 & m_2 & \cdots & m_n \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1} & m_n & \cdots & m_{2n-2} \end{bmatrix} \begin{bmatrix} b_n \\ b_{n-1} \\ \vdots \\ b_1 \end{bmatrix} = - \begin{bmatrix} m_n \\ m_{n+1} \\ \vdots \\ m_{2n-1} \end{bmatrix} \tag{53}$$
$$M_n \boldsymbol{b} = -\boldsymbol{m}$$

where M_n is the ***Hankel matrix*** and also known as a moment matrix. Since the moments m_j are known, $\boldsymbol{b}$ can be solved, given that M_n is nonsingular. Furthermore, a_j of $P_{n-1}(s)$ can now be computed according to

$$\begin{aligned} a_0 &= m_0 \\ a_1 &= m_0 b_1 + m_1 \\ &\vdots \\ a_{n-1} &= m_0 b_{n-1} + m_1 b_{n-2} + ... + m_{n-2} b_1 + m_{n-1} \end{aligned} \tag{54}$$

from where it is seen that $h_k(s)$ is a nth Padé approximant of $h(s)$ [9, 67]. The biggest drawback of the above method, also known as Asymptotic Waveform Evaluation [70], is that M_n quickly becomes ill conditioned for more accurate approximations ($n > 20$, [9]) which basically prevents the approximation of large frequency ranges. Additionally, compared to the already local nature of the moment matching approach, this greatly reduces the usability of MM methods.

As stated above, a more numerically stable method is required for obtaining better approximations and ultimately reductions. Such a method can be found by exploiting the connection between Padé approximants and the Lanczos process, and it is known as Padé via Lanczos (PVL) [67]. PVL is effective, because it avoids the computation of the moments directly, thus the numerically poor matrix M_n is not formed at all. To explain

the Lanczos process the meaning of Krylov subspaces has to first be established.

The Krylov subspace is spanned by the column vectors that result from the multiplication between a square matrix G and a starting vector $\boldsymbol{r}$, for example the matrix G and vector $\boldsymbol{r}$ defined earlier in the context of transfer functions in Equations (44) and (45). The nth order Krylov subspace is

$$K_n(G, \boldsymbol{r}) = \text{span}\{\boldsymbol{r}, G\boldsymbol{r}, G^2\boldsymbol{r}, .., G^{n-1}\boldsymbol{r}\} \tag{55}$$

also known as the right Krylov subspace, and if $G^T \neq G$, the left Krylov subspace is defined as

$$K_n(G, \boldsymbol{c}) = \text{span}\{\boldsymbol{c}, G^T\boldsymbol{c}, (G^T)^2\boldsymbol{c}, .., (G^T)^{n-1}\boldsymbol{c}\} \tag{56}$$

with a starting vector $\boldsymbol{c}$ that is the output vector of a SISO LTI system. With these selections, the Krylov subspace is connected to the previously defined moment matrix M_n through inner products between the left and right subspace as

$$m_{2j} = ((G^T)^j\boldsymbol{c})^T \cdot (G^j\boldsymbol{b})^T$$

where m is an element of M_n for $j = 1, 2, ..., n-1$. However, $G^j\boldsymbol{r}$ and $(G^T)^j\boldsymbol{c}$ lack the numerical qualities for being linearly independent basis vectors. The goal then becomes finding basis vectors (Lanczos vectors) $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ and $\{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_n\}$ that span the same Krylov subspaces $K_n(G, \boldsymbol{r})$, $K_n(G, \boldsymbol{c})$ defined earlier but with better numerical properties, which is what the Lanczos process does [9]. In summary, the Krylov subspace contains the necessary information to numerically robustly define more moments than by Padé approximation, and the Lanczos process [71] is a way to find these basis vectors that span these exact subspaces.

The Lanczos process is a step-wise process, which eventually tridiagonalizes the target square matrix G of size $n \times n$ [71]. It generates two matrices V_n and W_n for which the columns are exactly the respective Lanczos vectors of $\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{w}_i\}$.

Algorithm 1 depicts the Lanczos process to obtain a projection subspace for a SISO LTI system. In Algorithm 1 the inputs G and $\boldsymbol{r}$ are derived from the state space matrices at expansion point s_0 and $\boldsymbol{c}$ is the output vector while the outputs are two subspaces spanned by the linearly independent basis vectors in matrices V_n and W_n. Here $\boldsymbol{r}$ and $\boldsymbol{c}$ are used as starting vectors based on which biorthogonal $\boldsymbol{v}$ and $\boldsymbol{w}$ are calculated, so that a connection to the state space matrices is established [67]. Moreover, the process itself is iterative and runs at most for n steps that ultimately equals the dimension of the square state matrix A. However, the process might terminate prematurely, in which case the basis generation process is not complete and only reaches $k < n$ steps, resulting in V_k and W_k that have only k columns. On the other hand, the process might be voluntarily terminated early. The resulting two matrices can be used to project onto the subspaces $K_n(G, \boldsymbol{r})$ and $K_n(G^T, \boldsymbol{c})$.

Algorithm 1 Lanczos process

INPUT: $\boldsymbol{r} = ((s_0 I - A)^{-1})\boldsymbol{b}$, $\boldsymbol{c}$, $G = -(s_0 I - A)^{-1}$
OUTPUT: $V_n = v_1, ..., v_n$, $W_n = w_1, ..., w_n$
$\rho_1 = \|\boldsymbol{r}\|$, $\eta_1 = \|\boldsymbol{c}\|$, $\boldsymbol{v}_1 = \boldsymbol{r}/\rho_1$, $\boldsymbol{w}_1 = \boldsymbol{c}/\eta_1$
for $k = 1$ to n **do**
    $\delta_k = \boldsymbol{w}_k^T \boldsymbol{v}_k$
    $\alpha_k = \boldsymbol{w}_k^T G \boldsymbol{v}_k / \delta_k$
    $\beta_k = (\delta_k / \delta_{k-1})\eta_k$
    $\gamma_k = (\delta_k / \delta_{k-1})\rho_k$
    $\boldsymbol{v} = G\boldsymbol{v}_k - \boldsymbol{v}_k \alpha_k - \boldsymbol{v}_{k-1}\beta_k$
    $\boldsymbol{w} = G^T \boldsymbol{w}_k - \boldsymbol{w}_k \alpha_k - \boldsymbol{w}_{k-1}\gamma_k$
    $\rho_{k+1} = \|\boldsymbol{v}\|$
    $\eta_{k+1} = \|\boldsymbol{w}\|$
    $\boldsymbol{v}_{k+1} = \boldsymbol{v}/\rho_{k+1}$
    $\boldsymbol{w}_{k+1} = \boldsymbol{w}/\eta_{k+1}$
end for

Assuming a completed process, after the Lanczos matrices are constructed tridiagonalization is performed by similarity transformation as

$$V_n^{-1} G V_n = T_n \tag{57}$$

where

$$T_n = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \rho_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_n \\ & & \rho_n & \alpha_n \end{bmatrix} \tag{58}$$

and additionally

$$\tilde{T}_n^T = D_n T_n D_n^{-1}, \text{ where } D_n = W_n^T V_n.$$

In case the process encountered a termination at step k, the transformation is

$$W_k^T G V_k = W_k^T V_k T_k = D_k T_k. \tag{59}$$

If tridiagonalization with the Lanczos process is applied to matrix G of SISO LTI system, the transfer function becomes

$$h(s_0) = (\boldsymbol{c}^T \boldsymbol{r}) \boldsymbol{e}_1^T (I - (s_0 - s))T_n)^{-1} \boldsymbol{e}_1 \tag{60}$$

where $\boldsymbol{e}$ is the first column vector of I_n and which equals the original transfer function of the system. However, with large systems, the process is likely to encounter a breakdown at k steps, in which case a smaller dimensional T_k consisting of the leading $k \times k$ block

of T_n is obtained. This leads to a reduced transfer function

$$h_k(s_0) = (\boldsymbol{c}^T\boldsymbol{r})\boldsymbol{e}_1^T(I - (s_0 - s))T_k)^{-1}\boldsymbol{e}_1 \tag{61}$$

that corresponds to the Padé approximant of the original transfer function, since T_k is calculated in a subspace chosen so that the moments of the Krylov subspace transformed system match the original moments [9]. The difference is that the calculation method was more robust, allowing for a larger k and a better approximation and that instead of approximating with a polynomial function, a matrix representation was obtained. This resolves the problem of approximating very large systems with moment matching, since a good approximation that still has a smaller dimension compared to the original system is now obtainable.

The incomplete Lanczos projection matrices V_k and W_k can also be be applied for model reduction in time domain, although the basis vectors are still first calculated in the Laplace domain. They can be obtained by voluntarily interrupting the Krylov process or by calculating as many steps as possible. However, reaching n steps in the Algorithm 1 serves no purpose for model reduction. Under expansion at s_0 and further employing G and r from Equation (44) and Equation (45), the SISO LTI system is

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= -(I + s_0G)\boldsymbol{x}(t) + \boldsymbol{r}u(t), \\ y(t) &= \boldsymbol{c}\boldsymbol{x}(t) \end{aligned} \tag{62}$$

and a reduced form is obtained with the change of basis $V_k^T\boldsymbol{x}(t) = \boldsymbol{x}_k(t)$ and transformation by W_k^T which yields

$$\begin{aligned} \dot{\boldsymbol{x}}_k(t) &= -W_k^T(I + s_0G)V_k\boldsymbol{x}_k(t) + W_k^T\boldsymbol{r}u(t), \\ y(t) &= \boldsymbol{c}V_k\boldsymbol{x}_k(t) \end{aligned} \tag{63}$$

that is an $k < n$ dimensional system that matches as many moments of the transfer function of the original system, making it a Padé approximant. Additionally, the original inputs are preserved and a direct approximation of the outputs is obtained. The reduced matrices can be precomputed, so that in the online phase of simulating the model a smaller amount of differential equations has to be processed.

Generally MM methods are defined for SISO systems and applications to MIMO systems are a case requiring special consideration. There are special procedures for the MIMO case depending on whether B and C are symmetric and positive semidefinite (in which case a symmetric Lanczos process can be used [72]) [9, 50] or bilinear [24]. The general approach to MIMO systems with Krylov subspace is called Block-Krylov [73] and a universally applicable reduction procedure that handles all sizes of non-Hermitian input and output matrices based the block approach is given in [74], although the authors

note that the connection to Padé approximation still has to be verified. Additionally, the Lanczos algorithm is only one possibility to finding the Krylov subspaces, and for example the iterative Arnoldi algorithm or tangential interpolation methods might be more suitable in application where Lanczos fails.

One solution to the problem of very local approximations generated by moment matching methods is calculating approximations at several expansion points. The approach is called multipoint moment matching or multipoint interpolation [48]. However it is computationally more expensive compared to increasing the order k of a single approximation.

The error of the reduced model obtained from MM methods depends on the algorithm that is used to calculate the approximation of the transfer function. In PVL, the order of the Padé approximant, that is the column number of the Krylov projection subspace or number of steps taken in the Lanczos algorithm, determines the error [9]. If a kth order SISO LTI approximation is used, the error at expansion point s_0 is determined by

$$h(s) - h_k(s) = (\boldsymbol{c}^T\boldsymbol{r})(\frac{\rho_{k+1}\eta_{k+1}}{\gamma_k})[s_0^2\tau_{k1}(s_0)\tau_{1k}(s_0)]\gamma_{k+1}(s_0), \tag{64}$$

where $\tau_{k1}(s_0) = \boldsymbol{e}_1^T(I - s_0T_k)^{-1}\boldsymbol{e}_k$, $\tau_{1k}(s_0) = \boldsymbol{e}_k^T(I - s_0T_k)^{-1}\boldsymbol{e}_1$ and $\gamma_{k+1}(s_0) = \boldsymbol{w}_{k+1}^T(I - s_0G)^{-1}\boldsymbol{v}_{k+1}$ which are obtained from the Lanczos process in Algorithm 1 [75]. Moreover, the term $s_0^2\tau_{k1}(s_0)\tau_{1k}(s_0)$ is most dependent on k of the terms in the equation and can thus be used as an indication for the quality of the approximation. As a function of the number of iterations k in the Lanczos process, it should exhibit exponential decay as k increases, indicating a smaller error as the approximation quality improves.

PVL does not necessarily preserve asympotic stability of the model during the reduction process even if the original model is SISO LTI and stable [9, 76]. This can be confirmed from the eigenvalues of matrix T_k. However, due to the iterative nature of the process it is possible to inspect the real parts of the eigenvalues of T_k at each step during the algorithm and stop the process if stability is lost, assuming that there is a chance a stable reduced model could be found. Furthermore, an extension to the PVL method named PVLπ has been developed that attempts to stabilize an unstable PVL-reduced model [76].

Reducing time varying systems with moment matching method has also been studied. The LTV system has to be periodic for the model order reduction to be possible [77, 78].

Aside from linearization approaches, moment matching methods are not very advanced with regards to reducing nonlinear systems. The difficulty stems from the transfer function not being clearly defined for nonlinear systems. However, it is possible to determine moments for the nonlinear functions of a given system and approximate those with reduced models, resulting in a group of Padé approximations with an equal dimension to the original nonlinear system. The problem is that, as stated in the linear case, using a single larger Padé approximation is more efficient than having several small Padé approximations. Bilinear MIMO systems have been reduced in [24] using a block Krylov approach

and in [9] using Carleman bilinearization and a multidimensional Laplace transform.

2.5 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) is a model order reduction method that is in some sources used synonymously with Principal Component Analysis (PCA) [79, 80], Kosambi-Karhunen-Loeve Transformation (KLT) [81] or Singular Value Decomposition (SVD) [82]. The confusion around naming the method stems from it being highly flexible with regards to finding the projection space. Lumley originally introduced POD as a model order reduction method for studying the structure of turbulent flows [83]. Here the term POD actually refers to an empirical three step process that seeks to approximate the targeted behavior and dynamics of the system rather than all possible states. The method trades mathematical rigour for flexibility as it is applicable to nonlinear and time-varying systems, but gives no guarantee of stability, controllability or observability of the system.

A time-domain derivation of POD will be presented here. In essence, the process begins with the *method of snapshots* outlined by Sirovich [58], which includes simulation and sampling of the full dynamics of the system that is to be reduced, followed by SVD of these snapshots, and finally Galerkin projection of the original full dimension system to a mean squared error optimal reduced orthonormal subspace. The singular vectors that span this subspace are often called *POD modes*. Since the full system has to be simulated before POD can be applied, it is an a posteriori method and does not rely on any other prior knowledge of the system [84]. POD can then be summarised as a method for approximating the system dynamics in a lower dimension than the original system. This can be contrasted with other order reduction methods of control theory that are mostly concerned with maintaining the input-output relation of the system, such as BT.

POD is directly applicable to time-invariant nonlinear systems of Equation (2). The derivation here will be for the time-invariant case for simplicity. However, it is mandatory to separate the linear coefficients, nonlinear function and external inputs of the system that were previously grouped in $f(\boldsymbol{x}(t), \boldsymbol{u}(t))$, so that the state equation

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + F(\boldsymbol{x}(t)) + B\boldsymbol{u}(t) \tag{65}$$

contains the linear coefficients in the state matrix A and the nonlinear functions in the vector F and linear inputs grouped in the vector B. The same formalism can be applied to the output equation and will not be explicitly written for the rest of the section.

In order to obtain the reduced order model from POD the full dimensional system has to be simulated first. The purpose is to sample time points from the solutions of the simulation that the reduced model will approximate. This method of snapshots has two main reasons; the full solution space contains a lot of redundant information and processing a smaller data matrix (the snapshots) is efficient. Moreover, it gives the modeler

control over the desired features of the reduced model. Consequently, generation of the snapshots for computing the POD modes is a crucial part of the POD process, since the reduced order model will be built to approximate the dynamics contained in them.

A subsequently arising question considering the generation of the snapshots is the so called excitement function used to produce the sampled trajectories. This is highly dependent on the application of the reduced order model. In case the approximation should be accurate for a variety excitation, it is possible to use any different input functions and catenate the resulting data matrices together before selecting snapshots. An additional benefit achieved from catenation is that each input can be emphasized as wished by manipulating the length of the matrix. However, a very specific reduced model is obtained by using perhaps only one input function. Another consideration is the simulation time. If the original model exhibits a wide range of dynamics as time increases, then a longer simulation period is warranted.

After the original system has been simulated for data collection, there are several methods for finding the snapshots that minimize the error between the POD-solution and original trajectories. To write the above in mathematical notation, simulating the system grants a matrix $X \in \mathbb{R}^{n \times t}$, where n is the number of state variables or dimensions of the system and t is the number of time steps at which the system was solved. In other words, X contains the time trajectories of the solutions to the system from initial time t_0 to t_{end}. As per the method of snapshots, X is then sampled at every i:th index to obtain the matrix of snapshots $S = [\boldsymbol{x}_1, \boldsymbol{x}_i, ..., \boldsymbol{x}_s] \in \mathbb{R}^{n \times s}$, where s is the total number of snapshots.

Oftentimes, a very good approximation is obtained by simply choosing the snapshots with a constant interval. A suitable interval can be selected for example by boosting the information content of each snapshot by evaluating the eigenvalues of a snapshot covariance matrix [85, 86]. There, the focus is on maximizing the magnitude of the smallest eigenvalue. Moreover, the approximation quality can be increased by adding extra snapshots between the intervals [87]. Finally, a selection method that that uses a logarithmic interval by either emphasizing the early or late values has been suggested [85]. After the snapshots are obtained, in certain scenarios it might be beneficial to center them with a normalizing process, for which an algorithm is given in [88].

The snapshots will be used to compute a set of orthonormal basis vectors that span a reduction subspace for POD. Applying SVD to the snapshots S results in the singular values of the system as well as the singular vectors of the system. The singular vectors are a set of linearly independent orthonormal vectors that span the space in which S is observed. This means that the values of the snapshots, or the dynamics of the system, can be reconstructed from these basis vectors. SVD of the snapshot gives

$$S = V \Sigma D^* \tag{66}$$

where $\Sigma \in \mathbb{R}^{n\times i}$ is a diagonal matrix of singular values that are real and positive, and both $V \in \mathbb{C}^{n\times n}$ and $D \in \mathbb{C}^{i\times i}$ are square matrices whose columns are the orthonormal left and right singular vectors $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_n\}$ and $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_i\}$ of S, respectively. The name D is used here by convention and the matrix is not to be confused with the feedthrough matrix of the state space equation. Additionally, both V and D are unitary so that $VV^* = V^*V = I$. The singular values in Σ are ordered by their magnitude and can be interpreted as a measure of the importance of the corresponding singular vectors.

Next, from the result of SVD a set of POD modes that can best approximate the full system will be selected. Since the purpose is to be able to reproduce the most important dynamics of the system in a lower dimension, a subspace that is spanned by the most important singular vectors of the snapshots should theoretically be able to generate most of the states and outputs of the system. Because the singular values are already ordered by magnitude, and that magnitude is used as a metric for importance, the first k columns of V can then be chosen as a basis for a reduced subspace of the original system, so that they form a matrix $V_k = [\boldsymbol{v}_1, ...\boldsymbol{v}_k] \in \mathbb{Z}^{n\times k}$. The basis is optimal for reconstructing the snapshots of the original system, since it minimizes the mean squared error between the original system outputs y and the reduced solution [48]. The minimized optimization problem is

$$\min_{rank\{V_k\}=k} \sum_{j=1}^{n} \left\| \boldsymbol{y}_j - V_k V_k^* \boldsymbol{y}_j \right\|^2, \tag{67}$$

where $V_k^* V_k = I$ and this is solved in the POD process [89].

The accuracy of the reduced subspace of dimension k can be estimated from the decay of the singular values of the snapshots. If the singular values σ are interpreted as the "energy" the system contains, the energy contained in the k:th dimensional approximation is

$$e = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{n} \sigma_i} \tag{68}$$

where the singular values σ are sorted in a descending order of magnitude as they are when SVD is applied. Typically the decay of the singular values is exponential, in which case only a small portion of them is required to reach $e > 0.99$. SVD makes the assumption that the underlying data has a Gaussian distribution. If the singular values do not decay exponentially, this approximation might not hold, which tends to indicate poor results from POD model reduction. This problem will be elaborated on in the Discussion chapter.

The next step is applying Galerkin-projection to project the full dimensional system on the reduced subspace spanned by columns of V_k. With this linear transformation the matrices of the system are rescaled and shaped to smaller dimension. Since the projection basis is constructed from snapshots of the solutions to the system, the new reduced system

should maintain this quality. Consider a change of basis for the state vector $\boldsymbol{x}$ to a k dimensional subspace so that $\boldsymbol{x}_r = V_k^* \boldsymbol{x}$. The reduced system is then

$$\begin{aligned} \dot{\boldsymbol{x}}_k(t) &= \underbrace{V_k^* A V_k}_{k\times k} \boldsymbol{x}_k(t) + V_k^* \underbrace{F(V_k \boldsymbol{x}_r(t))}_{n\times 1} + \underbrace{V_k^* B}_{k\times r} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \underbrace{C V_k}_{q\times k} \boldsymbol{x}_k(t) + D\boldsymbol{u}(t) \end{aligned} \tag{69}$$

and due to the smaller size of the linear matrices, a computational and mathematical advantage is gained. The reduced matrices of the linear terms $V_k^* A V_k = A_k$, $V_k^* B = B_k$ and $C V_k = C_k$ can be precomputed in the offline stage of the simulation, which directly results in improved simulation time on the online stage since a smaller amount of simpler differential equations has to be solved. If the input matrix does not depend on time it can be reduced by precomputation as well. However, the nonlinear term in $V_k^* F(V_k \boldsymbol{x}_r(t))$ needs to be evaluated in the original dimension at each time step and in this sense the complexity of the system still depends on the original dimension n (see Subsection 2.1.3 Subspace Projection). Depending on the structure of the model, this format might actually increase simulation time if the complexity of the nonlinear term is not further reduced. After simulating the reduced system, an approximation of the original system is obtained by transforming the solutions back to the original space with $\tilde{X} \approx V_k X_r$, where X_r is the matrix of solutions at every time step the system was solved.

POD does not consider stability of the reduced order model [84, 90, 91]. That said, it does not necessarily turn a stable system unstable either. Moreover, POD approaches to enforce stability have been developed and they are applicable to linear and nonlinear systems [92]. In [92] a method for keeping stable continuous-time models bounded in asymptotically stable range during reduction has been extended to the discrete case that has been discussed in this thesis. However, forcing such boundary conditions comes at the cost of accuracy of the reduced model.

There are no constraints to applying POD to time-varying systems, after all the empirical basis contains information of the evolution of the system. Moreover, due to the empirical nature of the method obtaining the basis for time-varying systems is no different than for time-invariant systems. However, LTV systems have the same difficulty regarding reduction as the nonlinear term, as it is not possible to reduce them entirely on the offline stage. As such, much of the discussion regarding subspace projection in Subsection 2.1.3 and BT of time-varying systems applies here.

An interpolation approach can be used to reduced the complexity of the nonlinear term and remove the dependency on the original dimension. The following section will introduce the Discrete Empirical Interpolation Method that does precisely this. There, POD will be used on the nonlinear part of the model separately.

2.6 Discrete Empirical Interpolation Method

An empirical interpolation method for reducing the complexity of nonlinear functions was first proposed by Barrault, Maday, Nguyen and Patera [93] in 2004. The discrete version Discrete Empirical Interpolation Method (DEIM) was then introduced by Sorensen and Chaturantabut in 2010 [89]. The previous five years have seen DEIM developed further with localized, adaptive and stability conserving variants [25, 94, 95]. It is a method that complements POD by reducing the nonlinear term so that together the two arrive at a reduced model which no longer depends on the original dimension of the system. Alternatively, DEIM can be used for standalone reduction of nonlinear functions.

DEIM is a projection based method, like POD and BT, combined with an interpolation approach. Given a system in the format of Equation (65), the goal is to find an approximation for $F(\boldsymbol{x}(t))$ that does not depend on the original dimension n. From POD and BT the reduced format of the nonlinear term $V_k^* F(V_k \boldsymbol{x}_k(t))$ was obtained, which does not allow precomputation in the offline stage since $F(\boldsymbol{x}(t))$ is not defined in the reduced space. Consequently, $V_k \boldsymbol{x}_k$ has to be computed during simulation run-time in order to evaluate the function vector $F(\boldsymbol{x}(t))$ for subspace projection. A solution proposed with DEIM is to reduce the complexity of the nonlinear function by interpolating at specific indices of the function vector and projecting the result onto a basis that spans the space of the output values of the nonlinear function [89]. The basis vectors are the columns of a matrix denoted by U and the following will outline the calculations required for finding them along with the interpolation indices.

The first step in DEIM is calculating a projection subspace spanned by linearly independent orthonormal vectors $U_m = [\boldsymbol{u}_1, ..., \boldsymbol{u}_m] \in \mathbb{R}^{n \times m}$ also called DEIM modes. The process begins with applying POD to values of the nonlinear function at the solutions of the full-order system. In summary, as the full system is simulated for $[t_0, t_{end}]$, the outputs of $F(\boldsymbol{x}(t))$ are saved at every t to matrix $X_f = [F(\boldsymbol{x}(t_0)), F(\boldsymbol{x}(t_2)), ..., F(\boldsymbol{x}(t_i))]$. The values can be collected during simulation of the full system for POD, so no unnecessary computational burden is introduced. Afterwards, SVD is applied to snapshots of X_f in order to obtain the first $m \leq n$ left singular vectors $\{\boldsymbol{u}_1, ..., \boldsymbol{u}_m\} = U_m$ from $U \coloneqq V_{SVD}$. Again as with POD, the approximation error of the reduced system can be estimated from the magnitude of singular values [96]. Moreover, if DEIM is used in conjunction with POD, calculating separate subspaces for POD (spanned by columns of V_k) and DEIM (spanned by columns of U_m) has the additional benefit that an individual number of modes can be chosen for approximating the linear (POD) and nonlinear (DEIM) terms so that k does not necessarily equal m and $k, m \leq n$.

Once U_m is defined, it is possible to derive a reduced order model of the nonlinear function by reducing the number of nonlinear component functions in $F(\boldsymbol{x})$. Recall that columns of U_m span the subspace where (most of) the output values of the vector of

nonlinear functions $F(\boldsymbol{x}(t))$ exist. Assuming for a moment an m dimensional vector of functions selected from $F(x)$ exists in coefficient vector $f(x)$ and is available, an approximation of the original outputs is obtained by matrix multiplication

$$F(x) \approx U_m f(x), \tag{70}$$

which is a linear projection of the coefficients to the space spanned by columns of U_m. If the above is interpreted as an equation group of m equations and since $F(x)$ and U_m are known, it stands to reason that this equation could be used to define the coefficient vector $f(x)$. Commonly matrix equations of type $M\boldsymbol{\alpha} = \boldsymbol{\beta}$ with unknown $\boldsymbol{\alpha}$ are solved with the least squares method by $\boldsymbol{\alpha} = M^{-1}\boldsymbol{\beta}$. However, there are more equations in the rows of U than there are unknowns in $f(x)$, hence the system is overdetermined and cannot be uniquely solved. The least squares solution to overdetermined systems is

$$\begin{aligned} M^T M\boldsymbol{\alpha} &= M^T\boldsymbol{\beta} \\ \boldsymbol{\alpha} &= (M^T M)^{-1} M^T\boldsymbol{\beta}, \end{aligned} \tag{71}$$

that would lead to

$$f(x) = (U_m^T U)^{-1} U^T F(x), \tag{72}$$

where the multiplication between U^T and $F(x)$ results in a dependency on the original dimension n in the online phase of solving the system. This means that further processing is required to calculate a reduced nonlinear function vector.

The reduction method proceeds by selecting m rows from Equation (70), so that the system is no longer overdetermined. Let a matrix for extracting the rows be

$$P = [\boldsymbol{e}_{\wp_1}, ..., \boldsymbol{e}_{\wp_m}] \in \mathbb{R}^{n \times m}, \tag{73}$$

where $\boldsymbol{e}_{\wp_i}$ is the $\wp_i$th column of an $n \times n$ identity matrix I_n for $i = 1, ..., m$ [89]. Obtaining P and the interpolation indices $\{\wp_1, ..., \wp_m\}$ is an integral part of the DEIM procedure. As such, the subspace U_m is employed in the derivation of the indices as described in Algorithm 2 by maximizing variation in the result of multiplication with P.

In Algorithm 2 the input is a set of n dimensional column vectors, which now is U_m. The function $\max$ returns in ρ the largest value of the input and in $\wp$ the first corresponding index. In the initialization phase of steps 1 and 2, ρ and $\wp_1$ are obtained from the absolute values of the first column in U_m by $\max|\boldsymbol{u}_1|$, after which a temporary matrix U is initialized with $\boldsymbol{u}_1$, matrix P with the $\wp$th column of I, $\boldsymbol{e}_{\wp_1}$ and index vector $\boldsymbol{\wp}$ with $\wp_1$. Then, the m columns of U_m are processed. In step 4 a vector $\boldsymbol{c}$ of weights is determined from the columns processed so far. In step 5 it is employed for calculating a component-wise difference $\boldsymbol{r}$ between the current and processed entries. Step 6 then finds the magnitude and index of the most differing value. Before the loop restarts, U is ap-

Algorithm 2 Discrete Empirical Interpolation Method

INPUT: $\{\boldsymbol{u}_l\}_{l=1}^{m} \subset \mathbb{R}^n$ linearly independent
OUTPUT: $\boldsymbol{\wp} = [\wp_1, ..., \wp_m]^T \in \mathbb{R}^m$
1: $[|\rho|, \wp_1] = \max |u_1|$
2: $U = [u_1], P = [e_{\wp_1}], \overrightarrow{\wp} = [\wp_1]$
3: **for** $l = 2$ to m **do**
4: Solve $U\boldsymbol{c} = \boldsymbol{u}_l$ for c
5: $\boldsymbol{r} = \boldsymbol{u}_l - U\boldsymbol{c}$
6: $[|\rho|, \wp_l] = \max |r|$
7: $U \leftarrow \begin{bmatrix} U & u_l \end{bmatrix}, P \leftarrow \begin{bmatrix} P & e_{\wp_l} \end{bmatrix}, \boldsymbol{\wp} \leftarrow \begin{bmatrix} \boldsymbol{\wp} \\ \wp_l \end{bmatrix}$
8: **end for**

pended with the currently processed vector, P with the $\wp$th column of the identity matrix, and the index vector with $\wp$. According to this process, P will contain interpolation indices that select the most essential rows from U_m. It is also seen that $P^T U$ is nonsingular. [89]

Once P is obtained, left multiplication with P^T gives

$$P^T F(x) = (P^T U_m) f(x), \tag{74}$$

that can be solved assuming $P^T U_m$ is nonsingular. $P^T F(x)$ results in an $m \times 1$ vector and $P^T U_m$ in an $m \times m$ matrix, so that as a result of reducing the number of components in U_m and $F(x)$, the equation group is not overdetermined anymore. Although $F(x)$ is a vector of functions, by still assuming the above equation is solved for $f(x)$, the m coefficients become

$$f(x) = (P^T U_m)^{-1} P^T F(x), \tag{75}$$

where all terms on the right hand side are known. An important point to consider is that computing $P^T F(x)$ is interpreted as extracting rows of $F(x)$ at the interpolation points in P^T, in the current case leading to the reduced function $F_m(x) \coloneqq P^T F(x) \in \mathbb{R}^{m \times 1}$. Additionally, the extracted format $F_m(x)$ is defined in the offline stage of the simulation, which removes the dependency of the nonlinear term on n, further reducing computational expenses in the simulation phase. Finally, an approximation of the original nonlinear outputs at time t is obtained by projecting the output of the reduced function to the subspace spanned by columns of U_m as

$$F(x(t)) \approx U_m f(x) = U_m (P^T U_m)^{-1} F_m(x(t)), \tag{76}$$

and this format is employed later in the process to complete the model reduction.

Now that n and m dimensional approximations for the nonlinearity are established via interpolation, the reduction itself can be finalized by utilizing the POD basis V_k. This step

is necessary in order to process the system matrices in the same subspace and to make matrix dimensions of the model agree. Define the k dimensional reduced nonlinearity

$$F_k(\boldsymbol{x}_k(t)) = \underbrace{V_k^T U_m (P^T U_m)^{-1}}_{k \times m} \underbrace{F_m(V_k \boldsymbol{x}_k(t))}_{m \times 1}, \tag{77}$$

where $V_k^T U_m (P^T U_m)^{-1} = N_k$ can be precomputed in the offline stage since it does not depend on time. This equation can be interpreted as projection of the m point interpolation approximation of the nonlinearity to the k dimensional POD subspace.

In the special case that the function vector $F(\boldsymbol{x})$ evaluates the input vector $\boldsymbol{x}$ component wise so that the ith function in $F(\boldsymbol{x})$ depends only on the ith component of $\boldsymbol{x}$, the evaluation of the nonlinear term simplifies even further. The reason is that $F_m(\boldsymbol{x})$ will only need and use m components of $\boldsymbol{x}$. Although during simulation $\boldsymbol{x}_k$ still has to be transformed back to the original space, it becomes possible to precompute an extracted version of $V_k^P = P^T V_k$ that only calculates the required m components, similarly to how rows were selected from the function vector by P. V_k^P does not depend on n and thus has reduced complexity. Evaluation of the function vector then becomes

$$F_m(V_k^P \boldsymbol{x}_k(t)), \tag{78}$$

where $\boldsymbol{x}^P = V_k^P \boldsymbol{x}_k(t)$ only contains the m components the DEIM reduced function requires for evaluation and the unnecessary components are not computed at all so that computation speed is improved. However, in the general case this cannot be assumed feasible. Given that usually $m < n$, the sparse evaluation of F_m can be efficiently implemented with a compressed sparse roi data structure for indexing vector $\boldsymbol{x}$. [89]

DEIM can be used alone without multiplying the approximation of the nonlinear function with the POD basis. In this case, the result is just an n dimensional m point interpolatory approximation of the original system. When combined with POD, a reduced order model that does not depend on n is obtained. If desired, a separate W_p basis (comparable to N_k) for p interpolation points is first calculated for the output function $g(t, \boldsymbol{x}(t), \boldsymbol{u}(t))$. However, depending on the size and structure of the outputs of the system, this might not bring considerable computational savings. Recalling the POD result of Equation (69), the reduced form of the nonlinear system in Equation (2) is then

$$\begin{aligned} \dot{\boldsymbol{x}}_k(t) &= A_k(t)\boldsymbol{x}_k(t) + N_k F_m(V_k \boldsymbol{x}_k(t)) + B_k(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= C_k(t)\boldsymbol{x}(t) + W_p G_p(V_k \boldsymbol{x}_k(t)) + D(t)\boldsymbol{u}(t), \end{aligned} \tag{79}$$

Due to the empirical POD basis used in DEIM there are no guarantees of persevering stability in the reduced model. Additionally, the interpolation method poses an additional challenge for applying methods that guarantee POD stability to DEIM. However, a sta-

bility persevering reduction method based on non-negative matrix transformations called Non-Negative Discrete Empirical Interpolation Method (NNDEIM) exists and has been shown to be effective for model reduction of networks of neurons modeled with the cable-equation [95].

Although DEIM is a relatively recently developed algorithm, a modification for time varying systems has already been presented in [25, 95]. Both approaches are based on constructing multiple interpolation function sets and choosing the best approximation in the online phase of the simulation with efficient lookup methods.

3. CASE STUDY: SYNAPTIC PLASTICITY MODEL

In order to test the effectiveness of model reduction in biological context a mathematical model of signaling pathways in striatal synaptic plasticity by Kim et al. [3] was chosen for a case study. The model is specific for the basal ganglia area of the brain and it explains how certain molecules in intercellular information transfer points of neurons, synapses, are responsible for plasticity, which is presumably a prerequisite for learning, in the brain. It is a biophysical model that is based on experimental data, in contrast to being a phenomenological model. Originally the model was employed in studying the effects of different stimuli to the synapse and how they could direct plasticity. Additionally, the predictions from the model have been verified experimentally and the model itself is based on validated experimental data, which contributed to the decision of choosing this specific model.

This striatal synaptic plasticity model is based on chemical reactions of the molecules in the synapse. The stoichiometric equations obey the law of mass action, which leads to a deterministic system of ordinary differential equations. For the following analysis five biologically interesting species included in the model were chosen as outputs of the system to studied in more detail. These were 2-arachidonoylglycerol (2Ag), calcium (Ca), diacylglycerol (DAG), G protein with α, $beta$ and γ subunits (Gabg) and phospholipase C (PLC). Their behavior is significant as these substances can connect the current model to a larger, even more detailed model and they are known to be active influencers in LTP and LTD [4].

For this thesis the model of Kim et al. [3] was implemented based on the equations given in the original publication. This implementation of the model contains $n = 44$ ordinary differential equations, and further details regarding the model and the species it contains can be found in Appendix A.1. The output species 2AG, Ca, DAG, Gabg and PLC are modeled by equations x_1, x_2, x_{21}, x_{25} and x_{43}, respectively. Compared to the original model, the differential equation for external glutamate (Glu) was omitted in favor of having Glu as a purely external time-dependent stimulus, which is also justified by the fact that the model is tested as stand-alone system instead of being coupled to a larger system. In addition to Glu, the implementation contains external calcium stimulus as another time-dependent variable, which appears also to the power of two in some equations. The end result is a nonlinear control system of the format shown in Equation (2), and additionally the system has bilinear characteristics. The five first equations of the model

are

$$\begin{aligned}
\dot{x_1} &= k_{prodAG_c} * x_4 - k_{degAG_f} * x_1 \\
\dot{x_2} &= k_{PMCA_c} * x_{15} + k_{NCX_c} * x_{11} - k_{Leak_f} * x_2 * x_{36} + k_{Leak_b} * x_{10} \\
\dot{x_3} &= k_{buffer_f} * Ca_{post}(t) * x_{19} - k_{buffer_b} * x_3 \\
\dot{x_4} &= k_{prodAG_f} * x_{21} * x_7 - k_{prodAG_b} * x_4 - k_{prodAG_c} * x_4 \\
\dot{x_5} &= k_{DAG_3c} * x_8 - k_{DAG_4f} * x_5,
\end{aligned} \tag{80}$$

and they illustrate the nonlinearity of the system in equation of $\dot{x_2}$, where two state variables x_2 and x_{36} are multiplied together and the time-dependence of the system in the equation of $\dot{x_3}$ where a state variable is multiplied with calcium stimulus.

Model order reduction was performed with the Proper Orthogonal Decomposition and Discrete Empirical Interpolation Method (POD+DEIM) approach because of its applicability to nonlinear systems without having to linearize the original model. Additionally, the method is readily applicable to multiple-input multiple-output systems.. POD+DEIM requires that the linear and nonlinear terms are separated as in Equation (65) in order to reach the format of Equation 79. However, time-dependent variables posed an additional challenge. Without further processing of the state matrix, the reduction of the linearity could not have been precomputed because with the POD+DEIM method, the time-varying parameters need to be updated in the original dimension. (In effect, this requires the system to be in the full order format to calculate the updates of the terms that are multiplied with any time-dependent variables.) With these issues acknowledged, the equation system was written as

$$\frac{d\boldsymbol{x}(t)}{dt} = (A_0 + A_1 * Ca(t) + A_2 * Ca(t)^2 + A_3 * Glu(t))\boldsymbol{x}(t) + F(\boldsymbol{x}(t)) + B\boldsymbol{u}(t), \tag{81}$$

where the bilinear characteristics of the system $A(t) = A_0 + A_1 Ca(t) + A_2 Ca(t)^2 + A_3 Glu(t)$ are observable. Here $A_i \in \mathbb{R}^{n \times n}$ for $i = 0, 1, 2, 3$. The matrices $A_1, ..., A_3$ are sparse coefficient matrices for each of the time varying variables and were separated from A_0 so that precomputing their order reduction in the offline stage would become possible. Matrix $B \in \mathbb{R}^{n \times r}$ contains the coefficients for each input in rows of $\boldsymbol{u} \in \mathbb{R}^{r \times 1}$ and $\boldsymbol{u}(t)$ in this system only contains Glu, so $r = 1$. Due to the equation of x_{32} B is nonzero and the system is called *forced* in control theory terms. This is because Glu acts as an input to the system so that in some state equations it is not directly multiplied with state variables.

Recalling the ultimate reduced format of the nonlinear system, the reduced form of the Kim model obtained by projecting $\boldsymbol{x}$ onto V_k as $V_k^T \boldsymbol{x} = \boldsymbol{x}_k$ and thus the equation was written as

$$\begin{aligned}
\frac{d\boldsymbol{x}_k(t)}{dt} =& {V_k}^T (A_0 + A_1 * Ca(t) + A_2 * Ca(t)^2 + A_3 * Glu(t)) V_k \boldsymbol{x}_k(t) \\
&+ NF_m(V_k \boldsymbol{x}_k(t)) + {V_k}^T B\boldsymbol{u}(t)
\end{aligned} \tag{82}$$

where the reduced linear part by commutativity of matrices becomes

$$\begin{aligned} A_k =&(V_k{}^T A_0 V_k + V_k{}^T A_1 * Ca(t) V_k + V_k{}^T A_2 * Ca(t)^2 V_k + V_k{}^T A_3 * Glu(t) V_k)\boldsymbol{x}(t) \\ =&(A_{k0} + A_{k1} * Ca(t) + A_{k2} * Ca(t)^2 + A_{k3} * Glu(t))\boldsymbol{x}(t) \end{aligned} \tag{83}$$

where columns of V_k span the k-dimensional subspace obtained from POD, $N = U_m(P^T U_m)^{-1}$ where U_m and P are calculated during DEIM and F_m has m rows extracted from it by P^T. This allowed the precomputation of the reduced $k \times k$ square matrices $A_{k0} = V_k{}^T A_0 V_k$, $A_{k1} = V_k{}^T A_1 V_k$, $A_{k2} = V_k{}^T A_2 V$, $A_{k3} = V_k{}^T A_3 V_k$ and the $k \times r$ matrix $V_k{}^T B = B_k$, so that much of the computational load in the simulation phase was distributed to the offline stage. The only dependency of the system on the original dimension n was then the input to the nonlinear term $V_k \boldsymbol{x}_k$ and the reduced form allowed testing of different values of k and m effortlessly.

The time-varying variables depicting external stimulus to the system Ca and Glu were implemented as sine wave signals that had amplitudes in the physiological range. Stimulus was applied during time $t = [5, 10]$ whereas otherwise it was zero, so that

$$u(t) = \begin{cases} \frac{A}{2} + \frac{A}{2}\sin(-\pi * t(n)), & 5 \leq t \leq 10 \\ 0, & otherwise \end{cases}, \tag{84}$$

where $u(t)$ is the output amplitude at time t and A is the desired physiologically realistic maximum stimulus amplitude.

4. RESULTS

This chapter details how the effectiveness of model reduction was evaluated and what were the results of model reduction for the selected case study. First, the expected results are compared to empirical findings. Afterwards, several variations of the reduction and their results are presented. The code used for producing the results presented in this chapter is available in Appendix A.2.

4.1 Finding the Optimal Dimensions

To predict the successfulness of the reduction in advance the SVD of the solutions at each time step (POD SVs) and the SVD of the values of the nonlinear term (DEIM SVs) were calculated by simulating the system for $t = [0, 10000]$ seconds with the aforementioned stimuli. A constant snapshot interval of 5 was used, meaning that every fifth value of the solutions and nonlinear term values was included in the calculation of the SVD. Figure 4.1 shows the singular values obtained from SVD for both POD (on the left) and DEIM (right). The singular values are ordered and plotted on a base 10 logarithmic scale to better illustrate their behavior, since the difference between the largest and smallest value was orders of magnitude. Recall that an exponential decay on a linear scale implies that the system is possibly suitable for SVD based reduction methods. Here both POD and DEIM SVs were found to exhibit this radical decay, with the DEIM scale indicating 5 to 13 modes (dimensions) and the POD scale 4 to 10 modes as a sufficient selection to capture most of the dynamics of the system.

In order to compare the original model versus POD+DEIM reduced models the simulation speed and error of several subspace dimensions were measured. The original and reduced ordinary differential equation systems were simulated in Matlab for time span $t = [0, 10000]$ using the variable time step ODE15S solver for stiff differential equations. For each POD dimension $k = 2 : 2 : 40$ (Matlab notation), DEIM dimension $m = 5 : 5 : 30$ reduced models were calculated. For each combination, 20 simulations were performed and their average computation times and system solutions at each time step were stored. Before calculating the mean of the solutions from several simulation executions, the results obtained from ODE15S had to be interpolated into matching lengths. Since the solver uses a variable time step, the number and location of time steps the solution is evaluated at is not constant. The Matlab function DEVAL was employed for interpolation into time points $t = 0 : 10000$. Additionally, before the solutions of the

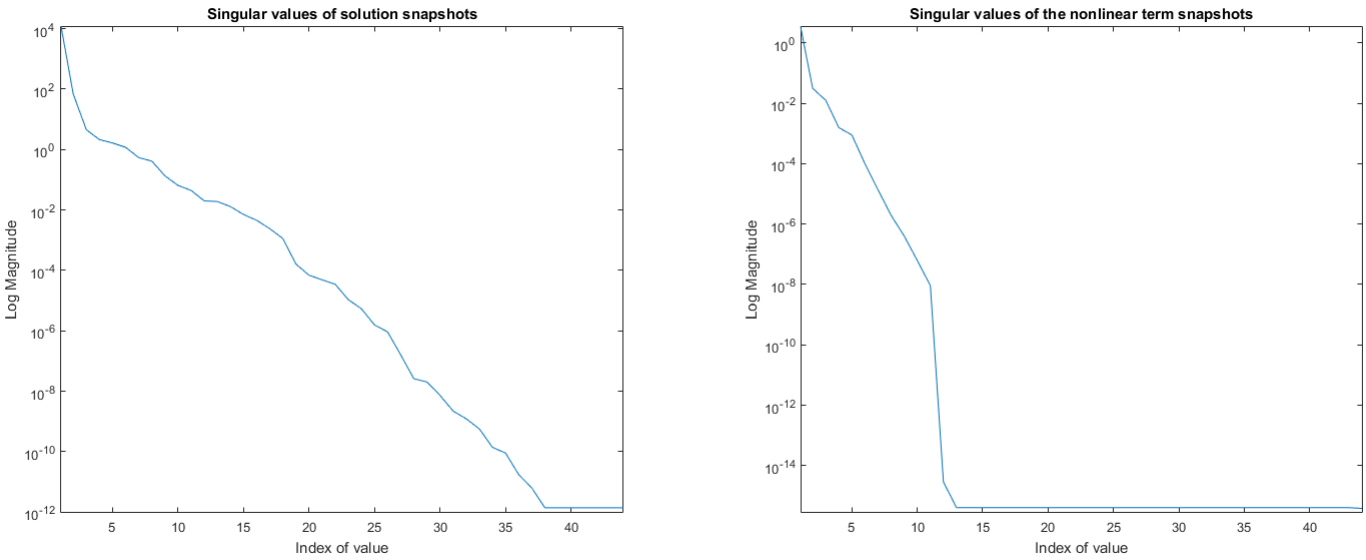

Figure 4.1: Decay of the singular values of solution snapshots of the system (left) and values of the nonlinear term (right). The y-axis uses a base 10 logarithmic scale to indicate the magnitude of each value on the x-axis.

reduced model were interpolated, they were transformed back into the original dimension of the system by $\tilde{Y} = V_k Y_{reduced}$, where $Y_{reduced}$ is the matrix of solutions obtained from the solver and columns of V_k span the subspace to which the model was projected. By doing this the approximations of all system variables given by the reduced model could be compared to the solutions calculated from the full dimensional model.

Figure 4.2 shows the root mean square (RMS) error between the full dimension model and different reduced models averaged from 20 simulations of each POD+DEIM combination. The y-axis contains the error values on a logarithmic scale, while x-axis indicates the number of POD dimensions. Each line in the plot corresponds to a DEIM dimension. For example the blue line with the highest error values has been calculated for all POD dimensions while DEIM dimension was locked to five. RMS error was calculated by

$$e_{RMS} = \sqrt{\frac{1}{k}\sum_{n=1}^{k}(Y - \tilde{Y})^2} \tag{85}$$

where Y is the interpolated matrix of solutions of the original system and k is the number of elements in the matrices. The approximation yielded by the reduced model is subtracted from the original solutions and the values in the difference matrix are squared. The mean is then calculated by squaring and summing the elements and dividing by the number of elements. Finally, the square root of this value is taken to display the error. It should be emphasised that this calculation is only possible after the simulation results of the reduced model are transformed back into the original dimension and interpolated to

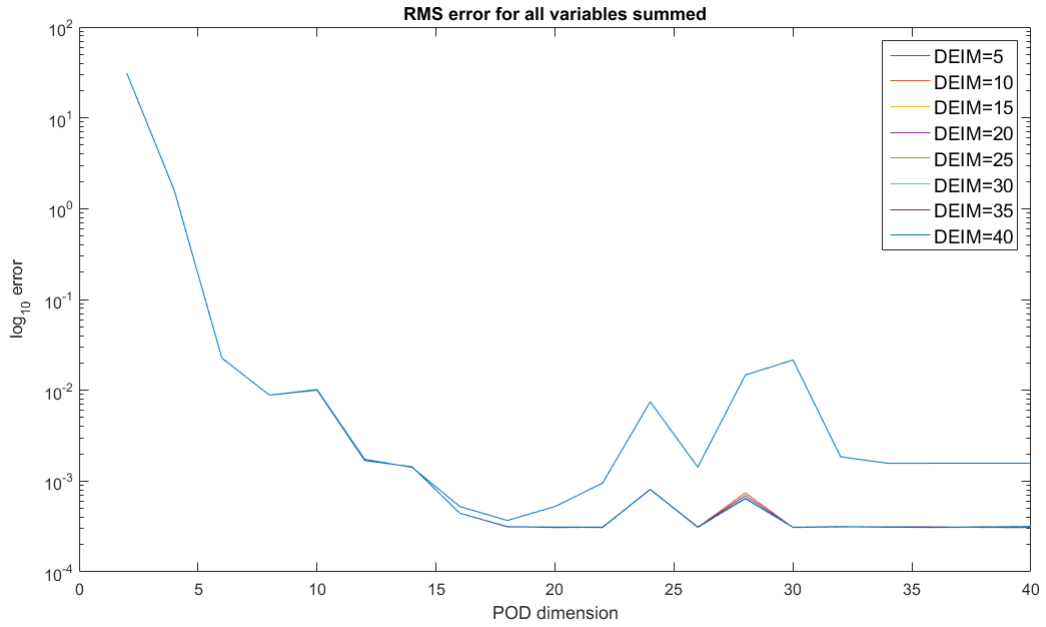

Figure 4.2: Root mean square error of the solutions at each time step. X-axis shows POD dimension and y-axis the error on logarithmic scale. Each distinctly colored plot corresponds to different dimensions used in DEIM.

matching size, otherwise the reduced model describes incomparable system and additionally the dimensions of the matrices will not agree.

From Figure 4.2 it is seen that regardless of the DEIM mode, or the nonlinear dimensionality, the error decays rapidly until POD dimension 15 is reached. This suggests that more than 15 POD dimensions is not beneficial, since the accuracy will not improve with further dimensions. Until 15 POD dimensions, the error diminishes exponentially. The slight increase in error between 25 and 30 POD modes is inconsistent with the expected behavior, and suggests that these dimensions introduce something to the system that increases numerical inaccuracy of the employed solver. This could be stiffness that is resolved again when more equations are added. Depending on the application, as little as five to ten dimensions could be sufficient for simulating this model while keeping the error tolerable. Moreover the RMS error is seen to not depend radically on the DEIM dimension. Increasing the DEIM modes from 5 to 10 reduces the error if the POD mode is already over 15. This suggests that the linear part of the model that is reduced with POD is dominant in terms of approximation error and that the interpolation approach to reducing the nonlinear complixity is effective.

Figure 4.3 displays the computational advantage gained from the reduced model in terms of simulation speed. The tests were performed on a Windows 7 desktop PC with Intel Core i5-4690K processor clocked at 3.50GHz and 8GB memory. Simulation times were obtained from Matlab using the TIC and TOC functions to record the time to execute ODE15S. In the figure, the simulation time of the original full dimension model is plotted

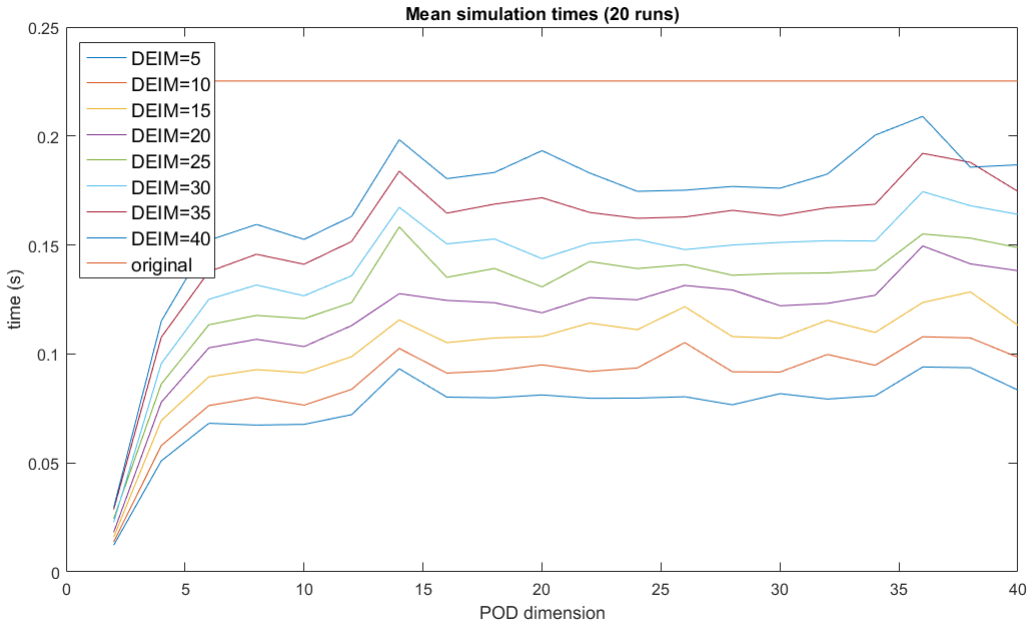

Figure 4.3: Mean simulation times of 20 executions of each POD+DEIM reduced model compared to the original model, plotted as a straight red line. Y-axis shows the simulation time in seconds, and x-axis shows the POD dimension. Each colored plot corresponds to a DEIM dimension. Simulation time interval was $t = [0, 10000]$.

as a straight red line. This simulation duration for solving the original system for time interval $t = 0 : 10000$ averaged to 0.23 seconds for 20 simulations. From Figure 4.3 it is seen that the simulation time is approximately halved by using a 20 DEIM modes, which also corresponds to roughly halving the dimension of the nonlinear term. Each time five dimensions are removed, the simulation speed is increased by between 0.01 and 0.02 seconds, starting from 40 DEIM modes. The simulation times seem to depend on POD reduction only when less than 15 modes are chosen. In summary, this suggests that for this specific model, the nonlinear term is largest computational burden, since reducing it has the largest effect on simulation times. However, if error is not of concern, the simulation can be made very fast by using less than 5 POD modes.

4.2 Analysis of the Dynamics of the Reduced Model

With the information obtained from Figure 4.2 and Figure 4.3 a reduced model was constructed and the resulting approximations of solutions were compared to the solutions calculated from the original model. The POD and DEIM modes were chosen so that the error was kept reasonable while simulation time would be maximally reduced. By reading Figure 4.2 it is seen that after ten POD modes, the approximation error does not significantly decrease, meaning that 10 POD modes were required to keep the error in reasonable bounds. Moreover, since at ten POD modes the error does not seem to depend

significantly on the number of DEIM modes, the number of dimensions for the DEIM reduced nonlinear term was chosen to be five. Judging from Figure 4.3 the simulation time for this reduced model should be approximately 0.075 seconds if the previous time span is maintained, which was confirmed to be true by performing the simulation with these dimensions.

Figure 4.4 displays how the dynamics of the output species given by the reduced order model (red line) compared to the original model (blue line) in the first 5000 seconds. Here y-axis shows the concentration of each substance and x-axis the time. Analyzing the time-series in this format is important, for the absolute error measured earlier does not take into account how the actual behavior of the model is affected by dimension reduction. In the context of neural models, it is important that the dynamics are preserved. For example information transmission via calsium signaling between astrocytes and neurons is known to be amplitude and frequency modulated, so even a slight defect might cause the higher level behavior of the model to change. However, different applications require different error tolerances. With this in mind, it is seen from Figure 4.4 that all the approximated outputs of the reduced model behave very similarly to the original model. In the beginning at $t = 5$ the reaction to the external stimulus is similar in both models while also the recovery that begins at $t = 10$ adheres to the same dynamics. The notable exception here is DAG_{post}, which is seen to exhibit slight unnecessary oscillations, although the oscillation amplitude is very small. Interestingly, if the chosen POD mode is increased to 15 while maintaining DEIM mode 5, these deviations from the original model are no longer observed.

An interesting observation with regards to the nonlinear equations chosen by DEIM for approximating the entire system. As there were 5 DEIM modes, 5 indices mapping to equations were obtained in a step-wise manner. The output was x_2, x_{25}, x_{34}, x_{12}, x_{22}, which correpond to the equations of biological processes of external calcium, postsynaptic Gabg, inositol trisphosphate (IP_3) degradation, Ca phosphatidylinositol PLC complex (CA-PIP-PLC) and DAG diacylglycerol kinase complex (DAG-DAGK). It is worth noting that these selections are specific to the stimulus used in creating the original results as well as to the length of the simulation. To elaborate on the result, for the present simulation these five equations provide the best approximation of all the original 44 equations when their output is projected onto the subspace spanned by columns of U_k.

Although the approximation with 10 POD modes and 5 DEIM modes was not perfect, another interesting question is does the long time behavior of the reduced system change significantly. For example a reasonable concern would be if the unnecessary oscillations continue to gain amplitude with time. This was studied by using an extended simulation time of $t = 5 * 10^5$ seconds. Figure 4.5 illustrates the error between the original and reduced order model at every ten time points with this longer simulation time. It can be interpreted that as time passes, the approximated solution differs increasingly from the

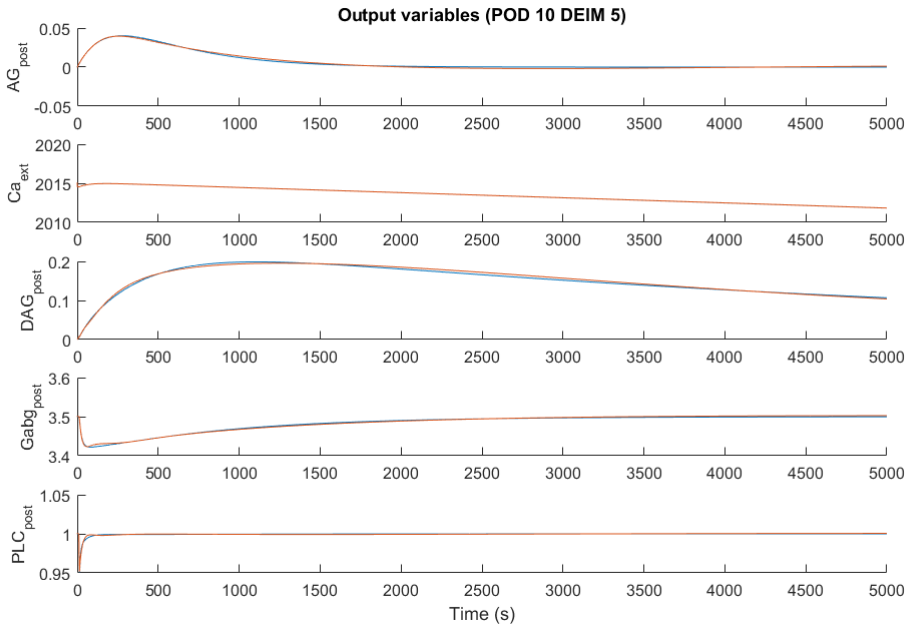

Figure 4.4: Time series of the dynamics of the biologically interesting output variables. Five species were tracked and their behavior plotted as a function of time. Y-axis displays the concentration of each ion/molecule. Blue line is the original model, and red is the approximation from the reduced order model with 10 POD and 5 DEIM modes.

original one, which suggests that whereas the original model reaches an equilibrium, the reduced one does not. Moreover, there is no reason to expect the solution to improve even with time. In the declared time frame, Ca_{post} performs the worst, with absolute error reaching over two thousand units. Given the initial calcium concentration of $2000\mu M$, this error is definitely intolerable. Although less radical in the other output species, the approximations are not satisfying. However, increasing the accuracy of the reduced model by employing more dimensions in POD, specifically 30 while keeping DEIM at 5, was the lowest dimension combination that produced more tolerable but still not perfect results in this time frame, while still radically reducing the simulation speed as estimated in Figure 4.3. (Simulation times between POD 10 and POD 30 were 0.0900 vs 0.0920, the increase corresponds to predicted times). An additional observation made during even longer simulations of the reduced model with $t > 5 * 10^5$ was that the solver could not meet error tolerances during integration, unless both POD and DEIM dimensions were increased along with the simulation time, which practically nullified the effectiveness of the reduction altogether. Moreover, there were no problems with any time frames simulating the original model.

Possible reasons for the erroneous approximation are the numerical accuracy of the solver that was used, inexact interpolation or insufficient capability of the reduced model to imitate the original model. The final point is the most probable reason given that in the above example, the POD and DEIM approximations were trained on only 10000 seconds.

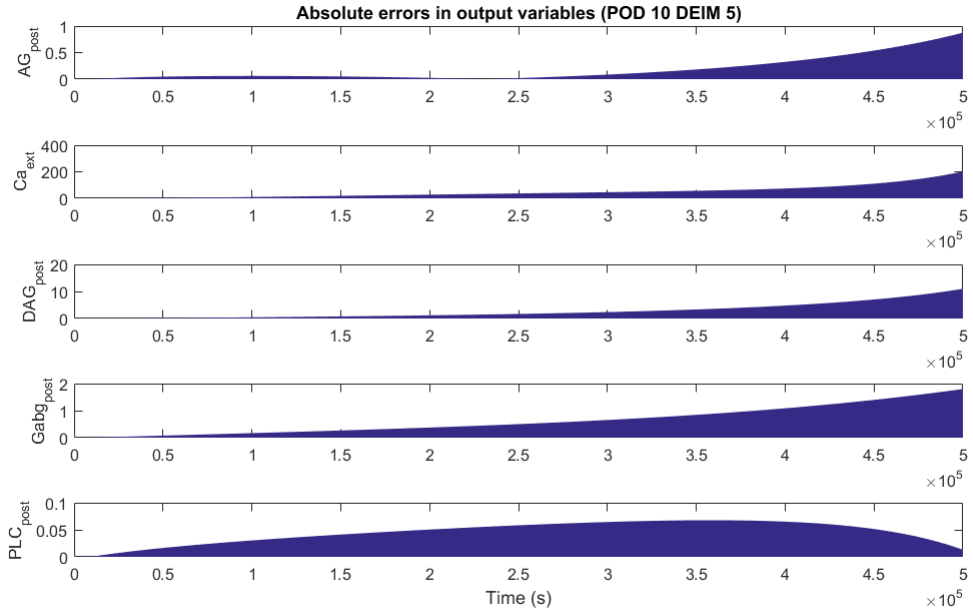

Figure 4.5: Absolute error between the reduced model (POD 10 DEIM 5) and the original model for each species at every tenth second, when the models were simulated for $5 * 10^5$ seconds.

The reduced model performs well when relatively many POD modes are used or when the reduced model is only used for prediction in the time span it was constructed in. Given the external stimulus in the early phase of simulation and a shorter time span of interest, it is logical that especially with only a few modes SVD emphasizes the dynamics caused by the stimuli.

In order to test a hypothesis that a low number of POD modes would be able to perform a near-correct approximation for a very long time span if the snapshots were also taken from a prolonged simulation, new reduced models were generated. (An additional reason is that the reduced model trained with 10k seconds was not numerically stable for long simulations even with high dimensions.) However, this time the snapshots were taken in the time span $t = [0, 5 * 10^9]$ while persevering the original stimuli. The snapshot interval was kept at 5 and the reduced model was constructed first with ten POD and five DEIM modes and afterwards with 30 POD and 10 DEIM modes.

The results of the longer simulations are shown in Figure 4.6. On the left side are the results with 10 POD and 5 DEIM modes, for which the DEIM interpolation indices were the same as in the shorter simulation. It seems that in this simulation all species in the original and reduced models now reach a steady state, although the approximation is not perfect. Especially in the beginning of the simulation the reduced model exhibits different dynamics than the original model. This is probably directly related to the length of the training time interval, which causes SVD to emphasize the equilibrium behavior. Moreover, $Gabg_{post}$ and PLC_{post} never seem to recover from the initial stimulus and thus

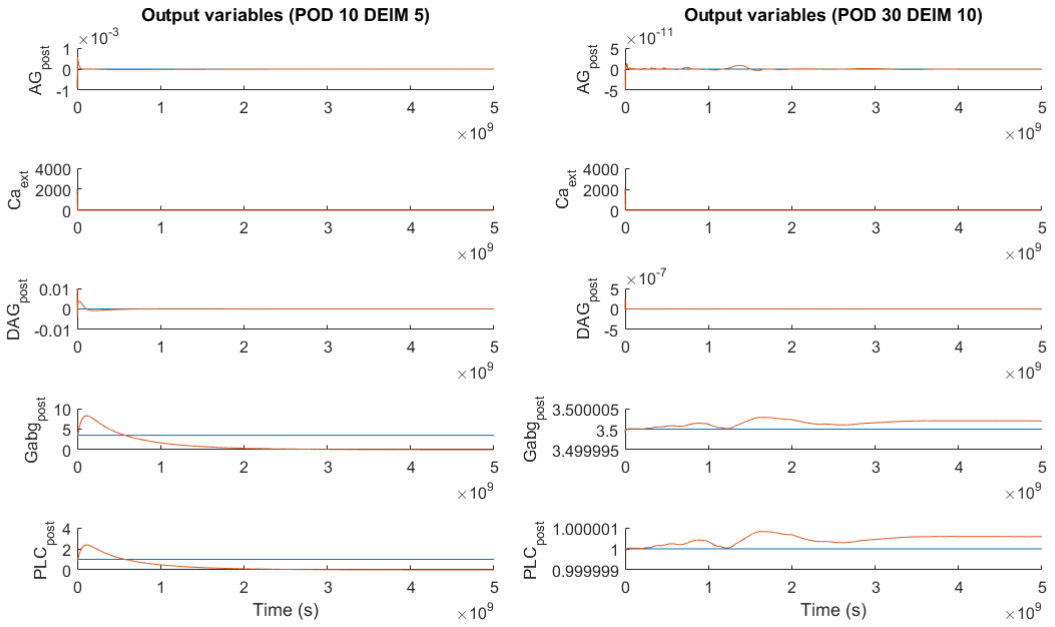

Figure 4.6: Behavior the reduced order model in a long duration simulation using ten POD and five DEIM modes on the left while 30 POD and 10 DEIM modes are used on the right hand side plot. Blue line is the original model and red is the reduced model for each output variable. The simulation time was $5 * 10^9$ seconds.

reach a different steady state than the original model, which is seen as a constant deviation of the blue plot from the red plot towards the end of the simulation in Figure 4.6 on the left. In the steady state, the highest error is seen at $Gabg_{post}$. The reduced model predicts it and PLC_{post} would both reach zero, which is not the case in the original model. For both molecules, the size of the error is roughly 100% of the true concentration, which would in most applications be devastating. However, a positive aspect is that the reduced model keeps the concentrations in physiologically meaningful bounds above or at zero. Additionally, the 10 POD 5 DEIM reduced model was three times faster to simulate than the full model.

Finally, a very good approximation was obtained with 30 POD and 10 DEIM modes, which is seen in Figure 4.6 on the right, while almost maintaining a simulation time of one third of the original model. Of the ten DEIM indices that the algorithm chose for the reduced order model, five first were identical to those chosen for the shorter simulation. The five additional ones were equations x_{29}, x_{21}, x_{40}, x_{37}, and x_{14}. These correspond to the equations of the biological species of postsynaptic G-protein with α subunit guanosine triphosphate PLC complex (Ga-GTP-PLC), DAG, phosphatidylinositol 4,5-bisphosphate (PIP_2), metabolic glutamate receptor activation (mGlur) and Ca-PLC complex, respectively. The reduced model has gained more pronounced unnecessary oscillations, although their amplitude is extremely low. Moreover, the steady state concentrations are

physiologically very close to the original and in an acceptable range considering the inherent errors a deterministic model such as the one studied here always has. Whether the errors seen here would affect the behavior of a multi-scale model, such as learning, remains a question for another study.

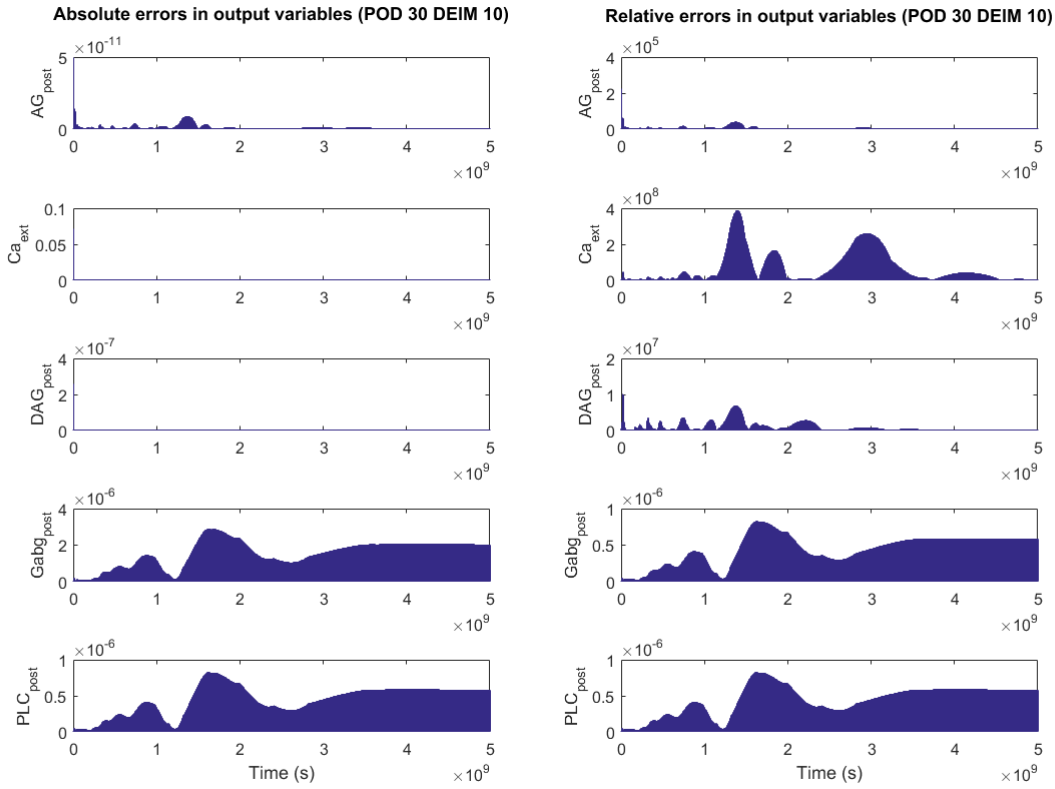

Figure 4.7: Absolute error (left) and relative error (right) between the 30 POD and 10 DEIM modes reduced model and the original model at every 10^6 seconds when simulated for $5 * 10^9$ seconds.

The magnitude of the errors with a long simulation time was further studied using the absolute and relative errors between the original model and the 30 POD 10 DEIM reduced model. After recovering the outputs from the reduced model by matrix multiplication and interpolation as described earlier, the absolute error was calculated as

$$e_{absolute} = |y_{reduced} - y_{original}| \tag{86}$$

and the relative error as

$$e_{relative} = \frac{|y_{reduced} - y_{original}|}{\epsilon + y_{original}}, \tag{87}$$

where ϵ was added to the denominator to prevent division by zero, as zero concentrations are abundant in the original model. The errors are visualized in Figure 4.7. As already hinted by Figure 4.6, the absolute errors are small, with the size being less than 10^{-6} for

all species except calcium, where the range is approximately 10^{-2}. The relative errors for PLC and $Gabg$ confirm that the observed variation is extremely small. The relative errors for AG, Ca and DAG on the other hand first display well the unnecessary oscillations the reduced model makes and second are in a completely different magnitude than the two other output species, going as high as 10^8. However, the magnitude can be explained by the epsilon used in the denominator, since the true concentration reaches zero at all points where a high error is seen, except the very beginning of the simulation where stimulus is applied. Moreover, the relative error proves that these three species actually correctly predict the steady state concentration, eventually, seen as the error degrading to zero.

5. DISCUSSION

Employing a mathematical description of the modeled system is an advantageous method in neuroscience. For example, most ordinary differential equation systems can be written in the control system formalism presented in this thesis, where inputs and state variables interact to produce an output. Describing the system in such a standardized manner is insightful from the perspective of the modeler in itself while also enhancing collaboration due to a common representation and allowing for additional mathematical analysis, such as optimal control studies or perturbation analysis. An important fact for model reduction is that the state space representation naturally leads to the problem of finding the minimal combination of state variables and inputs that produce the desired dynamics or outputs of the system. Such methods are extremely useful for connecting reduced small scale models via the inputs and outputs to form optimally performing large scale models [1]. Indeed methods exist that strive to preserve the input-output behavior, with balancing being the most studied one [45].

Although the need for simplifying models is often recognized in biological sciences, the methods for pursuing model reduction remain *ad hoc* and not enough attention is paid to mathematical properties of the simplified models. This trend is supported by recent model reduction publications in biosciences that strive to provide a reproducible methodology for the simplification process [26, 28, 29, 97, 98]. The consensus is that if the simulation results look like the original, the simplification was successful. However, since this approach is not concerned with actually reproducing the properties of the original system with a reduced order model, it has serious drawbacks when the models are applied in different time scales or as parts of larger simulations if the generalization capability of the model is not carefully confirmed. Model reduction has been studied extensively in applied mathematics which has, especially in the previous ten years, led to reduction methods that are also applicable to the highly nonlinear models arising from molecular chemistry and complex neural circuits [48].

Reduced order models have benefits in addition to less expensive computation and analysis. A good reduced order model increases the usage and thus the lifespan of a given model, since the reduced version has the possibility to be usable in a wider array of conditions such as real time applications, brain-inspired machine learning algorithms or brain-like computational hardware [50, 99, 100]. This allows for the model to evolve into a better one, since it will be expanded and modified to fit specific use cases and have

inherent errors corrected. Moreover, reduced order modeling leads to a better coverage of already modeled systems, which in turn promotes cost savings and increases available time for other research tasks as existing models can be recycled. Additionally, reduced order models will make more detailed models possible, as even more information can be included to large scale network studies [1]. These motivating factors should contribute to the reduction decision equally with computational savings.

5.1 Model Reduction Methods

Simplification is possible by eliminating reactions and variables from the system and thus obtaining a new, smaller model [26]. This occasionally leads to great results and is also the approach still taken in the field of neuroscience today (see e.g. [13]). However, the preferred model reduction approach would be one that does not lose details of any species in the system as a result of the dimensionality reduction. This is precisely where mathematical model order reduction methods such as those based on subspace projection are at their strongest, since the entire original system is approximated by a smaller dimensional one in the simulation and analysis phase. Additionally, the original inputs and outputs of the system are preserved, giving mathematical methods another advantage over elimination approaches.

Linearization as a simplification method was briefly mentioned in this thesis. It is a noteworthy method when an approximation that is valid in a very small parameter range is acceptable. An additional benefit is that the linear result can be further reduced with mathematical methods. However, locality of the result is the biggest drawback of linearizing models because the result lacks generalization and this leads to loss of nonlinear dynamics outside the linearization point [21].

In [101], a nonlinear model was divided into a subsystem and an environment, followed by linearization and reduction of the environment model with balanced truncation. This way the more interesting subsystem of a biochemical pathway model was modeled accurately while the reduced environment part provided computational efficiency. It is an interesting approach which should be considered if reducing the entire model produces erroneous results yet additional complexity can be tolerated. Moreover, large models might contain linear subsystems even without linearization and this could be exploited in model reduction.

Balanced Truncation (BT) [45] is a model order reduction method that is greatly recommended for linear models when optimal control of the reduced system is one of the primary goals of the reduction. In the context of neuroscience the greatest benefit of BT is arguably the insight it gives into the behavior of the system controllability and observability wise, since much could be learned by studying the controllability of neurons and neuronal networks [102]. While a great benefit of BT is the preservation of stability in the reduced order model, the stability of models in neuroscience remains a topic that

has received little concern, which possibly negatively affects the interest towards BT in neuroscience. Another concern is the efficiency and numerical stability of solving the grammians through Lyapunov equations for very large systems. Although approximative methods can be employed if numerical or performance restrictions occur, the quality of the reduced order model will likely suffer as a result [56].

Balanced Truncation has been applied to a model of a dendrite of a neuron with a combination of Hodgkin-Huxley, Connor-Stevens and Hoffman kinetics and 5000 compartments, which totalled at 50000 ordinary differential equations [103]. Each compartment modeled the membrane potential dynamics with nonlinear equations, which meant that linearization at resting potential was performed before BT. The study performed a parameter sweep analysis with the reduced order model and achieved comparable results to the full model while reducing the computation time to one fourth of the original.

Moment Matching (MM) [65, 66] methods rely on the Padé approximant theory for finding reduced order models and in this thesis the Padé via Lanczos algorithm [67] for MM was studied. While their computation is efficient, applicability of MM methods to different types of models, such as nonlinear models, is the most limited out of the methods presented in this thesis. The biggest difficulty with MM methods is finding a suitable expansion point, which often requires manual selection [50]. Although using many points relieves the need for a single point to be perfectly chosen, the efficacy of the reduced order model also suffers as a result [9].

Compared to BT, the MM reduced order model has weaker properties with regards to controllability and stability, but their computation is likely to be comparatively efficient and the resulting reduced order models achieve very low dimensions. Due to the locality of MM methods, they generally produce a weaker approximation of the original model compared to other methods, especially if evaluated far from the expansion point [50]. On the other hand, the calculation becomes more expensive if additional accuracy and better numerical qualities are desired, such as in the case of achieving stability of the reduced order model through a restarting scheme [50]. Moreover, the choice of algorithm plays a large role in determining the quality of the reduction. For example, Lanczos methods might not be able to reach the desired order of the model before unintended termination [67], while Arnoldi methods do not handle MIMO models as well [74] and do not generally manage to match as many moments as Lanczos methods [69].

An interesting thought is the use of MM methods for reducing spiking neuron models, since the transfer function has been shown to be effective in modeling neurons before [104]. Spiking neuron models are characterized by a certain rate of spiking, where the voltage of the cell membrane rapidly raises and falls when some stimulus is applied to the system [5]. The spiking is known as limit cycle dynamics. Moreover, neuroscience has an abundance of simple spiking neuron models of the integrate-and-fire type, but none of the simple models are able to model the exact waveform of the action potential

responsible for a spike accurately. As the strength of MM methods is in approximating the dynamics in a narrow frequency range, it makes sense to combine the two. Especially in network simulations there is a need for a variety of different neurons, which could be generated by using a variety of different expansion points [8].

Proper Orthogonal Decomposition (POD) [83] is a model order reduction method characterized by applicability to different types of systems, which include time-dependent and nonlinear systems. POD takes advantage of the observed dynamics of the [58] system in model order reduction by the method of snapshots [58]. However, it must be noted that the method achieves true independency of the original dimension only for linear systems and might even be counter productive for heavily nonlinear systems [89]. POD does not guarantee stability of the reduced order model and requires the full original system to be simulated before order reduction. Although the method might seem less mathematically rigorous compared to BT and MM, calculating the basis for subspace projection via this data driven approach is less likely encounter numerical instability, making POD a very accessible method altogether [105].

Discrete Empirical Interpolation Method has seen applications in reducing models of neuroscience, probably because it is one of the few suitable methods for reducing large nonlinear systems. However, the number of studies employing DEIM in neuroscience is surprisingly low given the potential of the method, and several more studies are needed to validate and popularize use of DEIM. A contributing fact to the non-utilization of mathematical model reduction methods might be a lack of mathematical training in biosciences. In neuroscience, DEIM has been used for reducing the dimension of the partial differential equation form of the FitzHugh-Nagumo model and the theoretical speed advantage and approximation error have been promising [89]. The time-varying form of the same equation has been reduced with a DEIM variation in [25]. In these two studies, the aim is proving the succesful order reduction rather than studying the results from a neuroscience point of view. Furthermore, spiking neurons modeled by the cable equation with simplified and realistic morphologies including several compartments and stimulation points have been reduced [106]. The reduced order models achieved six to thirty times faster simulation speeds with varying error rates. Additionally, a detailed compartmental neuron model with multiple dendritic branches and Hodgkin-Huxley kinetics has been reduced with DEIM [107]. In that study it is concluded that a detailed model of 879 compartments can be accurately reduced to only eight compartments. Very recently a non-negative variation of DEIM (NNDEIM) was used in reducing a neuronal network model constructed from cables with HH dynamics where a 20 fold speedup was obtained with slight approximation error [95]. All studies employed discretized partial differential equations in order to obtain the ordinary differential equation systems for model reduction.

Proper Orthogonal Decomposition and Discrete Empirical Interpolation Method are the only methods presented in this thesis that reduce the order of the model based on

the observed dynamics of the system by the method of snapshots [58]. The sampling approach minimizes an error between the snapshots and the trajectory of the original solution vector, instead of focusing on input-output characteristics as heavily as BT and MM. Additionally, this method of snapshots provides a possibility to tune the reduced order model for different stimuli, since the snapshots contain information of how the system reacts to excitation, or to the external inputs of the system [48]. It becomes possible to create very specific, very low order reduced model for a small number of stimuli in this way, which was done in this thesis. Moreover, snapshots of simulations with different excitation can be combined together in order to produce a reduced order model with considerable generalization capability so that it is accurate for an array of stimuli.

5.2 Approximation Error in Reduced Models

Obtaining an estimate of the error introduced in the reduction is an essential part of the reduction process. Consequently, established model reduction methods have analytical apriori error measures developed for controlling the result of the reduction. As an example, the accuracy of SVD based methods, such as POD and DEIM, can be estimated from the behavior of singular values calculated during the reduction process [89, 96]. Although error estimates are available for each method, the estimates are not necessarily comparable between methods. As such, they are best used for tuning the model under study at the time, rather than for deciding which reduction method to use. Additionally, error can reference multiple measurable quantities. For example, the sensitivity of the reduced system to perturbations may have changed, which in some applications can be considered reduction error. It is again the responsibility of the modeler to consider the importance of such factors and their reporting.

For all methods it is possible to evaluate the error from the result of the reduction compared to original time-series data of the state variables or outputs, or both. Additionally, the error calculation can include only some selected species of the model. Determining the size of an acceptable distance-based error is always dependent on the application. In very sensitive systems, such as the nervous system or the brain, even minor errors might lead to differences in observed behavior on a larger scale. On the other hand, there are also systems that are resistant to minor perturbations. As such, a larger error does not necessarily mean a worse reduction result, meaning that the context and future application of the reduced order model are important factors when evaluating error values. In this regard, a first consideration is that mathematical models of natural systems inherently contain inaccurate assumptions and measurement flaws, which makes chasing zero error a fruitless endeavour. Second, distance based error might be large for some species of the model and small for others, which is a detail that is hidden by averaged error measures, and the model could still be usable if the relevant species are approximated well.

5.3 Significance of Results and Future Work

In this thesis, Proper Orthogonal Decomposition and Discrete Empirical Interpolation Method (POD+DEIM) were applied to reduce a neural plasticity model that is an integral piece in modeling learning in the mammalian brain. POD+DEIM was chosen since it is directly applicable to nonlinear models, unlike most of the other methods which would require linearization. Additionally, the method scales to very large systems, which again is a property not possessed by all the existing methods. Further still, it allows for a purely mathematical approach, eliminating the need of converting the model to a simulator-specific format or hand-picking variables and equations for removal. Novel results were obtained since the previous studies employing DEIM have targeted models obtained from discretization of partial differential equations [89, 95, 106, 107], which is not the case here. Moreover, previously nonlinear systems in neuroscience have been reduced with linearization methods. Linearization was not used here, which theoretically makes the reduced order model more globally accurate. Additionally, this study marks the first time the dimension of the plasticity model from [3] is reduced. Although the model itself is rather small with 44 equations, remarkable performance gains were obtained. It could be hypothesized that larger models, such as those obtained from discretizing partial differential equations, are more compressible and reducible given the inherent linearities and correlations, which would lead to even better performance gains percentage wise.

DEIM is especially interesting in the sense that the original functions are employed in the interpolation, in contrast to projecting to a non-human comprehendable subspace. The indices picked by DEIM directly show which equations can be used for the best possible approximation, and from this it can be deduced that they are also the ones most responsible for the nonlinear behavior of the system. This allows the researcher to interpret the value of each nonlinear function for driving the dynamics of the system.

In the present study, the biological species whose nonlinear equation the DEIM algorithm selected for interpolation with five equations were external calcium, postsynaptic Gabg, IP3 degradation, calcium-PIP-PLC and DAG-DAGK (see Appendix A.1). The selection of the interpolation points depends on both the input to the system and the simulation time. In addition to the results presented in this thesis, it is interesting to note how the output of the DEIM algorithm changes in the scenario where no external stimuli to the system is employed. Basically this means that the system finds an equilibrium point with a set of initial values. Under those circumstances, the five selected species of equations selected for interpolation were external calcium and IP_3 degradation as before, and additionally three new species, postsynaptic DAG, Ca-DAG-diacylglycerol lipase complex and 2AG from equations x_{21}, x_4 and x_1 respectively (see Appendix A.1).

The greatest challenge with POD+DEIM was found to generalization to time scales outside of the time scale in which the training data was created. Here the ordinary ver-

sion of POD+DEIM was used, although DEIM has already been developed further so that the interpolation points change during the simulation. These methods are called Localized DEIM (LDEIM) [94] and Adaptive DEIM (ADEIM) [25] and they strive to provide greater accuracy while still maintaining a low number of DEIM modes. Moreover, despite the young age of the DEIM method, a variation that keeps the results positive and maintains stability has been developed, called NNDEIM [95]. The positivity guarantee is definitely a useful addition to the reduced order model, since in theory, the reduced order model might introduce errors with negative values, which are usually not meaningful in biological models. An interesting research question would be if these methods with increased complexity improve the accuracy of the original POD+DEIM while still maintaining the performance improvements.

Although the results presented in this thesis contained error introduced by the reduction method, the magnitudes of the deviations were not dramatically high. Additionally, the dynamics of the original model were not perfectly predicted in very long simulations, although the steady states after applying stimuli were eventually practically equal. Possible reasons for the erroneous approximation are the numerical accuracy of the solver that was used, inexact interpolation or insufficient capability of the reduced order model to imitate the original model. Even with the current small model used in this study, it was not possible to perform long time scale simulations with continuous time-varying input to the system due to performance reasons of the computation hardware. Future studies are needed to determine the usefulness of reduced order models as parts of increasingly detailed models, since with the current knowledge it is impossible to predict if the dynamics and emergent features of very large models would be altered by the inaccuracies introduced in the model order reduction process.

When reduced models are included in large network models, it is often important that they correctly react to many types of different stimuli, and this is made possible by POD and DEIM especially. To this end, the recently published Localized DEIM looks extremely promising for maintaining a low number of approximation modes throughout different environments, such as changing excitation [94], given that the inputs were present in the offline training phase of POD+DEIM. Moreover, the adaptive version ADEIM is able to react to unanticipated behavior on the online stage of a simulation by efficiently querying the original system [25]. Further studies are needed to evaluate the performance of these new methods in delivering reduced order models for extremely heterogeneous environments and multiscale simulations.

6. CONCLUSIONS

Model order reduction is an essential process for improving the scale and quality of future computational models of the human brain. Although many methods of simplifying exist, subspace projection methods studied especially in the field of control theory, show most promise for they can be automatically applied, have adjustable error bounds and scale to virtually any size of systems. Additionally, they are applicable to nonlinear systems, either directly or via linearization, which greatly increases their applicability to complex models in neuroscience.

In this thesis Proper Orthogonal Decomposition and Discrete Empirical Interpolation Method (POD+DEIM) was applied to a data driven biological model of plasticity in the brain. Five equations that model important molecules and ions were chosen for special analysis, since these species have the greatest potential to link the model to a larger system comprising more areas and features of the brain. This nonlinear system had a sparse linear part and included a time dependent stimulus.

Model reduction with POD+DEIM was found to significantly reduce the simulation time. An additional benefit is that the approximation can be tuned by adjusting the POD and DEIM dimensionality independently. However, the reduced order model did not perfectly reproduce the dynamics of the original model in long time scales and the steady states also had slight deviations from the results of the original model. Whether the observed error is tolerable depends on the final purpose of the model. All in all, subspace projections methods seem suitable for reducing the dimensionality of signaling pathway models in neuroscience.

REFERENCES

[1] W. Gerstner, H. Sprekeler, and G. Deco. Theory and simulation in neuroscience. *Science*, 338, October 2012.

[2] M.L. Linne and T. Jalonen. Astrocyte-neuron interactions: from experimental research-based models to translational medicine. *Progress in Molecular Biology and Translational Science*, 123:191–217, 2014.

[3] B. Kim, SL. Hawes, F. Gillani, LJ. Wallace, and KT. Blackwell. Signaling pathways involved in striatal synaptic plasticity are sensitive to temporal pattern and exhibit spatial specificity. *PLoS Comput Biol*, 9(3), March 2013.

[4] T. Manninen, K. Hituri, J. Hellgren-Kotaleski, K. Blackwell, and M.L. Linne. Postsynaptic signal transduction models for long-term potentiation and depression. *Frontiers in Computational Neuroscience*, 4(152):1–29, December 2010.

[5] A. Hodgkin and A. Huxley. A quantitave description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117(4):500–544, August 1952.

[6] U. Bhalla. Multiscale interactions between chemical and electric signaling in ltp induction, ltp reversal and dendritic excitability. *Neural Networks*, 24(9):943–949, November 2011.

[7] G. Einevoll, C. Kayser, N. Logothetis, and S. Panzeri. Modelling and analysis of local field potentials for studying the function of cortical circuits. *Nature Reviews Neuroscience*, 14(11), November 2013.

[8] H. Markram, E. Mulle, S. Ramaswamy, M. W. Reimann, M. Abdellah, C. Aguado Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever, G Antoine Atenekeng Kahou, T. K. Berger, A. Bilgili, N. Buncic, A. Chalimourda, G. Chindemi, J-D. Courcol, F. Delalondre, V Delattre, S Druckmann, R. Dumusc, J. Dynes, S. Eilemann, E. Gal, M. Gevaert, J-P. Ghobril, A. Gidon, J. W. Graham, A. Gupta, V. Haenel, E. Hay, T. Heinis, J. B. Hernando, M. Hines, L. Kanari, D. Keller, J. Kenyon, G. Khazen, Y. Kim, J. G. King, Z. Kisvarday, P. Kumbhar, S. Lasserre, J-V. Le Bé, B. R. C. Magalhães, A. Merchán-Pérez, J. Meystre, B. R. Morrice, J. Muller, A. Muñoz-Céspedes, S. Muralidhar, K. Muthurasa, D. Nachbaur, T. H. Newton, M. Nolte, A. Ovcharenko, J. Palacios, L. Pastor, R. Perin, R. Ranjan, I. Riachi, J-R. Rodríguez, J. L. Riquelme, C. Rössert, K Sfyrakis, Y. Shi, J. C. Shillcock, G. Silberberg, R. Silva, F. Tauheed, M. Telefont, M. Toledo-Rodriguez, T. Trönkler, W. Van Geit, J. Díaz, R. Walker, Y. Wang, S. M. Zaninetta,

J. DeFelipe, S. L. Hill, I. Segev, and F. Schürmann. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2), October 2015.

[9] Z. Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Applied Numerical Mathematics*, 43(1):9–44, October 2002.

[10] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, July 1961.

[11] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50:2061–2070, 1962.

[12] J. Hindmarsh and R. Rose. A model of neuronal bursting using three coupled first order differential equations. *Proceedings of the Royal Society, Series B: Biological Sciences*, 221(1222):87–210, March 1984.

[13] C. Diekman, C. Fall, J. Lechleiter, and D. Terman. Modeling the neuroprotective role of enhanced astrocyte mitochondrial metabolism during stroke. *Biophysical Journal*, 104(8):1635–1838, April 2013.

[14] T. Kepler, L. Abbot, and E. Marder. Reduction of conductance-based neuron models. *Biological Cybernetics*, 55(5):381–387, 1992.

[15] B. Woo, D. Shin, D. Yang, and J. Choi. Reduced model and simulation of neuron with passive dendritic cable: an eigenfunction expansion approach. *Journal of Computational Neuroscience*, 19(3):379–397, December 2005.

[16] M. Sorensen and S. DeWeerth. An algorithmic method for reducing conductance-based neuron models. *Biological Cybernetics*, 95(2):185–192, August 2006.

[17] D. Shin, D. Yang, and J. Choi. On the use of pseudo-spectral method in model reduction and simulation of active dendrites. *Computers in Biology and Medicine*, 39(4):340–345, April 2009.

[18] K. Ogata. *Modern Control Engineering*. Prentice Hall, 5 edition, 2008.

[19] J. Stewart. *Calculus Early Transcendentals*. Thomson Brooks/Cole, 6 edition, 2008.

[20] D. Jordan and P. Smith. *Nonlinear Ordinary Differential Equations*. Oxford University Press, 4 edition, 2007.

[21] M. Rewieński and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2), February 2003.

[22] W. Rugh. *Nonlinear System Theory*. The John Hopkins University Press, Baltimore, 1 edition, 1981.

[23] M. Condon and R. Ivanov. Krylov subspaces from bilinear representations of nonlinear systems. *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, 26(2):399–406, February 2007.

[24] Y. Lin, L. Bao, and Y. Wei. Order reduction of bilinear MIMO dynamical systems using new block Krylov subspaces. *Computers and Mathematics with Applications*, 58(6):1093–1102, October 2009.

[25] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM Journal of Scientific Computing*, 37(4), August 2015.

[26] E. Kutumova, A. Zinovyev, R. Sharipov, and F. Kolpakov. Model composition through model reduction: a combined model of CD95 and NF-κB signaling pathways. *BMC Systems Biology*, 13, July 2013.

[27] J. Whiteley. Model reduction using a posteriori analysis. *Mathematical Biosciences*, 225(1):44–52, May 2010.

[28] O Radulescu, A. N. Gorban, A. Zinovyev, and V. Noel. Reduction of dynamical biochemical reactions networks in computational biology. *Frontiers in Genetics*, 131(3), July 2012.

[29] Westm S., L. Bridge, M. White, P. Paszek, and V. Biktashev. A method of 'speed coefficients' for biochemical model reduction applied to the NF-κB system. *Journal of Mathematical Biology*, 70:591–620, March 2015.

[30] D. Lebiedz, D. Skanda, and M. Fein. Automatic Complexity Analysis and Model Reduction of Nonlinear Biochemical Systems. In M. Heiner and A. M. Uhrmacher, editors, *Computational Methods in Systems Biology*, pages 123–140. Springer Berlin Heidelberg, October 2008.

[31] A. Gorban and I. Karlin. Method of invariant manifold for chemical kinetics. *Chemical Engineering Science*, 58(21), November 2003.

[32] I. Surovtsova, N. Simus, K. Hübner, S. Sahle, and U. Kummer. Simplification of biochemical models: a general approach based on the analysis of the impact of individual species and reactions on the systems dynamics. *BMC Systems Biology*, 6(1), March 2012.

[33] Y. Tang, J. Stephenson, and G. Othmer. Simplification and analysis of models of calcium dynamics based on IP3-sensitive calcium channel kinetics. *Biophysical Journal*, 70(1), January 1996.

[34] J. Borghans and L. Segel. Extending the quasi steady state approximation by changing variables. *Bulletin of Mathematical Biology*, 58(1):43–63, 1996.

[35] E. H. Flach and S. Schnell. Use and abuse of the quasi-steady-state approximation. *Systems biology*, 153(4), July 2006.

[36] S. Schnell and P. K. Maini. A century of enzyme kinetics. Reliability of the K_M and v_{max} estimates. Comment. *Journal of Theoretical Biology*, 8:169–187, 2003.

[37] I. Stoleriu, F. A. Davidson, and J. L. Liu. Effects of periodic input on the quasi-steady state assumptions for enzyme-catalysed reactions. *Journal of Mathematical Biology*, 50:115–132, 2005.

[38] L. Michaelis and M.L. Menten. Die Kinetik der Invertinwirkung. *Biochemische Zeitschrift*, 49:333–369, 1913.

[39] A. Pettinen, T. Aho, Smolander O.P., T. Manninen, A. Saarinen, K.L. Taattola, O. Yli-Harja, and M.L. Linne. Simulation tools for biochemical networks: evaluation of performance and usability. *Bioinformatics*, 21(3):357–363, February 2005.

[40] R. Alves, F. Antunes, and A. Salvador. Tools for kinetic modeling of biochemical networks. *Nature Biotechnology*, 24(10):667–672, October 2006.

[41] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A.P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takashi, M. Tomita, J. Wagner, and J. Wang. The Systems Biology Markup Language (SBML): A medium for representation and exhange of biochemical network models. *Bioinformatics*, 19:524–531, October 2003.

[42] L. Calzone, F. Fages, and S. Soliman. BIOCHAM - an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[43] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI - a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, December 2006.

[44] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya. Logical modelling of regulatory networks with GINsim 2.3. *Biosystems*, 97(2):134–139, August 2009.

[45] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, February 1981.

[46] K. Glover. All optimal Hankel-norm approximations of linear multivariate systems and their l^{∞}-error bounds. *International Journal of Control*, 39(6):1115–1193, 1984.

[47] R. Kalman. Contributions to the theory of optimal control. *Boletín de la Sociedad Matemática Mexicana*, 5:102–119, 1960.

[48] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *Society for Industrial and Applied Mathematics Review*, 57(4):483–531, November 2015.

[49] K. Zhou, G. Salomon, and E. Wu. Balanced realization and model reduction for unstable systems. *International Journal of Robust Nonlinear Control*, 9(3):183–198, March 1999.

[50] C Antoulas, D. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary Mathematics*, 280:193–220, 2001.

[51] A. Laub, M. Heath, C. Paige, and R. Ward. Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Transactions on Automatic Control*, 32(2):115 – 122, February 1987.

[52] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *The American Institute of Aeronautics and Astronautics Journal*, 40(11):2323–2330, November 2002.

[53] N. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, May 1988.

[54] P. Opdenacker and E. Jonckheere. Lqg balancing and reduced lqg compensation of symmetric passive systems. *International Journal of Control*, 41(1), 1985.

[55] L. Pernebo and L. Silverman. Model reduction via balanced state space representations. *IEEE Transactions on Automatic Control*, 27(2):382–387, April 1982.

[56] P. Benner and J. Saak. Numerical solution of large and sparse continuous time algebraic matrix riccati and lyapunov equations A state of the art survey. *GAMM Mitteilungen*, 36(1):32–52, 2013.

[57] S. Shokoohi, L. Silverman, and P. Van Dooren. Linear time-variable systems: Balancing and model reduction. *IEEE Transactions on Automatic Control*, 28(8):810–822, August 1983.

[58] L. Sirovich. Turbulence and the dynamics of coherent structures. I-III. *Quarterly of Applied Mathematics*, 45(3):561–590, October 1987.

[59] Z. Ma, C. Rowley, and G. Tadmor. Snapshot-based balanced truncation for linear time-periodic systems. *IEEE Transactions on Automatic Control*, 55(2):469–473, January 2010.

[60] T. Stykel and A. Vasilyev. A two-step model reduction approach for mechanical systems with moving loads. *Journal of Computational and Applied Mathematics*, 297:85–97, May 2016.

[61] J. Scherpen. Balancing for nonlinear systems. *Systems & Control Letters*, 21:143–153, 1993.

[62] J. Bouvrie and B. Hamzi. Model reduction for nonlinear control systems using kernel subspace methods. *CoRR*, abs/1108.2903, 2011.

[63] O. Nilsson and A. Rantzer, editors. *A novel approach to balanced truncation of nonlinear systems*, August 2009.

[64] I. Dones, S. Skogestad, and H. Preisig. Application of balanced truncation to nonlinear systems. *Industrial & Engineering Chemistry Research*, 50(17):10093–10101, 2011.

[65] H. James, N. Nichols, and R. Phillips. *Theory of Servomechanisms*, chapter 7. McGraw-Hill, 1 edition, 1947.

[66] L. Meier. Approximation of linear constant systems. *IEEE Transactions on Automatic Control*, 12(5):585–588, 1967.

[67] P. Feldmann and R. Freund. Efficient linear circuit analysis by Padé approximation via the lanczos process. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(5):639–649, May 1995.

[68] W. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Auarterly of Applied Mathematics*, 9(1):17–29, April 1951.

[69] E. Grimme. *Krylov Projection Methods for Model Reduction*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[70] L. Pillage and R. Rohrer. Asymptotic Waveform Evaluation for timing analysis. *IEEE Transactions on Computer-Aided Design*, 9(4):352–366, April 1990.

[71] C. Lanczos. An iteration method for the solution of the eigenvalue problmen of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, October 1950.

[72] A. Ruhe. Implementation aspects of band lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Mathematics of Computation*, 33(146):680–687, 1979.

[73] R. Craig and A. Hale. Block-Krylov component synthesis method for structural model reduction. *Journal of Guidance, Control and Dynamics*, 11(6):562–570, November 1988.

[74] S. Gugercin. An iterative SVD-Krylov based method for model reduction of large-scale dynamical systems. *Linear Algebra and its Applications*, 428(8):1964–1986, April 2008.

[75] Z. Bai, R. Slone, W. Smith, and Q. Ye. Error bounds for reduced system model by Padé approximation via the Lanczos process. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(2):133–141, February 1999.

[76] Z. Bai and R. Freund. A partial Padé-via-lanczos method for reduced order modeling. *Linear Algebra and its Applications*, 332-334(1):139–164, August 2001.

[77] J. Roychowdhury. Reduced-order modeling of time-varying systems. *IEEE Transactions on Circuits and Systems*, 46(10), October 1999.

[78] R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):171–187, February 2003.

[79] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[80] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.

[81] D. Kosambi. Statistics in function space. *Journal of the Indian Mathematical Society*, 7:76–88, 1943.

[82] G. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[83] G. Berkooz, P. Holmes, and J. Lumley. The Proper Orthogonal Decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993.

[84] M. Rathinam and L. Petzold. A new look at Proper Orthogonal Decomposition. *SIAM Journal on Numerical Analysis*, 41(5):1893–1925, 2003.

[85] M. Kowalski and J.M. Jin. Model-order reduction of nonlinear models of electromagnetic phased-array hyperthermia. *IEEE Transactions on Biomedical Engineering*, 50(11):1243–1254, November 2003.

[86] A. Siade, M. Putti, and W.G. Yeh. Snapshot selection for groundwater model reduction using proper orthogonal decomposition. *Water Resources Research*, 46(8), 24 2010.

[87] K. Kunisch and S. Volkwein. Optimal snapshot location for computing POD basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(3):509–529, November 2010.

[88] S. Glavaski, J. Marsden, and R. Murray. Model reduction, centering, and the karhunen-loeve expansion. *IEEE Decision and Control*, 2:2071–2076, 1998.

[89] S. Chaturantabut and D. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal of Scientific Computing*, 32(5):2737–2764, September 2010.

[90] B. King. Nonuniform grids for reduced basis design of low order feedback controllers for nonlinear continuous systems. *Mathematical Models and Methods in Applied Sciences*, 8(7):1223–1241, November 1998.

[91] J. Atwell and B. King. Proper orthogonal decomposition for reduced basis feedback controllers for parabolic equations. *Mathematical and Computer Modelling*, 33(1):1–19, January 2001.

[92] I. Kalashnikova and M. Barone. Efficient non-linear proper orthogonal decomposition/galerkin reduced order models with stable penalty enforcement of boundary conditions. *International Journal for Numerical Methods in Engineering*, 90(11):1337–1362, June 2012.

[93] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematicue*, 339, June 2004.

[94] B. Peherstorfer, D. Butnaru, K. Willcox, and H.J. Bungartz. Localized discrete empirical interpolation method. *SIAM Journal of Scientific Computing*, 36(1), February 2014.

[95] D. Amsallem and J. Nordström. Energy stable model reduction of neurons by nonnegative discrete empirical interpolation. *SIAM Journal of Scientific Computing*, 38(2), April 2016.

[96] S. Chaturantabut and D. Sorensen. A state space error estimate for pod-deim nonlinear model reduction. *SIAM Journal on Numerical Analysis*, 50(1):46–63, January 2012.

[97] A. Saysel and Y. Barlas. Model simplification and validation with indirect structure validity tests. *System Dynamics Review*, 22(3):241–262, November 2006.

[98] S. Tewari, M. Gottipati, and V. Parpura. Mathematical modeling in neuroscience: neuronal activity and its modulation by astrocytes. *Frontiers in Integrative Neuroscience*, February 2016.

[99] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, November 1982.

[100] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber. SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor. In *IEEE International Joint Conference on Neural Networks*, pages 2849–2856, 1993.

[101] W. Liebermeister, U. Baur, and E. Klipp. Biochemical network models simplified by balanced truncation. *Federation of Europen Biochemical Societies Journal*, 272(16):4034–4043, August 2005.

[102] S. Gu, F. Pasqualetti, Q. Telesford, A. Yu, A. Kahn, J. Medaglia, J. Vettel, M. Miller, S. Grafton, and D. Bassett. Controllability of structural brain networks. *Nature Communications*, 6, October 2015.

[103] A. Kellems, D. Roos, N. Xioa, and S. Cox. Low-dimensional, morphologically accurate models of subthreshold membrane potential. *Journal of Computational Neuroscience*, 27(2):161–176, October 2009.

[104] S. Ostojic and N. Brunel. From spiking neuron models to linear-nonlinear models. *PLOS Computational Biology*, 7(1), January 2011.

[105] M. Safonov and R. Chiang. A schur method for balanced-truncation model reduction. *IEEE Transactions on Automatic Control*, 34(7):729–733, 1989.

[106] A. Kellems, S. Chaturantabut, D. Sorensen, and S. Cox. Morphologically accurate reduced order modeling of spiking neurons. *Journal of Computational Neuroscience*, 28(3), June 2010.

[107] B. Du, D. Sorensen, and S.J. Cox. Model reduction of strong-weak neurons. *Frontiers in Computational Neuroscience*, 8(164), December 2014.

A. APPENDIX

A.1 Synaptic Plasticity Model

A.1.1 Species in the Model

Chemical species	**Abbreviation**	**Variable**
2-arachidonoyl glycerol	2AG	x_1
Calcium	Ca	x_2
Calcium - calcium-binging protein complex	Ca-Calbindin	x_3
Calcium - diacylglycerol - diacylglycerol lipase complex	Ca-DAG-DAGL	x_4
Calcium - diacylglycerol - G-protein α-subunit - guanosine-5'-triphosphate - phospholipase C complex	Ca-DAG-GaGTP-PLC	x_5
Calcium - diacylglycerol - phospholipase C complex	Ca-DAG-PLC	x_6
Calcium - diacylglycerol lipase complex	Ca-DAGL	x_7
Calcium - G-protein α-subunit - phosphatidylinositol 4,5-bisphosphate - phospholipase C complex	Ca-GaGTP-PIP_2-PLC	x_8
Calcium - G-protein α-subunit - guanosine-5'-triphosphate - phospholipase C complex	Ca-GaGTP-PLC	x_9
External leak calcium	Ca-Leak	x_{10}
Calcium - sodium-calcium exchange protein complex	Ca-NCX	x_{11}
Calcium - phosphatidylinositol 4,5-bisphosphate - phospholipase C complex	Ca-PIP_2-PLC	x_{12}
Calcium - protein kinase C complex	Ca-PKC	x_{13}
Calcium - phospholipase C complex	Ca-PLC	x_{14}
Calcium - plasma membrane Ca^{2+} ATPase complex	Ca-PMCA	x_{15}
Calmodulin C-terminal domain - calcium		

complex	CaMC-Ca_2	x_{16}
Calmodulin N-terminal domain - calcium complex	CaMN-Ca_2	x_{17}
Calciumsaturated calmodulin	CaM-Ca_4	x_{18}
Calcium-binding protein	calbindin	x_{19}
Calmodulin	CaM	x_{20}
Diacylglycerol	DAG	x_{21}
Diacylglycerol - diacylglycerol kinase complex	DAG-DAGK	x_{22}
Diacylglycerol kinase	DAGK	x_{23}
Diacylglycerol lipase	DAGL	x_{24}
G-protein with α, β and γ subunits	Gabg	x_{25}
G-protein with α, β and γ subunits - glutamate - metabotropic glutamate receptor complex	Gabg-Glu-mGluR	x_{26}
G-protein α subunit - guanosine diphosphate complex	GaGDP	x_{27}
G-protein α subunit - Guanosine-5'-triphosphate complex	GaGTP	x_{28}
G-protein α subunit - Guanosine-5'-triphosphate - phospholipase C complex	GaGTP-PLC	x_{29}
Glutamate - metabotropic glutamate receptor complex	Glu-mGluR	x_{30}
Glutamate - desensitized metabotropic glutamate receptor complex	Glu-mGluRdesens	x_{31}
Inactive glutamate	GluInact	x_{32}
Inositol 1,4,5-trisphosphate	IP_3	x_{33}
Degraded inositol 1,4,5-trisphosphate	IP_3deg	x_{34}
Inositol 1,4,5-trisphosphate - phosphatidylinositol-4,5-bisphosphate 3-kinase complex	IP_3-PIKin	x_{35}
Leak channel protein	Leak	x_{36}
Metabotropic glutamate receptor	mGluR	x_{37}
Sodium-calcium exchange protein	NCX	x_{38}
Phosphatidylinositol-4,5-bisphosphate 3-kinase	PIKin	x_{39}
Phosphatidylinositol 4,5-bisphosphate	PIP_2	x_{40}
Active protein kinase C	PKCa	x_{41}

Inactive protein kinase C	PKCi	x_{42}
Phospholipase C	PLC	x_{43}
Plasma membrane Ca^{2+} ATPase complex	PMCA	x_{44}

Table A.1: Processes included in the plasticity model of [3]

A.1.2 Full Order Plasticity Model

The time varying parameters of the model are Ca_{post} and Glu_{ext}.

$$
\begin{aligned}
\dot{x_1} &= k_{prodAG_c}x_4 - k_{degAG_f}x_1 \\
\dot{x_2} &= k_{PMCA_c}x_{15} + k_{NCX_c}x_{11} - k_{Leak_f}x_2x_{36} + k_{Leak_b}x_{10} \\
\dot{x_3} &= k_{buffer_f}Ca_{post}(t)x_{19} - k_{buffer_b}x_3 \\
\dot{x_4} &= k_{prodAG_f}x_{21}x_7 - k_{prodAG_b}x_4 - k_{prodAG_c}x_4 \\
\dot{x_5} &= k_{DAG_3c}x_8 - k_{DAG_4f}x_5 \\
\dot{x_6} &= k_{DAG_1c}x_{12} - k_{DAG_2f}x_6 \\
\dot{x_7} &= k_{DAGL_f}Ca_{post}(t)x_{24} - k_{DAGL_b}x_7 - k_{prodAG_f}x_{21}x_7 + k_{prodAG_b}x_4 \\
&\quad + k_{prodAG_c}x_4 \\
\dot{x_8} &= k_{DAG_3f}x_9x_{40} - k_{DAG_3b}x_8 - k_{DAG_3c}x_8 \\
\dot{x_9} &= k_{G_PLC2_f}x_{28}x_{14} - k_{G_PLC2_b}x_9 + k_{Ca_PLC2_f}Ca_{post}(t)x_{29} - k_{Ca_PLC2_b}x_9 \\
&\quad - k_{DAG_3f}x_9x_{40} + k_{DAG_3b}x_8 + k_{DAG_4f}x_5 - k_{GAP_2f}x_9 \\
\dot{x_{10}} &= k_{Leak_f}x_2x_{36} - k_{Leak_b}x_{10} - k_{Leak_c}x_{10} \\
\dot{x_{11}} &= k_{NCX_f}Ca_{post}(t)x_{38} - k_{NCX_b}x_{11} - k_{NCX_c}x_{11} \\
\dot{x_{12}} &= k_{DAG_1f}x_{40}x_{14} - k_{DAG_1b}x_{12} - k_{DAG_1c}x_{12} \\
\dot{x_{13}} &= k_{Ca_PKC_f}Ca_{post}(t)x_{42} - k_{Ca_PKC_b}x_{13} - k_{DAG_PKC_f}x_{21}x_{13} + k_{DAG_PKC_b}x_{41} \\
\dot{x_{14}} &= k_{Ca_PLC1_f}Ca_{post}(t)x_{43} - k_{Ca_PLC1_b}x_{14} - k_{G_PLC2_f}x_{28}x_{14} \\
&\quad + k_{G_PLC2_b}x_9 - k_{DAG_1f}x_{40}x_{14} + k_{DAG_1b}x_{12} + k_{DAG_2f}x_6 + k_{GAP_2f}x_9 \\
\dot{x_{15}} &= k_{PMCA_f}Ca_{post}(t)x_{44} - k_{PMCA_b}x_{15} - k_{PMCA_c}x_{15} \\
\dot{x_{16}} &= k_{CaM_1f}Ca^2_{post}(t)x_{20} - k_{CaM_1b}x_{16} - k_{CaM_2f}Ca^2_{post}(t)x_{16} + k_{CaM_2b}x_{18} \\
\dot{x_{17}} &= k_{CaM_3f}Ca^2_{post}(t)x_{20} - k_{CaM_3b}x_{17} - k_{CaM_4f}Ca^2_{post}(t)x_{17} + k_{CaM_4b}x_{18} \\
\dot{x_{18}} &= k_{CaM_2f}Ca^2_{post}(t)x_{16} - k_{CaM_2b}x_{18} + k_{CaM_4f}Ca^2_{post}(t)x_{17} - k_{CaM_4b}x_{18} \\
\dot{x_{19}} &= -k_{buffer_f}Ca_{post}(t)x_{19} + k_{buffer_b}x_3 \\
\dot{x_{20}} &= -k_{CaM_1f}Ca^2_{post}(t)x_{20} + k_{CaM_1b}x_{16} - k_{CaM_3f}Ca^2_{post}(t)x_{20} + k_{CaM_3b}x_{17}
\end{aligned}
$$

$$\dot{x_{21}} = k_{DAG_2f}x_6 + k_{DAG_4f}x_5 - k_{prodAG_f}x_{21}x_7 + k_{prodAG_b}x_4 - k_{inacDAG_f}x_{21}x_{23} + k_{inacDAG_b}x_{22} - k_{DAG_PKC_f}x_{21}x_{13} + k_{DAG_PKC_b}x_{41}$$

$$\dot{x_{22}} = k_{inacDAG_f}x_{21}x_{23} - k_{inacDAG_b}x_{22} - k_{inacDAG_c}x_{22}$$

$$\dot{x_{23}} = -k_{inacDAG_f}x_{21}x_{23} + k_{inacDAG_b}x_{22}$$

$$\dot{x_{24}} = -k_{DAGL_f}Ca_{post}(t)x_{24} + k_{DAGL_b}x_7$$

$$\dot{x_{25}} = -k_{G_actt_f}x_{25}x_{30} + k_{G_actt_b}x_{26} + k_{regenG_f}x_{27}$$

$$\dot{x_{26}} = k_{G_actt_f}x_{25}x_{30} - k_{G_actt_b}x_{26} - k_{G_actt_c}x_{26}$$

$$\dot{x_{27}} = k_{GAP_1f}x_{29} + k_{GAP_2f}x_9 + k_{hydrG_f}x_{28} - k_{regenG_f}x_{27}$$

$$\dot{x_{28}} = k_{G_actt_c}x_{26} - k_{G_PLC2_f}x_{28}x_{14} + k_{G_PLC2_b}x_9 - k_{G_PLC1_f}x_{28}x_{43} + k_{G_PLC1_b}x_{29} - k_{hydrG_f}x_{28}$$

$$\dot{x_{29}} = k_{G_PLC1_f}x_{28}x_{43} - k_{G_PLC1_b}x_{29} - k_{Ca_PLC2_f}Ca_{post}(t)x_{29} + k_{Ca_PLC2_b}x_9 - k_{GAP_1f}x_{29}$$

$$\dot{x_{30}} = k_{mGluR_f}Glu_{ext}(t)x_{37} - k_{mGluR_b}x_{30} - k_{mGluR_desf}x_{30} + k_{mGluR_desb}x_{31} - k_{G_actt_f}x_{25}x_{30} + k_{G_actt_b}x_{26} + k_{G_actt_c}x_{26}$$

$$\dot{x_{31}} = k_{mGluR_desf}x_{30} - k_{mGluR_desb}x_{31}$$

$$\dot{x_{32}} = -k_{Glu_b}x_{32} + k_{Glu_f}Glu_{ext}(t)$$

$$\dot{x_{33}} = k_{DAG_1c}x_{12} + k_{DAG_3c}x_8 - k_{degIP3_f}x_{33}$$

$$\dot{x_{34}} = k_{degIP3_f}x_{33} - k_{PIP2_f}x_{34}x_{39} + k_{PIP2_b}x_{35}$$

$$\dot{x_{35}} = k_{PIP2_f}x_{34}x_{39} - k_{PIP2_b}x_{35} - k_{PIP2_c}x_{35}$$

$$\dot{x_{36}} = -k_{Leak_f}x_2x_{36} + k_{Leak_b}x_{10} + k_{Leak_c}x_{10}$$

$$\dot{x_{37}} = -k_{mGluR_f}Glu_{ext}(t)x_{37} + k_{mGluR_b}x_{30}$$

$$\dot{x_{38}} = -k_{NCX_f}Ca_{post}(t)x_{38} + k_{NCX_b}x_{11} + k_{NCX_c}x_{11}$$

$$\dot{x_{39}} = -k_{PIP2_f}x_{34}x_{39} + k_{PIP2_b}x_{35} + k_{PIP2_c}x_{35}$$

$$\dot{x_{40}} = -k_{DAG_1f}x_{40}x_{14} + k_{DAG_1b}x_{12} - k_{DAG_3f}x_9x_{40} + k_{DAG_3b}x_8 + k_{PIP2_c}x_{35}$$

$$\dot{x_{41}} = k_{DAG_PKC_f}x_{21}x_{13} - k_{DAG_PKC_b}x_{41}$$

$$\dot{x_{42}} = -k_{Ca_PKC_f}Ca_{post}(t)x_{42} + k_{Ca_PKC_b}x_{13}$$

$$\dot{x_{43}} = -k_{Ca_PLC1_f}Ca_{post}(t)x_{43} + k_{Ca_PLC1_b}x_{14} - k_{G_PLC1_f}x_{28}x_{43} + k_{G_PLC1_b}x_{29} + k_{GAP_1f}x_{29}$$

$$\dot{x_{44}} = -k_{PMCA_f}Ca_{post}(t)x_{44} + k_{PMCA_b}x_{15} + k_{PMCA_c}x_{15}$$

A.1.3 Constants of the Model

Constant	Value	Unit
k_{PMCA_f}	0.05	$1\ /\ (ms\mu M)$
k_{PMCA_b}	$7e-3$	$1\ /\ ms$
k_{PMCA_c}	$3.5e-3$	$1\ /\ ms$
k_{NCX_f}	0.0168	$1\ /\ (ms\mu M)$
k_{NCX_b}	$11.2e-3$	$1\ /\ ms$
k_{NCX_c}	$5.6e-3$	$1\ /\ ms$
k_{Leak_f}	0.0015	$1\ /\ (ms\mu M)$
k_{Leak_b}	$1.1e-3$	$1\ /\ ms$
k_{Leak_c}	$1.1e-3$	$1\ /\ ms$
k_{buffer_f}	0.028	$1\ /\ (ms\mu M)$
k_{buffer_b}	$19.6e-3$	$1\ /\ ms$
k_{CaM_1f}	0.006	$1\ /\ (ms\mu M)$
k_{CaM_1b}	$9.1e-3$	$1\ /\ ms$
k_{CaM_2f}	0.1	$1\ /\ (ms\mu M)$
k_{CaM_2b}	1	$1\ /\ ms$
k_{CaM_3f}	0.1	$1\ /\ (ms\mu M)$
k_{CaM_3b}	1	$1\ /\ ms$
k_{CaM_4f}	0.006	$1\ /\ (ms\mu M)$
k_{CaM_4b}	$9.1e-3$	$1\ /\ ms$
k_{Glu_f}	$2e-3$	$1\ /\ ms$
k_{Glu_b}	$2e-8$	$1\ /\ ms$
k_{mGluR_f}	0.0001	$1\ /\ (ms\mu M)$
k_{mGluR_b}	$10e-3$	$1\ /\ ms$
k_{mGluR_desf}	$0.25e-3$	$1\ /\ ms$
k_{mGluR_desb}	$0.001e-3$	$1\ /\ ms$
$k_{G_act_f}$	0.015	$1\ /\ (ms\mu M)$
$k_{G_act_b}$	$7.2e-3$	$1\ /\ ms$
$k_{G_act_c}$	$0.5e-3$	$1\ /\ ms$
$k_{Ca_PLC1_f}$	0.02	$1\ /\ (ms\mu M)$
$k_{Ca_PLC1_b}$	$120e-3$	$1\ /\ ms$
$k_{G_PLC2_f}$	0.1	$1\ /\ (ms\mu M)$
$k_{G_PLC2_b}$	$10e-3$	$1\ /\ ms$
$k_{G_PLC1_f}$	0.01	$1\ /\ (ms\mu M)$
$k_{G_PLC1_b}$	$12e-3$	$1\ /\ ms$
$k_{Ca_PLC2_f}$	0.08	$1\ /\ (ms\mu M)$
$k_{Ca_PLC2_b}$	$40e-3$	$1\ /\ ms$

$k_{DAG_1 f}$	0.0006	$1\ /\ (ms\mu M)$
$k_{DAG_1 b}$	$10e-3$	$1\ /\ ms$
$k_{DAG_1 c}$	$25e-3$	$1\ /\ ms$
$k_{DAG_2 f}$	$200e-3$	$1\ /\ ms$
$k_{DAG_3 f}$	0.015	$1\ /\ (ms\mu M)$
$k_{DAG_3 b}$	$75e-3$	$1\ /\ ms$
$k_{DAG_3 c}$	$250e-3$	$1\ /\ ms$
$k_{DAG_4 f}$	1	$1\ /\ ms$
k_{degIP3_f}	$10e-3$	$1\ /\ ms$
k_{PIP2_f}	0.002	$1\ /\ (ms\mu M)$
k_{PIP2_b}	$1e-3$	$1\ /\ ms$
k_{PIP2_c}	$1e-3$	$1\ /\ ms$
$k_{GAP_1 f}$	$30e-3$	$1\ /\ ms$
$k_{GAP_2 f}$	$30e-3$	$1\ /\ ms$
k_{hydrG_f}	$1e-3$	$1\ /\ ms$
k_{regenG_f}	$10e-3$	$1\ /\ ms$
k_{DAGL_f}	0.125	$1\ /\ (ms\mu M)$
k_{DAGL_b}	$50e-3$	$1\ /\ ms$
k_{prodAG_f}	0.0025	$1\ /\ (ms\mu M)$
k_{prodAG_b}	$1.5e-3$	$1\ /\ ms$
k_{prodAG_c}	$1e-3$	$1\ /\ ms$
k_{degAG_f}	$5e-3$	$1\ /\ ms$
$k_{inacDAG_f}$	0.0007	$1\ /\ (ms\mu M)$
$k_{inacDAG_b}$	$40e-3$	$1\ /\ ms$
$k_{inacDAG_c}$	$10e-3$	$1\ /\ ms$
$k_{Ca_PKC_f}$	0.02	$1\ /\ (ms\mu M)$
$k_{Ca_PKC_b}$	$50e-3$	$1\ /\ ms$
$k_{DAG_PKC_f}$	$1.5e-5$	$1\ /\ (ms\mu M)$
$k_{DAG_PKC_b}$	$0.15e-3$	$1\ /\ ms$

Table A.2: Constants included in the model presented in [3]

A.1.4 Non-Zero Initial Values

Variable	Value	Unit
x_2	2015.1	μM
x_3	7.648	μM
x_4	0.4	μM

x_{11}	0.784	μM
x_{15}	0.178	μM
x_{16}	0.06	μM
x_{17}	0.06	μM
x_{19}	153.290	μM
x_{20}	7.94	μM
x_{23}	1.4	μM
x_{24}	2.4	μM
x_{25}	3.5	μM
x_{32}	1019.1	μM
x_{34}	1.2	μM
x_{35}	0.8	μM
x_{36}	0.6	μM
x_{37}	5	μM
x_{38}	14.980	μM
x_{39}	0.6	μM
x_{40}	48	μM
x_{42}	15	μM
x_{43}	1	μM
x_{44}	0.659	μM

Table A.3: The employed initial values that are different from zero in the model developed in [3]

A.2 Matlab Code

The following Matlab codes have been used to produce the results presented in this thesis.

A.2.1 Kim Model Creation

```
%% Constants:

kPMCAf = .05;
kPMCAb = 7e-3;
kPMCAc = 3.5e-3;
kNCXf = .0168;
kNCXb = 11.2e-3;
kNCXc = 5.6e-3;
kLeakf = .0015;
kLeakb = 1.1e-3;
```

```
kLeakc = 1.1e-3;
kbufferf = .028;
kbufferb = 19.6e-3;
kCaM1f = .006;
kCaM1b = 9.1e-3;
kCaM2f = .1;
kCaM2b = 1;
kCaM3f = .1;
kCaM3b = 1;
kCaM4f = .006;
kCaM4b = 9.1e-3;
kGluf = 2e-3;
kGlub = 2e-8;
kmGluRf = .0001;
kmGluRb = 10e-3;
kmGluRdesf = .25e-3;
kmGluRdesb = .001e-3;
kGactf = .015;
kGactb = 7.2e-3;
kGactc = .5e-3;
kCaPLC1f = .02;
kCaPLC1b = 120e-3;
kGPLC2f = .1;
kGPLC2b = 10e-3;
kGPLC1f = .01;
kGPLC1b = 12e-3;
kCaPLC2f = .08;
kCaPLC2b = 40e-3;
kDAG1f = .0006;
kDAG1b = 10e-3;
kDAG1c = 25e-3;
kDAG2f = 200e-3;
kDAG3f = .015;
kDAG3b = 75e-3;
kDAG3c = 250e-3;
kDAG4f = 1;
kdegIP3f = 10e-3;
kPIP2f = .002;
kPIP2b = 1e-3;
kPIP2c = 1e-3;
kGAP1f = 30e-3;
kGAP2f = 30e-3;
khydrGf = 1e-3;
kregenGf = 10e-3;
kDAGLf = 0.125;
kDAGLb = 50e-3;
kprodAGf = .0025;
```

```
kprodAGb = 1.5e-3;
kprodAGc = 1e-3;
kdegAGf = 5e-3;
kinacDAGf = .0007;
kinacDAGb = 40e-3;
kinacDAGc = 10e-3;
kCaPKCf = .02;
kCaPKCb = 50e-3;
kDAGPKCf = 1.5e-5;
kDAGPKCb = .15e-3;

%% Initial states

x0 = sparse(44,1);
x0(2) = 2015.1;
x0(3) = 7.648;
x0(4) = .4;
x0(11) = .784;
x0(15) = .178;
x0(16) = .06;
x0(17) = .06;
x0(19) = 153.290;
x0(20) = 7.94;
x0(23) = 1.4;
x0(24) = 2.4;
x0(25) = 3.5;
x0(32) = 1019.1;
x0(34) = 1.2;
x0(35) = .8;
x0(36) = .6;
x0(37) = 5;
x0(38) = 14.980;
x0(39) = .6;
x0(40) = 48;
x0(42) = 15;
x0(43) = 1;
x0(44) = 0.659;


%%
% Linear autonomous part
A0 = sparse(44,44);

A0(1,[1 4]) = [-kdegAGf kprodAGc];
A0(2,[10 11 15]) = [kLeakb kNCXc kPMCAc];
A0(3,3) = -kbufferb;
A0(4,4) = -kprodAGb-kprodAGc;
```

```
A0(5,[5 8]) = [-kDAG4f kDAG3c];
A0(6,[6 12]) = [-kDAG2f kDAG1c];
A0(7,[4 7]) = [kprodAGb+kprodAGc -kDAGLb];
A0(8,8) = -kDAG3b-kDAG3c;
A0(9,[5 8 9]) = [kDAG4f kDAG3b -kGAP2f-kCaPLC2b-kGPLC2b];
A0(10,10) = -kLeakb-kLeakc;
A0(11,11) = -kNCXb-kNCXc;
A0(12,12) = -kDAG1b-kDAG1c;
A0(13,[13 41]) = [-kCaPKCb kDAGPKCb ];
A0(14,[6 9 12 14]) = [kDAG2f kGAP2f+kGPLC2b kDAG1b -kCaPLC1b ];
A0(15,15) = -kPMCAb-kPMCAc;
A0(16,[16 18]) = [-kCaM1b kCaM2b];
A0(17,[17 18]) = [-kCaM3b kCaM4b];
A0(18,18) = -kCaM2b-kCaM4b;
A0(19,3) = kbufferb;
A0(20,[16 17]) = [kCaM1b kCaM3b];
A0(21,[4 5 6 22 41]) = [kprodAGb kDAG4f kDAG2f kinacDAGb kDAGPKCb];
A0(22,22) = -kinacDAGb-kinacDAGc;
A0(23,22) = kinacDAGb;
A0(24,7) = kDAGLb;
A0(25,[26 27]) = [kGactb kregenGf];
A0(26,26) = -kGactb-kGactc;
A0(27,[9 27 28 29]) = [kGAP2f -kregenGf khydrGf kGAP1f];
A0(28,[9 26 28 29]) = [kGPLC2b kGactc -khydrGf kGPLC1b];
A0(29,[9 29]) = [kCaPLC2b -kGPLC1b-kGAP1f];
%A0(30,[30 31 33]) = [-kGluf kmGluRb kGlub];
A0(30,[26 30 31]) = [kGactb+kGactc -kmGluRb-kmGluRdesf kmGluRdesb];
A0(31,[30 31]) = [kmGluRdesf -kmGluRdesb];
A0(32,32) = -kGlub; %%MUUTA GLU
A0(33,[8 12 33]) = [kDAG3c kDAG1c -kdegIP3f];
A0(34,[33 35]) = [kdegIP3f kPIP2b];
A0(35,35) = -kPIP2b-kPIP2c;
A0(36,10) = kLeakb+kLeakc;
A0(37,30) = kmGluRb;
A0(38,11) = kNCXb+kNCXc;
A0(39,35) = kPIP2b+kPIP2c;
A0(40,[8 12 35]) = [kDAG3b kDAG1b kPIP2c];
A0(41,41) = -kDAGPKCb;
A0(42,13) = kCaPKCb;
A0(43,[14 29]) = [kCaPLC1b kGPLC1b+kGAP1f];
A0(44,15) = kPMCAb+kPMCAc;

%%
% Linear time-dependent part
% Coefficients for the first power of Capost(t)
A1 = sparse(44,44);
A1(3,19) = kbufferf;
```

```
A1(7,24) = kDAGLf;
A1(9,29) = kCaPLC2f;
A1(11,38) = kNCXf;
A1(13,42) = kCaPKCf;
A1(14,43) = kCaPLC1f;
A1(15,44) = kPMCAf;
A1(19,19) = -kbufferf;
A1(24,24) = -kDAGLf;
A1(29,29) = -kCaPLC2f;
A1(38,38) = -kNCXf;
A1(42,42) = -kCaPKCf;
A1(43,43) = -kCaPLC1f;
A1(44,44) = -kPMCAf;

% Coefficients for the second power of Capost(t)
A2 = sparse(44,44);
A2(16,[16 20]) = [-kCaM2f kCaM1f];
A2(17,[17 20]) = [-kCaM4f kCaM3f];
A2(18,[16 17]) = [kCaM2f kCaM4f];
A2(20,20) = -kCaM1f-kCaM3f;

% Coefficients for Glu_ext
A3 = sparse(44,44);
A3(30,37) = kmGluRf;
A3(37,37) = -kmGluRf;

Capost = @(t) 0;
Gluext = @(t) 0;

A_lin = @(t) A0+A1*Capost(t)+A2*Capost(t)^2+A3*Gluext(t);

%Input matrix B
B = sparse(44,1); %44 x r is 44 x 1 because Gluext is the only input
%u is r x 1, where r is amount of inputs r = 1
B(32,1) = kGluf;


%Put functions to cells for DEIM
func = {...
@(x)0; ... % 1
@(x)-kLeakf*x(2)*x(36); ... % 2
@(x)0; ... % 3
@(x)kprodAGf*x(7)*x(21); ... % 4
@(x)0; ... % 5
@(x)0; ... % 6
@(x)-kprodAGf*x(7)*x(21); ... % 7
@(x)kDAG3f*x(9)*x(40); ... % 8
```

```
@(x)kGPLC2f*x(14)*x(28)-kDAG3f*x(9)*x(40); ... % 9
@(x)kLeakf*x(2)*x(36); ... % 10
@(x)0; ... % 11
@(x)kDAG1f*x(14)*x(40); ... % 12
@(x)-kDAGPKCf*x(21)*x(13); ... % 13
@(x)-kGPLC2f*x(14)*x(28)-kDAG1f*x(14)*x(40); ... % 14
@(x)0; ... % 15
@(x)0; ... % 16
@(x)0; ... % 17
@(x)0; ... % 18
@(x)0; ... % 19
@(x)0; ... % 20
@(x)-kprodAGf*x(7)*x(21)-kinacDAGf*x(21)*x(23)-kDAGPKCf*x(13)*x(21); ... % 21
@(x)kinacDAGf*x(21)*x(23); ... % 22
@(x)-kinacDAGf*x(21)*x(23); ... % 23
@(x)0; ... % 24
@(x)-kGactf*x(25)*x(30); ... % 25
@(x)kGactf*x(25)*x(30); ... % 26
@(x)0; ... % 27
@(x)-kGPLC2f*x(14)*x(28)-kGPLC1f*x(28)*x(43); ... % 28
@(x)kGPLC1f*x(28)*x(43); ... % 29
%-kmGluRf*x(30)*x(38); ... % 30 vanha
@(x)-kGactf*x(25)*x(30); ... % 30
@(x)0; ... % 31
@(x)0; ... % 32
@(x)0; ... % 33
@(x)-kPIP2f*x(34)*x(39); ... % 34
@(x)kPIP2f*x(34)*x(39); ... % 35
@(x)-kLeakf*x(2)*x(36); ... % 36
@(x)-kmGluRf*x(30)*x(37); ... % 37
@(x)0; ... % 38
@(x)-kPIP2f*x(34)*x(39); ... % 39
@(x)-kDAG1f*x(14)*x(40)-kDAG3f*x(9)*x(40); ... % 40
@(x)kDAGPKCf*x(13)*x(21); ... % 41
@(x)0; ... % 42
@(x)-kGPLC1f*x(28)*x(43); ... % 43
@(x)0 ... % 44
};

F = @(y)cellfun(@(x)x(y),func);

%% Simulation

tspan = [0 10^4];

%tic
sol = ode15s(@(t,x) (A0+A1*Capost_sine(t)+A2*Capost_sine(t).^2+...
```

```
    A3*Gluext_sine(t))*x+F(x)+B*Gluext_sine(t),tspan,x0);
%disp('original = ', num2str(toc))

tt = linspace(tspan(1),tspan(2),100);
xx = deval(sol,tt);

plot(tt,xx(1,:));

%%
plot(tt,xx(20,:));


%% Analysis

figure(1);
spy(abs(A1)+abs(A2));

figure(2);
spy(A0);

eig(full(A0));

%%
semilogy(1:44,svd(full(A0)),'b.','Markersize',10);
```

A.2.2 Full Model Solver

```
function dy = Kim_full(t, x, A0, A1, A2, A3, F, B)
%   A linear part
%   A1 linear calciums
%   A2 linear calsium.^2
%   B input weights
%   F nonlinear part as a vector of functions

Ca = Capost_sine(t);
Glu = Gluext_sine(t);

dy = (A0+A1*Ca+A2*Ca.^2+A3*Glu)*x+F(x)+B*Glu;

end
```

A.2.3 Calcium Stimulus

```
function [ y ] = Capost_sine( t )
```

```
%CAPOST_SINE Amplitude of calcium stimulus at time t

if ~isvector(t) && ~isscalar(t)
   error('Input should be a vector or scalar');
end

startThreshold = 10;
endThreshold = 15;
stimulusAmplitude = 1.5 / 2; %muM
steps = numel(t);
y = zeros(1,steps);

for n = 1:steps
    if t(n) > startThreshold && t(n) < endThreshold
       y(n) = stimulusAmplitude+stimulusAmplitude*sin(-pi*t(n));
    end
end
end
```

A.2.4 Glutamate Stimulus

```
function [ y ] = Gluext_sine( t )
%%GLUTEXT_SINE Amplitude of glutamate stimulus at time t

if ~isvector(t) && ~isscalar(t)
   error('Input should be a vector or scalar');
end

startThreshold = 10;
endThreshold = 15;
stimulusAmplitude = 100 / 2; %muM
steps = numel(t);
y = zeros(1,steps);

for n = 1:steps
    if t(n) > startThreshold && t(n) < endThreshold
        y(n) = stimulusAmplitude+stimulusAmplitude*sin(-pi*t(n));
    end
end
end
```

A.2.5 POD Algorithm

```
function [ basis ] = calculate_POD( Y, order )
```

```
%CALCULATE_POD returns a POD projection basis from the give solution matrix
%   Y the solution matrix with timesteps in columns and variables in rows
%   order is the desired order of the projection basis

snapshot_interval = 5;

%Calculate the basis by the method of snapshots
snapshots = Y(:,1:snapshot_interval:end);
[U,V,~] = svd(snapshots);

% figure;
% plot(diag(V));
% title('Singular values of solution snapshots')
% xlabel('Index of value')
% ylabel('Magnitude')

basis = U(:,1:order);

end
```

A.2.6 DEIM Algorithm

```
function [ P, U, ind ] = calculate_DEIM( solutions, dimension, nonlinHandle )
%CALCULATE_DEIM Calculates the DEIM matrices and indices
%   SOLUTIONS solutions to the full system
%   DIMENSION desired dimension of the result
%   NONLINHANDLE function handle to the nonlinear part of the system

%Generate values for the nonlinear function
vals = zeros(size(solutions));
snapshot_step = 5;

for n = 1:size(solutions,2)

    vals(:,n) = nonlinHandle(solutions(:,n));

end

[U, V, ~] = svd(vals(:,1:snapshot_step:end));
U = U(:,1:dimension);
[P, ind] = deim(U);
P = sparse(P);

end
```

A.2.7 DEIM Reduced Model

```
function [ dy ] = Kim_reduced_DEIM( t, y, A_red, A1, A2, A3,...
B, F, V, N)
%   A_red reduced linear part
%   A1 linear calciums
%   A2 linear calsium.^2
%   B input weights
%   F nonlinear part as a vector of functions
%   V projection basis
%   N the nonlinear projection basis precomputed V'*U*inv(P'U)

Ca = Capost_sine(t);
Glu = Gluext_sine(t);

dy = (A_red + A1*Ca + A2*Ca.^2 + A3*Glu)*y + N*(F(V*y)) + B*Glu;

end
```

A.2.8 Predict Reduction Results

```
Kim_model_44_cells;
close all;

dimensions = 2:2:40;
DEIM_dimensions = 2:3:40;
num_runs = 20;
mean_times = zeros(numel(DEIM_dimensions), numel(dimensions));
sqr_errors = zeros(numel(DEIM_dimensions), numel(dimensions));
sol_rows = size(sol.y, 1);

%We need to interpolate the results in order to calculate errors
tFine = 0:tspan(2);
mean_original_res = zeros(sol_rows, numel(tFine));

%% Calculating groundtruths for simulation results and time

reduced = zeros(1,num_runs);
for k = 1:num_runs

tic
sol = ode15s(@(t,x) Kim_full(t, x, A0, A1, A2, A3, F, B),tspan,x0);
reduced(k) = toc;
%save the results for averaging since the solver is stochastic
```

```
mean_original_res = mean_original_res + deval(sol, tFine);

end
original_time = mean(reduced);
mean_original_res = mean_original_res./num_runs; %average concentrations

%% Calculating the errors and run times of the reduced models

for n = 1:numel(dimensions)
    disp(strcat(...
        'Calculating error and simulation time with dimension ', ...
        ' ', num2str(dimensions(n))));
    num_dims = dimensions(n);

    %Get the basis for this new dimension
    basis = calculate_POD(sol.y, num_dims);
    A_red = basis'*A0*basis;
    A1_red = basis'*A1*basis;
    A2_red = basis'*A2*basis;
    A3_red = basis'*A3*basis;
    B_red = basis'*B;

    %Run another loop to go through DEIM dimensions
    for m = 1:numel(DEIM_dimensions)
        mean_err_sol = zeros(44, numel(tFine));
        reduced = zeros(1,num_runs);

        [P, U, ind] = calculate_DEIM(sol.y, DEIM_dimensions(m), F);
        %Run a loop to get a mean of computing times for this POD dimension
        F_red = @(y)cellfun(@(x)x(y),func(ind));
        %pick the functions shown by DEIM
        N = basis'*U*inv(P'*U); %nonlinear basis
        for k = 1:num_runs

            tic
            sol_red = ode15s(@(t,y) Kim_reduced_DEIM(t,y,A_red, A1_red, ...
                A2_red, A3_red, B_red, F_red, basis, N), ...
                tspan, basis'*x0);
            reduced(k) = toc;
            %We need to interpolate to matching dimensions
            try
                mean_err_sol = mean_err_sol + basis*deval(sol_red, tFine);
            catch
                mean_err_sol = mean_err_sol + interp1(sol_red.x, ...
                    (basis*sol_red.y)', tFine, 'spline')';
            end
        end
```

```
        %Store the results of DEIM to the rows. Each column corresponds to
        %a POD dimension. So plotting the first row gives the same DEIM
        %dimension for all POD dimensions
        mean_times(m,n) = mean(reduced);
        %Calculate the total squared error
        sqr_errors(m,n) = sqrt(mean((mean_original_res(:) - (mean_err_sol(:)./nur
    end
end

%% plotting

figure;
plot(dimensions, mean_times);
title('Mean simulation times (20 runs)')
ylabel('time (s)')
xlabel('POD dimension')
hold on
plot(original_time*ones(1,max(dimensions)))
legendCell = cellstr(num2str(DEIM_dimensions', 'DEIM=%-d'));
legend([legendCell; {'original'}])
hold off
figure;
semilogy(dimensions, sqr_errors)
title('RMS error for all variables summed')
ylabel('error')
xlabel('POD dimension')
legend(legendCell)
```

A.2.9 Plot Results

```
clear all;
Kim_model_44_cells;
close all;

%% Test some dimensions
basis = calculate_POD(sol.y,10);
A_red = basis'*A0*basis;
A1_red = basis'*A1*basis;
A2_red = basis'*A2*basis;
A3_red = basis'*A3*basis;
B_red = basis'*B;

[P, U, ind] = calculate_DEIM(sol.y, 10, F);

F_red = @(y)cellfun(@(x)x(y),func(ind)); %pick the functions shown by DEIM
N = basis'*U*inv(P'*U); %nonlinear basis
```

```
%Using the time span of the original model
tFine = 0:tspan(2);

tic
sol_red = ode15s(@(t,y) Kim_reduced_DEIM(t,y,A_red, A1_red, ...
                A2_red, A3_red, B_red, F_red, basis, N), ...
                tspan, basis'*x0);
toc
tic
sol = ode15s(@(t,x) Kim_full(t, x, A0, A1, A2, A3, F, B),tspan,x0);
toc

yred = basis*deval(sol_red, tFine);
yorig = deval(sol, tFine);

figure;
subplot(5,1,1);
hold on
plot(tFine, yorig(1,:));
plot(tFine, yred(1,:));
title('Output variables')
ylabel('AG_{post}')
subplot(5,1,2);
hold on
plot(tFine, yorig(2,:));
plot(tFine, yred(2,:));
ylabel('Ca_{ext}')
subplot(5,1,3);
hold on
plot(tFine, yorig(21,:));
plot(tFine, yred(21,:));
ylabel('DAG_{post}')
subplot(5,1,4);
hold on
plot(tFine, yorig(25,:));
plot(tFine, yred(25,:));
ylabel('Gabg_{post}')
subplot(5,1,5);
hold on
plot(tFine,yorig(43,:));
plot(tFine,yred(43,:));
ylabel('PLC_{post}')
xlabel('Time (s)')
```

A.2.10 Plot Long Interval Results

```
Kim_model_44_cells;
close all;

%% plot errors for a really long time

PODdims = [10, 30];
DEIMdims = [5, 10];
lim = 5*10^4; %Set the time used for training AND simulation
time = [0 lim];
res = cell(1,2);
grid = time(1):10:time(2);

tic
sol2 = ode15s(@(t,x) Kim_full(t, x, A0, A1, A2, A3, F, B),time,x0);
toc
yorig2 = deval(sol2, grid);
indices = []
for n = 1:2

basis = calculate_POD(sol2.y, PODdims(n));
A_red = basis'*A0*basis;
A1_red = basis'*A1*basis;
A2_red = basis'*A2*basis;
A3_red = basis'*A3*basis;
B_red = basis'*B;

[P, U, ind] = calculate_DEIM(sol2.y, DEIMdims(n), F);

%Run a loop to get a mean of computing times for this POD dimension

F_red = @(y)cellfun(@(x)x(y),func(ind)); %pick the functions shown by DEIM
%F_red = getFunHandles(func, ind); %not for the cell version
N = basis'*U*inv(P'*U); %nonlinear basis

tic
sol_red2 = ode15s(@(t,y) Kim_reduced_DEIM(t,y,A_red, A1_red, ...
                A2_red, A3_red, B_red, F_red, basis, N), ...
                time, basis'*x0);
toc

yred2 = basis*deval(sol_red2, grid);
res{n} = yred2;
indices = [indices ind'];
end

yred2 = res{1};
```

```
figure;
subplot(5,2,1);
hold on
plot(grid, yorig2(1,:));
plot(grid, yred2(1,:));
title('Output variables (POD 10 DEIM 5)')
ylabel('AG_{post}')
subplot(5,2,3);
hold on
plot(grid, yorig2(2,:));
plot(grid, yred2(2,:));
ylabel('Ca_{ext}')
subplot(5,2,5);
hold on
plot(grid, yorig2(21,:));
plot(grid, yred2(21,:));
ylabel('DAG_{post}')
subplot(5,2,7);
hold on
plot(grid, yorig2(25,:));
plot(grid, yred2(25,:));
ylabel('Gabg_{post}')
subplot(5,2,9);
hold on
plot(grid,yorig2(43,:));
plot(grid,yred2(43,:));
ylabel('PLC_{post}')
xlabel('Time (s)')

yred2 = res{2};

subplot(5,2,2);
hold on
plot(grid, yorig2(1,:));
plot(grid, yred2(1,:));
title('Output variables (POD 30 DEIM 10)')
ylabel('AG_{post}')
subplot(5,2,4);
hold on
plot(grid, yorig2(2,:));
plot(grid, yred2(2,:));
ylabel('Ca_{ext}')
subplot(5,2,6);
hold on
plot(grid, yorig2(21,:));
plot(grid, yred2(21,:));
ylabel('DAG_{post}')
```

```
subplot(5,2,8);
hold on
plot(grid, yorig2(25,:));
plot(grid, yred2(25,:));
ylabel('Gabg_{post}')
subplot(5,2,10);
hold on
plot(grid,yorig2(43,:));
plot(grid,yred2(43,:));
ylabel('PLC_{post}')
xlabel('Time (s)')


figure;
subplot(5,2,2);
bar(grid, abs(yorig2(1,:)-yred2(1,:))./(eps + abs(yorig2(1,:))),1);
xlim(time)
title('Relative errors in output variables (POD 30 DEIM 10)')
ylabel('AG_{post}')
axis tight
subplot(5,2,4);
bar(grid, abs(yorig2(2,:)-yred2(2, :))./(eps + abs(yorig2(2,:))),1);
xlim(time)
ylabel('Ca_{ext}')
axis tight
subplot(5,2,6);
bar(grid, abs(yorig2(21,:)-yred2(21,:))./(eps + abs(yorig2(21,:))),1);
xlim(time)
ylabel('DAG_{post}')
subplot(5,2,8);
bar(grid, abs(yorig2(25,:)-yred2(25,:))./(eps + abs(yorig2(25,:))),1);
xlim(time)
ylabel('Gabg_{post}')
subplot(5,2,10);
bar(grid, abs(yorig2(43,:)-yred2(43,:))./(eps + abs(yorig2(43,:))),1);
xlim(time)
ylabel('PLC_{post}')
xlabel('Time (s)')

subplot(5,2,1);
bar(grid, abs(yorig2(1,:)-yred2(1,:)),1);
xlim(time)
title('Absolute errors in output variables (POD 30 DEIM 10)')
ylabel('AG_{post}')
subplot(5,2,3);
bar(grid, abs(yorig2(2,:)-yred2(2, :)),1);
xlim(time)
```

```
ylabel('Ca_{ext}')
subplot(5,2,5);
bar(grid, abs(yorig2(21,:)-yred2(21,:)),1);
xlim(time)
ylabel('DAG_{post}')
subplot(5,2,7);
bar(grid, abs(yorig2(25,:)-yred2(25,:)),1);
xlim(time)
ylabel('Gabg_{post}')
subplot(5,2,9);
bar(grid, abs(yorig2(43,:)-yred2(43,:)),1);
xlim(time)
ylabel('PLC_{post}')
xlabel('Time (s)')
```