LUMPING METHODS FOR MODEL REDUCTION

BLESSING EBERECHUKWU OKEKE
Bachelor of Science in Mathematics, University of Jos, 2006
Master of Science in Industrial Mathematics, University of Hamburg, 2010

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Chemistry and Biochemistry
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

Dedication

To my ever loving and supportive husband Dr. Onyekwelu U. Okeke and to my lovely daughters Chiamaka N. Okeke and Ebube O. Okeke. This would not have been possible without your love and support.

Abstract

Modelling a chemical or biochemical system involves the use of differential equations which often include both fast and slow time scales. After the decay of transients, the behaviour of these differential equations usually rests on a low-dimensional surface in the phase space which is called the slow invariant manifold (SIM) of the flow. A model has been effectively reduced if such a manifold can be obtained. In this study, we develop a method that introduces the lumping process (a technique whereby chemical species are grouped into pseudo-reagents (lumps) to simplify modelling) into the invariant equation method, and this new method effectively reduces complex models as well as preserving the structure and underlying mechanism of the original system. We also apply these methods to simple models of metabolic pathways. This method of model reduction would be of great importance for industrial applications.

Acknowledgments

I would like to express my profound gratitude to my supervisor Professor Marc R. Roussel. Thank you for your guidance, encouragement and support throughout this research. I would also like to thank my supervisory committee members, Dr. Stacey Wetmore and Dr. David Kaminski for their feedback.

My special thanks goes to my husband, sweetheart, lover, prince and my better half. Thank you so much for your love and support throughout this program. I would not have made it through without you. I love you so much, and may our good Lord bless you so abundantly. Thanks to my lovely daughters Chiamaka N. Okeke and Ebube O. Okeke for the joy and laughter you brought to my life. You made this journey pleasant and I love you two so much.

Many thanks to my family. I am indebted to my parents Rev. Kenneth and Mrs. Grace Amadi for standing by me throughout these years. To my brother and sisters, Comfort Ndubuisi, Emmanuel Amadi and Mercy Amadi, your prayers made it worthwhile. And to my mentor, Mama Duke for your prayers and support. May God bless you all.

I would also like to acknowledge the financial support provided by my supervisor, the University of Lethbridge, and the School of Graduate Studies. Funding this research is well appreciated.

My special thanks goes to God Almighty, my Father, Lord and Savior. Your love toward me is unending and everlasting. You are the portion of my inheritance and you maintain my lots (Ps. 16:5). All thanks and praise be to your name for you grace and favour through this program.

Contents

List of Figures

List of Abbreviations

CSP	Computational Singular Perturbation
IE	Invariance Equation
ILDM	Intrinsic Low-Dimensional Manifold
QSSA	Quasi-Steady-State Approximation
SIM	Slow Invariant Manifold

Chapter 1

Motivation

Biochemical models are often very large, thus it is desirable to reduce the large sets of differential equations that make up these models to smaller systems. As biochemical data continues to increase, the need for model reduction gains in importance. Model reduction is important for some of the following reasons:

1. Biochemical simulations generate hundreds of differential equations, which bring about computational difficulties especially when a spatially inhomogeneous system is considered [51, 43, 50].

2. There is a limitation in our ability to interpret simulation results and use them due to the difficulty associated with understanding the interrelationships between variables in higher-dimensional spaces.

3. Model reduction facilitates the prediction and control of species in a system [21, 60].

Lumping is the reduction in the number of variables of a system by the introduction of a new set of variables, which is a function of the variables in the original set [12]. The technique whereby chemical species with similar properties are grouped into pseudo-reagents (lumps) to simplify modelling is a historical approach to lumping [11, 34]. In mathematical terms, lumping can be described as the reduction of an N-dimensional system [33]

$$\frac{dy}{dt} = f(y), \;\; y \in \mathrm{R}^N \tag{1.1}$$

to an $\hat{N}$-dimensional lumped set

$$\frac{d\hat{y}}{dt} = \hat{f}(\hat{y}), \tag{1.2}$$

where $\hat{N} < N$ and

$$\hat{y} = h(y), \tag{1.3}$$

where h is some function of the original variables y. The lumping is called exact or approximate depending on whether the solution of the lumped differential equations system does or does not contain errors compared to that of the original system. Some desirable properties of a lumping scheme are the following:

1. The solutions of the reduced model obtained from the lumping scheme gives an accurate representation of the solutions of the original model.

2. There is a possibility of calculating values of the original variables from the lumped variables at any point in time.

3. In the lumping process, one has the freedom to select variables for the reduced model that would aid in giving a clearer chemical interpretation of the original system.

In this thesis, I present an efficient lumping method for model reduction that would be of great importance for industrial applications [21, 43, 54]. In the lumping process, a large number of possible lumping matrices exists. Thus instead of testing each reduced scheme to check whether or not its dynamics matches that of the original system, there is a need to incorporate the knowledge about the time scales involved in a reaction to the lumping process [57].

Lumping will be implemented by solving the invariance equation (IE). The success of the IE method is based upon the existence of fast and slow time scales in the dynamics of a biochemical system. The effectiveness of this new method will be illustrated using the Michaelis-Menten mechanism [10, 26, 37] and a linear metabolic pathway model [47, 50]. In chapter 2, I will discuss some methods used for model reduction. In subsequent chapters, I present the methodology used for the formulation of an iterative algorithm that solves the invariance equation, present results for different ways of arbitrarily splitting the variables into one or two lumps, compare the trajectories of the full and reduced models and then study the discretization error of the model.

Chapter 2

Methods for Model Reduction

In this chapter, I will discuss some of the methods used in model reduction such as the quasi-steady-state approximation, the computational singular perturbation method and the intrinsic low-dimensional manifold method. I will also discuss other approaches to lumping such as the linear and nonlinear lumping methods for model reduction.

2.1 Quasi-Steady-State Approximation

The quasi-steady-state approximation (QSSA) is used in the study of chemical reactions when certain species have short life spans with respect to other species [3]. It describes the kinetics in a system of ordinary differential equations in which, after the decay of the fast time scales, some of the variables are referred to as being in a steady state[1] [52]. The QSSA assumes that the rate of change among highly reactive species is zero, which reduces the system of equations used for modeling [3].

In the precomputer era, this method was used to obtain approximate analytic solutions for kinetic differential equations, but since the advent of computers and advanced software it has been suggested that the QSSA is a redundant technique and that its application should be discontinued [57, 58]. Côme emphasized that it is still needed since it has been used to clarify most reaction mechanisms and to determine many rate coefficients of elementary processes [9]. The quasi-steady state approximation is still useful for the conversion of stiff systems[2] to non-stiff forms and for the reduction in the number of variables to be solved, since the use of chemical mechanisms in reactive flow calculations leads to high demands

[1]A true steady state is a situation in which all the state variables of a system are constant.

[2]The degree of stiffness of a vector $r(c)$ is given by the ratio $S(c) = \frac{\max \Re(-\alpha_i)}{\min \Re(-\alpha_i)}$, where α_i are the eigenvalues of the Jacobian matrix $\frac{\partial r_i}{\partial c}$. A large value of $S(c)$ indicates that the time constants of a system are spread over many orders of magnitude [7].

on even present computer power [3, 57].

The quasi-steady-state approximation is the zero-order approximation to a slow manifold[3] [6, 23, 47, 50]. It involves time-scale separation, in particular finding those species that react in a short time and whose fast motion is adiabatically coupled to the slow time evolution.

Given the following ordinary differential equation:

$$\frac{dc}{dt} = f(c,k), \quad c(0) = c^0 \tag{2.1}$$

where c is an n-dimensional concentration vector and $f(c,k)$ is a function of the reaction rates, the application of the QSSA involves setting some components of the differential equation (2.1) to zero, i.e. those corresponding to the QSSA (highly reactive) species. Equation (2.1) is replaced by the following differential-algebraic equation [57]:

$$\frac{dc^{(1)}}{dt} = f^{(1)}(c,k), \tag{2.2}$$

$$0 = f^{(2)}(c,k), \tag{2.3}$$

where $c^{(1)}$ and $c^{(2)}$ define the concentrations of the non-QSSA and the QSSA species respectively, and the rate of change of the non-QSSA and QSSA species are given by $f^{(1)}$ and $f^{(2)}$. Setting $f^{(2)}$ to zero reduces the number of differential equation since some have been replaced by algebraic equations.

An important characteristic of the QSSA is that from the concentration of other species in equation (2.3), one can determine the concentrations of the QSSA species [57]. The choice of the QSSA species is an important aspect of applying this technique, and Frank-Kamenetskii [19] was the first to introduce this idea. The difference between the non-

[3]A low-dimensional surface in phase space on which solutions evolve according to the slower time scales is called a slow manifold. A phase space is a space in which all possible states of a system are represented. For example, in an isothermal system, the concentrations are known as the phase-space variables.

steady-state species (2.1) and the algebraic equation (2.3) is the instantaneous error of the quasi-steady-state approximation. This error is introduced by the application of the QSSA to a single species concentration c_i which can be used to identify the possible steady-state species and is defined by [58] :

$$\Delta c_i^s = \frac{1}{J_{ii}} \frac{dc_i}{dt}, \tag{2.4}$$

where the Jacobian matrix of the differential equation is

$$J_{ik} = \left[\frac{\partial f_i(c,k)}{\partial c_k} \right]. \tag{2.5}$$

The lifetime of a species i is equal to the reciprocal of the diagonal Jacobian element of the species, i.e.

$$\tau_i = \frac{-1}{J_{ii}}. \tag{2.6}$$

It is important to study lifetimes since they can indicate possible QSSA species, and small errors stem from small lifetimes and/or a slow rate of change for a species [57].

The unavailability of a method to calculate error estimates other than through a simulation of the mechanism with and without the application of the quasi-steady-state approximation is one of the major drawbacks of this technique [58]. Another drawback with this method is its limitation in usage, i.e. it can only be applied for specific reaction systems and for the QSSA species selected. Also any investigation done on a particular system cannot be transfered to other systems [58].

2.2 Computational Singular Perturbation Method

The computational singular perturbation (CSP) method developed by Lam is used for analyzing reaction mechanisms and it enables the user to establish quasi-steady-state and partial equilibrium relationships without detailed chemical knowledge [27, 28, 29]. This method uses mathematical analysis of the different time scales that occur in biochemical systems and also computes the slow manifold [56, 60].

Considering the rate equation (2.1), the n-dimensional rate vector f can be written as:

$$f = \sum_{j}^{R} v_j R_j, \tag{2.7}$$

where v_j is a vector of stoichiometric coefficients for reaction j, R is the number of elementary reactions incorporated in the mechanism and R_j is the jth reaction rate. Let $a_i, i = 1, 2, \ldots, N$ be a set of N linearly independent column basis vectors with a set of N row vectors b^i as inverses of a_i and N as the dimension of phase space. Specifically, a_i and b^i satisfy the orthonormal condition:

$$b^i \cdot a_j = \delta^i_j, \;\; i, j = 1, 2, \ldots, N, \tag{2.8}$$

where $\cdot$ is the dot product operator and δ^i_j is the Kronecker delta. Note that the corresponding b^i is computed once a_i is chosen. The physical representation of equation (2.7) is not unique and as such it can be written as

$$f = \sum_{i=1}^{N} a_i d^i, \tag{2.9}$$

where

$$a_i = \sum_{k=1}^{N} v_k A^k_i, \; i = 1, \ldots, N, \tag{2.10}$$

and

$$d^i = b^i \cdot f = \sum_{j=1}^{R} (b^i \cdot v_j) R_j,\ i = 1, \ldots, N. \tag{2.11}$$

The computational singular perturbation method basically chooses A_i^k in an optimal way so that the rate of each reaction group is associated with a single time scale [57]. These reaction groups are usually in ascending order with the fastest group at the beginning. As the reaction occurs, the fast modes become exhausted, i.e. the rate of reaction of a particular group approaches zero (become dead modes) and can be removed from the right-hand side of equation (2.9) [57].

For a mechanism with E chemical elements and M dead modes, the right-hand side of equation (2.9) becomes [57]:

$$a_{M+1} d^{M+1} + a_{M+2} d^{M+2} + \cdots + a_{N-E} d^{N-E} \tag{2.12}$$

which represents a reduced mechanism of $N-(M+E)$ steps and can be interpreted as an $N-(M+E)$ dimensional manifold in an N-dimensional space.

The computational singular perturbation method can be used for reducing reaction mechanisms given its ability to identify important species and reactions [57]. The information about fast modes enables this method to be used in the identification of quasi-steady-state species and partial equilibrium assumptions as well as the identification of potential rate controlling reactions [56, 57]. A major disadvantage of this method is that the reduced models contain systems that are mathematically transformed differential or differential algebraic equations (DAE) which do not relate one-to-one to biochemical species, thus limiting biochemical interpretation [56].

2.3 Intrinsic Low-Dimensional Manifold

The intrinsic low-dimensional manifold (ILDM) is a method that generates kinetic systems that are simplified based on the insights from dynamical systems theory [35, 45]. This method needs no prior knowledge about reactions that are assumed to be in partial equilibrium nor about species that are assumed to be in steady state. It only requires information about the kinetic mechanism and the number of degrees of freedom required in the simplified scheme [35]. This method also allows the decoupling of the fast time scales of the chemical system, reducing the dimension of the state space as well as the stiffness, both of which lead to increased computational efficiency [35, 57].

Consider equation (2.1) in terms of vectors [35]:

$$\frac{dc}{dt} = f(c). \tag{2.13}$$

The eigenvalues of f_c, the Jacobian of $f(c)$, describe the different time scales in the state space, while the corresponding eigenvectors describe the characteristic directions associated with those time scales [35]. The fact that one can have an ill-conditioned matrix (where almost degenerate eigenvectors exist) results in numerical difficulties, requiring work with a modified set of basis vectors [22]. Maas and Pope used the Schur vectors as their basis set.

The Schur decomposition of a matrix is a transformation that results in a matrix with the eigenvalues on the diagonals (or in the 2 by 2 blocks in the case of complex pairs) in order of descending real parts [57]. The Schur decomposition is defined by [35]:

$$Q^* f_c Q = N \tag{2.14}$$

with $*$ denoting the conjugate transpose, N the resulting triangular matrix, and Q the Schur matrix:

$$Q = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \dots & q_n \\ | & | & & | \end{pmatrix}, \tag{2.15}$$

where q_i are the Schur basis vectors which are mutually orthogonal and have the property that the first p vectors of the Schur matrix Q span the same subspace as the first p eigenvectors of the Jacobian. We can then define the low-dimensional manifold as the set of points in state space for which

$$0 = Q_L^*(c)f(c) = \begin{pmatrix} - & q^*_{2+n_e+n_c+1} & - \\ - & q^*_{2+n_e+n_c+2} & - \\ & \vdots & \\ - & q^*_n & - \end{pmatrix} f(c). \tag{2.16}$$

Q_L^* is the submatrix of Q^* corresponding to the $n - n_e - n_c$ fastest variables with n_e, n_c as the number of chemical elements and slow manifold dimension respectively. The slow manifold can be defined by assuming that the motion in the direction of the Schur vectors associated with the fast variables is zero, and this means that the system is in local equilibrium with respect to its fastest time scales [57].

A major drawback of the ILDM is that it is not an exact method since it neglects the curvature of the manifold, i.e. it is not the same as the invariant manifold [57].

2.4 Previous Approaches to Lumping

The need for reducing a high-dimensional system to a smaller one to facilitate modelling has led to the development of the lumping method. The lumping method is a technique whereby chemical species are grouped into pseudo-reagents (lumps) to simplify modelling

[57]. This reduction process is based on the representation of groups of species by a single variable, hence the term ”species lumping”. The new variable formed is related to the original system by the lumping function, which could be either linear or nonlinear.

According to Tomlin and coworkers, some questions that come to mind about lumping are the following [57]:

1. How small can the lower-dimensional space be while maintaining high accuracy?
2. What is the best way of choosing a lumping function that represents the original system adequately and gives the smallest reduced scheme?

The following approaches to lumping provide some answers to the above questions.

2.4.1 Linear Lumping

Linear lumping is a form of lumping where the new variables formed are linear combinations of the original ones [57]. It is the simplest form of lumping and can be represented as follows:

$$\hat{c} = Mc \tag{2.17}$$

where M is an $\hat{N} \times N$ real constant matrix called the lumping matrix with $\hat{N} \ll N$.

The new $\hat{N}$ set of ordinary differential equations for the lumped system is given by:

$$\frac{d\hat{c}}{dt} = Mf(c). \tag{2.18}$$

For exact lumping, $Mf(c)$ must be a function of $\hat{c}$ so that the reduced system can be expressed as a function of the new variables. We need to know the pseudo-inverse of M since from equation (2.17) we have that

$$c = M^{-1}\hat{c}. \tag{2.19}$$

The inverse mapping from the $\hat{c}$ to c space is as important as the forward mapping because it provides a link between the original and lumped species [57]. Its existence is also a necessary and sufficient condition for exact lumping [33, 57].

Li and Rabitz [30, 31] as well as Wei and Kuo [59] have stated the conditions and given examples of techniques that involve exact and approximate linear lumping methods. For a linear system, it involves finding an appropriate lumping matrix of a given dimension and its pseudo-inverse, or finding an invariant subspace of the original equation [57]. The eigenvalues of the reduced system at a fixed point form a subset of the eigenvalues for the full equations and by choosing which eigenvalues are retained one can relate the full to the reduced system [57]. For example, a full system exhibiting an oscillatory behaviour [17, 18, 42] cannot have a lumped system with all the complex eigenvalues lumped together because the qualitative dynamics of the system would not be adequately represented in the reduced system.

In combustion where we have nonlinear systems, the transpose of the Jacobian $J^T(c)$ is not a constant matrix and hence it is difficult to convert it into a standard form. $J^T(c)$ can be expressed as:

$$J^T(c) = A_0 + \sum_{k=1}^{N} A_k a_k(c), \tag{2.20}$$

where $a_k(c)$ are functions of c which for non-isothermal systems will be simple polynomials. Li and Rabitz proved that the invariant subspaces of $J^T(c)$ are also invariant subspaces of A_0 and A_k [30]. The simplest procedure for finding invariant subspaces is to determine eigenvalues and eigenvectors of $\sum_{k=0}^{N} A_k$. Given an eigenvector matrix:

$$X = (x_1, x_2, \ldots, x_N), \tag{2.21}$$

where $x_1, x_2, \ldots, x_N$ are the columns of X, then the subspaces are given by the span of these columns. The lumping matrices M of different dimensions can then be formed by taking the columns of X or any linear combinations of them [57].

2.4.2 Nonlinear Lumping

Given that combustion systems are highly nonlinear, linear lumping does not provide the required degree of reduction that gives an efficient model [57]. There are two approaches to such cases: Firstly, we can consider the system to be linear locally and then apply the linear technique over short time periods. Here, different lumped schemes would be needed for different time periods. The disadvantage of this approach is that for ignition systems which are highly nonlinear, a large number of lumping schemes would be required to cover a desired reaction period. As a result of this, one would need to switch between different reduced schemes and this may slow down the calculation to the point where using a lumping scheme would no longer save computational time [57].

Secondly, one can develop a nonlinear lumping scheme which would give more flexibility on how the lumped species can be represented as a function of the original species. This technique provides a possibility of producing a reduced scheme that can be applied over the whole reaction zone. The disadvantage of using this approach is that the nonlinear analysis will involve complicated analytical theory. Li *et al.* [33] described some developments in this direction.

From equation (2.1), the new lumped variable is defined by an $\hat{n}$-dimensional nonlinear transformation $\hat{c} = h(c)$ and the new $\hat{n}$-dimensional equation system given as:

$$\frac{d\hat{c}}{dt} = \hat{f}(\hat{c}(t)). \tag{2.22}$$

If we define the Jacobian of the transformation $h(c)$ as:

$$D_{h,c}(c) = \frac{\partial h}{\partial c}, \tag{2.23}$$

then according to Li and coworkers [33],

$$D_{h,c}(c)f(c) = D_{h,c}(\bar{h}(h(c)))f(\bar{h}(h(c))) \tag{2.24}$$

is a necessary and sufficient condition for exact lumping with $\bar{h}$ as the generalized inverse transformation. Equation (2.24) is not trivial due to the fact that the existence of an $\bar{h}$ with this property is not guaranteed. Since h is nonlinear, the calculation of $\bar{h}$ becomes difficult for high-dimensional systems. If we redefine the system using a partial differential operator A the comparison between the linear and nonlinear case becomes clearer. Defining the operator by:

$$A = \sum_{i=1}^{n} f_i(c(t)) \frac{\partial}{\partial c_i}, \tag{2.25}$$

the original differential system becomes:

$$\frac{dc}{dt} = Ac. \tag{2.26}$$

Thus, finding a nonlinear lumping function h depends on finding canonical forms for the operator A. This contrasts to the linearized case where canonical forms for the Jacobian and its invariant subspaces are searched for [57]. One way of finding canonical forms for the operator A in the linear case is to obtain eigenfunctions relating to eigenvalues which are no longer constant but functions of $c(t)$. For a nonlinear system, finding a full space of eigenfunctions is not an easy task as already shown by Li *et al* [33].

Chapter 3

Methodology

3.1 Preliminary Considerations

In this research, I used the invariance equation (IE) method. The success of this method is based upon the existence of fast and slow time variations in the dynamics of a stiff system [7, 8]. Stiff systems often arise in areas like biochemistry, chemical kinetics, nonlinear dynamics, life sciences and chemical engineering. They occur due to the fast variations in certain components of a particular system during a short period. In the longer period remaining, the solution evolves on a low-dimensional surface in the phase space called the Slow Invariant Manifold (SIM) according to the slower time scales, and the model under consideration has been effectively reduced if an equation for the SIM can be obtained [23].

The simplification of a stiff system is based on the accurate identification of the slow invariant manifold which aids in tackling the numerical difficulties typical of stiff systems, i.e. a large number of unknowns and a vector field containing fast time scales [23]. The simplified system and availability of the slow invariant manifold sheds light on the core of the problem under consideration by identifying the processes necessary for the development of the SIM [23] .

For a complete mechanism of a chemical reaction, we can derive a set of mass-action ordinary differential equations of the following form [50]:

$$\begin{aligned} \dot{y} &= f(y;k) \\ &= R(k)h(y), \end{aligned} \tag{3.1}$$

where y is the column vector of N chemical concentration variables, f is the vector of rates and can be written as a product of a matrix $R(k)$ whose elements are linear combinations of rate constants, and of a vector of monomial basis elements $h(y)$,[1] and k is the parameter vector of specific rate constants corresponding to a model mechanism.

Given (3.1), if there exists a nontrivial constant matrix L with positive entries whose rows are linearly independent such that [50]:

$$L\dot{y} = 0, \tag{3.2}$$

then there is a set of conservation laws

$$Ly = c, \tag{3.3}$$

where c is an M-dimensional constant vector. Using equations (3.1) and (3.3), we get a new dynamical system of $N - M$ ordinary differential equations,

$$\dot{\tilde{y}} = \tilde{R}(k,c)\tilde{h}(\tilde{y}), \tag{3.4}$$

where $\tilde{R}(k,c)$ and $\tilde{h}(\tilde{y})$ are the rate matrix and monomial basis for the reduced system and the components of $\tilde{y}$ are a subset of the components of y.

3.2 Derivation of the Invariance Equation

The invariance equation method, sometimes called the functional iteration method [20, 39, 49], is based on functional equations derived from the governing differential equations, resulting in reduced systems whose solutions are special solutions of the original system,

[1]The monomial basis $h(y)$ includes, in principle the degree zero element ($''1''$, required to treat open systems), monomials of degree one $\{y_1, y_2, \dots, y_N\}$, monomials of degree two $\{y_1^2, y_1y_2, \dots, y_1y_N, y_2^2, y_2y_3, \dots, y_N^2\}$, etc.

i.e. those that operate on the slow time scale of the original system [50]. It involves an iterative algorithm for the solution of the invariance equation. In its original formulation, the algorithm requires splitting the unknowns into two sets, one of which parametrizes the slow invariant manifold [23].

Let us consider the N-dimensional stiff system [23]:

$$\frac{ds}{dt} = f(s), \tag{3.5}$$

where the state vector s and the vector field f are N-dimensional column vectors; $s = [s^1, \ldots, s^N]^T$ and $f = [f^1, \ldots, f^N]^T$, with T indicating the matrix transpose. Let the slow invariant manifold be parametrized by z^i $(i = 1, 2, \ldots, d)$ a set of smooth functions of s, where $d = N - M$ with M as the fast time scales and $d < N$:

$$\begin{aligned} z^i &= z^i(s) \\ &= z^i(s^1, \ldots, s^N), \quad i = 1, \ldots, d. \end{aligned} \tag{3.6}$$

Assuming we have suitable choices of z^i and d, s on the SIM can be written as:

$$\begin{aligned} s^j &= s^j(z) \\ &= s^j(z^1, \ldots, z^d), \quad j = 1, \ldots, N \end{aligned} \tag{3.7}$$

where $z = [z^1, \ldots, z^d]^T$. Differentiating (3.7) with respect to time gives:

$$\frac{ds}{dt} = S_z \frac{dz}{dt} = f(s), \tag{3.8}$$

where

$$\frac{dz}{dt} = Z_s \frac{ds}{dt} = Z_s f(s). \tag{3.9}$$

Substituting equations (3.9) into (3.8), we obtain a system of algebraic equations:

$$\begin{aligned} S_z Z_s f(s) &= f(s) \\ f(s) - S_z Z_s f(s) &= 0 \\ \Rightarrow [I_N^N - S_z Z_s] f(s) &= 0. \end{aligned} \tag{3.10}$$

Equation (3.10) is the invariance equation first derived by Goussis and Valorani [23]. Only the M components of equation (3.10) are linearly independent, which is sufficient for the description of the SIM [23]. I_N^N is an $N \times N$ identity matrix, S_z, Z_s are $N \times d$, $d \times N$ matrices respectively, defined as:

$$S_z = \begin{bmatrix} \frac{\partial s^1}{\partial z^1} & \cdots & \frac{\partial s^1}{\partial z^d} \\ \vdots & & \vdots \\ \frac{\partial s^N}{\partial z^1} & \cdots & \frac{\partial s^N}{\partial z^d} \end{bmatrix}, \tag{3.11}$$

and

$$Z_s = \begin{bmatrix} \frac{\partial z^1}{\partial s^1} & \cdots & \frac{\partial z^1}{\partial s^N} \\ \vdots & & \vdots \\ \frac{\partial z^d}{\partial s^1} & \cdots & \frac{\partial z^d}{\partial s^N} \end{bmatrix}. \tag{3.12}$$

They also satisfy the relation:

$$Z_s S_z = I_d^d. \tag{3.13}$$

We can then solve equation (3.10) by iteration (to be explained later).

3.3 Lumping and the Formulation of the Invariance Equation

Our method involves incorporating the lumping process into the invariance equation method. For $0 \leq m < N-1$, with two lumped variables, one possible parametrization of the SIM is:

$$z_0 = \sum_{i=0}^{m} s_i = s_0 + s_1 + \cdots + s_m, \tag{3.14}$$

$$z_1 = \sum_{i=m+1}^{N-1} s_i = s_{m+1} + \cdots + s_{N-1}. \tag{3.15}$$

The invariance equation for this case can be written as:

$$[I_N^N - S_z Z_s] f(s) = 0, \tag{3.16}$$

where

$$S_z = \begin{pmatrix} \frac{\partial s_0}{\partial z_0} & \frac{\partial s_0}{\partial z_1} \\ \vdots & \vdots \\ \frac{\partial s_m}{\partial z_0} & \frac{\partial s_m}{\partial z_1} \\ \vdots & \vdots \\ \frac{\partial s_N}{\partial z_0} & \frac{\partial s_N}{\partial z_1} \end{pmatrix}, \tag{3.17}$$

$$Z_s = \begin{pmatrix} \frac{\partial z_0}{\partial s_0} & \cdots & \frac{\partial z_0}{\partial s_m} & \cdots \frac{\partial z_0}{\partial s_N} \\ \frac{\partial z_1}{\partial s_0} & \cdots & \frac{\partial z_1}{\partial s_m} & \cdots \frac{\partial z_1}{\partial s_N} \end{pmatrix}, \tag{3.18}$$

and

$$f(s) = \begin{pmatrix} \dot{s}_0 & \cdots & \dot{s}_m & \cdots & \dot{s}_N \end{pmatrix}^T. \tag{3.19}$$

S_z consists of unknown partial derivatives of the SIM which should be solved for, while Z_s is known and follows from equations (3.14) and (3.15). The invariance equations for six species, (i.e. $N = 6$) with two lumped variables $z_0 = s_0 + s_1 + s_2$ and $z_1 = s_3 + s_4 + s_5$ are given by:

$$\dot{s}_0\left(1-\frac{\partial s_0}{\partial z_0}\right)-\dot{s}_1\frac{\partial s_0}{\partial z_0}-\dot{s}_2\frac{\partial s_0}{\partial z_0}-\dot{s}_3\frac{\partial s_0}{\partial z_1}-\dot{s}_4\frac{\partial s_0}{\partial z_1}-\dot{s}_5\frac{\partial s_0}{\partial z_1}=0 \tag{3.20a}$$

$$-\dot{s}_0\frac{\partial s_1}{\partial z_0}+\dot{s}_1\left(1-\frac{\partial s_1}{\partial z_0}\right)-\dot{s}_2\frac{\partial s_1}{\partial z_0}-\dot{s}_3\frac{\partial s_1}{\partial z_1}-\dot{s}_4\frac{\partial s_1}{\partial z_1}-\dot{s}_5\frac{\partial s_1}{\partial z_1}=0 \tag{3.20b}$$

$$-\dot{s}_0\frac{\partial s_2}{\partial z_0}-\dot{s}_1\frac{\partial s_2}{\partial z_0}+\dot{s}_2\left(1-\frac{\partial s_2}{\partial z_0}\right)-\dot{s}_3\frac{\partial s_2}{\partial z_1}-\dot{s}_4\frac{\partial s_2}{\partial z_1}-\dot{s}_5\frac{\partial s_2}{\partial z_1}=0 \tag{3.20c}$$

$$-\dot{s}_0\frac{\partial s_3}{\partial z_0}-\dot{s}_1\frac{\partial s_3}{\partial z_0}-\dot{s}_2\frac{\partial s_3}{\partial z_0}+\dot{s}_3\left(1-\frac{\partial s_3}{\partial z_1}\right)-\dot{s}_4\frac{\partial s_3}{\partial z_1}-\dot{s}_5\frac{\partial s_3}{\partial z_1}=0 \tag{3.20d}$$

$$-\dot{s}_0\frac{\partial s_4}{\partial z_0}-\dot{s}_1\frac{\partial s_4}{\partial z_0}-\dot{s}_2\frac{\partial s_4}{\partial z_0}-\dot{s}_3\frac{\partial s_4}{\partial z_1}+\dot{s}_4\left(1-\frac{\partial s_4}{\partial z_1}\right)-\dot{s}_5\frac{\partial s_4}{\partial z_1}=0 \tag{3.20e}$$

$$-\dot{s}_0\frac{\partial s_5}{\partial z_0}-\dot{s}_1\frac{\partial s_5}{\partial z_0}-\dot{s}_2\frac{\partial s_5}{\partial z_0}-\dot{s}_3\frac{\partial s_5}{\partial z_1}-\dot{s}_4\frac{\partial s_5}{\partial z_1}+\dot{s}_5\left(1-\frac{\partial s_5}{\partial z_1}\right)=0 \tag{3.20f}$$

We seek to obtain a special solution, i.e. the slow manifold, from equations (3.20). The independent variables are z_0 and z_1 and the (unknown) dependent variables are the s_i's. Solutions of (3.20) are sheets of trajectories of (3.5) parametrized by (z_0, z_1). From experience [47, 49, 48, 50, 51] and based on some theoretical work done by Kaper and Kaper [25] , we know that iterative methods of solution of equations in this family, if they converge at all, converge on the slow manifold. The basic problem is to find $\frac{\partial s_i}{\partial z_j}$ for the manifold. If we knew these derivatives, then (3.20) is just an algebraic equation for the s_i's. Given reasonable estimates of these derivatives, it should be possible to obtain an iterative method that converges on the slow manifold. In the next section, we formulate an algorithm that computes the slow manifold iteratively.

3.3.1 Formulation of the Iterative Algorithm

The algorithm is given here for the case $N = 6, d = 2$, i.e. the case to which equations (3.20) apply.

1. Discretize z_0 and z_1 for a given number of mesh points.

2. Generate initial values for $s_0, s_1, s_2, \cdots, s_5$ at each mesh point.

3. Compute $\frac{\partial s_0}{\partial z_0}, \frac{\partial s_1}{\partial z_0}, \cdots, \frac{\partial s_5}{\partial z_0}$ and $\frac{\partial s_0}{\partial z_1}, \frac{\partial s_1}{\partial z_1}, \cdots, \frac{\partial s_5}{\partial z_1}$ by finite differences.

4. Solve the first invariance equation (3.20a) at each mesh point to obtain a new value for s_0.

5. Solve the second invariance equation (3.20b) at each mesh point using the updated value for s_0 to obtain a new value for s_1 and so on.

6. Iterate the algorithm until further iterates change $s_0, s_1, s_2, \cdots s_5$ negligibly and the convergence criterion is satisfied.

Chapter 4

A Toy Model: The Michaelis-Menten Mechanism

4.1 The Model

The Michaelis-Menten mechanism is one of the most important chemical reaction mechanisms in biochemistry and a basic building block for all enzyme modelling [10, 26, 37, 47, 50]. This model is used to illustrate the methods described in chapter 3. In this model, an enzyme reacts with a substrate and reversibly forms an intermediate complex which then results in a product and the original enzyme. The reverse reaction k_{-2} is omitted since it is usually negligible in an *in vitro* experiment, but in an *in vivo* experiment it is not always so. This model is represented as:

$$\mathrm{E} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{ES} \xrightarrow{k_2} \mathrm{E} + \mathrm{P}, \tag{4.1}$$

where S represents substrate, E the enzyme, ES the enzyme-substrate complex, and P the product. k_1, k_{-1} and k_2 are the reaction rate constants. From the Michaelis-Menten mechanism (4.1), we have the following ordinary differential equations derived from the law of mass action [24]:

$$\dot{e} = (k_{-1} + k_2)c - k_1 se, \tag{4.2a}$$

$$\dot{s} = k_{-1}c - k_1 se, \tag{4.2b}$$

$$\dot{c} = k_1 se - (k_{-1} + k_2)c, \tag{4.2c}$$

$$\dot{p} = k_2 c, \tag{4.2d}$$

where lower-case letters represent the concentrations of the corresponding chemical species, with $c = [\text{ES}]$, and the dot represents the time derivative. The appropriate *in vitro* initial conditions are $s(0) = s_0$, $e(0) = e_0$, $c(0) = 0$ and $p(0) = 0$.

4.1.1 Conservation Laws and Planar Reduction

Some important conservation laws can be deduced from the system of four ordinary differential equations (4.2) [26, 37]. Equation (4.2) yields two constants of the motion called the total enzyme and total substrate which are obtained as follows: Firstly, we add equations $(4.2a)$ and $(4.2c)$ to obtain

$$\frac{d(e+c)}{dt} = 0 \tag{4.3}$$

and then integrate with respect to time. We have the following:

$$c(t) + e(t) = e_0 \tag{4.4}$$

which represents the total enzyme concentration. Secondly, we add equations $(4.2b)$, $(4.2c)$, and $(4.2d)$ to obtain

$$\frac{d(s+c+p)}{dt} = 0 \tag{4.5}$$

and after integrating with respect to time we have the following:

$$s(t) + c(t) + p(t) = s_0, \tag{4.6}$$

which represents the total substrate concentration. We can ignore equation $(4.2d)$ since none of the other three rate equations depend explicitly on $p(t)$. We can also ignore equation $(4.2a)$ given the conservation law in equation (4.4) which means that $e(t)$ can be determined from $c(t)$. Hence the four differential equations in (4.2) have been reduced to two differential equations:

$$\dot{s} = k_{-1}c - k_1 s(e_0 - c), \tag{4.7a}$$

$$\dot{c} = k_1 s(e_0 - c) - (k_{-1} + k_2)c. \tag{4.7b}$$

4.2 Lumping/Formulation of the Invariance Equation

For the Michaelis-Menten mechanism, we now have two differential equations (i.e. a two-dimensional model), and we want to reduce it to a one-dimensional model. We already know that this mechanism has a one-dimensional slow manifold [6, 13, 14, 39]. We can lump this model in the following form:

$$z = s + c, \tag{4.8}$$

where z is the lumped variable. Our choice of this lumped variable is because $s+c$ represents the unreacted substrate. We can now formulate the invariance equation as follows:

$$[I_2^2 - Y_z Z_y]f(y) = 0, \tag{4.9}$$

where I_2^2 is a 2×2 identity matrix, and Y_z, Z_y and $f(y)$ are as follows:

$$Y_z = \begin{pmatrix} \frac{\partial s}{\partial z} \\ \frac{\partial c}{\partial z} \end{pmatrix}, \tag{4.10}$$

$$\begin{aligned} Z_y &= \begin{pmatrix} \frac{\partial z}{\partial s} & \frac{\partial z}{\partial c} \end{pmatrix}, \\ &= \begin{pmatrix} 1 & 1 \end{pmatrix} \end{aligned} \tag{4.11}$$

and

$$f(y) = \begin{pmatrix} \dot{s} \\ \\ \dot{c} \end{pmatrix}. \tag{4.12}$$

Z_y follows from (4.8) and is known while Y_z consists of unknown partial derivatives of the SIM and should be solved for. The invariance equation (4.9) gives us the following differential system:

$$\dot{s} - \dot{s}\frac{\partial s}{\partial z} - \dot{c}\frac{\partial s}{\partial z} = 0, \tag{4.13a}$$

$$\dot{c} - \dot{s}\frac{\partial c}{\partial z} - \dot{c}\frac{\partial c}{\partial z} = 0. \tag{4.13b}$$

At each step of the iterative calculation described in section 3.3.1, the unknowns $\frac{\partial s}{\partial z}$ and $\frac{\partial c}{\partial z}$ are estimated using central difference approximations. We will solve the above system iteratively in Matlab for $s(z)$ and $c(z)$. Once we have these, we obtain the reduced model by differentiating equation (4.8) with respect to time:

$$\frac{dz}{dt} = \frac{ds}{dt} + \frac{dc}{dt}. \tag{4.14}$$

Since $\frac{ds}{dt}$ and $\frac{dc}{dt}$ are functions of s and c (equation 4.7), if $s(z)$ and $c(z)$ are known, this is an autonomous ordinary differential equation for $z(t)$.

We would need a good initial function to begin the iteration. To achieve this, we can take a quasi-steady-state approximation, i.e. after the decay of the fast transients, $\dot{c}(t) \approx 0$ [10, 20, 26, 37, 39] . From equation (4.7b), we have:

$$k_1 s(t)[e_0 - c(t)] \approx (k_{-1} + k_2)c(t). \tag{4.15}$$

Solving for $c(t)$, we obtain

$$c(t) \approx \frac{e_0 s(t)}{K_m + s(t)} \tag{4.16}$$

where K_m is called the Michaelis-Menten constant denoted as:

$$K_m = \frac{k_{-1} + k_2}{k_1}. \tag{4.17}$$

4.3 Numerical Computations and Results

In this section, we illustrate the numerical technique used to obtain the SIM of the Michaelis-Menten mechanism using Matlab. The iterative algorithm that solves the system (4.13) is as follows:

1. Discretize the lumped variable z for a given number of mesh points.
2. Generate initial conditions for s and c. Here we use equation (4.16) with s a vector of the discretized points.
3. Calculate $z = s + c$.
4. Compute $\frac{\partial s}{\partial z}$ and $\frac{\partial c}{\partial z}$ using central differences.
5. Solve equation (4.13a) at each mesh point to obtain a new value for s.
6. Solve equation (4.13b) at each mesh point using the updated value of s to obtain a new value for c.

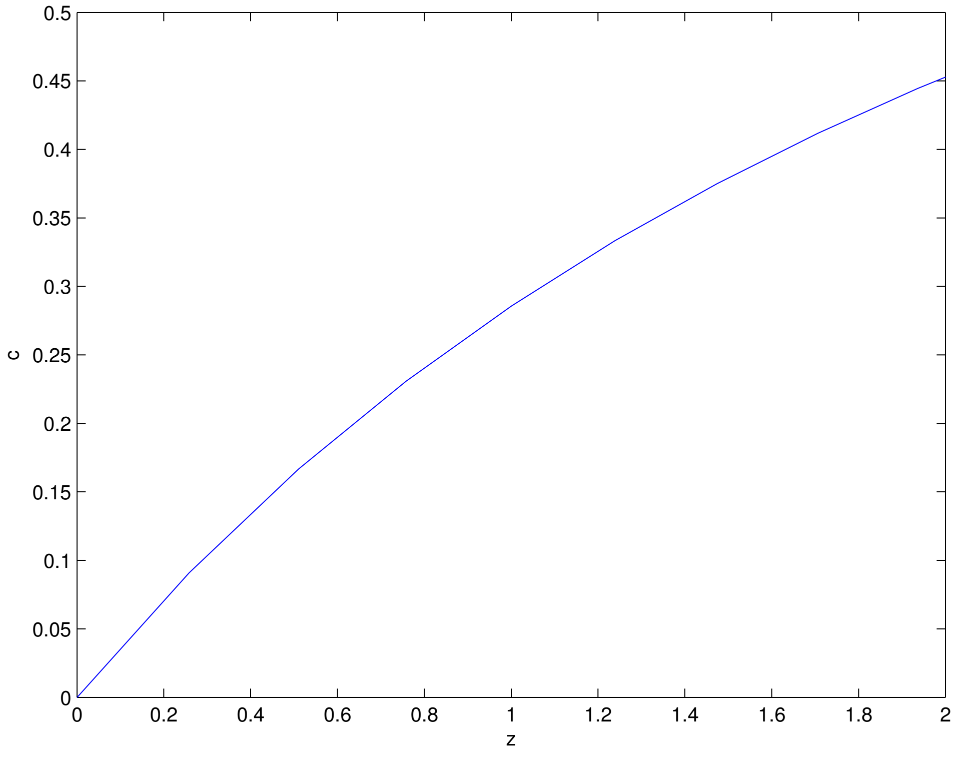

Figure 4.1: The enzyme-substrate complex concentration with respect to the lumped variable ($c(z)$) of the Michaelis-Menten mechanism with $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$ and $e_0 = 1$ for 10 mesh points. The calculation converges after 3 iterates with a tolerance value of 10^{-3}.

7. Recalculate $z = s + c$.

8. Iterate back to step 4 until further iterates change s and c negligibly and the error denoted by:

$$E = \max |s - s_{\text{new}}| + \max |c - c_{\text{new}}| \tag{4.18}$$

where $s_{\text{new}}, c_{\text{new}}$ represents the updated s and c, satisfies the convergence criterion:

$$E < \text{tolerance}. \tag{4.19}$$

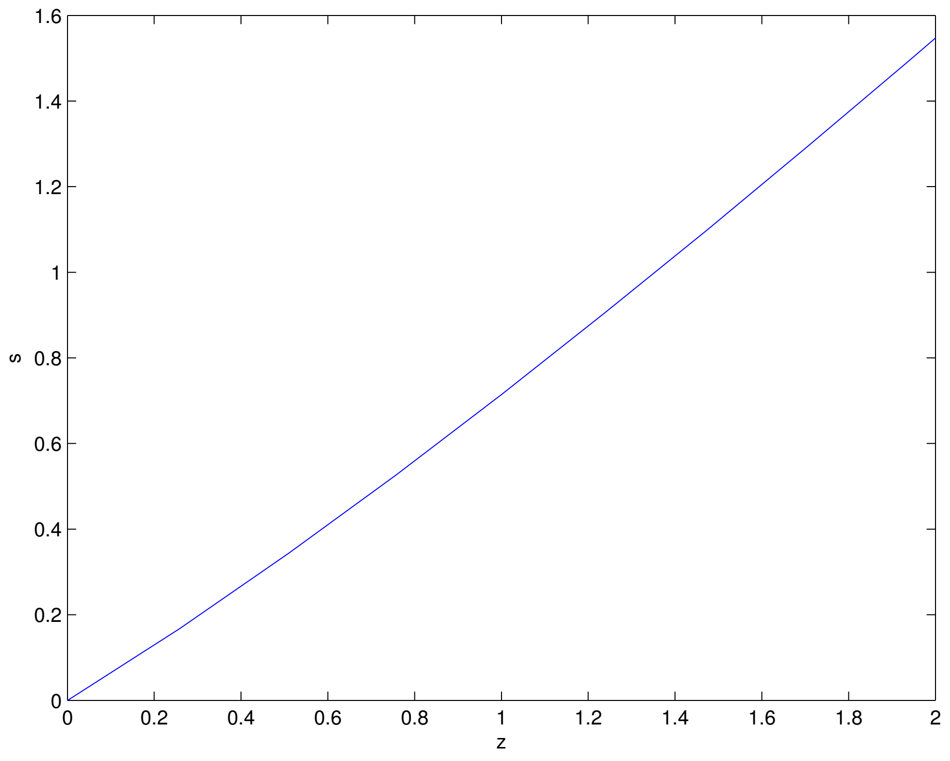

Figure 4.2: The substrate concentration with respect to the lumped variable (s(z)) of the Michaelis-Menten mechanism with the same parameters as figure 4.1.

Figures 4.1 and 4.2 show the enzyme-substrate complex and substrate concentrations with respect to the lumped variable. Figure 4.3 shows the vector field of the model approaching the SIM, which is a one-dimensional curve in the two-dimensional phase plane. The SIM attracts the flow and approaches the equilibrium point which is $(s,c) = (0,0)$. The results obtained here are identical to those in previous papers that computed the SIM for this model, for example Roussel [47].

In this chapter, we have illustrated the lumping method in detail using the Michaelis-Menten mechanism. We formulated the invariance equation, calculated the SIM and reproduced the results of previous studies on this system [20, 39, 47]. In principle, one could integrate equation (4.14) numerically to obtain the time evolution on the slow time scale.

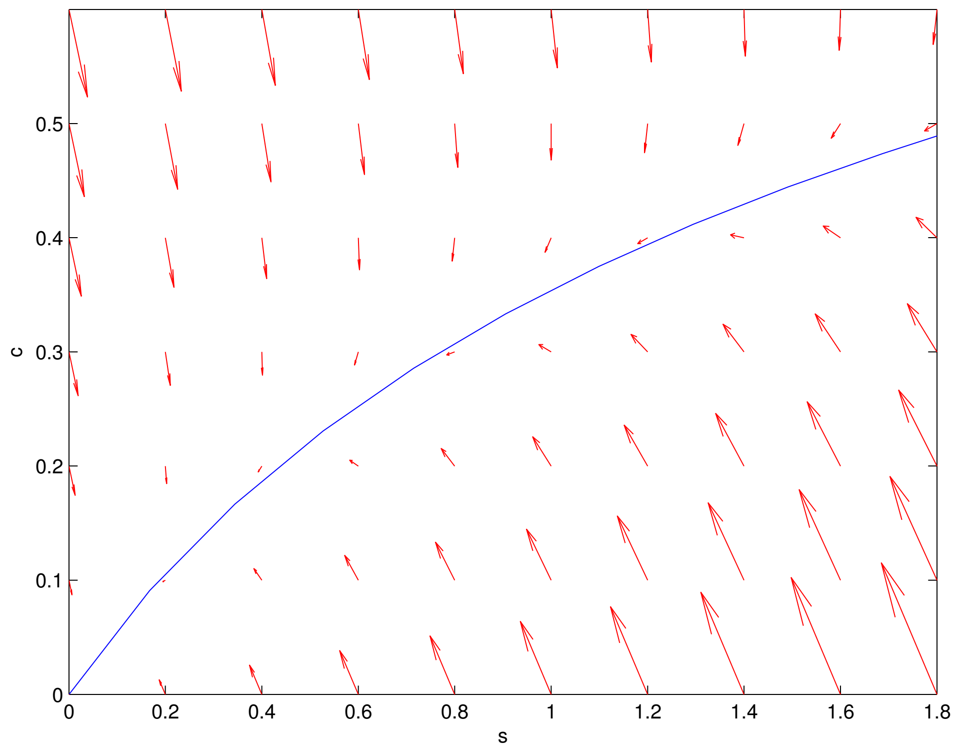

Figure 4.3: Vector field of the Michaelis-Menten mechanism plotted with the computed SIM with the same parameters as figure 4.1.

An example of this procedure will be presented for the more complex model studied in the next chapter.

Chapter 5

The Linear Pathway Model

5.1 The Model

In this chapter, we consider a metabolic transformation in which the product of one reaction is the reactant of the next. Each step is catalyzed by a reversible Michaelian enzyme [50]:

$$\mathrm{S_{i-1}} \underset{v_{i-}}{\overset{v_{i+}}{\rightleftharpoons}} \mathrm{S_i}, \quad i = 1, 2, \ldots, \mathrm{N}, \tag{5.1}$$

with

$$\begin{aligned} v_{i+} &= \frac{V_{i+} S_{i-1}/K_{i+}}{1 + S_{i-1}/K_{i+} + S_i/K_{i-}}, \\ v_{i-} &= \frac{V_{i-} S_i/K_{i-}}{1 + S_{i-1}/K_{i+} + S_i/K_{i-}}, \end{aligned} \tag{5.2}$$

where $V_{i\pm}$ are the maximum velocities for each direction of an enzyme-catalyzed reaction step and $K_{i\pm}$ are the Michaelis constants of the appropriate enzyme-substrate complexes.

We have the rate equations for the model as follows [50]:

$$\begin{aligned} \frac{dS_0}{dt} &= v_{1-} - v_{1+}, \\ \frac{dS_i}{dt} &= v_{i+} - v_{i-} - v_{(i+1)+} + v_{(i+1)-}, \quad i = 1, 2, \ldots, N-1, \\ \frac{dS_N}{dt} &= v_{N+} - v_{N-}. \end{aligned} \tag{5.3}$$

Note that

$$\begin{aligned}\sum_{i=0}^{N}\frac{dS_i}{dt} &= 0\\ \Rightarrow \sum_{i=0}^{N} S_i &= S_T,\end{aligned}\tag{5.4}$$

where S_T is a constant, the total amount of substrate. Then we can obtain S_N as follows:

$$S_N = S_T - \sum_{i=0}^{N-1} S_i.\tag{5.5}$$

The substrate conservation relation (5.4) provides us with a convenient scale to measure the concentrations of all the substrates in the chain given that $0 \le S_i \le S_T$. We can define the following dimensionless variables and parameters [50]:

$$\begin{aligned} s_i &= \frac{S_i}{S_T}, & \tau &= \frac{V_{1+}t}{K_{1+}},\\ \alpha_i &= \frac{S_T}{K_{i+}}, & \beta_i &= \frac{S_T}{K_{i-}},\\ \eta_i &= \frac{V_{i-}K_{1+}}{V_{1+}K_{i-}}, & \gamma_i &= \frac{V_{i+}K_{1+}}{V_{1+}K_{i+}}.\end{aligned}\tag{5.6}$$

We then obtain the following dimensionless differential equations:

$$\begin{aligned}\dot{s}_0 &= \frac{\eta_1 s_1}{1+\alpha_1 s_0+\beta_1 s_1} - \frac{s_0}{1+\alpha_1 s_0+\beta_1 s_1},\\ \dot{s}_i &= \frac{\gamma_i s_{i-1}}{1+\alpha_i s_{i-1}+\beta_i s_i} - \frac{\eta_i s_i}{1+\alpha_i s_{i-1}+\beta_i s_i}\\ &\quad - \frac{\gamma_{i+1} s_i}{1+\alpha_{i+1} s_i+\beta_{i+1} s_{i+1}} + \frac{\eta_{i+1} s_{i+1}}{1+\alpha_{i+1} s_i+\beta_{i+1} s_{i+1}},\\ &\quad i = 1,2,\ldots,N-1.\end{aligned}\tag{5.7}$$

By definition $\gamma_1 = 1$ and the system is closed by the transformed mass conservation relation:

$$s_N = 1 - \sum_{i=0}^{N-1} s_i. \tag{5.8}$$

5.2 The One-Dimensional Manifold

In this section, we consider a reduction of the model to one dimension. We initially consider a case in which the rates of the reversible steps have the following relations:

$$\mathrm{S}_0 \overset{\text{fast}}{\rightleftharpoons} \mathrm{S}_1 \overset{\text{fast}}{\rightleftharpoons} \mathrm{S}_2 \overset{\text{fast}}{\rightleftharpoons} \mathrm{S}_3 \overset{\text{fast}}{\rightleftharpoons} \mathrm{S}_4 \overset{\text{fast}}{\rightleftharpoons} \mathrm{S}_5 \overset{\text{slowest}}{\rightleftharpoons} \mathrm{S}_6\,. \tag{5.9}$$

Here, we have a one-dimensional lump represented as:

$$z = s_0 + s_1 + s_2 + s_3 + s_4 + s_5. \tag{5.10}$$

The above lumped variable is a good choice because, given that the last step is the slowest, s_0 to s_5 equilibrates faster than the slow conversion to s_6. The slow variable is therefore associated with the formation of s_6 and one way to obtain such a variable is by taking the complementary quantity z (all the values that are not s_6) as our slow variable.

5.2.1 Formulation of the Invariance Equation

The invariance equation for $N = 6$ is given as:

$$[I_6^6 - S_z Z_s] f(s) = 0, \tag{5.11}$$

where

$$S_z = \begin{pmatrix} \frac{\partial s_0}{\partial z} & \frac{\partial s_1}{\partial z} & \frac{\partial s_2}{\partial z} & \frac{\partial s_3}{\partial z} & \frac{\partial s_4}{\partial z} & \frac{\partial s_5}{\partial z} \end{pmatrix}^T, \tag{5.12}$$

$$\begin{aligned} Z_s &= \begin{pmatrix} \frac{\partial z}{\partial s_0} & \frac{\partial z}{\partial s_1} & \frac{\partial z}{\partial s_2} & \frac{\partial z}{\partial s_3} & \frac{\partial z}{\partial s_4} & \frac{\partial z}{\partial s_5} \end{pmatrix}, \\ &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \end{aligned} \tag{5.13}$$

and

$$f(s) = \begin{pmatrix} \dot{s}_0 & \dot{s}_1 & \dot{s}_2 & \dot{s}_3 & \dot{s}_4 & \dot{s}_5 \end{pmatrix}^T. \tag{5.14}$$

Note that (5.12) consists of unknown partial derivatives of the SIM while (5.13) follows from (5.10). The invariance equation is now:

$$\dot{s}_0\left(1-\frac{\partial s_0}{\partial z}\right)-\dot{s}_1\frac{\partial s_0}{\partial z}-\dot{s}_2\frac{\partial s_0}{\partial z}-\dot{s}_3\frac{\partial s_0}{\partial z}-\dot{s}_4\frac{\partial s_0}{\partial z}-\dot{s}_5\frac{\partial s_0}{\partial z}=0, \tag{5.15a}$$

$$-\dot{s}_0\frac{\partial s_1}{\partial z}+\dot{s}_1\left(1-\frac{\partial s_1}{\partial z}\right)-\dot{s}_2\frac{\partial s_1}{\partial z}-\dot{s}_3\frac{\partial s_1}{\partial z}-\dot{s}_4\frac{\partial s_1}{\partial z}-\dot{s}_5\frac{\partial s_1}{\partial z}=0, \tag{5.15b}$$

$$-\dot{s}_0\frac{\partial s_2}{\partial z}-\dot{s}_1\frac{\partial s_2}{\partial z}+\dot{s}_2\left(1-\frac{\partial s_2}{\partial z}\right)-\dot{s}_3\frac{\partial s_2}{\partial z}-\dot{s}_4\frac{\partial s_2}{\partial z}-\dot{s}_5\frac{\partial s_2}{\partial z}=0, \tag{5.15c}$$

$$-\dot{s}_0\frac{\partial s_3}{\partial z}-\dot{s}_1\frac{\partial s_3}{\partial z}-\dot{s}_2\frac{\partial s_3}{\partial z}+\dot{s}_3\left(1-\frac{\partial s_3}{\partial z}\right)-\dot{s}_4\frac{\partial s_3}{\partial z}-\dot{s}_5\frac{\partial s_3}{\partial z}=0, \tag{5.15d}$$

$$-\dot{s}_0\frac{\partial s_4}{\partial z}-\dot{s}_1\frac{\partial s_4}{\partial z}-\dot{s}_2\frac{\partial s_4}{\partial z}-\dot{s}_3\frac{\partial s_4}{\partial z}+\dot{s}_4\left(1-\frac{\partial s_4}{\partial z}\right)-\dot{s}_5\frac{\partial s_4}{\partial z}=0, \tag{5.15e}$$

$$-\dot{s}_0\frac{\partial s_5}{\partial z}-\dot{s}_1\frac{\partial s_5}{\partial z}-\dot{s}_2\frac{\partial s_5}{\partial z}-\dot{s}_3\frac{\partial s_5}{\partial z}-\dot{s}_4\frac{\partial s_5}{\partial z}+\dot{s}_5\left(1-\frac{\partial s_5}{\partial z}\right)=0. \tag{5.15f}$$

We then need to solve equations (5.15) iteratively and we do this as follows:

1. Discretize z for a given number of mesh points, with z bounded between 0 and 1.

2. Generate initial values for $s_0, s_1, s_2, \cdots, s_5$ at each mesh point.

3. Compute $\frac{\partial s_0}{\partial z}, \frac{\partial s_1}{\partial z}, \cdots, \frac{\partial s_5}{\partial z}$ by central differences.

4. Treating s_0 as an unknown, and the functions $s_1(z), \cdots, s_5(z)$ as well as the derivatives computed in steps 3 as known quantities, solve the first invariance equation (5.15a) at each mesh point to obtain a new set of values for s_0.

5. Similarly, solve the second invariance equation (5.15b) at each mesh point using the updated value for s_0 to obtain a new value for s_1, and so on.

6. Return to step 3 and iterate the algorithm until further iterates change $s_0, s_1, s_2, \cdots, s_5$ negligibly. Iteration stops when the error defined by:

$$E = \sum_{i=0}^{5} \max_j |s_{i,j}^{(n+1)} - s_{i,j}^{(n)}|, \tag{5.16}$$

   where j represents mesh points, i species, n iterates and $s_{i,j}^{(n)}, s_{i,j}^{(n+1)}$ the preceding and current iterate, satisfies the convergence criterion:

$$E < \text{tolerance}. \tag{5.17}$$

In order to obtain a correct manifold, we need a good initial function to start the iteration. One way to do this is the following: Given that $z = s_0 + s_1 + s_2 + s_3 + s_4 + s_5$

1. Solve the dimensionless equivalent of $v_{i+} = v_{i-}$ for $i = 1, 2, \cdots, 5$. Note that this is a local equilibrium approximation. Solving for $s_0, \cdots, s_5$ from equations (5.7) we

have:

$$
\begin{aligned}
s_0 &= \eta_1 s_1, && (5.18a)\\
s_2 &= \frac{\gamma_2 s_1}{\eta_2}, && (5.18b)\\
s_3 &= \frac{\gamma_3 s_2}{\eta_3} = \frac{\gamma_3\gamma_2 s_1}{\eta_2\eta_3}, && (5.18c)\\
s_4 &= \frac{\gamma_4 s_3}{\eta_4} = \frac{\gamma_4\gamma_3\gamma_2 s_1}{\eta_2\eta_3\eta_4}, && (5.18d)\\
s_5 &= \frac{\gamma_5 s_4}{\eta_5} = \frac{\gamma_5\gamma_4\gamma_3\gamma_2 s_1}{\eta_2\eta_3\eta_4\eta_5}. && (5.18e)
\end{aligned}
$$

2. Substitute the values of $s_0,\cdots,s_5$ from equations (5.18) into equation (5.10). We have:

$$z = \eta_1 s_1 + s_1 + \frac{\gamma_2 s_1}{\eta_2} + \frac{\gamma_3\gamma_2 s_1}{\eta_2\eta_3} + \frac{\gamma_4\gamma_3\gamma_2 s_1}{\eta_2\eta_3\eta_4} + \frac{\gamma_5\gamma_4\gamma_3\gamma_2 s_1}{\eta_2\eta_3\eta_4\eta_5}. \tag{5.19}$$

3. Solve for s_1 from equation (5.19). Solving for s_1 gives:

$$s_1 = \frac{z\eta_2\eta_3\eta_4\eta_5}{\eta_1\eta_2\eta_3\eta_4\eta_5 + \eta_2\eta_3\eta_4\eta_5 + \eta_3\eta_4\eta_5\gamma_2 + \eta_4\eta_5\gamma_2\gamma_3 + \eta_5\gamma_2\gamma_3\gamma_4 + \gamma_2\gamma_3\gamma_4\gamma_5}. \tag{5.20}$$

Note the dependence on z.

4. Solve for s_0, s_2, s_3, s_4 and s_5 with respect to z to obtain the remaining initial functions. To do this, substitute the value of s_1 above into equations (5.18a) to (5.18e) respectively.

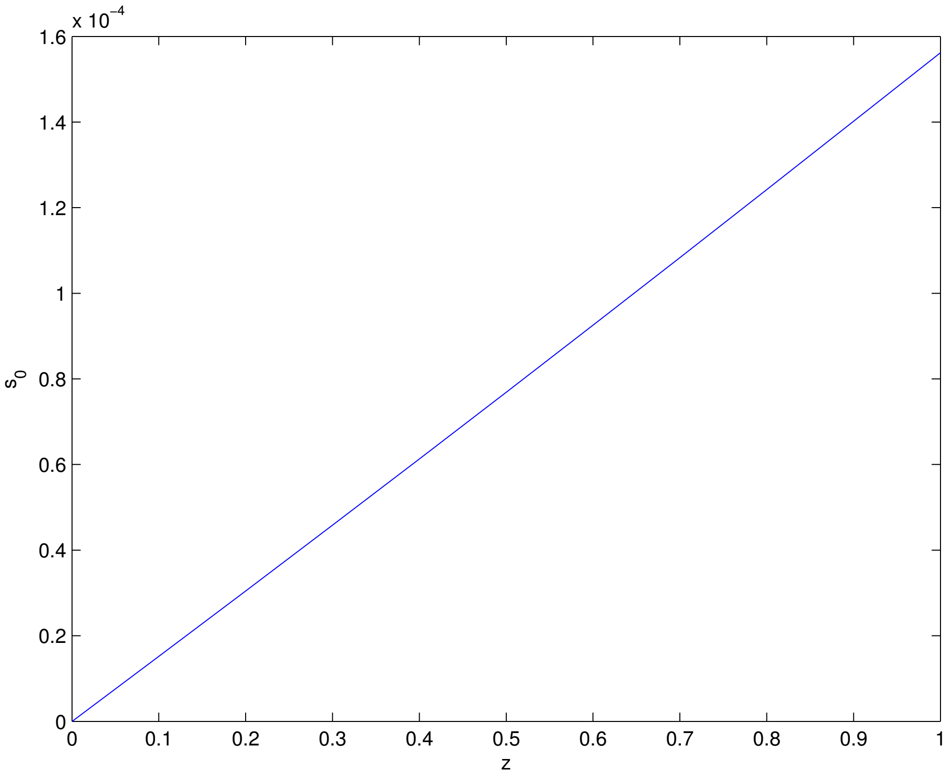

Figure 5.1: The one-dimensional manifold for species s_0 with 10 mesh points, and $\alpha_i = \beta_i = 1, i = 1, \ldots, 6$, $\gamma_2 = \gamma_4 = \gamma_5 = 100$, $\eta_1 = \eta_4 = 1$, $\eta_2 = \eta_5 = 10$, $\eta_3 = \gamma_3 = 0.01, \gamma_6 = 1$ and $\eta_6 = 0$. The solution converges after 2 iterates with a tolerance value of 10^{-3}.

5.2.2 Results

Figures 5.1 and 5.2 represent the SIM for the one-dimensional manifold for species s_0 and s_5 with respect to the lumped variable z. These manifolds are almost linear due to the choice of a good lumping variable, which is in contrast to the noticeably curved manifold parameterized by s_0 found by Roussel and Fraser [50]. However, they are not exactly linear as illustrated in figures 5.3 and 5.4. The calculation converges after 2 iterates, and even for a small number of mesh points (10) we get rapid convergence. Convergence is still obtained if one of the steps prior to the last step is as slow as the last one. For example with $\gamma_3 = \eta_3 = \gamma_6 = \eta_6 = 10^{-2}$, the calculation converges after two iterates. We also observe

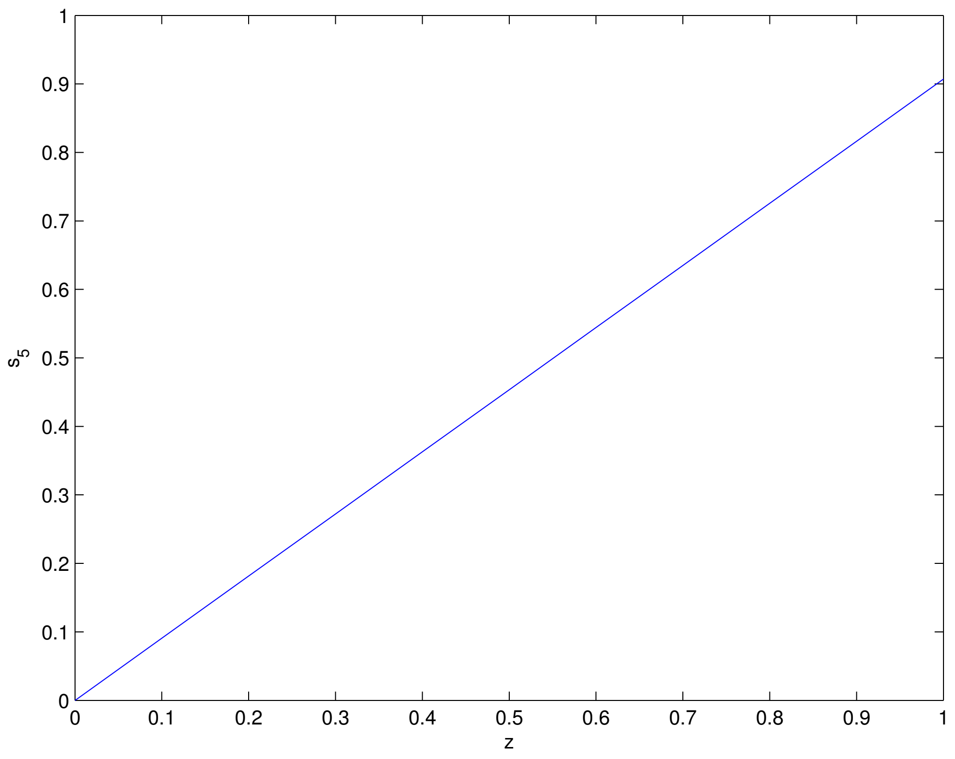

Figure 5.2: The one-dimensional manifold for species s_5 with 10 mesh points using the same parameters as in figure 5.1.

that $\gamma_6, \eta_6 \leq 1$ to obtain a convergent solution, any value greater than these would result in a divergent scheme because the assumptions made in choosing our lumped variable (z) break down since, implicitly, $\gamma_1 = 1$.

When $\eta_6 = 0$, the model represents a reaction with irreversible product formation. The lumped model then corresponds to $Z \rightarrow S_6$ with a nonlinear evolution equation given by equation (5.21). Figure 5.5 represents the rate of reaction with respect to the lumped variable z. S_6 is the product and the shape of this curve contrasts that of the Michaelis-Menten hyperbola obtained as the steady-state rate law in many models of simple pathways [10].

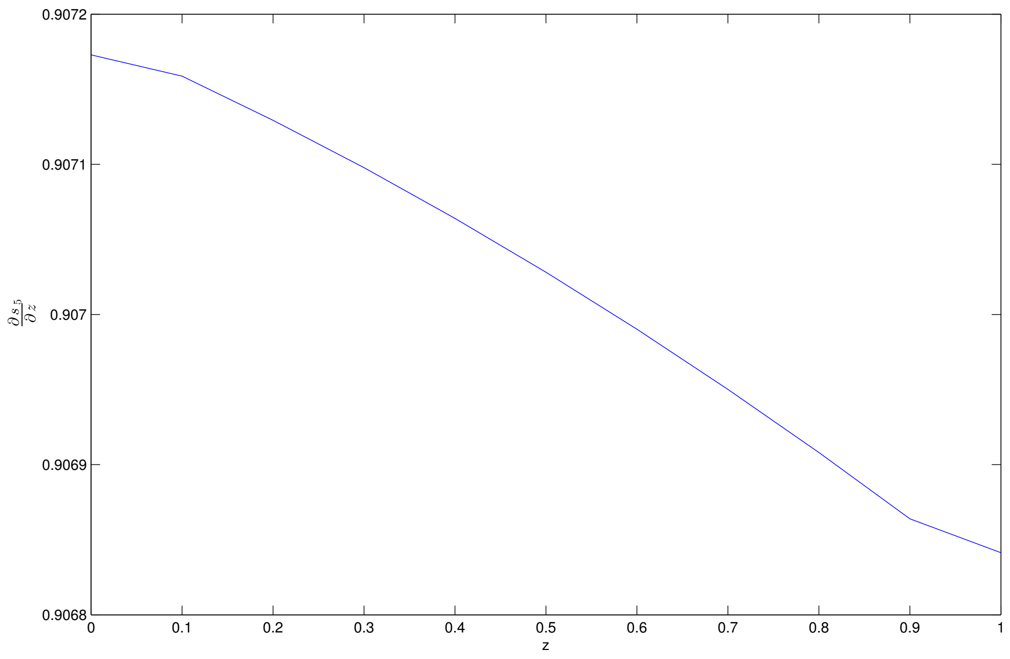

Figure 5.3: Evidence of manifold curvature using the same parameters as in figure 5.1.

5.2.3 *Comparing the Full and Reduced Models*

In this subsection, we compare the full and reduced models. We do this to verify that we have a reasonable representation of the invariant manifold and that the manifold computed is attracting. This shows that the reduced model provides a faithful representation of the full model. To do this, we proceed as follows:

1. Pick an initial condition (z_0) for the reduced model.
2. Compute the coordinates $s_0(z_0), s_1(z_0), \cdots, s_5(z_0)$ on the manifold and use them as initial conditions for the full model, i.e start the integration of the full model on the slow manifold.
3. Compute the trajectory of the reduced model by numerically integrating the differential equation:

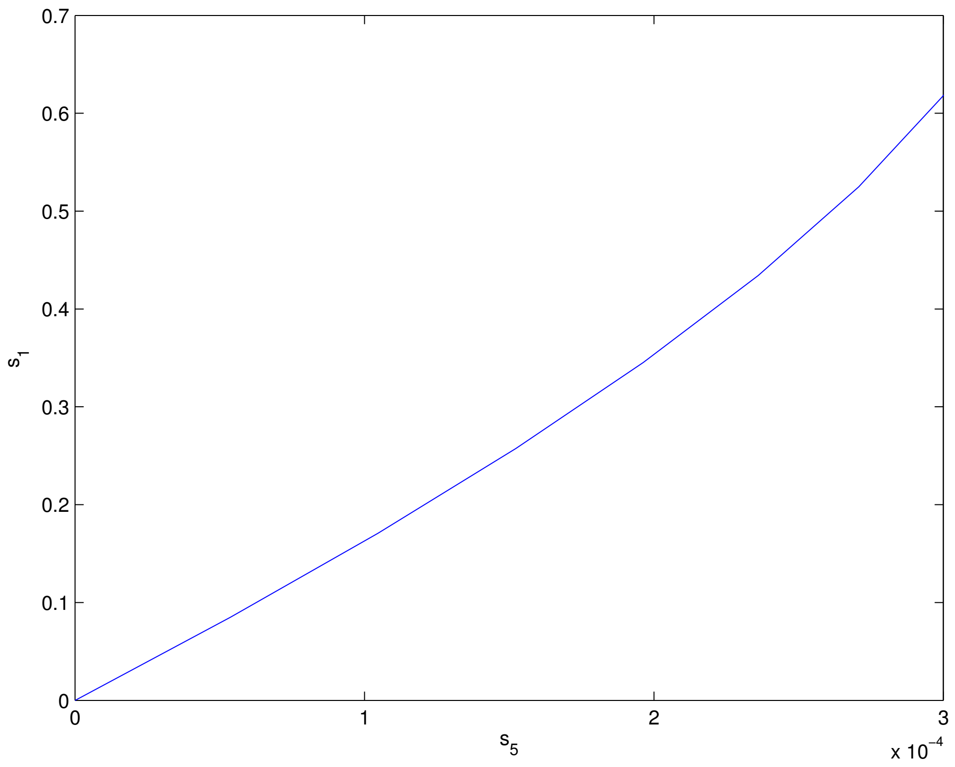

Figure 5.4: Evidence of manifold curvature with $\alpha_i = \beta_i = 1, i = 1, \ldots, 6$, $\gamma_2 = \gamma_4 = \gamma_5 = 100$, $\eta_1 = \eta_4 = 0.1$, $\eta_2 = \eta_5 = 10$, $\eta_3 = \gamma_3 = 0.01, \gamma_6 = 1$ and $\eta_6 = 0$.

$$\frac{dz}{dt} = \frac{ds_0}{dt} + \frac{ds_1}{dt} + \frac{ds_2}{dt} + \frac{ds_3}{dt} + \frac{ds_4}{dt} + \frac{ds_5}{dt}. \tag{5.21}$$

We compute the s_i at a set of discrete mesh points and then interpolate linearly to obtain values between the mesh points during integration. $\dot{s}_i(s_0, \ldots, s_5)$ is given by equation (5.7), and since we know $s_0(z), \cdots, s_5(z)$, then the above equation is an autonomous ordinary differential equation in z since the independent variable z does not appear explicitly in the equation.

From figure 5.6, we observe that the full and reduced models are identical within the resolution of the figure. Figure 5.7 represents solutions of the reduced model and of the full model starting from off-manifold initial conditions. For example, here we have the

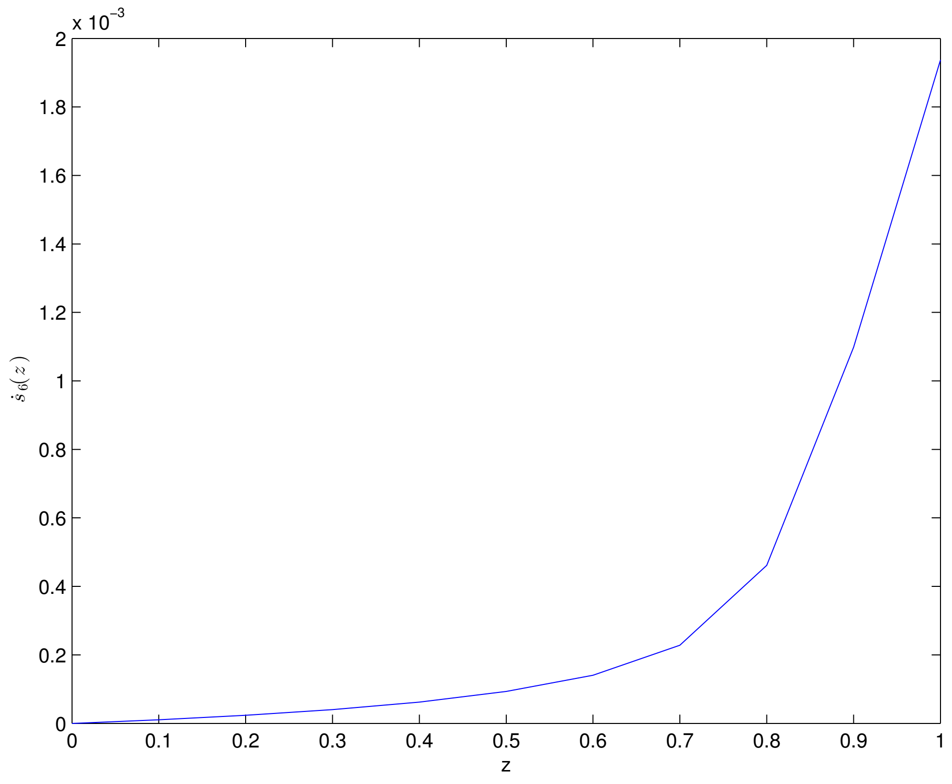

Figure 5.5: Rate of product formation for 10 mesh points with $\alpha_i = \beta_i = 100, i = 1, \ldots, 6$, $\gamma_2 = \gamma_4 = \gamma_5 = \eta_1 = \eta_4 = \eta_2 = \eta_5 = 100$, $\eta_3 = \gamma_3 = 0.01$, $\gamma_6 = 1$ and $\eta_6 = 0$.

off-manifold initial conditions as $s_0(0) = 0.8001, s_i(0) = 0$ for $i \geq 1$. We observe that even when we start the integration of the full model off the manifold, its trajectory is still indistinguishable from that of the reduced model. This shows that the reduced model can be used as a representation of the full model after the decay of transients and that the neglected transients have a negligible effect on the eventual trajectory. From figure 5.8, we see that at very small time the full and reduced models are different, but after a long period of time they become indistinguishable, as seen in figure 5.7.

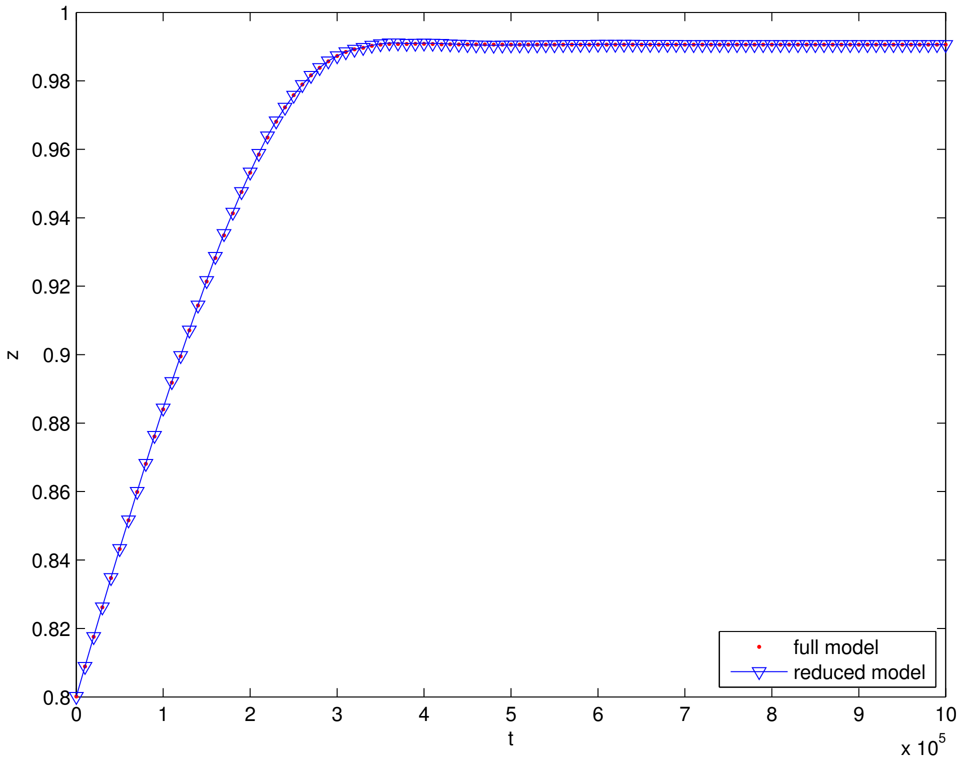

Figure 5.6: The full and reduced models integrated from initial conditions on the manifold. The manifold was computed with 10 mesh points using $\alpha_i = \beta_i = 100, i = 1,\ldots,6$, $\gamma_2 = \gamma_4 = \gamma_5 = \eta_1 = \eta_4 = \eta_2 = \eta_5 = 100$, $\eta_3 = \gamma_3 = 0.01$, $\gamma_6 = \eta_6 = 10^{-4}$. The initial condition for the full model is $s_0(0) = 0.7620, s_i(0) = 0.0076, i \geq 1$ and for the reduced model $z(0) = 0.8001$.

5.2.4 Discretization Error

The process of discretizing a differential equation usually leads to errors and some questions that come to mind regarding the accuracy of a numerical solution are the following:

1. How does this error arise in a numerical calculation?

2. How can we reduce this error?

Discretization error arises whenever we represent a continuous function by a discrete set of points [5, 55]. This error is in contrast to round-off error[1] which is always present.

[1]The error that comes from representing a real number as a floating point number on a computer.

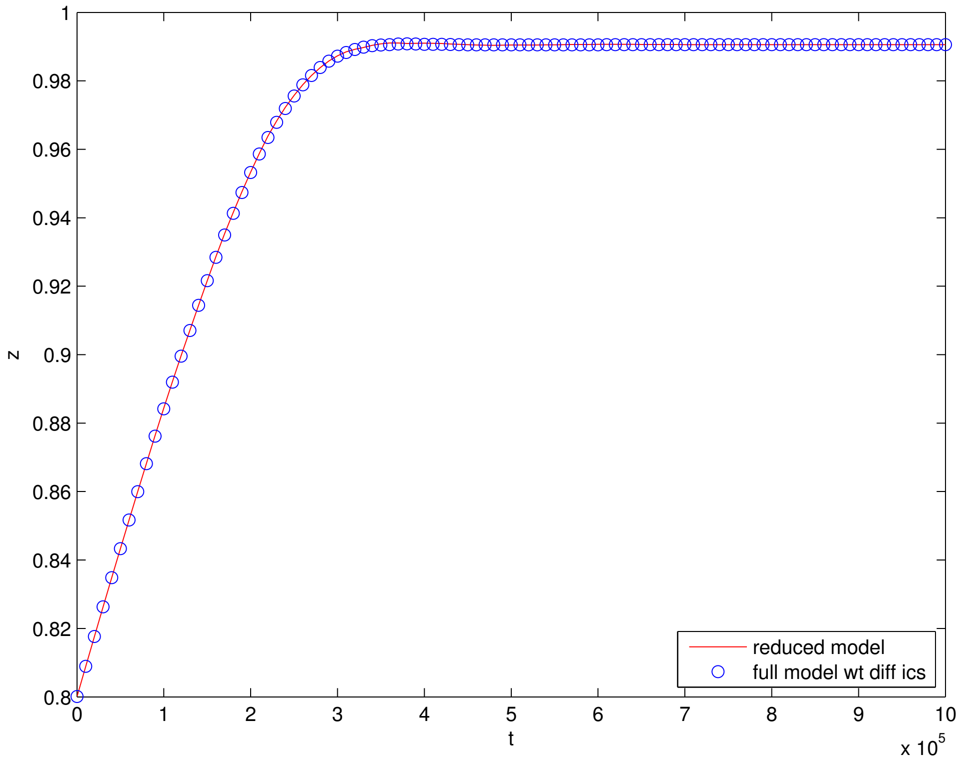

Figure 5.7: The reduced model integrated from initial conditions on the manifold and the full model started from off-manifold initial conditions. The manifold was computed with 10 mesh points using the same parameters as in figure 5.6. The initial condition for the full model is $s_0(0) = 0.8001, s_i(0) = 0, i \geq 1$ and for the reduced model $z(0) = 0.8001$.

Discretization error can be reduced by decreasing the grid spacing, (i.e. increasing the number of mesh points). As the grid spacing is decreased, the error will get small. This has a disadvantage of computational cost since more calculations would be done [5, 55]. Also there is a limit to how small the grid spacing should be before the round-off error will start negatively affecting the computation. The negative effect of the round-off error is due to the fact that we have derivatives and when taking the difference of two similar numbers we run into the problem of finite precision.

A discretization error (D) for the one-dimensional linear metabolic pathway model can be defined by:

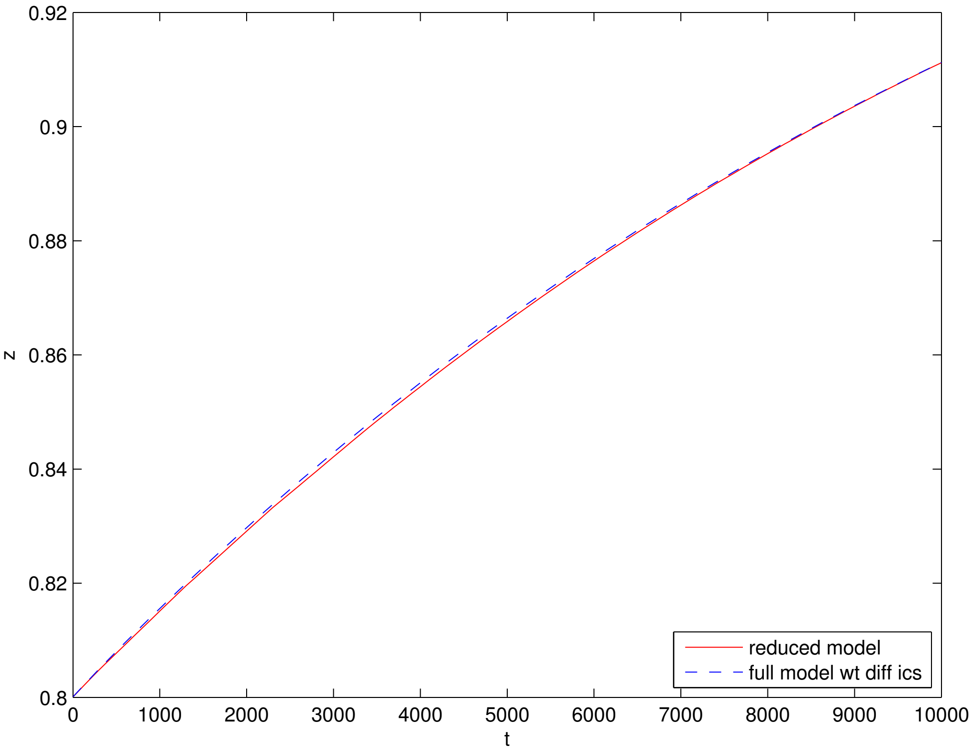

Figure 5.8: The reduced model integrated from initial conditions on the manifold and the full model started from off-manifold initial conditions for short time. The manifold was computed with 10 mesh points using $\alpha_i = \beta_i = 1, i = 1, \dots, 6$, $\gamma_2 = \gamma_4 = \gamma_5 = \eta_1 = \eta_4 = \eta_2 = \eta_5 = 100$, $\eta_3 = \gamma_3 = 0.01$, $\gamma_6 = \eta_6 = 10^{-4}$.

$$D = \sum_i (z^i_{(red)} - z^i_{(full)})^2 |z^{i-1}_{(red)} - z^i_{(red)}|. \tag{5.22}$$

where $z^i_{(red)}, z^i_{(full)}$ represent the ith time points of the full and reduced models respectively. The first part of this equation, i.e. $\left[(z^i_{(red)} - z^i_{(full)})^2\right]$ represents the error, i.e. the difference between the full and reduced model. The absolute value represents the distance travelled along a trajectory. This is an approximation of the path integral along the trajectory.

From figure 5.9, we observe that the discretization error is very small (of order 10^{-8}). The weak variation of the discretization error with the number of mesh points shows that the result is essentially fully converged in this regime. This is no doubt due to the near-

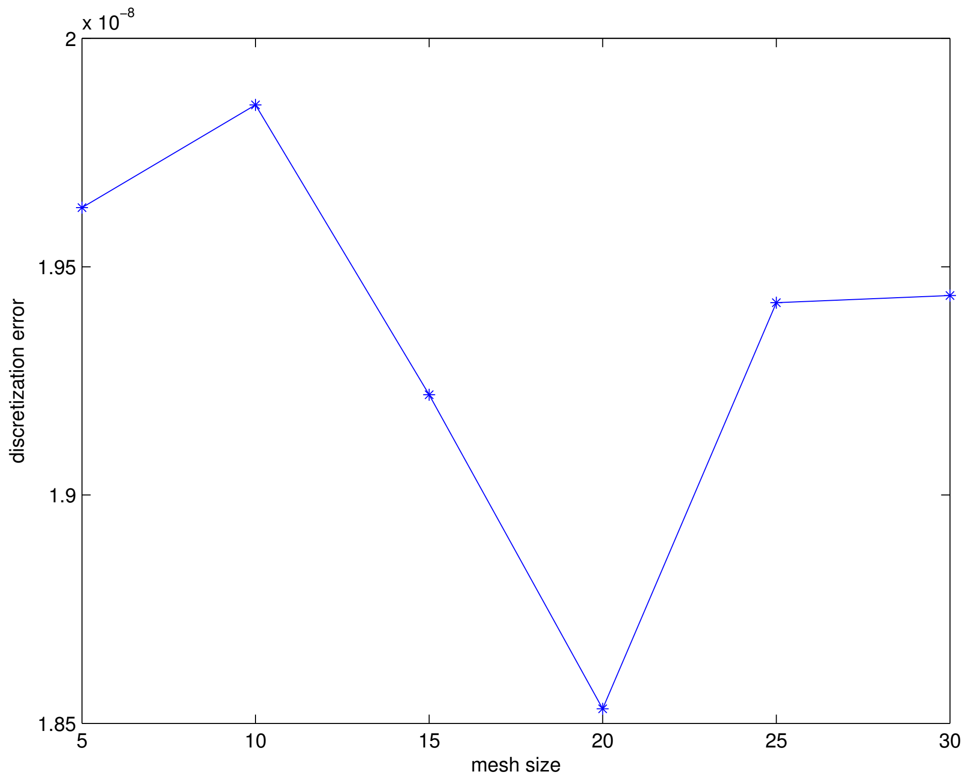

Figure 5.9: Discretization error versus mesh size, using the same parameters as in figure 5.1

linearity of the manifold observed earlier.

5.3 The Case with a Slow Step Somewhere Other Than at the End of the Chain

In this section, we investigate the case where the slow step is somewhere other than at the end of the chain. As an example, we consider the following case:

$$S_0 \xrightleftharpoons{\text{fast}} S_1 \xrightleftharpoons{\text{fast}} S_2 \xrightleftharpoons{\text{fast}} S_3 \xrightleftharpoons{\text{fast}} S_4 \xrightleftharpoons{\text{slowest}} S_5 \xrightleftharpoons{\text{fast}} S_6 \, . \tag{5.23}$$

Here, we can choose a single lumped variable:

$$z = s_0 + s_1 + s_2 + s_3 + s_4. \tag{5.24}$$

Choosing the lump in this way enables us to capture all the information contained in the slowest connection, i.e. the motion along the slow manifold. Roughly speaking, the lumped model can be thought of as $Z_0 \rightleftharpoons Z_1$, where z_0 is given by equation (5.24) and $z_1 = s_5 + s_6$. We only need one variable to parameterize a one-dimensional slow manifold, and chose z_0.

The invariance equation for $N = 6$ is given as:

$$[I_6^6 - S_z Z_s] f(s) = 0, \tag{5.25}$$

where

$$S_z = \begin{pmatrix} \frac{\partial s_0}{\partial z} & \frac{\partial s_1}{\partial z} & \frac{\partial s_2}{\partial z} & \frac{\partial s_3}{\partial z} & \frac{\partial s_4}{\partial z} & \frac{\partial s_5}{\partial z} \end{pmatrix}^T, \tag{5.26}$$

$$\begin{aligned} Z_s &= \begin{pmatrix} \frac{\partial z}{\partial s_0} & \frac{\partial z}{\partial s_1} & \frac{\partial z}{\partial s_2} & \frac{\partial z}{\partial s_3} & \frac{\partial z}{\partial s_4} & \frac{\partial z}{\partial s_5} \end{pmatrix}, \\ &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}, \end{aligned} \tag{5.27}$$

and

$$f(s) = \begin{pmatrix} \dot{s}_0 & \dot{s}_1 & \dot{s}_2 & \dot{s}_3 & \dot{s}_4 & \dot{s}_5 \end{pmatrix}^T. \tag{5.28}$$

The invariance equation is now:

$$\begin{aligned}
\dot{s}_0\left(1-\frac{\partial s_0}{\partial z}\right)-\dot{s}_1\frac{\partial s_0}{\partial z}-\dot{s}_2\frac{\partial s_0}{\partial z}-\dot{s}_3\frac{\partial s_0}{\partial z}-\dot{s}_4\frac{\partial s_0}{\partial z} &= 0,\\
-\dot{s}_0\frac{\partial s_1}{\partial z}+\dot{s}_1\left(1-\frac{\partial s_1}{\partial z}\right)-\dot{s}_2\frac{\partial s_1}{\partial z}-\dot{s}_3\frac{\partial s_1}{\partial z}-\dot{s}_4\frac{\partial s_1}{\partial z} &= 0,\\
-\dot{s}_0\frac{\partial s_2}{\partial z}-\dot{s}_1\frac{\partial s_2}{\partial z}+\dot{s}_2\left(1-\frac{\partial s_2}{\partial z}\right)-\dot{s}_3\frac{\partial s_2}{\partial z}-\dot{s}_4\frac{\partial s_2}{\partial z} &= 0,\\
-\dot{s}_0\frac{\partial s_3}{\partial z}-\dot{s}_1\frac{\partial s_3}{\partial z}-\dot{s}_2\frac{\partial s_3}{\partial z}+\dot{s}_3\left(1-\frac{\partial s_3}{\partial z}\right)-\dot{s}_4\frac{\partial s_3}{\partial z} &= 0,\\
-\dot{s}_0\frac{\partial s_4}{\partial z}-\dot{s}_1\frac{\partial s_4}{\partial z}-\dot{s}_2\frac{\partial s_4}{\partial z}-\dot{s}_3\frac{\partial s_4}{\partial z}+\dot{s}_4\left(1-\frac{\partial s_4}{\partial z}\right) &= 0,\\
-\dot{s}_0\frac{\partial s_5}{\partial z}-\dot{s}_1\frac{\partial s_5}{\partial z}-\dot{s}_2\frac{\partial s_5}{\partial z}-\dot{s}_3\frac{\partial s_5}{\partial z}-\dot{s}_4\frac{\partial s_5}{\partial z}+\dot{s}_5 &= 0
\end{aligned} \tag{5.29}$$

We then need to the compute initial conditions for s_0, s_1, s_2, s_3 and s_4 as in the previous case (section 5.2) but with $s_5 = 0$ since it is not included in the lumped variable.

Figures 5.10 and 5.11 represent the SIM for the one-dimensional manifold for species s_0 and s_5 with respect to the lumped variable z for the case where the slow step is not at the end of the chain. We observe that the calculation converges after 3 iterates and also when the rate of the slowest step approaches the rate of the next slowest step we still get a convergent solution.

5.4 Two-Dimensional Manifold

In this section, we calculate a two-dimensional manifold with the rates of the reversible steps similar to the one-dimensional manifold in section (5.2) with the following connections:

$$S_0 \overset{\text{fast}}{\rightleftharpoons} S_1 \overset{\text{fast}}{\rightleftharpoons} S_2 \overset{\text{slow}}{\rightleftharpoons} S_3 \overset{\text{fast}}{\rightleftharpoons} S_4 \overset{\text{fast}}{\rightleftharpoons} S_5 \overset{\text{slow}}{\rightleftharpoons} S_6\,. \tag{5.30}$$

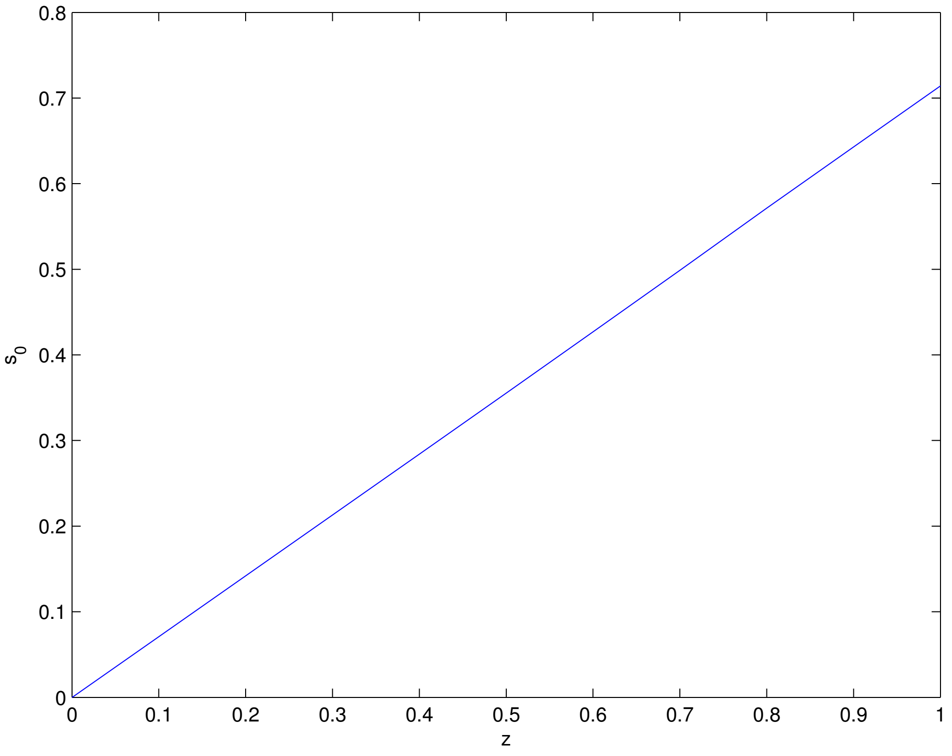

Figure 5.10: The manifold for the case with a slow step somewhere other than at the end of the chain for species s_0. The calculation was carried out with 10 mesh points with $\alpha_i = \beta_i = 100, i = 1, \ldots, 6$, $\gamma_2 = \gamma_4 = \gamma_6 = \eta_1 = \eta_2 = \eta_4 = \eta_6 = 10$, $\eta_3 = \gamma_3 = 0.1$ and $\gamma_5 = \eta_5 = 0.01$. The solution converges after 3 iterates.

In order to get a correct slow invariant manifold, we need to lump the species correctly. To do this, we need to have the fast connections within the lumps and the slow connection between the lumps as illustrated in figure 5.12. This choice enables us to capture all the information about the system contained in the slow connections. Lumping incorrectly results in the slow connection being within the lumps. The dynamical modes eliminated by the lumping are then not purely fast, so that the lumped variables do not evolve purely on the slow time scales. Note that the information contained in the fast connections often cannot be captured experimentally due to instrumental limitations.

We then have the two lumps as follows:

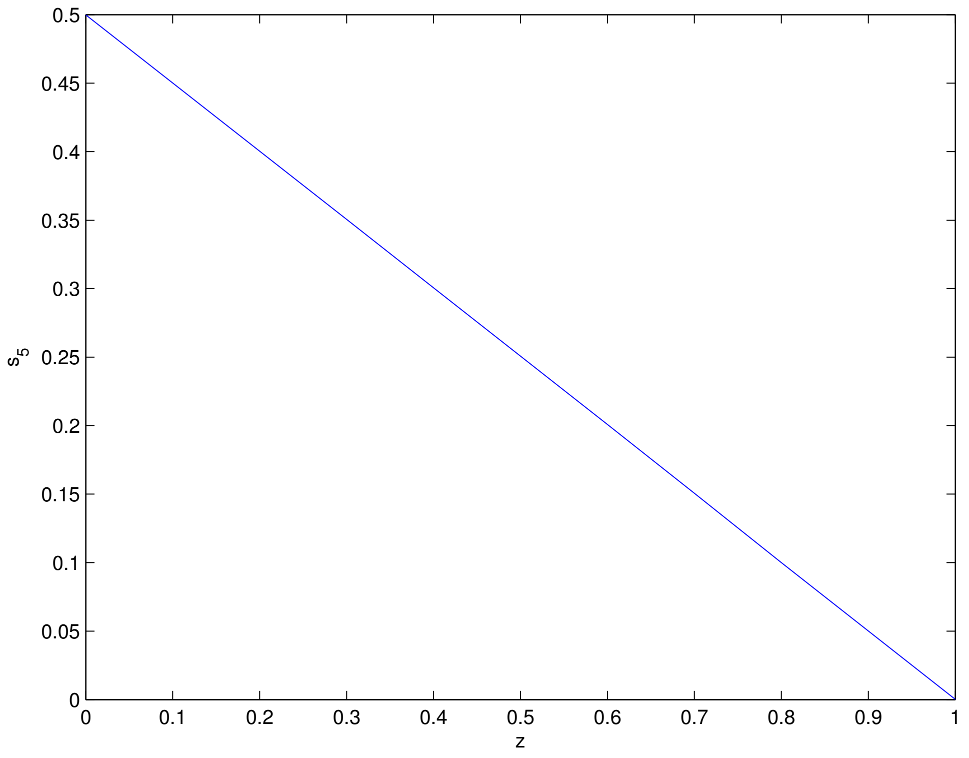

Figure 5.11: The manifold for the case with a slow step somewhere other than at the end of the chain for species s_5. The calculation was carried out with 10 mesh points with the same parameters as in figure 5.10. The solution converges after 3 iterates.

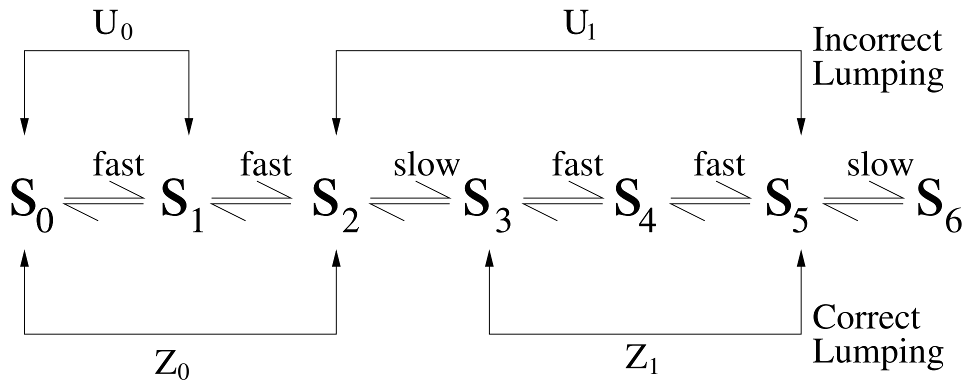

Figure 5.12: Correct and incorrect lumping

$$z_0 = s_0 + s_1 + s_2, \tag{5.31}$$

and

$$z_1 = s_3 + s_4 + s_5. \tag{5.32}$$

5.4.1 Formulation of the Invariance Equation

The invariance equation for $N = 6$ is given as:

$$[I_6^6 - S_z Z_s] f(s) = 0, \tag{5.33}$$

where

$$S_z = \begin{pmatrix} \frac{\partial s_0}{\partial z_0} & \frac{\partial s_0}{\partial z_1} \\ \frac{\partial s_1}{\partial z_0} & \frac{\partial s_1}{\partial z_1} \\ \frac{\partial s_2}{\partial z_0} & \frac{\partial s_2}{\partial z_1} \\ \frac{\partial s_3}{\partial z_0} & \frac{\partial s_3}{\partial z_1} \\ \frac{\partial s_4}{\partial z_0} & \frac{\partial s_4}{\partial z_1} \\ \frac{\partial s_5}{\partial z_0} & \frac{\partial s_5}{\partial z_1} \end{pmatrix}, \tag{5.34}$$

$$\begin{aligned} Z_s &= \begin{pmatrix} \frac{\partial z_0}{\partial s_0} & \frac{\partial z_0}{\partial s_1} & \frac{\partial z_0}{\partial s_2} & \frac{\partial z_0}{\partial s_3} & \frac{\partial z_0}{\partial s_4} & \frac{\partial z_0}{\partial s_5} \\ \frac{\partial z_1}{\partial s_0} & \frac{\partial z_1}{\partial s_1} & \frac{\partial z_1}{\partial s_2} & \frac{\partial z_1}{\partial s_3} & \frac{\partial z_1}{\partial s_4} & \frac{\partial z_1}{\partial s_5} \end{pmatrix}, \\ &= \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \end{aligned} \tag{5.35}$$

and

$$f(s) = \begin{pmatrix} \dot{s}_0 & \dot{s}_1 & \dot{s}_2 & \dot{s}_3 & \dot{s}_4 & \dot{s}_5 \end{pmatrix}^T. \tag{5.36}$$

The invariance equation is now:

$$\dot{s}_0\left(1-\frac{\partial s_0}{\partial z_0}\right) - \dot{s}_1\frac{\partial s_0}{\partial z_0} - \dot{s}_2\frac{\partial s_0}{\partial z_0} - \dot{s}_3\frac{\partial s_0}{\partial z_1} - \dot{s}_4\frac{\partial s_0}{\partial z_1} - \dot{s}_5\frac{\partial s_0}{\partial z_1} = 0, \tag{5.37a}$$

$$-\dot{s}_0\frac{\partial s_1}{\partial z_0} + \dot{s}_1\left(1-\frac{\partial s_1}{\partial z_0}\right) - \dot{s}_2\frac{\partial s_1}{\partial z_0} - \dot{s}_3\frac{\partial s_1}{\partial z_1} - \dot{s}_4\frac{\partial s_1}{\partial z_1} - \dot{s}_5\frac{\partial s_1}{\partial z_1} = 0, \tag{5.37b}$$

$$-\dot{s}_0\frac{\partial s_2}{\partial z_0} - \dot{s}_1\frac{\partial s_2}{\partial z_0} + \dot{s}_2\left(1-\frac{\partial s_2}{\partial z_0}\right) - \dot{s}_3\frac{\partial s_2}{\partial z_1} - \dot{s}_4\frac{\partial s_2}{\partial z_1} - \dot{s}_5\frac{\partial s_2}{\partial z_1} = 0, \tag{5.37c}$$

$$-\dot{s}_0\frac{\partial s_3}{\partial z_0} - \dot{s}_1\frac{\partial s_3}{\partial z_0} - \dot{s}_2\frac{\partial s_3}{\partial z_0} + \dot{s}_3\left(1-\frac{\partial s_3}{\partial z_1}\right) - \dot{s}_4\frac{\partial s_3}{\partial z_1} - \dot{s}_5\frac{\partial s_3}{\partial z_1} = 0, \tag{5.37d}$$

$$-\dot{s}_0\frac{\partial s_4}{\partial z_0} - \dot{s}_1\frac{\partial s_4}{\partial z_0} - \dot{s}_2\frac{\partial s_4}{\partial z_0} - \dot{s}_3\frac{\partial s_4}{\partial z_1} + \dot{s}_4\left(1-\frac{\partial s_4}{\partial z_1}\right) - \dot{s}_5\frac{\partial s_4}{\partial z_1} = 0, \tag{5.37e}$$

$$-\dot{s}_0\frac{\partial s_5}{\partial z_0} - \dot{s}_1\frac{\partial s_5}{\partial z_0} - \dot{s}_2\frac{\partial s_5}{\partial z_0} - \dot{s}_3\frac{\partial s_5}{\partial z_1} - \dot{s}_4\frac{\partial s_5}{\partial z_1} + \dot{s}_5\left(1-\frac{\partial s_5}{\partial z_1}\right) = 0. \tag{5.37f}$$

We can solve equations (5.37) iteratively as follows:

1. Discretize z_0 and z_1 for a given number of mesh points.
2. Generate initial values for $s_0, s_1, s_2, \cdots, s_5$ at each mesh point.
3. Compute $\frac{\partial s_0}{\partial z_0}, \frac{\partial s_1}{\partial z_0}, \cdots, \frac{\partial s_5}{\partial z_0}$ and $\frac{\partial s_0}{\partial z_1}, \frac{\partial s_1}{\partial z_1}, \cdots, \frac{\partial s_5}{\partial z_1}$ by central differences.
4. Using the partial derivative estimates from step 3 and the arrays $s_1, \cdots, s_5$, solve the first invariance equation (5.37a) at each mesh point to obtain a new value for s_0.
5. Similarly, solve the second invariance equation (5.37b) at each mesh point using the updated value for s_0 to obtain a new value for s_1 and so on.
6. Iterate the algorithm until further iterates change $s_0, s_1, s_2, \cdots s_5$ negligibly and the convergence criterion (5.17) is satisfied.

In order to obtain a correct manifold, we need a good initial function to start the iteration. One way to do this is the following: Given that $z_0 = s_0 + s_1 + s_2$ and $z_1 = s_3 + s_4 + s_5$

1. Take a quasi-equilibrium approximation of the fast steps, i.e. solve the dimensionless equivalents of $v_{1+} = v_{1-}, v_{2+} = v_{2-}$ and $v_{4+} = v_{4-}, v_{5+} = v_{5-}$ for $s_0, \ldots, s_5$. We obtain:

$$s_0 = \eta_1 s_1, \tag{5.38a}$$

$$s_2 = \frac{\gamma_2 s_1}{\eta_2}, \tag{5.38b}$$

$$s_3 = \frac{\eta_4 s_4}{\gamma_4}, \tag{5.38c}$$

$$s_5 = \frac{\gamma_5 s_4}{\eta_5}. \tag{5.38d}$$

2. Substitute the values of s_0, s_2, s_3, s_5 from equations (5.38) into equations (5.31) and (5.32) . We have:

$$z_0 = \eta_1 s_1 + s_1 + \frac{\gamma_2 s_1}{\eta_2}, \tag{5.39}$$

and

$$z_1 = \frac{\eta_4 s_4}{\gamma_4} + s_4 + \frac{\gamma_5 s_4}{\eta_5}. \tag{5.40}$$

3. Solve for s_1 and s_4 from equations (5.39) and (5.40) respectively. Solving for s_1 and s_4 gives:

$$s_1 = \frac{z_0 \eta_2}{\eta_1 \eta_2 + \eta_2 + \gamma_2}, \tag{5.41}$$

and

$$s_4 = \frac{z_1 \gamma_4 \eta_5}{\eta_4 \eta_5 + \gamma_4 \eta_5 + \gamma_5 \gamma_4}. \tag{5.42}$$

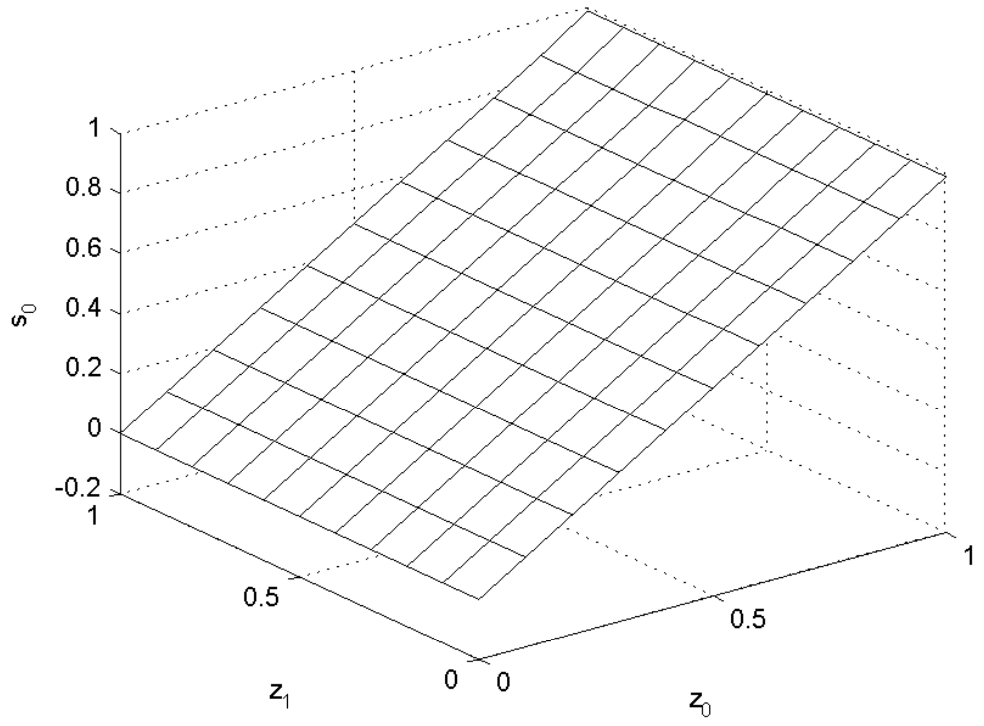

Figure 5.13: The slow invariant manifold for the linear metabolic pathway model for species s_0 with 10 mesh points along each coordinate axis using $\alpha_1,\ldots,\alpha_6 = \beta_1,\ldots,\beta_6 = 100, \gamma_2 = \gamma_4 = \gamma_5 = \eta_1 = \eta_2 = \eta_4 = \eta_5 = 100, \eta_3 = \gamma_3 = 0.01, \gamma_6 = 0.01$ and $\eta_6 = 0$. The solution converges after 3 iterates with a tolerance value of 10^{-3}.

4. Solve for the remaining initial conditions s_0, s_2, s_3 and s_5 with respect to z_0 and z_1.

   To do this, substitute the values of s_1 and s_4 above into equations (5.38a) to (5.38d).

5.4.2 Results

Here we present results obtained using the lumping scheme we developed in the section above.

Figures 5.13 and 5.14 represent the slow invariant manifold for the linear metabolic pathway model for the species s_0 and s_5 respectively. This manifold is a discretized approximation to the solution of the invariance equation, derived from the differential equations

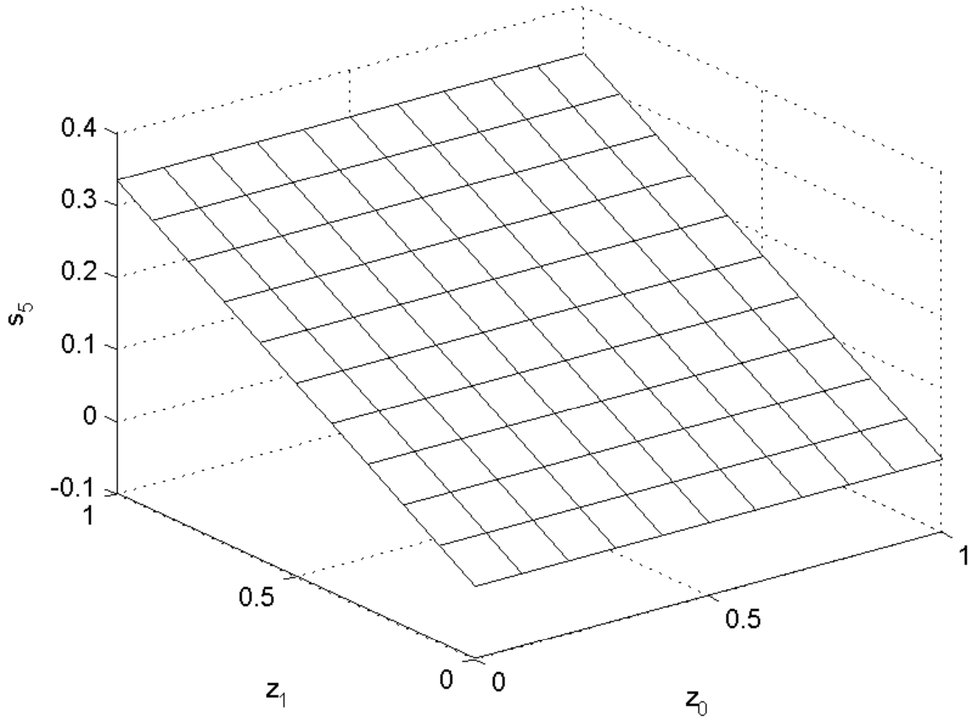

Figure 5.14: The slow invariant manifold for the linear metabolic pathway model for species s_5 with 10 mesh points along each coordinate axis using the same parameters as in figure 5.13. The solution converges after 3 iterates with a tolerance value of 10^{-3}.

for the reaction. The calculation converges after 3 iterates.

5.4.3 Comparing the Full and Reduced Models

To compare the full and reduced models, we can proceed as follows:

1. Pick an initial condition $(z_0^{(0)}, z_1^{(0)})$ for the reduced model.

2. Compute the coordinates $s_0, s_1, \cdots, s_5$ on the manifold, and use them as initial conditions for the full model, i.e. start the integration of the full model on the slow manifold.

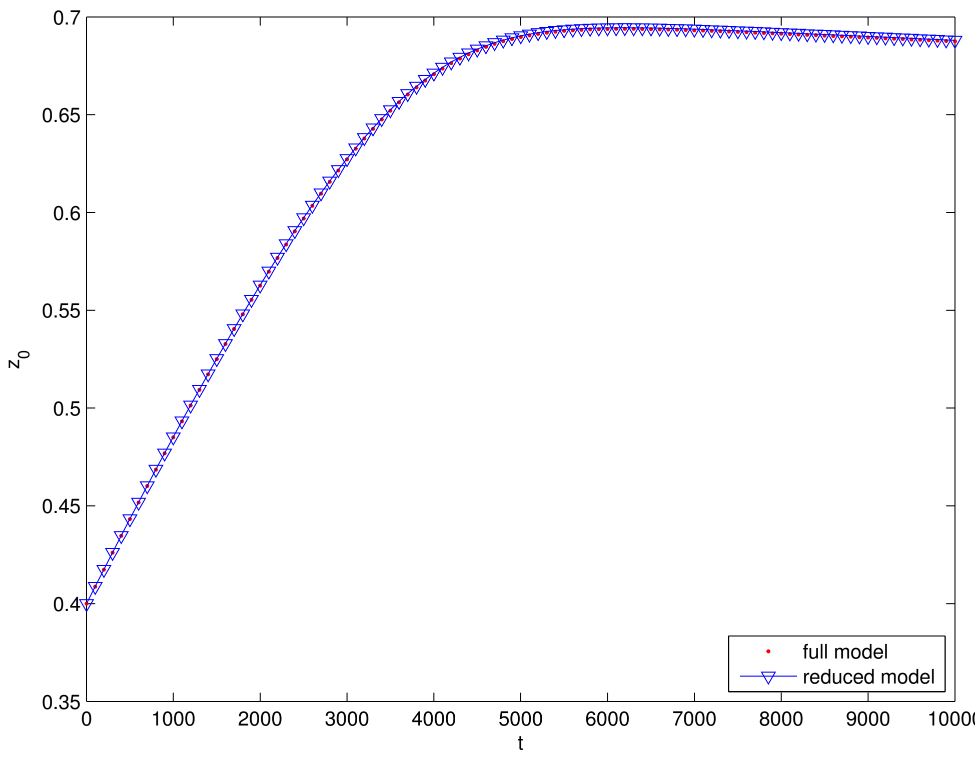

Figure 5.15: z_0 from the full and reduced model for 10 mesh points using the same parameters as in figure 5.13

3. Compute the trajectory of the reduced model by numerically integrating the differential equations:

$$\frac{dz_0}{dt} = \frac{ds_0}{dt} + \frac{ds_1}{dt} + \frac{ds_2}{dt}, \tag{5.43}$$

$$\frac{dz_1}{dt} = \frac{ds_3}{dt} + \frac{ds_4}{dt} + \frac{ds_5}{dt}, \tag{5.44}$$

obtained from equations (5.31) and (5.32). We compute the s_i at a set of discrete mesh points and then interpolate linearly to obtain values between the mesh points during integration.

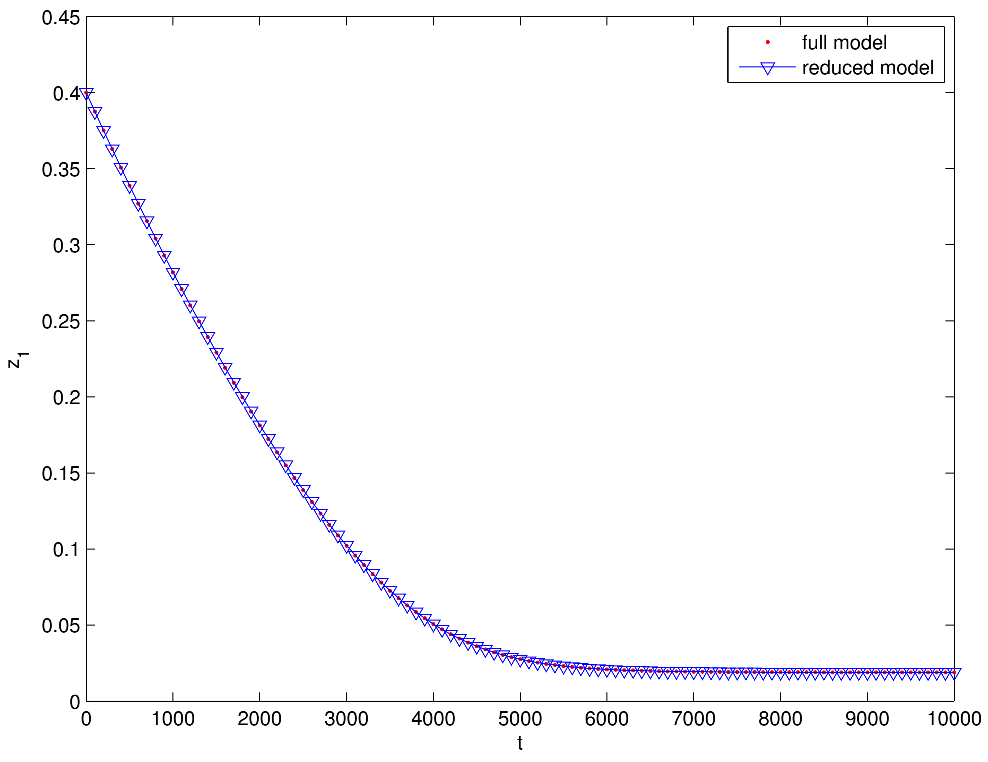

Figure 5.16: z_1 from the full and reduced model for 10 mesh points using the same parameters as in figure 5.13

We observe that the trajectories of the full and reduced models in figures 5.15 and 5.16 are identical. This shows that the reduced model can be used as a representation of the full model after the decay of transients.

5.4.4 Effects of Kinetic Parameters on the Model

In this subsection, we studied the effects of the kinetic parameters on the behaviour of the iterative solution. The kinetic parameters were chosen randomly and for $\alpha_i, \beta_i, i = 1, \dots, 6$ values were generated between $(0.01, 100)$. The fast time scale parameters, i.e. $\eta_1, \eta_2, \eta_4, \eta_5, \gamma_2, \gamma_4, \gamma_5$, were also generated from values between $(1, 100)$. We also generated random numbers between $(0.01, 0.1)$ for the slow time scales, i.e. $\eta_3, \gamma_3, \gamma_6$, and set η_6 to zero. Again, we obtained a convergent solution approximating the slow invariant manifold.

Secondly, fixing all the parameters except $\eta_3, \gamma_3, \gamma_6$ and setting $\gamma_3 = \gamma_6 = 0.01$ while varying η_3, we observed that $\eta_3 \leq 0.01$ is necessary to obtain a convergent solution. Values of η_3 greater than this lead to a divergent scheme. This is so because the assumptions made in choosing our lumps break down, i.e. the concentrations combined within a lump no longer interconvert on the fastest time scales of the system. Similarly, setting $\gamma_3 = \eta_3 = 0.01$ and varying γ_6 and also setting $\eta_3 = \gamma_6 = 0.01$ and varying γ_3, I observed similar behaviour. I also observed that $\gamma_6, \gamma_3 \leq 0.01$ to obtain a convergent solution, and values of γ_6, γ_3 greater than this lead to a divergent scheme.

5.4.5 Discretization Error

The discretization error (D) for the two-dimensional linear metabolic pathway model is given by:

$$D = \sum_i \left[(z^i_{0_{(red)}} - z^i_{0_{(full)}})^2 + (z^i_{1_{(red)}} - z^i_{1_{(full)}})^2 \right] \sqrt{(z^{i-1}_{0_{(red)}} - z^i_{0_{(red)}})^2 + (z^{i-1}_{1_{(red)}} - z^i_{1_{(red)}})^2} \tag{5.45}$$

where $z^i_{0_{(red)}}, z^i_{0_{(full)}}, z^i_{1_{(red)}}, z^i_{1_{(full)}}$ represents the ith time point of z_0 and z_1 for the full and reduced models respectively. The first part of this equation, i.e. $\left[(z^i_{0_{(red)}} - z^i_{0_{(full)}})^2 + (z^i_{1_{(red)}} - z^i_{1_{(full)}})^2 \right]$, represents the error, i.e. the difference between the full and reduced model for the z_0 and z_1 component. The value in the square root represents the distance travelled along a trajectory. The sum is an approximation of a path integral along the trajectory.

As we can see from figure 5.17, the discretization error of this problem is very small

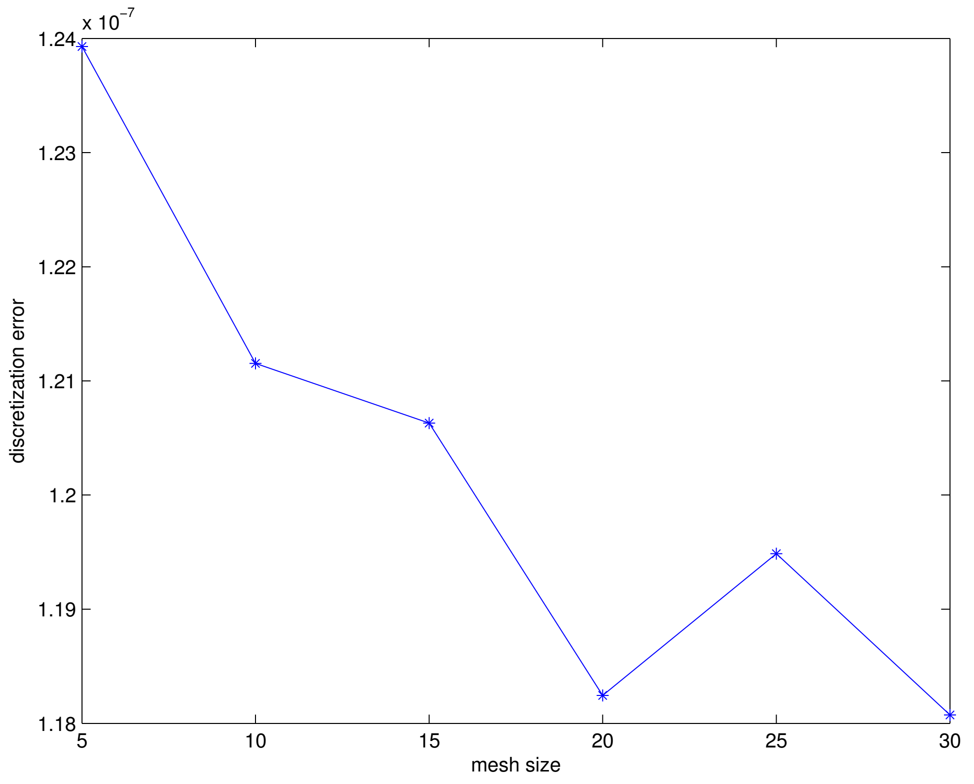

Figure 5.17: Discretization error for 30 mesh points, using the same parameters as in figure 5.13. The tolerance value is 10^{-3}.

(of order 10^{-7}), indicating that the manifold is very accurately approximated.

5.5 Chapter Summary

In this chapter, we illustrated the lumping method in more detail using the linear pathway model and considered a reduction of the model to one or two dimensions. We formulated the invariance equations, obtained the reduced model with numerical examples, compared the full and reduced models, studied the effect of the kinetic parameters and the discretization error. Therefore, we can say that the lumping method can be used to reduce complex models and still fully represent the original model.

Chapter 6

Conclusions and Future Directions

6.1 Summary and Conclusion

We have developed a lumping method for model reduction and illustrated it using the Michaelis-Menten mechanism and the linear metabolic pathway model. For the Michaelis-Menten mechanism, we derived the invariance equation and solved it iteratively using Matlab. We then obtained the slow invariant manifold, a low-dimensional surface on which the system evolves according to the slower time scale, which is a one-dimensional curve in the two-dimensional phase plane for this mechanism.

For the linear metabolic pathway model, we considered both one- and two-dimensional reductions. For the one-dimensional reduction, we obtained a convergent solution for the SIM. Thus instead of studying a six-dimensional model, we are working with a one-dimensional model which is easier to understand and analyse. We compared the full and reduced models starting from initial conditions on the manifold and observed that they are identical. We also compared the reduced model from initial conditions on the manifold with the full model starting from off-manifold initial conditions. We observed that the trajectory of the full model starting at different initial conditions is identical to the reduced model starting from initial conditions on the manifold. This tells us that the reduced model accurately represents the full model after the decay of transients. We considered the case where the model represents a reaction with irreversible product formation and observed that the rate of product formation was very small. We also studied the discretization error and observed that it is very small. This shows that the manifold is very accurately approximated.

For the two-dimensional reduction, we obtained a convergent solution for the slow invariant manifold and observed that the reduced model converges rapidly even for a small number of mesh points (e.g $N = 10$). In order to get a correct slow invariant manifold, we need to lump the species in the model correctly. To do this, we need to have the fast time scales within the lumps and the slow time scales between the lumps as illustrated in figure 5.12, otherwise our lumping scheme leads to nonconvergent iterative processes.

Comparison of the full and reduced models was also done and we observed that they are identical. This means that using the lumping method for model reduction we can relate the full and reduced models, and the reduced model fully represents the original system on the slow time scale(s). We studied the effect of the discretization error, and obtained a small discretization error which decreases as the mesh size increases.

For both the one- and two-dimensional reductions, generating a good initial function to start off the iteration is essential to obtain a convergent solution. We do this by using a quasi-equilibrium approximation to the manifold (explained in detail in chapter 5).

In conclusion, the lumping method we developed in this research is an efficient method for model reduction that would be of great importance for industrial application both locally and abroad, e.g. in ecological modelling [1], hydrocarbon combustion [36, 41], and enzyme kinetics[16].

6.2 Future Directions

Further work can be done by illustrating the relevance of the lumping method to more complex models as given below:

6.2.1 Two Linear Pathways Model

Here one could consider a two linear pathways model that interact in one point. Some questions that may arise here would be whether we can treat such a system using a lumped variable to represent each linear pathway, or rather when, (i.e. under what conditions) can this be done.

6.2.2 Biochemical Network Models

The study of biochemical network models with dense local connections and sparse global connections would be another class of complex models that could be used to illustrate the lumping method we developed in this research. An example is the process of photosynthesis where the Calvin cycle is coupled to the light-induced electron transport in the photosystems through the NADPH and NADP levels, even though each is a subsystem that can be studied in its own right. Here we can imagine a (lumped) variable specifying the state of each subsystem. The rate equations for the coupled system would then be derived from our solution of the invariance equation (3.10).

Bibliography

[1] Pierre Auger and Rafael Bravo de la Parra. Methods of aggregation of variables in population dynamics. *C. R. Acad. Sci., Ser. III*, 323(8):665–674, 2000.

[2] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, England, 1994.

[3] Marzia Bisi, Flammetta Conforto, and Laurent Desvillettes. Quasi-steady-state approximation for reaction-diffusion equations. *Bull. Inst. Math. Acad. Sinica (New series)*, 2(4):823–850, 2007.

[4] J.R. Bowen, A. Acrivos, and A.K. Oppenheim. Singular perturbation refinement to quasi-steady state approximation in chemical kinetics. *Chem. Eng. Sci.*, 18(3):177 – 188, 1963.

[5] Richard L. Burden and Douglas J. Faires. *Numerical Analysis*. PWS-Kent, Boston, 4th edition, 1981.

[6] S. Matt Calder and David Siegel. Properties of the Michaelis-Menten mechanism in phase space. *J. Math. Anal. Appl.*, 339:1044–1064, 2008.

[7] Guy Marie Côme. Radical reaction mechanisms. mathematical theory. *J. Phys. Chem.*, 81(25):2560–2563, 1977.

[8] Guy Marie Côme. Mechanistic modelling of homogeneous reactors: A numerical method. *Comput. & Chem. Eng.*, 3(1-4):603 – 609, 1979.

[9] Guy Marie Côme. *The Use of Computers in the Analysis and Simulation of Complex Reactions*, volume 24 of *Comprehensive Chemical Kinetics*. Elsevier, 1983.

[10] Athel Cornish-Bowden. *Fundamentals of Enzyme Kinetics, 3rd ed.* Portland press, 1995.

[11] Pamela G. Coxson and Kenneth B. Bischoff. Lumping strategy. 1. Introductory techniques and applications of cluster analysis. *Ind. Eng. Chem. Res.*, 26(6):1239–1248, 1987.

[12] Pamela G. Coxson and Kenneth B. Bischoff. Lumping strategy. 2. A system theoretical approach. *Ind. Eng. Chem.*, 26(10):2151–2157, 1987.

[13] S. Philip Crooke, D. Robert Tanner, and Rutherford Aris. The role of dimensionless parameters in the Briggs-Haldane and Michaelis-Menten approximations. *Chem. Eng. Sci.*, 34:1354–1357, 1979.

[14] I. G. Darvey and R. F. Matlak. An investigation of a basic assumption in enzyme kinetics using results of the geometric theory of differential equations. *Bull. Math. Biophys.*, 29:335–341, 1967.

[15] Michael J. Davis and Rex T. Skodje. Geometrical investigation of low-dimensional manifolds in systems approaching equilibrium. *J. Chem. Phys.*, 111(3):859–874, 1999.

[16] Allan Fersht. *Enzyme Structure and Mechanism.* Freeman, New York, 2nd edition, 1975.

[17] Richard J. Field, Endre Koros, and Richard M. Noyes. Oscillations in chemical systems. II. Thorough analysis of temporal oscillation in the bromate-cerium-malonic acid system. *J. Am. Chem. Soc.*, 94(25):8649–8664, 1972.

[18] Richard J. Field and Richard M. Noyes. Oscillations in chemical systems. IV. Limit cycle behavior in a model of a real chemical reaction. *J. Chem. Phys.*, 60(5), 1974.

[19] D.A. Frank-Kamenetskii. Conditions for the application of the Bodenstein method in chemical kinetics (in Russian). *Zh. Fiz. Him.*, 14:695–700, 1940.

[20] Simon J. Fraser. The steady state and equilibrium approximations: A geometrical picture. *J. Chem. Phys.*, 88(8):4732–4738, 1988.

[21] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):335 – 348, 1989.

[22] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, 3rd ed.* Johns Hopkins University Press, 1996.

[23] Dimitris A. Goussis and Mauro Valorani. An efficient iterative algorithm for the approximation of the fast and slow dynamics of stiff systems. *J. Comput. Phys.*, 214(1):316–346, May 2006.

[24] Steinfeld I. Jeffrey, Francisco S. Joseph, and Hase L. William. *Chemical Kinetics and Dynamics, 2nd ed.* Prentice Hall, 1998.

[25] Hans G. Kaper and Tasso J. Kaper. Asymptotic analysis of two reduction methods for systems of chemical reactions. *Physica D*, 165(1-2):66 – 93, 2002.

[26] Keith J. Laidler and Peter S. Bunting. *The Chemical Kinetics of Enzyme Action, 2nd ed.* Clarendon, Oxford, 1973.

[27] S. H. Lam. Using CSP to understand complex chemical kinetics. *Combust. Sci. and Tech.*, 89(5-6):375 – 404, 1993.

[28] S. H. Lam and D. A. Goussis. Understanding complex chemical kinetics with computational singular perturbation. *Symposium (International) on Combustion*, 22(1):931 – 941, 1989.

[29] S. H. Lam and D. A. Goussis. The CSP method for simplifying kinetics. *Int. J. Chem. Kinet.*, 26(4):461–486, 1994.

[30] Genyuan Li and Herschel Rabitz. A general analysis of exact lumping in chemical kinetics. *Chem. Eng. Sci.*, 44(6):1413 – 1430, 1989.

[31] Genyuan Li and Herschel Rabitz. A general analysis of approximate lumping in chemical kinetics. *Chem. Eng. Sci.*, 45(4):977 – 1002, 1990.

[32] Genyuan Li and Herschel Rabitz. Combined symbolic and numerical approach to constrained nonlinear lumping- with application to an H_2/O_2 oxidation model. *Chem. Eng. Sci.*, 51(21):4801–4816, 1996.

[33] Genyuan Li, Herschel Rabitz, and János Tóth. A general analysis of exact nonlinear lumping in chemical kinetics. *Chem. Eng. Sci.*, 49(3):343–361, 1994.

[34] Genyuan Li, Alison S. Tomlin, Herschel Rabitz, and János Tóth. Determination of approximate lumping schemes by a singular perturbation method. *J. Chem. Phys.*, 99(5):3562–3574, 1993.

[35] U. Maas and S.B. Pope. Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space. *Combustion and Flame*, 88(3-4):239 – 264, 1992.

[36] U. Mass, R.W. Dibble, J. Warnatz, and E. Zwicker. *Combusion: Physical and Chemical Fundamentals, Modeling and Simulation, Experiments, Pollutant Formation*. Springer, Berlin, 2nd edition, 1999.

[37] Leonor Michaelis and Maud L. Menten. Die kinetik der invertinwirkung. *Biochem. Z.*, 49(333-369):352, 1913.

[38] J. Nafe and U. Maas. A general algorithm for improving ILDMs. *Combust. Theory Model*, 6(4):697–709, 2002.

[39] An Hoang Nguyen and Simon J. Fraser. Geometrical picture of reaction in enzyme kinetics. *J. Chem. Phys.*, 91(1):186–193, 1989.

[40] David J.M. Park. The hierarchical structure of metabolic networks and the construction of efficient metabolic simulators. *J. Theor. Biol.*, 46(1):31 – 74, 1974.

[41] N. Peters and B. Rogg. *Reduced Kinetic Mechanism for Applications in Combustion Systems*. Springer, Berlin, 2nd edition, 1993.

[42] Valery Petrov, Stephen K. Scott, and Kenneth Showalter. Mixed-mode oscillations in chemical systems. *J. Chem. Phys.*, 97(9), 1992.

[43] Linda Petzold and Wenjie Zhu. Model reduction for chemical kinetics: An optimization approach. *AIChE Journal*, 45(4):869–886, 1999.

[44] J. Davis Philip and Rabinowitz Philip. *Methods of Numerical Integration, 2nd ed.* Academic press, 2007.

[45] S.B. Pope and U. Maas. Implementation of simplified chemical kinetics based on intrinsic low-dimensional manifolds. *Symposium (International) on Combustion*, 24(1):103 – 112, 1992.

[46] Christopher V. Rao and Adam P. Arkin. Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the gillespie algorithm. *J. Chem. Phys.*, 118(11):4999–5010, 2003.

[47] Marc R. Roussel. Forced convergence iterative schemes for the approximation of invariant manifolds. *J. Math. Chem.*, 21:385–393, 1997.

[48] Marc R. Roussel and Simon J. Fraser. Geometry of the steady state approximation: Perturbation and accelerated convergence methods. *J. of Chem. Phys.*, 93(2), 1990.

[49] Marc R. Roussel and Simon J. Fraser. On the geometry of transient relaxation. *J. Chem. Phys.*, 94(11):7106–7113, 1991.

[50] Marc R. Roussel and Simon J. Fraser. Invariant manifold methods for metabolic model reduction. *CHAOS*, 11(1):196–206, 2001.

[51] Marc R. Roussel and Rui Zhu. Reducing a chemical master equation by invariant manifold methods. *J. Chem. Phys.*, 121(18):8716–8730, 2004.

[52] Lee A. Segel and Marshall Slemrod. The quasi-steady-state assumption: A case study in perturbation. *SIAM Review*, 31(3):pp. 446–477, 1989.

[53] Rex T. Skodje and Michael J. Davis. Geometrical simplification of complex kinetic systems. *J. Phys. Chem. A*, 105(45):10356–10365, 2001.

[54] Michael Spence. Cost reduction, competition, and industry performance. *Econometrica*, 52(1):101–122, 1984.

[55] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

[56] Irina Surovtsova, Natalia Simus, Katrin Hübner, Sven Sahle, and Ursula Kummer. Simplification of biochemical models: A general approach based on the analysis of the impact of individual species and reaction on the system dynamics. *BMC Syst. Biol.*, 6(14):1–16, 2012.

[57] Alison S. Tomlin, Tamás Turányi, and Michael J. Pilling. Mathematical tools for the construction, investigation and reduction of combustion mechanisms. In Michael J. Pilling, editor, *Low-Temperature Combustion and Autoignition*, pages 342–375. Elsevier, 1997.

[58] T. Turanyi, A. S. Tomlin, and M. J. Pilling. On the error of the quasi-steady-state approximation. *J. Phys. Chem.*, 97(1):163–172, 1993.

[59] James Wei and J. C. W. Kuo. Lumping analysis in monomolecular reaction systems. analysis of the exactly lumpable system. *Indust. Eng. Chem. Fundam.*, 8(1):114–123, 1969.

[60] A. Zagaris, H. G. Kaper, and T. J. Kaper. Analysis of the computational singular perturbation reduction method for chemical kinetics. *J. Nonlinear Sci.*, 14(1):59 – 91, 2004.

Appendix

Matlab Code for the Michaelis-Menten Mechanism

The Matlab program for the Michaelis-Menten mechanism as illustrated in chapter 4.

```
%Blessing Okeke 25/07/2013

%generating an array of initial conditions for s and c

global k1 k2 km1 e0 iz dsdz dcdz s c snew cnew


%parameters
k1=1;k2=1;km1=1;e0=1;
Km = (km1+k2)/k1;
n=10;
zmax = 2;
h=zmax/n; %step size
zz0=0:h:zmax;


%generating the initial functions
s = zz0;%creates a array
%s=z;
c=e0.*s./(Km+s);
z=c+s;

%computing the finite difference approximation for the derivatives

delta=1;
tol=10^-3;%eps;
itr=0;

while delta>tol
        itr=itr+1;
        for iz=2:n
                 %finite within the mesh
                dsdz(iz)=(s(iz+1)-s(iz-1))/(z(iz+1)-z(iz-1));
                dcdz(iz)=(c(iz+1)-c(iz-1))/(z(iz+1)-z(iz-1));
        end

        %finite difference at initial & boundary condition
        dsdz(1) = (s(2)-s(1))/(z(2)-z(1));
        dsdz(n+1) = (s(n+1)-s(n))/(z(n+1)-z(n));
        dcdz(1) =(c(2)-c(1))/(z(2)-z(1));
        dcdz(n+1) = (c(n+1)-c(n))/(z(n+1)-z(n));


        for iz=1:n+1
                snew(iz)=fzero(@FE_s,s(iz));
                %cnew(iz)=z(iz)-snew(iz);
        end

        for iz=1:n+1
               cnew(iz)=fzero(@FE_c,c(iz));
               %cnew(iz)=z(iz)-snew(iz);
        end

        %condition for determining convergence
        delta=max(abs(s)-abs(snew))+max(abs(c)-abs(cnew));
        s = snew;
        c= cnew;
z=s+c;

end
figure(1)
plot(z,snew)
ylabel('s')
xlabel('z')
xlim([0 2])

figure(2)
plot(z,cnew)
ylabel('c')
xlabel('z')
xlim([0 2])

figure(3)
plot(snew,cnew)
axis([0 1.8 0 0.6])
xlabel('s')
ylabel('c')
%xlim([0 2])

hold on
```

```
[s,c]=meshgrid(0:.2:2, 0:.1:.6);
sdot = -k1.*(e0-c).*s+km1.*c;
cdot = k1.*(e0-c).*s-km1.*c-k2.*c;
quiver(s,c,sdot, cdot,'r')


%numerical integration to obtain the reduced model
n1=10;
t=(0:0.01:n1);
y0=[0.5 0.5];
[t,zz] = ode45(@mm_int,t,y0);
z_new=zz(:,1)+zz(:,2);

v=k2*cnew;

figure(4)
%plot(t,z_new)
plot(z,v)
ylabel('v=k_2*c(z)')
xlabel('z')
xlim([0 2])
```

Matlab functions for the code above.

```
function result=cdot(s,c)
    global k1 km1 k2 e0;
    result=k1*(e0-c)*s-(km1+k2)*c;
end

function result=sdot(s,c)
    global k1 km1 e0;
    result=-k1*(e0-c)*s+km1*c;
end

function result=FE_c(c)
    global dcdz iz snew
    result=cdot(snew(iz),c)*(1-dcdz(iz))-sdot(snew(iz),c)*dcdz(iz);
end

function result=FE_s(s)
    global dsdz iz c
    result=sdot(s,c(iz))*(1-dsdz(iz))-cdot(s,c(iz))*dsdz(iz);
```

```
end
```

Numerical integration for the Michealis-Menten mechanism.

```
%numerical integration of the full model

 function result=mm_int(t,zz)

    global k1 k2 km1 e0

    sdot=-k1*(e0-zz(2))*zz(1)+km1*zz(2);
    cdot=k1*(e0-zz(2))*zz(1)-km1*zz(2)-k2*zz(2);

    result=[sdot;cdot];
 end
```

Matlab Code for the One-Dimensional Linear Pathway Model

Matlab code for the linear pathway model as illustrated in chapter 5.

```
%Blessing Okeke 25/07/2013

%the iterative method for the multistep enzymic conversion
%s(i-1)->s(i), i=1,2..N
%the case with 1 lump and 6 substrate

global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
global gamma1 gamma2 gamma3 gamma4 eta1 eta2 eta3 eta4
global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6
global ds0dz0 ds1dz0 ds2dz0 ds3dz0  ds4dz0 ds5dz0
global s0new s1new s2new s3new s4new s5new
global s0 s1 s2 s3 s4 s5 i z0



n=5:5:30;
m=length(n);
itr_store=zeros(m,1);
for k=1:m

    %matrices containing initial condition at each mesh point

    %n=10;
    z0max = 1;
```

```
h0=z0max/n(k); %step size
zz0=0:h0:z0max;

%generating the initial functions
z0 = zz0';%creates a array

denom = eta2*eta3*eta4*eta5+eta3*eta4*eta5*gamma2+eta4*eta5*gamma2*gamma3...
+eta5*gamma2*gamma3*gamma4+gamma2*gamma3*gamma4*gamma5...
+eta1*eta2*eta3*eta4*eta5;
for i=1:n(k)+1
        s0(i) = (eta1*eta2*eta3*eta4*eta5*z0(i))/denom;
        s1(i) = (eta2*eta3*eta4*eta5*z0(i))/denom;
        s2(i) = (eta3*eta4*eta5*gamma2*z0(i))/denom;
        s3(i) = (eta4*eta5*gamma2*gamma3*z0(i))/denom;
        s4(i) = (eta5*gamma2*gamma3*gamma4*z0(i))/denom;
        s5(i) = (gamma2*gamma3*gamma4*gamma5*z0(i))/denom;
 end

delta=1;
tol=1e-3;
itr=0;


while delta>tol
    tic;
    itr=itr+1; %counts the number of iteration
    itr_store(k)=itr;

    %computing the finite difference approximation for the derivatives
    %inside the array
    for i=2:n(k)
         ds0dz0(i)=(s0(i+1)-s0(i-1))/(2*h0);%finite within the array
         ds1dz0(i)=(s1(i+1)-s1(i-1))/(2*h0);
         ds2dz0(i)=(s2(i+1)-s2(i-1))/(2*h0);
         ds3dz0(i)=(s3(i+1)-s3(i-1))/(2*h0);
         ds4dz0(i)=(s4(i+1)-s4(i-1))/(2*h0);
         ds5dz0(i)=(s5(i+1)-s5(i-1))/(2*h0);
    end

    %finite difference at initial & boundary condition
    ds0dz0(1) = (s0(2)-s0(1))/h0;
    ds0dz0(n(k)+1) = (s0(n(k)+1)-s0(n(k)))/h0;
```

```
ds1dz0(1) = (s1(2)-s1(1))/h0;
ds1dz0(n(k)+1) = (s1(n(k)+1)-s1(n(k)))/h0;

ds2dz0(1) = (s2(2)-s2(1))/h0;
ds2dz0(n(k)+1) = (s2(n(k)+1)-s2(n(k)))/h0;

ds3dz0(1) = (s3(2)-s3(1))/h0;
ds3dz0(n(k)+1) = (s3(n(k)+1)-s3(n(k)))/h0;

ds4dz0(1) = (s4(2)-s4(1))/h0;
ds4dz0(n(k)+1) = (s4(n(k)+1)-s4(n(k)))/h0;

ds5dz0(1) = (s5(2)-s5(1))/h0;
ds5dz0(n(k)+1) = (s5(n(k)+1)-s5(n(k)))/h0;

%computes the solution at each array point
for i=1:n(k)+1
    s0new(i)=fsolve(@FE_s0,s0(i));
end

for i=1:n(k)+1
    s1new(i)=fsolve(@FE_s1,s1(i));
end

for i=1:n(k)+1
    s2new(i)=fsolve(@FE_s2,s2(i));
end

for i=1:n(k)+1
    s3new(i)=fsolve(@FE_s3,s3(i));
end

for i=1:n(k)+1
    s4new(i)=fsolve(@FE_s4,s4(i));
end

for i=1:n(k)+1
    s5new(i)=fsolve(@FE_s5,s5(i));
end



delta = max(abs(s0-s0new))+max(abs(s1-s1new))...
```

```
        +max(abs(s2-s2new))+max(abs(s3-s3new))...
        +max(abs(s4-s4new))+max(abs(s5-s5new));

    %e(itr)=delta; % iterative error
    s0=s0new;
    s1= s1new;
    s2=s2new;
    s3=s3new;
    s4=s4new;
    s5=s5new;

    %s0_store(:,itr)=s0new;
    %s1_store(:,itr)=s1new;
    %s2_store(:,itr)=s2new;
    %s3_store(:,itr)=s3new;
    %s4_store(:,itr)=s4new;
    %s5_store(:,itr)=s5new;
end
tElapsed = toc;

%plot of the SIM

figure(1)
plot(z0,s0new)
xlabel('z')
ylabel('s_0')

figure(2)
plot(z0,s5new)
xlabel('z')
ylabel('s_5')


%========= comparing the full and reduced model======================

%numerical integration of the full model
na=100000;
t=(0:100:na);
a0=interp1(z0,s0new,0.8001);
a1=interp1(z0,s1new,0.8001);
a2=interp1(z0,s2new,0.8001);
a3=interp1(z0,s3new,0.8001);
a4=interp1(z0,s4new,0.8001);
```

```
a5=interp1(z0,s5new,0.8001);
x0=[a0 a1 a2 a3 a4 a5];

[t,s] = ode15s(@num_int_full,t,x0);

%the full model: suming up the s’s to obtain the correpsonding z0 and z1
z_1=s(:,1)+s(:,2)+s(:,3)+s(:,4)+s(:,5)+s(:,6);

%numerical integration of the reduced model
n1=na;
t1=(0:100:n1);
y0=0.8001;
[t1,z] = ode15s(@num_int2_z,t1,y0);

%numerical integration of the full model starting from different initial
%condition on the manifold
n2=na;
t2=(0:100:n2);
xx0=[0.8001 0 0 0 0 0];

[t2,ss] = ode15s(@num_int_full,t2,xx0);

%the full model: suming up the s’s to obtain the correpsonding z0 and z1
zz_1=ss(:,1)+ss(:,2)+ss(:,3)+ss(:,4)+ss(:,5)+ss(:,6);


%plots to compare the full and reduced model wrt to t
figure(3)
plot(t,z_1,’.r’)
hold on
plot(t1,z,’-v’)
xlabel(’t’)
ylabel(’z’)
legend(’full model’,’reduced model’)
xlim([0 na])


%plots to compare the full and reduced model wrt to t within then manifold
%and wt diff ics on the manifold
figure(4)
plot(t,z_1,’.r’)
hold on
plot(t1,z,’-v’)
```

```
    plot(t2,zz_1,'-')
    xlabel('t')
    ylabel('z')
    legend('full model','reduced model','full model wt diff ics')
    xlim([0 na])




    %=====================================================
    %checking the case where the model represents a reaction with irreversible
    %product formation i.e eta6=0

    s6_update=1-(s0new+s1new+s2new+s3new+s4new+s5new);

    s6_dot=(gamma6.*s5new)./(1+alpha6.*s5new+beta6.*s6_update);

    figure(6)
    plot(s6_dot,z0)
    xlabel('$\dot{s}_{6}(z)$','interpreter','latex')
    ylabel('z')
    %legend('\dot{s_6}','z')


     %==========discretization error================

    error1 = zeros(length(z),1);
    for kk=2:length(z)
        distance=sqrt((z(kk-1)-z(kk))^2) ;
        z_part=(z(kk)-z_1(kk))^2 ;
        error1(kk)=z_part*distance;
    end
    error(k)=sum(error1);



end

figure(5)
plot(n,itr_store,'*-')
xlabel('mesh size')
ylabel('number of iterations')
```

```
figure(6)
plot(n,error,'*-')
xlabel('mesh size')
ylabel('discretization error')
```

Parameters file for the above code.

```
global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4 eta1 eta2 eta3 eta4
global gamma2 gamma3 gamma4
global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6

alpha1=100;  alpha2 =100; alpha3 =100; alpha4 =100; alpha5=100; alpha6=100;
beta1 = 100; beta2 = 100;  beta3 =100; beta4 = 100; beta5=100; beta6=100;
gamma2 = 100;  gamma4 =100; gamma5=100; %1e-4

eta1 = 1; eta2 =10;    eta4 =1; eta5=10;
gamma3 =1e-2;  eta3 =1e-2; gamma6=1e-4;  eta6=1e-4;
```

Matlab functions for the code above.

```
function result=FE_s0(s0)
    global ds0dz0 i s1 s2 s3 s4 s5
    result=s0dot(s0,s1(i))*(1-ds0dz0(i))-s1dot(s0,s1(i),s2(i))*ds0dz0(i)...
        -s2dot(s1(i),s2(i),s3(i))*ds0dz0(i)-s3dot(s2(i),s3(i),s4(i))*ds0dz0(i)...
        -s4dot(s3(i),s4(i),s5(i))*ds0dz0(i)...
        -s5dot(s0,s1(i),s2(i),s3(i),s4(i),s5(i))*ds0dz0(i);
end

function result=FE_s1(s1)
    global ds1dz0 i s0new s2 s3 s4 s5
    result=-s0dot(s0new(i),s1)*ds1dz0(i)+s1dot(s0new(i),s1,s2(i))*(1-ds1dz0(i))...
     -s2dot(s1,s2(i),s3(i))*ds1dz0(i)-s3dot(s2(i),s3(i),s4(i))*ds1dz0(i)...
     -s4dot(s3(i),s4(i),s5(i))*ds1dz0(i)...
     -s5dot(s0new(i),s1,s2(i),s3(i),s4(i),s5(i))*ds1dz0(i);
end

function result=FE_s2(s2)
    global ds2dz0 i s0new s1new s3 s4 s5
    result=-s0dot(s0new(i),s1new(i))*ds2dz0(i)...
    -s1dot(s0new(i),s1new(i),s2)*ds2dz0(i)...
    +s2dot(s1new(i),s2,s3(i))*(1-ds2dz0(i))-s3dot(s2,s3(i),s4(i))*ds2dz0(i)...
    -s4dot(s3(i),s4(i),s5(i))*ds2dz0(i)...
    -s5dot(s0new(i),s1new(i),s2,s3(i),s4(i),s5(i))*ds2dz0(i);
end

function result=FE_s3(s3)
    global ds3dz0 i s0new s1new s2new s4 s5
    result=-s0dot(s0new(i),s1new(i))*ds3dz0(i)...
    -s1dot(s0new(i),s1new(i),s2new(i))*ds3dz0(i)...
    -s2dot(s1new(i),s2new(i),s3)*ds3dz0(i)...
    +s3dot(s2new(i),s3,s4(i))*(1-ds3dz0(i))...
    -s4dot(s3,s4(i),s5(i))*ds3dz0(i)...
    -s5dot(s0new(i),s1new(i),s2new(i),s3,s4(i),s5(i))*ds3dz0(i);
end

function result=FE_s4(s4)
    global ds4dz0 i s0new s1new s2new s3new s5
    result=-s0dot(s0new(i),s1new(i))*ds4dz0(i)...
    -s1dot(s0new(i),s1new(i),s2new(i))*ds4dz0(i)...
    -s2dot(s1new(i),s2new(i),s3new(i))*ds4dz0(i)...
    -s3dot(s2new(i),s3new(i),s4)*ds4dz0(i)...
    +s4dot(s3new(i),s4,s5(i))*(1-ds4dz0(i))...
    -s5dot(s0new(i),s1new(i),s2new(i),s3new(i),s4,s5(i))*ds4dz0(i);
end

function result=FE_s5(s5)
    global ds5dz0 i s0new s1new s2new s3new s4new
    result=-s0dot(s0new(i),s1new(i))*ds5dz0(i)...
    -s1dot(s0new(i),s1new(i),s2new(i))*ds5dz0(i)...
    -s2dot(s1new(i),s2new(i),s3new(i))*ds5dz0(i)...
    -s3dot(s2new(i),s3new(i),s4new(i))*ds5dz0(i)...
    -s4dot(s3new(i),s4new(i),s5)*ds5dz0(i)...
    +s5dot(s0new(i),s1new(i),s2new(i),s3new(i),s4new(i),s5)*(1-ds5dz0(i));
end

function result=s0dot(s0,s1)
    global alpha1 beta1 eta1
    result=(eta1*s1)/(1+alpha1*s0+beta1*s1)-s0/(1+alpha1*s0+beta1*s1);
end

function result=s1dot(s0,s1,s2)
    global alpha1 beta1 eta1 alpha2 gamma2 eta2 beta2
    result=(s0)/(1+alpha1*s0+beta1*s1)-(eta1*s1)/(1+alpha1*s0+beta1*s1)...
        -(gamma2*s1)/(1+alpha2*s1+beta2*s2)+(eta2*s2)/(1+alpha2*s1+beta2*s2);
```

```
end

function result=s2dot(s1,s2,s3)
    global alpha2 gamma2 eta2 beta2 gamma3 alpha3 eta3 beta3
    result=(gamma2*s1)/(1+alpha2*s1+beta2*s2)-(eta2*s2)/(1+alpha2*s1+beta2*s2)...
        -(gamma3*s2)/(1+alpha3*s2+beta3*s3)+(eta3*s3)/(1+alpha3*s2+beta3*s3);
end

function result=s3dot(s2,s3,s4)
    global gamma3 alpha3 eta3 beta3 gamma4 alpha4 beta4 eta4
    result=(gamma3*s2)/(1+alpha3*s2+beta3*s3)-(eta3*s3)/(1+alpha3*s2+beta3*s3)...
        -(gamma4*s3)/(1+alpha4*s3+beta4*s4)+(eta4*s4)/(1+alpha4*s3+beta4*s4);
end

function result=s4dot(s3,s4,s5)
    global  gamma4 alpha4 beta4 eta4 gamma5 alpha5 eta5 beta5
    result=(gamma4*s3)/(1+alpha4*s3+beta4*s4)-(eta4*s4)/(1+alpha4*s3+beta4*s4)...
        -(gamma5*s4)/(1+alpha5*s4+beta5*s5)+(eta5*s5)/(1+alpha5*s4+beta5*s5);
end

function result=s5dot(s0,s1,s2,s3,s4,s5)
    s6=1-s0-s1-s2-s3-s4-s5;
    global  gamma5 alpha5 eta5 beta5 gamma6 alpha6 beta6 eta6
    result=(gamma5*s4)/(1+alpha5*s4+beta5*s5)-(eta5*s5)/(1+alpha5*s4+beta5*s5)...
        -(gamma6*s5)/(1+alpha6*s5+beta6*s6)+(eta6*s6)/(1+alpha6*s5+beta6*s6);
end
```

Numerical integration of the full and reduced model.

```
% the reduced model

function result=num_int2_z(t1,z)
    global s0new s1new s2new s3new s4new s5new z0
    global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
    global gamma2 gamma3 gamma4 eta1 eta2 eta3 eta4
    global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6


    s0_in=interp1(z0,s0new,z);
    s1_in=interp1(z0,s1new,z);
    s2_in=interp1(z0,s2new,z);
    s3_in=interp1(z0,s3new,z);
    s4_in=interp1(z0,s4new,z);
    s5_in=interp1(z0,s5new,z);


    s6_in=1-s0_in-s1_in-s2_in-s3_in-s4_in-s5_in;


    %computes the rhs of the rate equations

    ds0dt=(eta1*s1_in)/(1+alpha1*s0_in+beta1*s1_in)...
    -s0_in/(1+alpha1*s0_in+beta1*s1_in);

    ds1dt=(s0_in)/(1+alpha1*s0_in+beta1*s1_in)...
    -(eta1*s1_in)/(1+alpha1*s0_in+beta1*s1_in)...
    -(gamma2*s1_in)/(1+alpha2*s1_in+beta2*s2_in)...
    +(eta2*s2_in)/(1+alpha2*s1_in+beta2*s2_in);

    ds2dt=(gamma2*s1_in)/(1+alpha2*s1_in+beta2*s2_in)...
    -(eta2*s2_in)/(1+alpha2*s1_in+beta2*s2_in)...
    -(gamma3*s2_in)/(1+alpha3*s2_in+beta3*s3_in)...
    +(eta3*s3_in)/(1+alpha3*s2_in+beta3*s3_in);

    ds3dt=(gamma3*s2_in)/(1+alpha3*s2_in+beta3*s3_in)...
    -(eta3*s3_in)/(1+alpha3*s2_in+beta3*s3_in)...
    -(gamma4*s3_in)/(1+alpha4*s3_in+beta4*s4_in)...
    +(eta4*s4_in)/(1+alpha4*s3_in+beta4*s4_in);

    ds4dt=(gamma4*s3_in)/(1+alpha4*s3_in+beta4*s4_in)...
    -(eta4*s4_in)/(1+alpha4*s3_in+beta4*s4_in)...
    -(gamma5*s4_in)/(1+alpha5*s4_in+beta5*s5_in)...
    +(eta5*s5_in)/(1+alpha5*s4_in+beta5*s5_in);

    ds5dt=(gamma5*s4_in)/(1+alpha5*s4_in+beta5*s5_in)...
    -(eta5*s5_in)/(1+alpha5*s4_in+beta5*s5_in)...
    -(gamma6*s5_in)/(1+alpha6*s5_in+beta6*s6_in)...
    +(eta6*s6_in)/(1+alpha6*s5_in+beta6*s6_in);

   result=ds0dt+ds1dt+ds2dt+ds3dt+ds4dt+ds5dt;


end
```

```
%numerical integration of the full model

function result=num_int_full(t,s)


    global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
    global gamma2 gamma3 gamma4 eta1 eta2 eta3 eta4
    global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6



    s(7)=1-s(1)-s(2)-s(3)-s(4)-s(5)-s(6);

    result=[(eta1*s(2))/(1+alpha1*s(1)+beta1*s(2))...
    -s(1)/(1+alpha1*s(1)+beta1*s(2));
    (s(1))/(1+alpha1*s(1)+beta1*s(2))...
    -(eta1*s(2))/(1+alpha1*s(1)+beta1*s(2))...
    -(gamma2*s(2))/(1+alpha2*s(2)+beta2*s(3))...
    +(eta2*s(3))/(1+alpha2*s(2)+beta2*s(3));
    (gamma2*s(2))/(1+alpha2*s(2)+beta2*s(3))...
    -(eta2*s(3))/(1+alpha2*s(2)+beta2*s(3))...
    -(gamma3*s(3))/(1+alpha3*s(3)+beta3*s(4))...
    +(eta3*s(4))/(1+alpha3*s(3)+beta3*s(4));
    (gamma3*s(3))/(1+alpha3*s(3)+beta3*s(4))...
    -(eta3*s(4))/(1+alpha3*s(3)+beta3*s(4))...
    -(gamma4*s(4))/(1+alpha4*s(4)+beta4*s(5))...
    +(eta4*s(5))/(1+alpha4*s(4)+beta4*s(5));
    (gamma4*s(4))/(1+alpha4*s(4)+beta4*s(5))...
    -(eta4*s(5))/(1+alpha4*s(4)+beta4*s(5))...
    -(gamma5*s(5))/(1+alpha5*s(5)+beta5*s(6))...
    +(eta5*s(6))/(1+alpha5*s(5)+beta5*s(6));
    (gamma5*s(5))/(1+alpha5*s(5)+beta5*s(6))...
    -(eta5*s(6))/(1+alpha5*s(5)+beta5*s(6))...
    -(gamma6*s(6))/(1+alpha6*s(6)+beta6*s(7))...
    +(eta6*s(7))/(1+alpha6*s(6)+beta6*s(7))];


end
```

Matlab Code for the Two-Dimensional Linear Pathway Model

Matlab code for the two-dimensional linear pathway model.

```
%Blessing Okeke 25/07/2013
%main file

%the iterative method for the multistep enzymic conversion
%s(i-1)->s(i), i=1,2..N
%the case where we have 2 lumps  and 6 substrates. The lumps are splitted
%equally with discretization error included

global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4 eta1 eta2 eta3 eta4
global gamma2 gamma3 gamma4
global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6
global ds0dz0 ds0dz1 ds1dz0 ds1dz1 ds2dz0 ds2dz1 ds3dz0 ds3dz1
global ds4dz0 ds4dz1 ds5dz0 ds5dz1
global s0new s1new s2new s3new s4new s5new
global s0 s1 s2 s3 s4 s5 i j k
global z0 z1


n=5:5:30;
m=length(n);
itr_store=zeros(m,1); %counts number of iterates
for k=1:m

    z0max =1;
    z1max =1;
    h0=z0max/n(k); %step size
    h1=z1max/n(k); %step size
    zz0=0:h0:z0max; %grid on the x axis representing z0=s0+s2+s4
    zz1=0:h1:z1max; %grid on the y axis representing z1=s1+s3+s5
    tic; %setting timer

    %generating the initial functions

    [z0,z1] = meshgrid(zz0,zz1);%creates a mesh of z0 and z1

    for i=1:n(k)+1
        tStart=tic; %start timing
        for j=1:n(k)+1
         s0(i,j) = (eta1*eta2*z0(i,j))/(eta1*eta2+eta2+gamma2);
         s1(i,j) = (eta2*z0(i,j))/(eta1*eta2+eta2+gamma2);
```

```
     s2(i,j) = (gamma2*z0(i,j))/(eta1*eta2+eta2+gamma2);
     s3(i,j) = (eta4*eta5*z1(i,j))/(eta4*eta5+gamma4*eta5+gamma5*gamma4);
     s4(i,j) = (gamma4*eta5*z1(i,j))/(eta4*eta5+gamma4*eta5+gamma5*gamma4);
     s5(i,j) = (gamma4*gamma5*z1(i,j))/(eta4*eta5+gamma4*eta5+gamma5*gamma4);
    end
end

delta=1;
tol=1e-3;
itr=0;

while delta>tol

    itr=itr+1;
    itr_store(k)=itr;
    for i=2:n(k) %represents columns
        for j=2:n(k) %represents rows
           %finite difference within the mesh
           ds0dz0(i,j)=(s0(i,j+1)-s0(i,j-1))/(2*h0);
           ds0dz1(i,j)=(s0(i+1,j)-s0(i-1,j))/(2*h1);

           ds1dz0(i,j)=(s1(i,j+1)-s1(i,j-1))/(2*h0);
           ds1dz1(i,j)=(s1(i+1,j)-s1(i-1,j))/(2*h1);

           ds2dz0(i,j)=(s2(i,j+1)-s2(i,j-1))/(2*h0);
           ds2dz1(i,j)=(s2(i+1,j)-s2(i-1,j))/(2*h1);

           ds3dz0(i,j)=(s3(i,j+1)-s3(i,j-1))/(2*h0);
           ds3dz1(i,j)=(s3(i+1,j)-s3(i-1,j))/(2*h1);

           ds4dz0(i,j)=(s4(i,j+1)-s4(i,j-1))/(2*h0);
           ds4dz1(i,j)=(s4(i+1,j)-s4(i-1,j))/(2*h1);

           ds5dz0(i,j)=(s5(i,j+1)-s5(i,j-1))/(2*h0);
           ds5dz1(i,j)=(s5(i+1,j)-s5(i-1,j))/(2*h1);
        end
    end

%outer edges and corners for s0
    for j=1:n(k)
        ds0dz0(1,j)=(s0(1,j+1)-s0(1,j))/h0;
        ds0dz0(n(k)+1,j)=(s0(n(k)+1,j+1)-s0(n(k)+1,j))/h0;
        ds0dz0(j,1)=(s0(j,2)-s0(j,1))/h0;
        ds0dz0(j,n(k)+1)=(s0(j,n(k)+1)-s0(j,n(k)))/h0;
        ds0dz1(j,1)=(s0(j+1,1)-s0(j,1))/h1;
        ds0dz1(j,n(k)+1)=(s0(j+1,n(k)+1)-s0(j,n(k)+1))/h1;
        ds0dz1(1,j)=(s0(2,j)-s0(1,j))/h1;
        ds0dz1(n(k)+1,j)=(s0(n(k)+1,j)-s0(n(k),j))/h1;
    end
    ds0dz0(n(k)+1,n(k)+1)=(s0(n(k)+1,n(k)+1)-s0(n(k)+1,n(k)))/h0;
    ds0dz1(n(k)+1,n(k)+1)=(s0(n(k)+1,n(k)+1)-s0(n(k),n(k)+1))/h1;


    %outer edges and corners for s1
    for j=1:n(k)
        ds1dz0(1,j)=(s1(1,j+1)-s1(1,j))/h0;
        ds1dz0(n(k)+1,j)=(s1(n(k)+1,j+1)-s1(n(k)+1,j))/h0;
        ds1dz0(j,1)=(s1(j,2)-s1(j,1))/h0;
        ds1dz0(j,n(k)+1)=(s1(j,n(k)+1)-s1(j,n(k)))/h0;
        ds1dz1(j,1)=(s1(j+1,1)-s1(j,1))/h1;
        ds1dz1(j,n(k)+1)=(s1(j+1,n(k)+1)-s1(j,n(k)+1))/h1;
        ds1dz1(1,j)=(s1(2,j)-s1(1,j))/h1;
        ds1dz1(n(k)+1,j)=(s1(n(k)+1,j)-s1(n(k),j))/h1;
    end
    ds1dz0(n(k)+1,n(k)+1)=(s1(n(k)+1,n(k)+1)-s1(n(k)+1,n(k)))/h0;
    ds1dz1(n(k)+1,n(k)+1)=(s1(n(k)+1,n(k)+1)-s1(n(k),n(k)+1))/h1;

    %outer edges and corners for s2
    for j=1:n(k)
        ds2dz0(1,j)=(s2(1,j+1)-s2(1,j))/h0;
        ds2dz0(n(k)+1,j)=(s2(n(k)+1,j+1)-s2(n(k)+1,j))/h0;
        ds2dz0(j,1)=(s2(j,2)-s2(j,1))/h0;
        ds2dz0(j,n(k)+1)=(s2(j,n(k)+1)-s2(j,n(k)))/h0;
        ds2dz1(j,1)=(s2(j+1,1)-s2(j,1))/h1;
        ds2dz1(j,n(k)+1)=(s2(j+1,n(k)+1)-s2(j,n(k)+1))/h1;
        ds2dz1(1,j)=(s2(2,j)-s2(1,j))/h1;
        ds2dz1(n(k)+1,j)=(s2(n(k)+1,j)-s2(n(k),j))/h1;
    end
    ds2dz0(n(k)+1,n(k)+1)=(s2(n(k)+1,n(k)+1)-s2(n(k)+1,n(k)))/h0;
    ds2dz1(n(k)+1,n(k)+1)=(s2(n(k)+1,n(k)+1)-s2(n(k),n(k)+1))/h1;



    %----------------------------next lump(z1)-----------------------------

    %outer edges and corners for s3
```

```
 for j=1:n(k)
     ds3dz0(1,j)=(s3(1,j+1)-s3(1,j))/h0;
     ds3dz0(n(k)+1,j)=(s3(n(k)+1,j+1)-s3(n(k)+1,j))/h0;
     ds3dz0(j,1)=(s3(j,2)-s3(j,1))/h0;
     ds3dz0(j,n(k)+1)=(s3(j,n(k)+1)-s3(j,n(k)))/h0;
     ds3dz1(1,j)=(s3(2,j)-s3(1,j))/h1;
     ds3dz1(n(k)+1,j)=(s3(n(k)+1,j)-s3(n(k),j))/h1;
     ds3dz1(j,1)=(s3(j+1,1)-s3(j,1))/h1;
     ds3dz1(j,n(k)+1)=(s3(j+1,n(k)+1)-s3(j,n(k)+1))/h1;
end
ds3dz0(n(k)+1,n(k)+1)=(s3(n(k)+1,n(k)+1)-s3(n(k)+1,n(k)))/h0;
ds3dz1(n(k)+1,n(k)+1)=(s3(n(k)+1,n(k)+1)-s3(n(k),n(k)+1))/h1;


%outer edges and corners for s4
 for j=1:n(k)
     ds4dz0(1,j)=(s4(1,j+1)-s4(1,j))/h0;
     ds4dz0(n(k)+1,j)=(s4(n(k)+1,j+1)-s4(n(k)+1,j))/h0;
     ds4dz0(j,1)=(s4(j,2)-s4(j,1))/h0;
     ds4dz0(j,n(k)+1)=(s4(j,n(k)+1)-s4(j,n(k)))/h0;
     ds4dz1(1,j)=(s4(2,j)-s4(1,j))/h1;
     ds4dz1(n(k)+1,j)=(s4(n(k)+1,j)-s4(n(k),j))/h1;
     ds4dz1(j,1)=(s4(j+1,1)-s4(j,1))/h1;
     ds4dz1(j,n(k)+1)=(s4(j+1,n(k)+1)-s4(j,n(k)+1))/h1;
end
ds4dz0(n(k)+1,n(k)+1)=(s4(n(k)+1,n(k)+1)-s4(n(k)+1,n(k)))/h0;
ds4dz1(n(k)+1,n(k)+1)=(s4(n(k)+1,n(k)+1)-s4(n(k),n(k)+1))/h1;


%outer edges and corners for s5
 for j=1:n(k)
     ds5dz0(1,j)=(s5(1,j+1)-s5(1,j))/h0;
     ds5dz0(n(k)+1,j)=(s5(n(k)+1,j+1)-s5(n(k)+1,j))/h0;
     ds5dz0(j,1)=(s5(j,2)-s5(j,1))/h0;
     ds5dz0(j,n(k)+1)=(s5(j,n(k)+1)-s5(j,n(k)))/h0;
     ds5dz1(1,j)=(s5(2,j)-s5(1,j))/h1;
     ds5dz1(n(k)+1,j)=(s5(n(k)+1,j)-s5(n(k),j))/h1;
     ds5dz1(j,1)=(s5(j+1,1)-s5(j,1))/h1;
     ds5dz1(j,n(k)+1)=(s5(j+1,n(k)+1)-s5(j,n(k)+1))/h1;
end
ds5dz0(n(k)+1,n(k)+1)=(s5(n(k)+1,n(k)+1)-s5(n(k)+1,n(k)))/h0;
ds5dz1(n(k)+1,n(k)+1)=(s5(n(k)+1,n(k)+1)-s5(n(k),n(k)+1))/h1;
```

```
%iteration for the invariance equation

for i=1:n(k)+1
 for j=1:n(k)+1
     s0new(i,j)=fsolve(@FE_s0,s0(i,j));
 end
end

for i=1:n(k)+1
 for j=1:n(k)+1
     s1new(i,j)=fsolve(@FE_s1,s1(i,j));
 end
end



for i=1:n(k)+1
 for j=1:n(k)+1
     s2new(i,j)=fsolve(@FE_s2,s2(i,j));
 end
end


for i=1:n(k)+1
 for j=1:n(k)+1
     s3new(i,j)=fsolve(@FE_s3,s3(i,j));
 end
end


 for i=1:n(k)+1
  for j=1:n(k)+1
     s4new(i,j)=fsolve(@FE_s4,s4(i,j));
  end
 end

 for i=1:n(k)+1
  for j=1:n(k)+1
     s5new(i,j)=fsolve(@FE_s5,s5(i,j));
  end
 end
```

```
   delta = max(max(abs(s0-s0new)))+max(max(abs(s1-s1new)))...
   +max(max(abs(s2-s2new)))+max(max(abs(s3-s3new)))...
   +max(max(abs(s4-s4new)))+max(max(abs(s5-s5new)));

   %error(itr)=delta; %iterative error
   s0=s0new;
   s1=s1new;
   s2=s2new;
   s3=s3new;
   s4=s4new;
   s5=s5new;


   %store the the values of the iterates at each point in computations
   %s0_store(:,:,itr)=s0new;
   %s1_store(:,:,itr)=s1new;
   %s2_store(:,:,itr)=s2new;
   %s3_store(:,:,itr)=s3new;
   %s4_store(:,:,itr)=s4new;
   %s5_store(:,:,itr)=s5new;

end
tElapsed = toc(tStart);

figure(1)
h=mesh(s0new);
set(h,'LineWidth',1)
xlabel('z_0')
ylabel('z_1')
zlabel('s_0')



figure(2)
b=mesh(s5new);
set(b,'LineWidth',1)
xlabel('z_0')
ylabel('z_1')
zlabel('s_5')

%========= comparing the full and reduced model=======================
%numerical integration of the full model

na=10000;
t=(0:100:na);
x0=[0.1568 0.1352 0.1080 0.1026 0.1278 0.1697];
[t,s] = ode15s(@num_int_full,t,x0);



%the full model: suming up the s’s to obtain the correpsonding z0 and z1
z_1=s(:,1)+s(:,2)+s(:,3);
z_2=s(:,4)+s(:,5)+s(:,6);


%numerical integration of the reduced model

testoptions=odeset(’RelTol’,1e-8,’AbsTol’,1e-8);
n1=na;
t1=(0:100:n1);
y0=[0.4000 0.4001];
[t1,z] = ode15s(@num_int2_z,t1,y0);

%plots to compare the full and reduced model wrt to t
figure(3)
plot(t,z_1,’.r’)
hold on
plot(t1,z(:,1),’-v’)
xlabel(’t’)
ylabel(’z_0’)
legend(’full model’,’reduced model’,’Location’,’SouthEast’)
xlim([0 na])

figure(4)
plot(t,z_2,’.r’)
hold on
plot(t1,z(:,2),’-v’)
xlabel(’t’)
ylabel(’z_1’)
legend(’full model’,’reduced model’)
xlim([0 na])


%==========discretization error================
    error1 = zeros(length(z),1);
    for kk=2:length(z)
        distance=sqrt((z(kk-1,1)-z(kk,1))^2 + (z(kk-1,2)-z(kk,2))^2);

        z1_part=(z(kk,2)-z_2(kk))^2;
        z0_part=(z(kk,1)-z_1(kk))^2 ;
        error1(kk)=(z0_part + z1_part)*distance;
    end
    error(k)=sum(error1);

end

%number of iteration versus mesh size
figure(5)
plot(n,itr_store,'-*')
xlabel('mesh size')
ylabel('number of iterates')

%discretization error plot versus mesh size
figure(6)
plot(n,error,'-*')
xlabel('mesh size')
ylabel('discretization error')
```

Parameter file for the code above.

```
%multistep paramater file

global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
global eta1 eta2 eta3 eta4 gamma2 gamma3 gamma4
global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6

alpha1= 100;  alpha2 = 100; alpha3 = 100; alpha4 = 100; alpha5=100; alpha6=100;
beta1 = 100; beta2 = 100;  beta3 =100; beta4 = 100; beta5=100; beta6=100;
gamma2 = 100; gamma4 =100; gamma5=100;
eta1 = 100; eta2 = 100; eta4 =100; eta5=100;
gamma3 =1e-2; eta3 =1e-2; gamma6=1e-2; eta6=0;
```

Functions for the above program.

```
function result=FE_s0(s0)
```

```
    global ds0dz0 ds0dz1 i j s1 s2 s3 s4 s5
    result=s0dot(s0,s1(i,j))*(1-ds0dz0(i,j))...
    -s1dot(s0,s1(i,j),s2(i,j))*ds0dz0(i,j)...
    -s2dot(s1(i,j),s2(i,j),s3(i,j))*ds0dz0(i,j)...
    -s3dot(s2(i,j),s3(i,j),s4(i,j))*ds0dz1(i,j)...
    -s4dot(s3(i,j),s4(i,j),s5(i,j))*ds0dz1(i,j)...
    -s5dot(s0,s1(i,j),s2(i,j),s3(i,j),s4(i,j),s5(i,j))*ds0dz1(i,j);
end

function result=FE_s1(s1)
    global ds1dz0 ds1dz1 i j s0new s2 s3 s4 s5
    result=-s0dot(s0new(i,j),s1)*ds1dz0(i,j)...
    +s1dot(s0new(i,j),s1,s2(i,j))*(1-ds1dz0(i,j))...
    -s2dot(s1,s2(i,j),s3(i,j))*ds1dz0(i,j)...
    -s3dot(s2(i,j),s3(i,j),s4(i,j))*ds1dz1(i,j)...
    -s4dot(s3(i,j),s4(i,j),s5(i,j))*ds1dz1(i,j)...
    -s5dot(s0new(i,j),s1,s2(i,j),s3(i,j),s4(i,j),s5(i,j))*ds1dz1(i,j);
end

function result=FE_s2(s2)
    global ds2dz0 ds2dz1 i j s0new s1new s3 s4 s5
    result=-s0dot(s0new(i,j),s1new(i,j))*ds2dz0(i,j)...
    -s1dot(s0new(i,j),s1new(i,j),s2)*ds2dz0(i,j)...
    +s2dot(s1new(i,j),s2,s3(i,j))*(1-ds2dz0(i,j))...
    -s3dot(s2,s3(i,j),s4(i,j))*ds2dz1(i,j)...
     -s4dot(s3(i,j),s4(i,j),s5(i,j))*ds2dz1(i,j)...
     -s5dot(s0new(i,j),s1new(i,j),s2,s3(i,j),s4(i,j),s5(i,j))*ds2dz1(i,j);
end

function result=FE_s3(s3)
    global ds3dz0 ds3dz1 i j s0new s1new s2new s4 s5
    result=-s0dot(s0new(i,j),s1new(i,j))*ds3dz0(i,j)...
    -s1dot(s0new(i,j),s1new(i,j),s2new(i,j))*ds3dz0(i,j)...
    -s2dot(s1new(i,j),s2new(i,j),s3)*ds3dz0(i,j)...
    +s3dot(s2new(i,j),s3,s4(i,j))*(1-ds3dz1(i,j))...
    -s4dot(s3,s4(i,j),s5(i,j))*ds3dz1(i,j)...
    -s5dot(s0new(i,j),s1new(i,j),s2new(i,j),s3,s4(i,j),s5(i,j))*ds3dz1(i,j);
end

function result=FE_s4(s4)
    global ds4dz0 ds4dz1 i j s0new s1new s2new s3new s5
    result=-s0dot(s0new(i,j),s1new(i,j))*ds4dz0(i,j)...
    -s1dot(s0new(i,j),s1new(i,j),s2new(i,j))*ds4dz0(i,j)...
```

```
     -s2dot(s1new(i,j),s2new(i,j),s3new(i,j))*ds4dz0(i,j)...
     -s3dot(s2new(i,j),s3new(i,j),s4)*ds4dz1(i,j)...
     +s4dot(s3new(i,j),s4,s5(i,j))*(1-ds4dz1(i,j))...
     -s5dot(s0new(i,j),s1new(i,j),s2new(i,j),s3new(i,j),s4,s5(i,j))*ds4dz1(i,j);
end

function result=FE_s5(s5)
global ds5dz0 ds5dz1 i j s0new s1new s2new s3new s4new
result=-s0dot(s0new(i,j),s1new(i,j))*ds5dz0(i,j)...
-s1dot(s0new(i,j),s1new(i,j),s2new(i,j))*ds5dz0(i,j)...
-s2dot(s1new(i,j),s2new(i,j),s3new(i,j))*ds5dz0(i,j)...
-s3dot(s2new(i,j),s3new(i,j),s4new(i,j))*ds5dz1(i,j)...
-s4dot(s3new(i,j),s4new(i,j),s5)*ds5dz1(i,j)...
+s5dot(s0new(i,j),s1new(i,j),s2new(i,j),s3new(i,j),s4new(i,j),s5)*(1-ds5dz1(i,j));
end

function result=s0dot(s0,s1)
    global alpha1 beta1 eta1
    result=(eta1*s1)/(1+alpha1*s0+beta1*s1)-s0/(1+alpha1*s0+beta1*s1);
end

function result=s1dot(s0,s1,s2)
    global alpha1 beta1 eta1 alpha2 gamma2 eta2 beta2
    result=(s0)/(1+alpha1*s0+beta1*s1)-(eta1*s1)/(1+alpha1*s0+beta1*s1)...
        -(gamma2*s1)/(1+alpha2*s1+beta2*s2)+(eta2*s2)/(1+alpha2*s1+beta2*s2);
end

function result=s2dot(s1,s2,s3)
    global alpha2 gamma2 eta2 beta2 gamma3 alpha3 eta3 beta3 k
    result=(gamma2*s1)/(1+alpha2*s1+beta2*s2)-(eta2*s2)/(1+alpha2*s1+beta2*s2)...
        -(gamma3(k)*s2)/(1+alpha3*s2+beta3*s3)+(eta3*s3)/(1+alpha3*s2+beta3*s3);
end

function result=s3dot(s2,s3,s4)
    global gamma3 alpha3 eta3 beta3 gamma4 alpha4 beta4 eta4 k
    result=(gamma3(k)*s2)/(1+alpha3*s2+beta3*s3)...
    -(eta3*s3)/(1+alpha3*s2+beta3*s3)...
        -(gamma4*s3)/(1+alpha4*s3+beta4*s4)+(eta4*s4)/(1+alpha4*s3+beta4*s4);
end

function result=s4dot(s3,s4,s5)
    global  gamma4 alpha4 beta4 eta4 gamma5 alpha5 eta5 beta5
    result=(gamma4*s3)/(1+alpha4*s3+beta4*s4)-(eta4*s4)/(1+alpha4*s3+beta4*s4)...
```

```
        -(gamma5*s4)/(1+alpha5*s4+beta5*s5)+(eta5*s5)/(1+alpha5*s4+beta5*s5);
end

function result=s5dot(s0,s1,s2,s3,s4,s5)
    global  gamma5 alpha5 eta5 beta5 gamma6 alpha6 beta6 eta6
    s6=1-s0-s1-s2-s3-s4-s5;
    result=(gamma5*s4)/(1+alpha5*s4+beta5*s5)-(eta5*s5)/(1+alpha5*s4+beta5*s5)...
        -(gamma6*s5)/(1+alpha6*s5+beta6*s6)+(eta6*s6)/(1+alpha6*s5+beta6*s6);
end
```

Numerical integration for the full and reduced model.

```
%numerical integration of the full model

function result=num_int_full(t,s)

    global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
    global gamma2 gamma3 gamma4 eta1 eta2 eta3 eta4
    global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6

    s(7)=1-s(1)-s(2)-s(3)-s(4)-s(5)-s(6);

    result=[(eta1*s(2))/(1+alpha1*s(1)+beta1*s(2))...
    -s(1)/(1+alpha1*s(1)+beta1*s(2));
        (s(1))/(1+alpha1*s(1)+beta1*s(2))...
        -(eta1*s(2))/(1+alpha1*s(1)+beta1*s(2))...
        -(gamma2*s(2))/(1+alpha2*s(2)+beta2*s(3))...
        +(eta2*s(3))/(1+alpha2*s(2)+beta2*s(3));
        (gamma2*s(2))/(1+alpha2*s(2)+beta2*s(3))...
        -(eta2*s(3))/(1+alpha2*s(2)+beta2*s(3))...
        -(gamma3*s(3))/(1+alpha3*s(3)+beta3*s(4))...
        +(eta3*s(4))/(1+alpha3*s(3)+beta3*s(4));
        (gamma3*s(3))/(1+alpha3*s(3)+beta3*s(4))...
        -(eta3*s(4))/(1+alpha3*s(3)+beta3*s(4))...
        -(gamma4*s(4))/(1+alpha4*s(4)+beta4*s(5))...
        +(eta4*s(5))/(1+alpha4*s(4)+beta4*s(5));
        (gamma4*s(4))/(1+alpha4*s(4)+beta4*s(5))...
        -(eta4*s(5))/(1+alpha4*s(4)+beta4*s(5))...
        -(gamma5*s(5))/(1+alpha5*s(5)+beta5*s(6))...
        +(eta5*s(6))/(1+alpha5*s(5)+beta5*s(6));
        (gamma5*s(5))/(1+alpha5*s(5)+beta5*s(6))...
        -(eta5*s(6))/(1+alpha5*s(5)+beta5*s(6))...
        -(gamma6*s(6))/(1+alpha6*s(6)+beta6*s(7))...
        +(eta6*s(7))/(1+alpha6*s(6)+beta6*s(7))];
```

```
end

% numerical integration of the reduced model

function result=num_int2_z(t1,z)
    global s0new s1new s2new s3new s4new s5new z0 z1
    global alpha1 alpha2 alpha3 alpha4 beta1 beta2 beta3 beta4
    global gamma2 gamma3 gamma4 eta1 eta2 eta3 eta4
    global alpha5 alpha6 beta5 beta6 eta5 eta6 gamma5 gamma6


    s0_in=interp2(z0,z1,s0new,z(1),z(2));
    s1_in=interp2(z0,z1,s1new,z(1),z(2));
    s2_in=interp2(z0,z1,s2new,z(1),z(2));
    s3_in=interp2(z0,z1,s3new,z(1),z(2));
    s4_in=interp2(z0,z1,s4new,z(1),z(2));
    s5_in=interp2(z0,z1,s5new,z(1),z(2));
    s6_in=1-s0_in-s1_in-s2_in-s3_in-s4_in-s5_in;


    %computes the rhs of the rate equations

    ds0dt=(eta1*s1_in)/(1+alpha1*s0_in+beta1*s1_in)...
    -s0_in/(1+alpha1*s0_in+beta1*s1_in);

    ds1dt=(s0_in)/(1+alpha1*s0_in+beta1*s1_in)...
    -(eta1*s1_in)/(1+alpha1*s0_in+beta1*s1_in)...
        -(gamma2*s1_in)/(1+alpha2*s1_in+beta2*s2_in)...
        +(eta2*s2_in)/(1+alpha2*s1_in+beta2*s2_in);

    ds2dt=(gamma2*s1_in)/(1+alpha2*s1_in+beta2*s2_in)...
    -(eta2*s2_in)/(1+alpha2*s1_in+beta2*s2_in)...
    -(gamma3*s2_in)/(1+alpha3*s2_in+beta3*s3_in)...
    +(eta3*s3_in)/(1+alpha3*s2_in+beta3*s3_in);

    ds3dt=(gamma3*s2_in)/(1+alpha3*s2_in+beta3*s3_in)...
    -(eta3*s3_in)/(1+alpha3*s2_in+beta3*s3_in)...
    -(gamma4*s3_in)/(1+alpha4*s3_in+beta4*s4_in)...
    +(eta4*s4_in)/(1+alpha4*s3_in+beta4*s4_in);

    ds4dt=(gamma4*s3_in)/(1+alpha4*s3_in+beta4*s4_in)...
    -(eta4*s4_in)/(1+alpha4*s3_in+beta4*s4_in)...
    -(gamma5*s4_in)/(1+alpha5*s4_in+beta5*s5_in)...
    +(eta5*s5_in)/(1+alpha5*s4_in+beta5*s5_in);

    ds5dt=(gamma5*s4_in)/(1+alpha5*s4_in+beta5*s5_in)...
    -(eta5*s5_in)/(1+alpha5*s4_in+beta5*s5_in)...
    -(gamma6*s5_in)/(1+alpha6*s5_in+beta6*s6_in)...
    +(eta6*s6_in)/(1+alpha6*s5_in+beta6*s6_in);

    result=[ds0dt+ds1dt+ds2dt;ds3dt+ds4dt+ds5dt];
end
```