MULTIPLE INTERVAL METHODS FOR ODES WITH AN OPTIMIZATION CONSTRAINT

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Xinli Yu
May,2020

ABSTRACT

MULTIPLE INTERVAL METHODS FOR ODES WITH AN OPTIMIZATION CONSTRAINT

Xinli Yu

DOCTOR OF PHILOSOPHY

Temple University, May,2020

Professor Daniel B. Szyld, Chair

We are interested in numerical methods for the optimization constrained second order ordinary differential equations arising in biofilm modelling. This class of problems is challenging for several reasons.

One of the reasons is that the underlying solution has a steep slope, making it difficult to resolve. We propose a new numerical method with techniques such as domain decomposition and asynchronous iterations for solving certain types of ordinary differential equations more efficiently. In fact, for our class of problems after applying the techniques of domain decomposition with overlap we are able to solve the ordinary differential equations with a steep slope on a larger domain than previously possible. After applying asynchronous iteration techniques, we are able to solve the problem with less time. We provide theoretical conditions for the convergence of each of the techniques.

The other reason is that the second order ordinary differential equations are coupled with an optimization problem, which can be viewed as the constraints. We propose a numerical method for solving the coupled problem and show that it converges under certain conditions.

An application of the proposed methods on biofilm modeling is discussed. The numerical method proposed is adopted to solve the biofilm problem, and we are able to solve the problem with larger thickness of the biofilm than possible before as is shown in the numerical experiments.

ACKNOWLEDGEMENTS

First and foremost, I would like to give the deepest and most sincere gratitude possible to my advisor Prof. Daniel B. Szyld. He has always been very patient and enthusiastic and I have found his advice and support invaluable. I feel very fortunate and privileged for being a student of such a great mentor and mathematician. Without his coaching, this research project would have never been completed.

I would like to thank Prof. Isaac Klapper. He has been a constant source of inspiration and guidance. A considerable amount of my knowledge comes from Prof. Klapper.

I also owe my special thanks to my thesis committee members Prof. Gillian Queisser, Prof. Bettina Buttaro, and the Director of Graduate Studies, Prof. David Futer, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I would like to thank Dr. Tianyu Zhang from Center of Biofilm Engineering at Montana State University for helping me with thesis related knowledge. I would also like to thank Dr. Cristal Zúñiga from the University of California San Diego for helping me with data collection and sending me the metabolic network models for performing flux balanced analysis.

I would also like to thank all the Temple math faculty who I have encountered in some way or another. My knowledge of mathematics has grown so much in the past years and I owe that to the professors of the many classes I have taken as a graduate student at Temple.

I would like to thank my friends, Dr. Fayçal Chaouqui, Dr. Francisco Villarroya for being supportive during my studies.

Finally, I would like to thank my parents and my entire family for their love and support.

To my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this thesis, we are interested in numerical methods for the optimization constrained second order ordinary differential equations (ODEs) arising in microbial biofilm modelling [76]. The underlying solution of the ordinary differential equations arising in microbial biofilm modelling has a steep slope, making it difficult to resolve. In this thesis, we propose a new numerical method with techniques such as domain decomposition and asynchronous iterations for solving ordinary differential equations more efficiently. We provide theoretical conditions for the convergence of each of the techniques. In biofilm modelling, the ordinary differential equations are often coupled with an optimization problem, we present an iteration algorithm to solve the coupled problem and show that it converges under certain conditions.

My contribution is to use domain decomposition methods and in particular the introduction of overlap in the discretization for the ODE. With this novel idea we are able to solve bigger multi-scale problem with steeper slope than previously possible. We also provide theoretical developments giving conditions for the convergence of the proposed methods.

To solve the second order ordinary differential equations coupled with an optimization problem, we use an alternating scheme in which we fix the value of the constraints, solve the ODE, then use this solution as data for the optimization problem, giving rise to new constraints and we repeat the process.

Again, we propose conditions under which this process converges.

An application is the kinetic free modeling of the biofilms [76], communities of microbes living and interacting at close quarters in self-secreted polymeric matrices [17]. Using the proposed method, we are able to solve the model with larger thickness of the biofilm than possible before.

1.1 Motivation of the Problem

Microbial biofilms are defined as clusters of microbial cells living in self-produced extracellular polymeric substances (EPS), which are always attached to various kinds of surfaces. Models of microbial community dynamics generally rely on a sub-scale model for microbial metabolisms. In systems such as distributed multispecies communities like biofilms, where it is not reasonable to simplify to a small number of limiting substrates, tracking the large number of active metabolites likely requires measurement or estimation of large numbers of kinetic and regulatory parameters. Alternatively, a largely kinetics-free methodology is proposed in [76] combining cellular level constrained, steady state metabolic flux analysis with macroscale microbial community models. The methodology easily allows coupling of macroscale information, including measurement data, with cell-scale metabolism.

The ODE problem arising in the biofilm modelling is challenging because the slope of the biofilm modeling solution is sometimes steeper than in other applications. We review some possible methods for the boundary value problem in section 1.2.

1.2 Boundary Value Problems

In the field of differential equations, a boundary value problem is a differential equation together with a set of additional constraints, called the boundary conditions [74]. A solution to a boundary value problem is a solution to the

differential equation which also satisfies the boundary conditions. In the next two subsections, we review two methods for solving boundary value problems.

1.2.1 Numerical Approaches: The Shooting Method

The shooting method [4, 16, 32, 38, 39, 57, 62] is a method for solving a boundary value problem by reducing it to the solution of an initial value problem. Roughly speaking, we "shoot" out trajectories in different directions until we find a trajectory that has the desired boundary value. This will be clarified by the following illustration of the shooting method.

We will first introduce the shooting method for the boundary value problem as follows, then we will introduce the multiple shooting method.

Let $x(z;C)$ denote the solution of the initial value problem

$$\begin{cases} \ddot{x}(z) = f(x(z)) \\ x(a) = A \\ \dot{x}(a) = C \end{cases} \tag{1.1}$$

Define the function $F(C)$ as the difference between $x(b;C)$ and the specified boundary value B with $F(C) = x(b;C) - B$.

If F has a root C^* then the solution $x(z;C^*)$ of the corresponding initial value problem is also a solution of the boundary value problem. Conversely, if the boundary value problem has a solution $x(z)$, then $x(z)$ is also the unique solution $x(z;C^*)$ of the initial value problem where $C^* = \dot{x}(a)$, thus C^* is a root of F.

This root can be found by any root-finding method given that certain method-dependent prerequisites are satisfied. This often will require an initial value for C. Typically, finding the root analytically is impossible and iterative methods such as Newton's method are used for this task.

The application of the shooting method for the numerical solution of boundary value problems suffers from several drawbacks.

For a given initial value C the solution of the initial value problem (1.1) obviously must exist on the interval $[a, b]$ so that we can evaluate the function

F whose root is sought.

For highly nonlinear or unstable ODEs, this requires the initial value of C to be extremely close to an actual but unknown solution. Initial values that are chosen slightly off the true solution may lead to singularities or breakdown of the ODE solver method. Choosing such solutions is inevitable in an iterative root-finding method, however. Even the computation of the initial value to full machine accuracy does not guarantee that C can be determined accurately [68].

Finite precision numerics may make it impossible to find initial values that allow for the solution of the ODE on the whole time interval. The nonlinearity of the ODE effectively becomes a nonlinearity of F, and requires a root-finding technique capable of solving highly nonlinear systems. Such methods typically converge slower as nonlinearities become more severe. The boundary value problem solver's performance suffers from this. Even stable and well-conditioned ODEs may make for unstable and ill-conditioned BVPs. A slight alteration of the initial value of C may generate an extremely large step in the ODEs solution and thus in the values of the function F whose root is sought. Non-analytic root-finding methods can seldom cope with this behavior [35].

This might be improved by the multiple shooting method [15, 16, 44]. The multiple shooting method partitions the interval (a, b) by introducing additional grid points

$$z_a = z_0 < z_1 < \cdots < z_N = z_b$$

The method starts by providing initial values of y at all grid points z_k with $1 \leq k \leq N$.

The influence of inaccurate initial data can be made arbitrary small by a reduction of the interval size. The dillemma is that one does not know the initial values of y at points $z_k, 1 \leq k \leq N$.

Denote these values by C_k. Let $y^{(k)}(z)$ denote the solution emanating from the k-th grid point, that is, the solution of the initial value problem

$$\begin{cases} \dot{y}^{(1)}(z) = f(y^{(1)}(z)), y^{(1)}(z_1) = A \\ \dot{y}^{(2)}(z) = f(y^{(2)}(z)), y^{(2)}(z_2) = C_1 \\ \vdots \\ \dot{y}^{(N)}(z) = f(y^{(N)}(z)), y^{(N)}(z_N) = C_{N-1} \\ \dot{y}^{(N+1)}(z) = f(y^{(N+1)}(z)), y^{(N+1)}(z_N + 1) = C_N. \end{cases}$$

All these solutions can be pieced together to form a continuous trajectory if the values y match at the grid points. Thus, solutions of the boundary value problem correspond to solutions of the following system of N equations:

$$\begin{cases} \dot{y}^{(1)}(z_1; a, A) = C_1 \\ \vdots \\ \dot{y}^{(N-1)}(z_{N-1}; z_{N-2}, C_{N-2}) = C_{N-1} \\ \dot{y}^{(N)}(z_N; z_{N-1}, C_{N-1}) = C_N \\ \dot{y}^{(N+1)}(b; z_N, C_N) = B. \end{cases}$$

The central $N - 2$ equations are the matching conditions, and the first and last equations are the conditions $y(z_0) = A$ and $y(z_{N+1}) = B$ from the boundary value problem. The multiple shooting method solves the boundary value problem by solving this system of equations. Typically, a modification of Newton's method is used for the latter task.

However, some of the same drawbacks for single shooting method we discussed above still exists and it is even more expensive than the single shooting method.

1.2.2 Numerical Approaches: The Finite Difference Method

In numerical analysis, finite-difference methods (FDMs) are discretizations used to solve differential equations by approximating them with difference equations that approximate the derivatives.

FDMs convert a linear ordinary differential equations (ODEs) or nonlinear partial differential equations (PDEs) into a system of equations that can be

solved by matrix algebra techniques. The reduction of the differential equation to a system of algebraic equations makes the problem of finding the solution to a given ODE/PDE ideally suited to modern computers [31].

The error in a method's solution is defined as the difference between the approximation and the exact analytical solution. The two sources of error in finite difference methods are round-off error, the loss of precision due to computer rounding of decimal quantities, and truncation error or discretization error, the difference between the exact solution of the original differential equation and the exact quantity assuming exact arithmetic (that is, assuming no round-off).

The finite difference method relies on discretizing a function on a grid. To use a finite difference method to approximate the solution to a problem, one must first discretize the problem's domain. This is usually done by dividing the domain into a uniform grid. This means that finite-difference methods produce sets of discrete numerical approximations to the solution of the ODE and its derivative, often in a "time-stepping" manner.

An expression of general interest is the local truncation error of a method. Typically expressed using big-O notation, local truncation error refers to the error from a single application of a method. That is, it is the quantity $f'(z_i) - f'_i$ if $f'(z_i)$ refers to the exact value and f'_i to the numerical approximation. The remainder term of a Taylor polynomial is convenient for analyzing the local truncation error. Using the Lagrange form of the remainder from the Taylor polynomial for $f(z_0 + h)$, which is

$$R_n(z_0 + h) = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1},$$

where $z_0 < \xi < z_0 + h$,

The dominant term of the local truncation error can be calculated, for example, again using the forward-difference formula for the first derivative, knowing that

$$f(z_i) = f(z_0 + ih),$$

$$f(z_0 + ih) = f(z_0) + f'(z_0)ih + \frac{f''(\xi)}{2!}(ih)^2,$$

and with some algebraic manipulation, this leads to

$$\frac{f(z_0 + ih) - f(z_0)}{ih} = f'(z_0) + \frac{f''(\xi)}{2!}ih,$$

and further noting that the quantity on the left is the approximation from the finite difference method and that the quantity on the right is the exact quantity of interest plus a remainder, clearly that remainder is the local truncation error. A final expression of this example and its order is:

$$\frac{f(z_0 + ih) - f(z_0)}{ih} = f'(z_0) + O(h).$$

This means that, in this case, the local truncation error is proportional to the step size. The quality and time complexity of simulated FDM solutions depend on the discretization equation selection and the step sizes (time and space steps). The data quality and time complexity increase significantly with smaller step size [36]. Therefore, a reasonable balance between data quality and simulation duration is necessary for practical usage. Large time steps are useful for increasing simulation speed in practice. However, time steps which are too large may create instabilities and affect the data quality [34, 60].

The von Neumann and Courant-Friedrichs-Lewy criteria are often evaluated to determine the numerical method stability [34, 60].

1.3 Boundary Value Problem Coupled with Optimization Problem

The modelling of biofilm often consists coupled system of differential equations and optimization. A detailed biofilm system example is described in Chapter 5, and the biofilm system model described in Chapter 5 is a system

of ODEs coupled with an optimization problem. Next we give a brief review about optimization.

Mathematical optimization or mathematical programming is the selection of a best element (with regard to some criterion) from some set of available alternatives. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries [5, 26, 45].

Optimization theory provides algorithms to solve well-structured optimization problems along with the analysis of those algorithms. This analysis includes necessary and sufficient conditions for the existence of optimal solutions. Optimization problems are expressed in terms of variables (degrees of freedom) and the domain; objective function to be optimized; and, possibly, constraints. The generalization of optimization theory and techniques to other formulations constitutes a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain (or input), including a variety of different types of objective functions and different types of domains.

1.3.1 Nonlinear Programming

In particular, we shall consider the nonlinear programming. Nonlinear programming (NLP) is the process of solving an optimization problem where some of the constraints or the objective function are nonlinear. An optimization problem is one of calculation of the extrema (maxima, minima or stationary points) of an objective function over a set of unknown real variables and conditional to the satisfaction of a system of equalities and inequalities, collectively termed constraints. It is the sub-field of mathematical optimization that deals with problems that are not linear.

Let n, m, and p be positive integers. Let X be a subset of $\mathbb{R}^n$, let f, g_i, and h_j be real-valued functions on X for each i in $\{1, \ldots, m\}$ and each j in

$\{1, \ldots, p\}$, with at least one of f, g_i, and h_j being nonlinear.

A nonlinear maximization problem is an optimization problem of the form

$$\begin{aligned} & \max f(z) \\ \text{subject to } & g_i(z) \leq 0 \text{ for each } i \in \{1, \ldots, m\} \\ & h_j(z) = 0 \text{ for each } j \in \{1, \ldots, p\} \\ & z \in X. \end{aligned}$$

A nonlinear minimization problem is defined in a similar way. There are several possibilities for the nature of the constraint set, also known as the feasible set or feasible region.

An infeasible problem is one for which no set of values for the choice variables satisfies all the constraints. That is, the constraints are mutually contradictory, and no solution exists; the feasible set is the empty set.

A feasible problem is one for which there exists at least one set of values for the choice variables satisfying all the constraints.

An unbounded problem is a feasible problem for which the objective function can be made to be better than any given finite value. Thus there is no optimal solution, because there is always a feasible solution that gives a better objective function value than does any given proposed solution.

If the objective function f is linear and the constrained space is a polytope, the problem is a linear programming problem, which may be solved using well-known linear programming techniques such as the simplex method [5, 26, 45].

If the objective function is concave (maximization problem), or convex (minimization problem) and the constraint set is convex, then the program is called convex and general methods from convex optimization can be used in most cases [5, 26, 45].

If the objective function is quadratic and the constraints are linear, quadratic programming techniques are used.

If the objective function is a ratio of a concave and a convex function (in the maximization case) and the constraints are convex, then the problem can be

transformed to a convex optimization problem using fractional programming techniques.

Several methods are available for solving nonconvex problems. One approach is to use special formulations of linear programming problems. Another method involves the use of branch and bound techniques, where the program is divided into subclasses to be solved with convex (minimization problem) or linear approximations that form a lower bound on the overall cost. With subsequent divisions, at some point an actual solution will be obtained whose cost is equal to the best lower bound obtained for any of the approximate solutions. This solution is optimal, although possibly not unique. The algorithm may also be stopped early, with the assurance that the best possible solution is within a tolerance from the best point found; such points are called ϵ-optimal. Terminating to ϵ-optimal points is typically necessary to ensure finite termination. This is especially useful for large, difficult problems and problems with uncertain costs or values where the uncertainty can be estimated with an appropriate reliability estimation [5, 45].

Under differentiability and constraint qualifications, the Karush–Kuhn–Tucker (KKT) conditions provide necessary conditions for a solution to be optimal. Under convexity, these conditions are also sufficient. If some of the functions are non-differentiable, subdifferential versions of KKT conditions are available [45, 56].

In Chapter 2 we propose a method for the solution of boundary value problems including a Green's function approach, and in Chapter 3 we apply the domain decomposition technique on the numerical method proposed in Chapter 2. In Chapter 4 we consider asynchronous methods and apply the asynchronous technique to the numerical method proposed in Chapter 3. In Chapter 5 we solve the optimization constrained boundary value problem arising in biofilm modelling and propose a numerical algorithm to decouple the ODE and the optimization and show that the algorithm converges with theoretical guarantee.

CHAPTER 2

A METHOD FOR THE BOUNDARY VALUE PROBLEM

In this chapter, we discuss solving the second order boundary value problem in $\mathbb{R}^n$. We show that we can use solutions of an homogeneous problem to build the solution of the inhomogeneous problem.

2.1 Problem Formulation

We consider the solution of second order boundary value problem, i.e.,

$$\ddot{\mathbf{x}}(z) = \mathbf{f}(\mathbf{x}(z)),$$

where $\mathbf{x} : \mathbb{R} \to \mathbb{R}^n$ and

$$\mathbf{x}(z) = (x_1(z), x_2(z), ..., x_n(z))^T$$

$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, i.e.,

$$\mathbf{f}(\mathbf{x}(z)) = (f_1(x_1(z), \ldots, x_n(z)), f_2(x_1(z), \ldots, x_n(z)), \ldots, f_n(x_1(z), \ldots, x_n(z)))^T.$$

The boundary conditions are

$$x_i(a) = A_i,\ x_i(b) = B_i,\ i = 1, 2, ..., n.$$

Note that the explicit form of f may be unknown, but we can evaluate f at any point. Thus we have

$$\begin{cases} \ddot{x}_1(z) = f_1(x_1(z), x_2(z), \ldots, x_n(z)) \\ \ddot{x}_2(z) = f_2(x_1(z), x_2(z), \ldots, x_n(z)) \\ \qquad\qquad \vdots \\ \ddot{x}_n(z) = f_n(x_1(z), x_2(z), \ldots, x_n(z)) \end{cases} \tag{2.1}$$

with boundary conditions $x_i(a) = A_i, x_i(b) = B_i, i = 1, \ldots, n$. Denote by

$$\mathbf{A} = (A_1, \ldots, A_n)^T$$

$$\mathbf{B} = (B_1, \ldots, B_n)^T.$$

The boundary value problem of a system of ordinary differential equations can be expressed as

$$\begin{cases} \ddot{\mathbf{x}}(z) = \mathbf{f}(\mathbf{x}(z)) \\ \mathbf{x}(a) = \mathbf{A} \\ \mathbf{x}(b) = \mathbf{B}. \end{cases} \tag{2.2}$$

2.2 Equivalent Form of an Homogeneous Problem

For the boundary value problem

$$\begin{cases} \ddot{\mathbf{y}}(z) = \mathbf{f}(\mathbf{y}(z)) \\ \mathbf{y}(a) = \mathbf{0}, \mathbf{y}(b) = \mathbf{0}, \end{cases} \tag{2.3}$$

we can consider integral equation formulation (IEF) obtained from Green's function approaches [8, 11, 42, 21]. For example, Green's function for isothermal linear reaction in a sphere is obtained and the resulting Fredholm integral equation is solved via a successive approximation technique [58]. The time-marching process for nonlinear dynamic analysis is carried out through an algorithm based on the Green's function of the mechanical system in nodal coordinates [61].

The solution for (2.3) can be expressed equivalently as the Integral equation form by

$$\mathbf{y}(z) = \int_a^b G(z,s)\mathbf{f}(\mathbf{y}(s))ds,$$

where the Green's function $G(z,s)$ for this system is given by (see, e.g., [10])

$$G(z,s) = \begin{cases} \frac{(z-a)(s-b)}{b-a} & a \le z \le s \le b \\ \frac{(z-b)(s-a)}{b-a} & a \le s \le z \le b. \end{cases} \tag{2.4}$$

However, homogeneous boundary conditions are rarely encountered in practice. The idea is to use a shift function $\mathbf{h}$ to convert the boundary condition

$$x_i(a) = A_i,\ x_i(b) = B_i,\ i = 1, \ldots, n$$

to homogeneous boundary conditions

$$y_i(a) = 0,\ y_i(b) = 0,\ i = 1, \ldots, n.$$

Consider the linear mapping

$$\mathbf{h}_{\mathbf{A},\mathbf{B}}(z) = \frac{a\mathbf{B} - b\mathbf{A} + (\mathbf{A}-\mathbf{B})z}{a-b}. \tag{2.5}$$

The mapping satisfies

$$\mathbf{h}_{\mathbf{A},\mathbf{B}}(a) = \mathbf{A},\ \mathbf{h}_{\mathbf{A},\mathbf{B}}(b) = \mathbf{B}$$

and

$$\ddot{\mathbf{h}}_{\mathbf{A},\mathbf{B}}(z) = 0.$$

If $\mathbf{x}(z)$ is the solution of (2.2), with the change of variables

$$\mathbf{y}(z) = \mathbf{x}(z) - \mathbf{h}_{\mathbf{A},\mathbf{B}}(z),$$

we can write (2.2) in terms of $\mathbf{y}(z)$ as

$$\begin{cases} \ddot{\mathbf{y}}(z) + \ddot{\mathbf{h}}_{\mathbf{A},\mathbf{B}}(z) = \mathbf{f}(\mathbf{y}(z) + \mathbf{h}_{\mathbf{A},\mathbf{B}}(z)) \\ \mathbf{y}(a) + \mathbf{h}_{\mathbf{A},\mathbf{B}}(a) = \mathbf{A} \\ \mathbf{y}(b) + \mathbf{h}_{\mathbf{A},\mathbf{B}}(b) = \mathbf{B} \end{cases} \tag{2.6}$$

Simplifying with the properties of $\mathbf{h_{A,B}}$, the equation becomes

$$\begin{cases} \ddot{\mathbf{y}}(z) = \mathbf{f}(\mathbf{y}(z) + \mathbf{h_{A,B}}(z)) \\ \mathbf{y}(a) = \mathbf{0}, \mathbf{y}(b) = \mathbf{0}. \end{cases} \tag{2.7}$$

Let

$$\mathbf{F}(\mathbf{y}(z)) = \mathbf{f}((\mathbf{y} + \mathbf{h_{A,B}})(z)). \tag{2.8}$$

Thus we can convert the non-homogeneous boundary conditions problem into a new problem of homogeneous boundary conditions and in this chapter, we will only consider homogeneous boundary conditions.

$$\begin{cases} \ddot{\mathbf{y}}(z) = \mathbf{F}(\mathbf{y}(z)) \\ \mathbf{y}(a) = \mathbf{0}, \mathbf{x}(b) = \mathbf{0} \end{cases} \tag{2.9}$$

and the expression for the solution $\mathbf{y}$ is

$$\mathbf{y}(z) = \int_a^b G(z,s)\mathbf{F}(\mathbf{y}(s))ds. \tag{2.10}$$

2.3 Fixed Point Iteration and Proof of Convergence

Consider the expression (2.10) for the solution of the homogeneous ODE (2.9), we can define a fixed point iteration

$$\mathbf{y}^{(k+1)}(z) = \int_a^b G(z,s)\mathbf{F}(\mathbf{y}^{(k)}(s))ds, \tag{2.11}$$

where $\mathbf{y}^{(k)}$ is the $k-th$ iteration and $\mathbf{F}$ is defined in (2.8).

Define the operator T as

$$\mathbf{y}^{(k+1)} = T(\mathbf{y}^{(k)}) = \int_a^b G(z,s)\mathbf{F}(\mathbf{y}^{(k)}(s))ds\cdot$$

Theorem 2.1 *Let $K = \max_i K_i$ when K_i is the constant such that the following inequality holds,*

$$(F_i(y_1(s), ..., y_n(s)) - F_i((Ty)_1(s), ..., (Ty)_n(s))) \leq K_i \max_i |y_i(s) - (Ty)_i(s)|,$$

then the fixed point iteration (2.11) *is convergent for any initial function* $\mathbf{y}^{(0)}(z)$ *and converges to a unique solution iff*

$$\frac{K(b-a)^2}{8} < 1 \ .$$

Proof. To show convergence of the fixed point iteration, we consider a general vector $\mathbf{y} = (y_1, ..., y_n)$ and its image under T, $T\mathbf{y} = ((T\mathbf{y})_1, ..., (T\mathbf{y})_n)$.

Since K_i is the constant for F_i such that the following inequality holds,

$$(F_i(y_1(s), ..., y_n(s)) - F_i((T\mathbf{y})_1(s), ..., (T\mathbf{y})_n(s))) \leq K_i \max_i |y_i(s) - (T\mathbf{y})_i(s)| \, .$$

Then the following inequality holds,

$$\begin{aligned}
&|(T\mathbf{y})_i(z) - (T \circ T\mathbf{y})_i(z)| \\
&= \left| \int_a^b G(z,s) \left(F_i(y_1(s), ..., y_n(s)) - F_i((Ty)_1(s), ..., (Ty)_n(s))\right) ds \right| \\
&\leq \int_a^b |G(z,s)| \, K_i \max_i |y_i(s) - (Ty)_i(s)| \, ds \\
&\leq K \|\mathbf{y} - T\mathbf{y}\| \int_a^b |G(z,s)| \, ds,
\end{aligned}$$

where $K = \max_i K_i$ and $\|\mathbf{y}\| = \max_i [\max_{a \leq t \leq b} |y_i(t)|]$. Furthermore,

$$\begin{aligned}
\int_a^b |G(z,s)| ds &= \left| \frac{z-a}{b-a} \int_z^a (s-b) ds \right| + \left| \frac{z-b}{b-a} \int_a^z (s-a) ds \right| \\
&= \frac{(z-a)(z-b)^2 + (b-z)(z-a)^2}{2(b-a)} \\
&= \frac{(z-a)(z-b)[(z-b) - (z-a)]}{2(b-a)} \\
&= \frac{(z-a)(b-z)}{2} \leq \frac{(b-a)^2}{8}.
\end{aligned} \tag{2.12}$$

Thus,

$$|(T\mathbf{y})_i(z) - (T \circ T\mathbf{y})_i(z)| \leq \frac{K(b-a)^2}{8} \|\mathbf{y} - T\mathbf{y}\|$$

Taking maximum of i and $a \leq z \leq b$ on both sides, we obtain

$$\|T\mathbf{y} - T \circ T\mathbf{y}\| \leq \frac{K(b-a)^2}{8}\|\mathbf{y} - T\mathbf{y}\|\cdot$$

So the iteration converges under the norm

$$\|\mathbf{y}\| = \max_i \left[\max_{a\leq t\leq b} |y_i(z)|\right]$$

if

$$\frac{K(b-a)^2}{8} < 1.$$

Next, we show that it converges to a unique solution. Suppose we have two different fixed point iteration solution $\mathbf{x}$ and $\mathbf{y}$ such that $T\mathbf{x} = \mathbf{x}$ and $T\mathbf{y} = \mathbf{y}$ so we have

$$\|T\mathbf{x} - T\mathbf{y}\| = \|\mathbf{x} - \mathbf{y}\|. \tag{2.13}$$

We have

$$\begin{aligned}
&|(T\mathbf{x})_i(z) - (T\mathbf{y})_i(z)| \\
&= \left|\int_a^b G(z,s)\left(f_i(x_1(s),...,x_n(s)) - f_i(y_1(s),...,y_n(s))\right) ds\right| \\
&\leq \int_a^b |G(z,s)|\, K_i \max_i |x_i(s) - y_i(s)|\, ds \\
&\leq K\|\mathbf{x} - \mathbf{y}\| \int_a^b |G(z,s)|\, ds\cdot
\end{aligned}$$

Thus by (2.12),

$$|(T\mathbf{x})_i(z) - (T\mathbf{y})_i(z)| \leq \frac{K(b-a)^2}{8}\|\mathbf{x} - \mathbf{y}\|\cdot$$

Taking maximum of i and $a \leq t \leq b$ on both sides, we get

$$\|T\mathbf{x} - T\mathbf{y}\| \leq \frac{K(b-a)^2}{8}\|\mathbf{x} - \mathbf{y}\|\cdot \tag{2.14}$$

From (2.13) and (2.14), we have

$$\|\mathbf{x} - \mathbf{y}\| \leq \frac{K(b-a)^2}{8}\|\mathbf{x} - \mathbf{y}\|$$

From the hypothesis, we have

$$\frac{K(b-a)^2}{8} < 1,$$

so this gives

$$\|\mathbf{x}-\mathbf{y}\| \leq \frac{K(b-a)^2}{8}\|\mathbf{x}-\mathbf{y}\| < \|\mathbf{x}-\mathbf{y}\|,$$

which is a contradiction. Thus the fixed point iteration is unique. For the converse, assuming that

$$\frac{K(b-a)^2}{8} \geq 1,$$

one can easily find $\mathbf{y}^{(0)}$ so that the method will not converge. ■

2.4 Implementation Details

The Green's function formulation is a suitable framework to derive numerical schemes incorporating both accuracy and non-local information in the whole domain to solve nonlinear differential equations [1, 33]. In this section, we will discuss some implementation details.

As we have shown in (2.11), the fixed point iteration to evaluate is

$$\mathbf{y}^{(k+1)}(z) = \int_a^b G(z,s)\mathbf{F}(\mathbf{y}^{(k)}(s))ds. \tag{2.15}$$

First discretize the interval (a,b) into $N+1$ sub-intervals with step-size

$$\frac{b-a}{N+1}\,.$$

Denote the discretization points by

$$\{a, z_1, z_2, ..., z_N, b\}$$

and the solution at the discretization points $z_1, z_2, ..., z_N$ results in a matrix of size $n \times N$ and denote this matrix by Y, where N is the number of discretization points and n is the number of equations; see (2.1).

Thus $Y^{(k)}$ is the solution values for $\mathbf{y}(z)$ at discretization points $z_1, .., z_N$ at iteration k. The matrix $Y^{(k)}$ is of size $n \times N$, and the i-th column denotes the value of $\mathbf{y}^{(k)}(z)$ at the discretization points z_i.

$$
\begin{aligned}
Y^{(k)} &= \left(\ \mathbf{y}^{(k)}(z_1), \ldots, \mathbf{y}^{(k)}(z_N) \ \right)_{n \times N} \\
&= \begin{pmatrix} y_1^{(k)}(z_1) & y_1^{(k)}(z_2) & \cdots & y_1^{(k)}(z_N) \\ y_2^{(k)}(z_1) & y_2^{(k)}(z_2) & \cdots & y_2^{(k)}(z_N) \\ \vdots & \vdots & \ddots & \vdots \\ y_n^{(k)}(z_1) & y_n^{(k)}(z_2) & \cdots & y_n^{(k)}(z_N) \end{pmatrix}_{n \times N}
\end{aligned}
$$

We should then use the recurrence (2.15) to get $Y^{(k+1)}$ which is the discretization of $\mathbf{y}^{(k+1)}(z)$ at points $\{z_1, z_2, ..., z_N\}$, i.e.,

$$
\begin{aligned}
Y^{(k+1)} &= \left(\ \mathbf{y}^{(k+1)}(z_1), \ldots, \mathbf{y}^{(k+1)}(z_N) \ \right)_{n \times N} \\
&= \begin{pmatrix} y_1^{(k+1)}(z_1) & y_1^{(k+1)}(z_2) & \cdots & y_1^{(k+1)}(z_N) \\ y_2^{(k+1)}(z_1) & y_2^{(k+1)}(z_2) & \cdots & y_2^{(k+1)}(z_N) \\ \vdots & \vdots & \ddots & \vdots \\ y_n^{(k+1)}(z_1) & y_n^{(k)}(z_2) & \cdots & y_n^{(k+1)}(z_N) \end{pmatrix}_{n \times N} .
\end{aligned}
$$

We can obtain the i-th column of the $Y^{(k+1)}$ by the letting the $z = z_i$ in formula (2.15). This result in

$$
Y^{(k+1)}(:, i) = \mathbf{y}^{(k+1)}(z_i) = \int_a^b G(z_i, s)\mathbf{F}(\mathbf{y}^{(k)}(s))ds. \tag{2.16}
$$

To approximate the integration, many different quadrature rules can be used, for example, midpoint rule, trapezoidal rule, Simpson's rule or other Newton-Cotes rules etc.

Table 2.1: Newton-Cotes Formulas

f corresponds to $G(z_i, s)\mathbf{F}(\mathbf{y}^{(k)}(s))$ above		
Common Name	Formula	Error Term $\mathbf{e}$
Trapezoidal Rule	$\frac{h}{2}(f_0 + f_1)$	$-\frac{1}{12}h^3 f^{(2)}(\xi)$
Simpson's Rule	$\frac{h}{3}(f_0 + 4f_1 + f_2)$	$-\frac{1}{90}h^5 f^{(4)}(\xi)$
Simpson's 3/8 Rule	$\frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3)$	$-\frac{3}{80}h^5 f^{(4)}(\xi)$
Boole's Rule	$\frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4)$	$-\frac{8}{945}h^7 f^{(6)}(\xi)$

The only values we know from the last iteration k of $(\mathbf{y}^{(k)}(s))$ are the values at the discretization points, which are in the matrix $Y^{(k)}$.

Thus we also use these point values $\{a = z_0, z_1, z_2, ..., z_N, b = z_{N+1}\}$ for the numerical integration.

$$\begin{aligned}&\int_a^b G(z_i, s)\mathbf{F}(\mathbf{y}^{(k)}(s))ds\\ =&h\left(\sum_{j=0}^{N} w_j G(z_i, z_j)\mathbf{F}(\mathbf{y}^{(k)}(z_j))\right) + \mathbf{e},\end{aligned} \tag{2.17}$$

where $h = \frac{b-a}{N+1}$, w_j are weights from the Newton-Cotes formula and $\mathbf{e}$ is the error term for the chosen Newton-Cotes formula. Some examples of closed Newton-Cotes Formulas are given in the Table 2.1.

Since for any i, by definition of $G(z, s)$

$$G(z_i, a) = \frac{(z_i - b)(a - a)}{b - a} = 0$$

$$G(z_i, b) = \frac{(z_i - a)(b - b)}{b - a} = 0,$$

thus in (2.17), when $j = 0$ and $j = N$ we have the summand is 0. When implemented, the following formula start from $j = 1$ to $j = N - 1$ can be used

$$Y^{(k+1)}(:, i) = h\sum_{j=1}^{N-1} w_j G(z_i, z_j)\mathbf{F}(\mathbf{y}^{(k)}(z_j). \tag{2.18}$$

We can write out the form of the matrix $\mathbf{G}$ explicitly shown as in the following steps, and we write each matrix element in the matrix $\mathbf{G}$ explicitly.

For $i \leq j$, we have

$$\begin{aligned} G(z_i, z_j) = G(z_j, z_i) &= \frac{(z_i - a)(z_j - b)}{b - a} \\ &= \frac{(a + ih - a)(a + jh - b)}{b - a} = \frac{ih(a - b + jh)}{b - a} \\ &= -\frac{ih(N + 1 - j)h}{(N + 1)h} = -\frac{i(N + 1 - j)h}{N + 1}. \end{aligned}$$

The formula for (i, j)-th element of the matrix $\mathbf{G}$ is then

$$\mathbf{G} = \begin{cases} -\frac{i(N+1-j)h}{N+1} & \text{when } i \leq j \\ -\frac{j(N+1-i)h}{N+1} & \text{when } i > j \end{cases},$$

$$\mathbf{G} = \frac{h}{N+1} \begin{pmatrix} -N & -N+1 & \dots & -2 & -1 \\ -N+1 & -2N+2 & \ddots & -4 & -2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -2 & -4 & \ddots & -2N+2 & -N+1 \\ -1 & -2 & \dots & -N+1 & -N \end{pmatrix}.$$

Denote by $\mathbf{W}$ the diagonal matrix with $w_j h$ on the diagonal. We have

$$Y^{(k+1)} = \mathbf{GW} \cdot \mathbf{F}(Y^{(k)}). \tag{2.19}$$

For non-homogeneous boundary conditions, we can do a similar implementation as follows

$$X^{(k+1)} = \mathbf{GW} \cdot \mathbf{f}(X^{(k)}) + \mathbf{H}_{\mathbf{A},\mathbf{B}}, \tag{2.20}$$

where the i-th row of the matrix $\mathbf{H}_{\mathbf{A},\mathbf{B}}$ is given by

$$\begin{aligned} \mathbf{H}_{\mathbf{A},\mathbf{B}}(i,:) &= \frac{a\mathbf{B} - b\mathbf{A} + (\mathbf{A} - \mathbf{B})z_i}{a - b}(1, 1, \dots, 1) \\ &= \left[\frac{a - z_i}{a - b}\mathbf{B} + \frac{z_i - b}{a - b}\mathbf{A}\right](1, 1, \dots, 1). \end{aligned}$$

2.5 Convergence of the Implementation

In this section, we discuss convergence of the numerical method we implemented.

From [7, 48] we obtain the following Lemma 2.1,

Lemma 2.1 *The eigenvalues of* $\mathbf{G}$ *are the reciprocal of eigenvalues of a tridiagonal matrix* $\mathbf{T}$ *which is given by*

$$\mathbf{T} = \begin{pmatrix} -\frac{2}{h} & \frac{1}{h} & & \\ \frac{1}{h} & -\frac{2}{h} & \ddots & \\ & \ddots & \ddots & \frac{1}{h} \\ & & \frac{1}{h} & -\frac{2}{h} \end{pmatrix}_{N\times N},$$

where h *is the step size with*

$$h = \frac{b-a}{N+1}.$$

Proof: For any

$$i = 2, ..., N-1,$$

the (i,i)-th element of $G^{(s)}$ is

$$\frac{-i(N+1-i)h}{N+1},$$

the $(i,i+1)$-th element of $G^{(s)}$ is

$$\frac{-i(N+1-i-1)h}{N+1}$$

and the $(i,i-1)$-th element of $G^{(s)}$ is

$$\frac{-(i-1)(N+1-i)h}{N+1},$$

and we can check that

$$(-2)\frac{-i(N+1-i)h}{N+1} + \frac{-i(N+1-i-1)h}{N+1} + \frac{-(i-1)(N+1-i)h}{N+1} = h.$$

For $i = 1$, we only need to consider the (i,i)-th and $(i,i+1)$-th element of $G^{(s)}$, we have

$$(-2)\frac{(-N)h}{N+1} + \frac{(-N+1)h}{N+1} = h.$$

For $i = N$, we only need to consider the (i, i)-th and $(i, i-1)$-th element of $G^{(s)}$, we have

$$\frac{-(-1+N)h}{N+1} - 2\frac{N(-1)h}{N+1} = h.$$

For

$$j = 2, ..., N-2$$

and $i < j$, the (i, j)-th element is

$$\frac{-i(N+1-j)h}{N+1},$$

the $(i, j+1)$-th element is

$$\frac{-i(N+1-j-1)h}{N+1},$$

and the $(i, j+2)$-th element is

$$\frac{-i(N+1-j-2)h}{N+1}.$$

So we have

$$\frac{-i(N+1-j)h}{N+1} - 2\frac{-i(N+1-j-1)h}{N+1} + \frac{-i(N+1-j-2)h}{N+1} = 0$$

For any $i < j$ with $j = N-1$, we only need to consider (i, j)-th and $(i, j+1)$-th element, which are

$$-\frac{i(N+1-(N+1))h}{N+1}$$

and

$$-\frac{i(N+1-N)h}{N+1}$$

respectively, and we have

$$-\frac{i(N+1-(N+1))h}{N+1} + 2\frac{i(N+1-N)h}{N+1} = 0\cdot$$

For $j = 3, ..., N-1$ and $i > j$, the (i, j)-th element is

$$\frac{-j(N+1-i)h}{N+1},$$

the $(i, j-1)$-th element is

$$\frac{-(j-1)(N+1-i)h}{N+1}$$

and the $(i, j-2)$-th element is

$$\frac{-(j-2)(N+1-i)h}{N+1}.$$

So we have

$$\frac{-i(N+1-j)h}{N+1} - 2\frac{-i(N+1-j-1)h}{N+1} + \frac{-i(N+1-j-2)h}{N+1} = 0.$$

For any $i > j$ with $j = 2$, we only need to consider (i, j)-th and $(i, j-1)$-th element, which are

$$\frac{-2(N+1-i)h}{N+1}$$

and

$$-\frac{(N+1-i)h}{N+1}$$

respectively, and we have

$$2\frac{(N+1-i)h}{N+1} + \frac{-2(N+1-i)h}{N+1} = 0.$$

So we proved the eigenvalues of $\mathbf{G}$ is the reciprocal of eigenvalues of a tridiagonal matrix $\mathbf{T}$. ■

Corollary 2.1 *From Lemma 2.1 we can conclude* $\mathbf{GT} = \mathbf{I}$, *i.e.,* $\mathbf{T}^{-1} = \mathbf{G}$.

Theorem 2.2 *The eigenvalues of* $\mathbf{G}$ *are*

$$\frac{h}{2}\frac{1}{(\cos(\frac{k\pi}{N+1}) - 1)}, k = 1, \ldots, J,$$

where h *is the step size with*

$$h = \frac{b-a}{N+1}.$$

Proof. Let p_N be the characteristic polynomial of

$$h\mathbf{T} = h \begin{pmatrix} -\frac{2}{h} & \frac{1}{h} & & \\ \frac{1}{h} & -\frac{2}{h} & \ddots & \\ & \ddots & \ddots & \frac{1}{h} \\ & & \frac{1}{h} & -\frac{2}{h} \end{pmatrix}_{N\times N} = \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}_{N\times N}.$$

The matrix $\mathbf{T}$ is a special case of Toeplitz matrix, and there are many results about the eigenvalues and eigenvectors of Toeplitz matrices [6, 51]. It is well known that the tridiagonal Toeplitz matrix has eigenvalues which are related to zeros of the Chebyshev polynomials of the second kind. Thus, after a simple transformation, the characteristic equations of successive orders of the tridiagonal Toeplitz matrix satisfy the three point Chebyshev recurrence formula [24, 30].

The p_N satisfies the following equations:

$$p_0(x) = 1, p_1(x) = x + 2$$

and by co-factor expansion

$$p_N(x) = (x + 2)p_{N-1}(x) - p_{N-2}(x).$$

The Chebyshev polynomials of the second kind [41, 23] are defined by the recurrence relation

$$U_0(y) = 1, \quad U_1(y) = 2y,$$

$$U_N(y) = 2yU_{N-1}(y) - U_{N-2}(y).$$

By a change of variable $y = x/2+1$, p_N is related to the Chebyshev polynomials of the second kind [41, 23] by

$$p_N(x) = U_N(x/2 + 1).$$

For $k = 1, \ldots, N$, the roots of the Chebyshev polynomials are given by

$$\cos\left(\frac{k\pi}{N+1}\right).$$

Thus by the change of variable formula, the eigenvalues of $h\mathbf{T}$ are

$$2\left(\cos\left(\frac{k\pi}{N+1}\right)-1\right).$$

That is the eigenvalues of $\mathbf{T}$ are

$$\frac{2}{h}\left(\cos\left(\frac{k\pi}{N+1}\right)-1\right).$$

Thus the eigenvalues of $\mathbf{G}$, which are the reciprocal of the eigenvalues of $\mathbf{G}$ are

$$\frac{h}{2}\frac{1}{\left(\cos(\frac{k\pi}{N+1})-1\right)}, k=1,\ldots,N. \quad \blacksquare$$

Theorem 2.3 *Let* $\mathbf{K}$ *be such that for any* Y_1, Y_2 *the following inequality holds,*

$$|\mathbf{F}(Y_1)-\mathbf{F}(Y_2)| \leq \mathbf{K}\cdot|Y_1-Y_2|.$$

The numerical method given by the iteration (2.19)

$$Y^{(k+1)} = \mathbf{GW}\cdot\mathbf{F}(Y^{(k)}).$$

converges when

$$\rho(\mathbf{K}) < \left(1-\cos\left(\frac{\pi}{N+1}\right)\right)\frac{2(N+1)^2}{(b-a)^2 w}\ , \tag{2.21}$$

where $(b-a)$ *is the interval size and* w *is the largest weight element of the quadrature rule, i.e.,*

$$w = \max_{1\leq j\leq N} w_j \leq 1.$$

Proof. By Lemma 2.1, the eigenvalues of $\mathbf{G}$ are

$$\frac{h}{2}\frac{1}{(\cos(\frac{k\pi}{N+1})-1)}, k=1,\ldots,N, \tag{2.22}$$

where h is the step size with

$$h=\frac{b-a}{N+1}\ .$$

We want to find the spectral radius of $\mathbf{G}$, it is known that the spectral radius of a square matrix is the largest absolute value of its eigenvalues. Thus spectral radius of $\mathbf{G}$ is attained at the maximum value

$$\max_{k=1,\ldots,N} \left| \frac{h}{2} \frac{1}{\left(\cos\left(\frac{k\pi}{N+1}\right) - 1\right)} \right|.$$

The above maximum is attained at $k = 1$ and the maximum value is

$$\frac{h}{2} \frac{1}{(1 - \cos(\frac{\pi}{N+1}))}.$$

Thus the spectral radius of $\mathbf{G}$ is

$$\begin{aligned} \rho(\mathbf{G}) &= \frac{h}{2} \frac{1}{\left(1 - \cos\left(\frac{\pi}{N+1}\right)\right)} \\ &= \frac{(b-a)}{2(N+1)} \frac{1}{\left(1 - \cos\left(\frac{\pi}{N+1}\right)\right)}. \end{aligned}$$

Assume that there exists a matrix $\mathbf{K}$, such that for any Y_1, Y_2 the following inequality holds,

$$|\mathbf{F}(Y_1) - \mathbf{F}(Y_2)| \leq \mathbf{K} \cdot |Y_1 - Y_2|.$$

So we have

$$\begin{aligned} \left|Y^{(k+1)} - Y^{(k)}\right| &= \left|\mathbf{GW} \cdot \mathbf{F}(Y^{(k)}) - \mathbf{GW} \cdot \mathbf{F}(Y^{(k-1)})\right| \\ &\leq \mathbf{GW} \cdot \mathbf{K} \cdot \left|Y^{(k)} - Y^{(k-1)}\right|. \end{aligned}$$

Since we have

$$\begin{aligned} \rho(\mathbf{GW} \cdot \mathbf{K}) &\leq \rho(\mathbf{G})\rho(\mathbf{W})\rho(\mathbf{K}) \\ &= \frac{(b-a)}{2(N+1)} \frac{1}{\left(1 - \cos\left(\frac{\pi}{N+1}\right)\right)} \rho(\mathbf{W})\rho(\mathbf{K}) \end{aligned}$$

From the assumption of the quadrature rules, it follows that

$$\rho(\mathbf{W}) = wh = \frac{w(b-a)}{N+1},$$

where w is some constant. Thus

$$\rho(\mathbf{GW} \cdot \mathbf{K}) \leq \frac{(b-a)^2 w}{2(N+1)^2} \frac{1}{\left(1 - \cos\left(\frac{\pi}{N+1}\right)\right)} \rho(\mathbf{K}).$$

The iteration converges if

$$\rho(\mathbf{GW} \cdot \mathbf{K}) < 1.$$

A sufficient condition for the iteration converges is

$$\frac{(b-a)^2 w}{2(N+1)^2} \frac{1}{(1-\cos(\frac{\pi}{N+1}))} \rho(\mathbf{K}) < 1,$$

i.e.,

$$\rho(\mathbf{K}) < \left(1 - \cos(\frac{\pi}{N+1})\right) \frac{2(N+1)^2}{(b-a)^2 w} \cdot \quad \blacksquare$$

In practice, if we apply the algorithm on a large interval, i.e., $b-a$ is a large number, We can use the domain decomposition method by dividing the large interval (a, b) into several smaller intervals and solve independently on each smaller interval.

The challenge is that we do not know the boundary values of the smaller intervals if we divide (a, b) into smaller intervals, since the only known boundary conditions are at points a and b. We will address this challenge by adding a small overlap between the adjacent intervals and use the result from last round iteration in the neighboring intervals as the boundary values for the next iteration.

CHAPTER 3

DOMAIN DECOMPOSITION METHODS

3.1 Background and Motivation

Domain decomposition methods solve a boundary value problem by splitting it into smaller boundary value problems on subdomains and iterating to coordinate the solution between adjacent subdomains. A coarse problem with one or few unknowns per subdomain is used to further coordinate the solution between the subdomains globally. The problems on the subdomains are independent, which makes domain decomposition methods suitable for parallel computing. Domain decomposition methods are typically used as preconditioners for Krylov space iterative methods, such as the conjugate gradient method or GMRES [66].

The Schwarz alternating method is an overlapping domain decomposition method which was first formulated by H.A.Schwarz and served as a theoretical tool: its convergence for general second order elliptic partial differential equations was first proved much later, in 1951, by Solomon Mikhlin. The problem considered by Schwarz was a Dirichlet problem (with Laplace's equation) on a domain consisting of a circle and a partially overlapping rectangle as shown in Figure 3.1. To solve the Dirichlet problem on one of the two subdomains

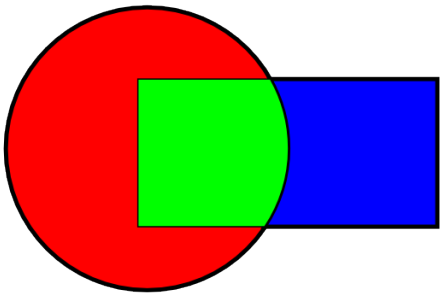

Figure 3.1: Representation of the problem considered by H. A. Schwarz. Available from: URL:Public Domain, `https://commons.wikimedia.org/wiki/File:Ddm_original_logo.png#/media/File:Ddm_original_logo.png`

(the rectangle or the circle), the value of the solution must be known on the border: since a part of the border is contained in the other subdomain, the Dirichlet problem must be solved jointly on the two subdomains. An iterative algorithm is introduced:

1. Make a first guess of the solution on the circle's boundary part that is contained in the square.
2. Solve the Dirichlet problem on the circle.
3. Use the solution in step 2 to approximate the solution on the square's boundary.
4. Solve the Dirichlet problem on the square.
5. Use the solution in step 4 to approximate the solution on the circle's boundary, then go to step 2.

In non-overlapping methods, the subdomains intersect only on their interface. In primal methods, such as Balancing Domain Decomposition and BDDC, the continuity of the solution across subdomain interfaces is enforced by representing the value of the solution on all neighboring subdomains by the same unknown. In dual methods, such as FETI, the continuity of the solution across the subdomain interface is enforced by Lagrange multipliers. The FETI-DP method is a hybrid between a dual and a primal method [25].

Non-overlapping domain decomposition methods are also called iterative substructuring methods.

Mortar methods are discretization methods for partial differential equations, which use separate discretizations on nonoverlapping subdomains. The meshes on the subdomains do not match on the interface, and the equality of the solution is enforced by Lagrange multipliers, judiciously chosen to preserve the accuracy of the solution. In engineering practice in the finite element method, continuity of solutions between non-matching subdomains is implemented by multiple-point constraints.

Finite element simulations of moderate size models require solving linear systems with millions of unknowns. Several hours per time step is an average sequential run time, therefore, parallel computing is a necessity. Domain decomposition methods embody large potential for a parallelization of the finite element methods, and serve a basis for distributed, parallel computations [20].

3.2 Domain Decomposition Method Application

To apply the domain decomposition method for the boundary value problem on a large interval size $b-a$, the idea is to partition the interval $[a, b]$ into p overlapping sub-intervals. The illustration is shown in Figure 3.2. Let us consider the solutions on each of the p overlapping sub-intervals

$$[a, z_{r_1}], [z_{l_2}, z_{r_2}], \ldots, [z_{l_p}, b]$$

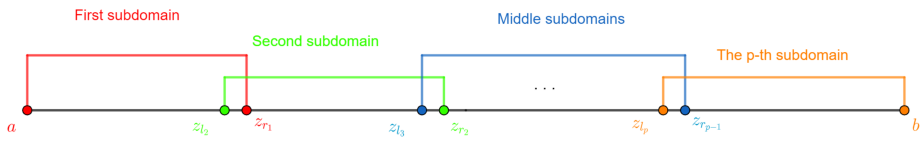

Figure 3.2: Domain Decomposition on the interval $[a, b]$

where the z_{l_j} denotes the left boundary point on the j-th subdomain and z_{r_j} denotes the right boundary point on the j-th subdomain. We will consider the boundary value problems on the subdomains as follows

$$\begin{cases} \ddot{\mathbf{x}}_1(z) = \mathbf{f}(\mathbf{x}_1(z)) \\ \mathbf{x}_1(a) = \mathbf{A} \\ \mathbf{x}_1(z_{r_1}) = \mathbf{x}_2(z_{r_1}) \end{cases} \tag{3.1}$$

$$\begin{cases} \ddot{\mathbf{x}}_2(z) = \mathbf{f}(\mathbf{x}_2(z)) \\ \mathbf{x}_2(z_{l_2}) = \mathbf{x}_1(z_{l_2}) \\ \mathbf{x}_2(z_{r_2}) = \mathbf{x}_3(z_{r_2}) \end{cases}$$

$$\begin{cases} \ddot{\mathbf{x}}_3(z) = \mathbf{f}(\mathbf{x}_3(z)) \\ \mathbf{x}_3(z_{l_3}) = \mathbf{x}_2(z_{l_3}) \\ \mathbf{x}_3(z_{r_3}) = \mathbf{x}_4(z_{r_3}) \end{cases}$$

$$\vdots$$

$$\begin{cases} \ddot{\mathbf{x}}_p(z) = \mathbf{f}(\mathbf{x}_p(z)) \\ \mathbf{x}_p(z_{l_p}) = \mathbf{x}_{p-1}(z_{l_p}) \\ \mathbf{x}_p(b) = \mathbf{B}. \end{cases}$$

To solve the boundary value problems (3.1) on each subdomains

$$[a, z_{r_1}], [z_{l_2}, z_{r_2}], \ldots, [z_{l_p}, b]$$

when implemented, we apply the formula (2.20) in Chapter 2. We discretize each subdomain such that there are N interrior discretization points in each subdomain and an illustration is shown in Figure 3.3. Hence the grid points

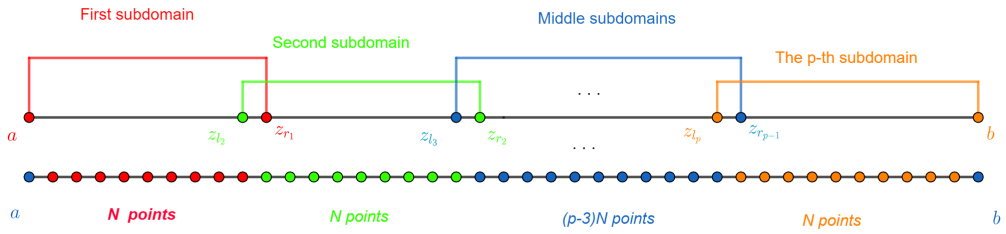

Figure 3.3: Representation of discretization

are

$$a < z_0 < z_1 < \cdots < z_{pN} < b.$$

We can handle discretizations with different numbers of nodes in each subdomain, but for simplicity, we develop our method with the same number of points N.

Denote the total number of discretization points in the original domain $[a, b]$ by J, then we have $J = pN$, where p is the number of subdomains and N is number of interior discretization points on each subdomain. We solve the boundary value problem on each sub-interval using the method in Chapter 2. As in Figure 3.3, we have

$$\begin{aligned} z_{l_1} &= a \quad z_{r_1} = z_{N+1} \\ z_{l_2} &= z_N \quad z_{r_2} = z_{2N+1} \\ z_{l_3} &= z_{2N} \quad z_{r_3} = z_{3N+1} \\ &\cdots \\ z_{l_p} &= z_{pN} \quad z_{r_p} = b. \end{aligned}$$

We now describe the domain decomposition numerical methods as the following equations, i.e., the equations solved at iteration $k+1$ are

$$\begin{cases} \ddot{\mathbf{x}}_1(z) = \mathbf{f}(\mathbf{x}_1(z)) \\ \mathbf{x_1}(a) = \mathbf{A} \\ \mathbf{x_1}(z_{N+1}) = \mathbf{x_2}(z_{N+1}) \end{cases} \tag{3.2}$$

$$\begin{cases} \ddot{\mathbf{x}}_2(z) = \mathbf{f}(\mathbf{x}_2(z)) \\ \mathbf{x}_2(z_N) = \mathbf{x}_1(z_N) \\ \mathbf{x}_2(z_{2N+1}) = \mathbf{x}_3(z_{2N+1}) \end{cases}$$

$$\begin{cases} \ddot{\mathbf{x}}_3(z) = \mathbf{f}(\mathbf{x}_3(z)) \\ \mathbf{x}_3(z_{2N}) = \mathbf{x}_2(z_{2N}) \\ \mathbf{x}_3(z_{3N+1}) = \mathbf{x}_4(z_{3N+1}) \end{cases}$$

$$\vdots$$

$$\begin{cases} \ddot{\mathbf{x}}_p(z) = \mathbf{f}(\mathbf{x}_p(z)) \\ \mathbf{x}_p(z_{(p-1)N}) = \mathbf{x}_{p-1}(z_{(p-1)N}) \\ \mathbf{x}_p(b) = \mathbf{B}. \end{cases}$$

For the ODE on each subdomain, we implement the iteration method (2.20) that we presented in Chapter 2 for nonhomogeneous boundary value conditions. Now apply (2.20) on each subdomain. Note that in (2.20) the subscript of $\mathbf{H}_{\mathbf{A},\mathbf{B}}$, $\mathbf{A}$ is the left boundary values and $\mathbf{B}$ is the right boundary value. So when we apply (2.20) on the first subdomain, the left boundary value is $\mathbf{A}$ and the right boundary value is $\mathbf{x}_2^k(z_{N+1})$, i.e., the solution of the second subdomain $\mathbf{x}_2^k(\cdot)$ evaluated at the right end point of the first subdomain (z_{N+1}). So consider the iteration (2.20), the approximation to the solution for the first subdomain can be represented as

$$X_1^{k+1} = \mathbf{GW} \cdot \mathbf{F}(X_1^k) + \mathbf{H}_{\mathbf{A},\mathbf{x}_2^k(z_{N+1})}.$$

Similar reasoning holds for all other subdomains. Therefore we can write the iterative method with the multiple domains as

$$\begin{cases} X_1^{k+1} = \mathbf{GW} \cdot \mathbf{F}(X_1^k) + \mathbf{H}_{\mathbf{A},\mathbf{x}_2^k(z_{N+1})} \\ X_2^{k+1} = \mathbf{GW} \cdot \mathbf{F}(X_2^k) + \mathbf{H}_{\mathbf{x}_1^k(z_N),\mathbf{x}_3^k(z_{2N+1})} \\ X_3^{k+1} = \mathbf{GW} \cdot \mathbf{F}(X_3^k) + \mathbf{H}_{\mathbf{x_2}^k(z_{2N}),\mathbf{x}_4^k(z_{3N+1})} \\ \vdots \\ X_p^{(k+1)} = \mathbf{GW} \cdot \mathbf{F}(X_p^k) + \mathbf{H}_{\mathbf{x}_p^k(z_{pN}),\mathbf{B}} \end{cases} \tag{3.3}$$

with

$$X_s = \left(\mathbf{x}(z_{(s-1)N+1}), \mathbf{x}(z_{(s-1)N+2}), \ldots, \mathbf{x}(z_{(s+1)N})\right); \quad s = 1, 2, \ldots, p.$$

Define matrices $\hat{\mathbf{G}}$ and $\hat{\mathbf{W}}$ as follows,

$$\hat{\mathbf{G}} = \begin{pmatrix} \mathbf{G} & & & & \\ & \ddots & & & \\ & & \mathbf{G} & & \\ & & & \ddots & \\ & & & & \mathbf{G} \end{pmatrix},$$

$$\hat{\mathbf{W}} = \begin{pmatrix} \mathbf{W} & & & & \\ & \ddots & & & \\ & & \mathbf{W} & & \\ & & & \ddots & \\ & & & & \mathbf{W} \end{pmatrix}.$$

Let us define the following two matrices $\mathbf{C}$ and $\hat{\mathbf{H}}$,

$$\mathbf{C} = \begin{pmatrix} NA & NA & \ldots & NA & NA \\ (N-1)A & (N-1)A & \ldots & (N-1)A & (N-1)A \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ A & A & \ldots & A & A \\ 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 \\ B & B & \ldots & B & B \\ 2B & 2B & \ldots & 2B & 2B \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ NB & NB & \ldots & NB & NB \end{pmatrix}_{J \times J}.$$

$$\hat{\mathbf{H}} = \frac{1}{N+1}\begin{pmatrix}
\begin{matrix} & 1 \\ & 2 \\ & \vdots \\ & N \end{matrix} & & & & \\
& \begin{matrix} N & & 1 \\ \vdots & & 2 \\ 2 & & \vdots \\ 1 & & N \end{matrix} & & & \\
& & \begin{matrix} & N & & 1 \\ & \vdots & & 2 \\ & 2 & & \vdots \\ & 1 & & N \end{matrix} & & \\
& & & \begin{matrix} & & \ddots \\ & & \\ & & \\ \ddots & & \end{matrix} & \\
& & & & \begin{matrix} & N \\ & \vdots \\ & 2 \\ & 1 \end{matrix}
\end{pmatrix}.$$

Theorem 3.1 *The domain decomposition iterative method* (3.3) *converges for any initial values* X^0 *if*

$$\rho((\mathbf{K} + (\hat{\mathbf{G}}\hat{\mathbf{W}})^{-1}\hat{\mathbf{H}})) < \frac{2(pN+1)}{(b-a)}\frac{N+1}{w}\left(1 - \cos\left(\frac{\pi}{N+1}\right)\right), \qquad (3.4)$$

where p *is the total number of subdomains,* N *is number of discretization points on each subdomain,* $(b-a)$ *is the interval size and* w *is the largest weight element of the quadrature rule, i.e.,*

$$w = \max_{1 \le j \le N} w_j \le 1.$$

and $\mathbf{K}$ *is such that for any* X_1, X_2 *the following inequality holds,*

$$|\mathbf{F}(X_1) - \mathbf{F}(X_2)| \leq \mathbf{K} \cdot |X_1 - X_2| \,. \tag{3.5}$$

Proof. Consider (3.3) and by the definition of

$$\begin{aligned}\mathbf{H}_{\mathbf{A},\mathbf{B}}(i,:) &= \frac{a\mathbf{B} - b\mathbf{A} + (\mathbf{A}-\mathbf{B})z_i}{a-b}(1,1,\ldots,1) \\ &= \left[\frac{a-z_i}{a-b}\mathbf{B} + \frac{z_i-b}{a-b}\mathbf{A}\right](1,1,\ldots,1).\end{aligned}$$

Denote by $\mathbf{1}^T = (1,1,\ldots,1)$, we have for $1 \leq i \leq N$, the following holds

$$\begin{cases} \mathbf{H}_{\mathbf{A},\mathbf{x_2}^{(k)}(z_{N+1})}(i,:) = \left(\frac{a-z_i}{a-z_{N+1}}\mathbf{x_2}^{(k)}(z_{N+1}) + \frac{z_i-z_{N+1}}{a-z_{N+1}}\mathbf{A}\right)\mathbf{1^T} \\ \mathbf{H}_{\mathbf{x_1}^{(k)}(z_N),\mathbf{x_3}^{(k)}(z_{2N+1})}(i,:) = \left(\frac{z_N-z_{i+N}}{z_N-z_{2N+1}}\mathbf{x_3}^{(k)}(z_{2N+1}) + \frac{z_{i+N}-t_{2N+1}}{z_N-z_{2N+1}}\mathbf{x_1}^{(k)}(z_N)\right)\mathbf{1^T} \\ \mathbf{H}_{\mathbf{x_2}^{(k)}(z_{2N}),\mathbf{x_4}^{(k)}(z_{3N+1})}(i,:) = \left(\frac{z_{2N}-z_{i+2N}}{z_{2N}-z_{3N+1}}\mathbf{x_4}^{(k)}(z_{3N+1}) + \frac{z_{i+2N}-z_{3N+1}}{z_{2N}-z_{3N+1}}\mathbf{x_2}^{(k)}(z_{2N})\right)\mathbf{1^T} \\ \vdots \\ \mathbf{H}_{\mathbf{x_p}^{(k)}(z_{pN}),\mathbf{B}}(i,:) = \left(\frac{z_{pN}-z_{i+pN}}{z_{pN}-b}\mathbf{B} + \frac{z_{i+pN}-b}{z_{pN}-b}\mathbf{x_p}^{(k)}(z_{pN})\right)\mathbf{1^T}. \end{cases}$$

Simplify

$$\begin{cases} \mathbf{H}_{\mathbf{A},\mathbf{x_2}^{(k)}(z_{N+1})}(i,:) = \left(\frac{i}{N+1}\mathbf{x_2}^{(k)}(z_{N+1}) + \frac{i-N-1}{N+1}\mathbf{A}\right)\mathbf{1^T} \\ \mathbf{H}_{\mathbf{x_1}^{(k)}(z_N),\mathbf{x_3}^{(k)}(z_{2N+1})}(i,:) = \left(\frac{i}{N+1}\mathbf{x_3}^{(k)}(z_{2N+1}) + \frac{i-N-1}{N+1}\mathbf{x_1}^{(k)}(z_N)\right)\mathbf{1^T} \\ \mathbf{H}_{\mathbf{x_2}^{(k)}(z_{2N}),\mathbf{x_4}^{(k)}(z_{3N+1})}(i,:) = \left(\frac{i}{N+1}\mathbf{x_4}^{(k)}(z_{3N+1}) + \frac{i-N-1}{N+1}\mathbf{x_2}^{(k)}(z_{2N})\right)\mathbf{1^T} \\ \vdots \\ \mathbf{H}_{\mathbf{x_p}^{(k)}(z_{pN}),\mathbf{B}}(i,:) = \left(\frac{i}{N+1}\mathbf{B} + \frac{i-N-1}{N+1}\mathbf{x_p}^{(k)}(z_{pN})\right)\mathbf{1^T}. \end{cases}$$

Define

$$X^k = (X_1^k, X_2^k, ..., X_p^k)^T.$$

Now (3.3) can be represented as

$$X^{k+1} = \hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{F}\left(X^k\right) + \hat{\mathbf{H}}^k + \mathbf{C}. \tag{3.6}$$

From the hypothesis (3.5) we have

$$\begin{aligned}\left|X^{k+1} - X^k\right| &\leq \left(\hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{K} + \hat{\mathbf{H}}\right) \cdot \left|\left(X^k - X^{k-1}\right)\right| \\ &= \hat{\mathbf{G}}\hat{\mathbf{W}}(\mathbf{K} + (\hat{\mathbf{G}}\hat{\mathbf{W}})^{-1}\hat{\mathbf{H}})\left|\left(X^k - X^{k-1}\right)\right|\end{aligned}$$

The method converges if the spectral radius of $\hat{\mathbf{G}}\hat{\mathbf{W}}(\mathbf{K}+(\hat{\mathbf{G}}\hat{\mathbf{W}})^{-1}\hat{\mathbf{H}})$ is less than 1. Thus using the fact that $\rho(\mathbf{MN}) = \rho(\mathbf{M})\rho(\mathbf{N})$, the method converges if

$$\rho((\mathbf{K} + (\hat{\mathbf{G}}\hat{\mathbf{W}})^{-1}\hat{\mathbf{H}})) < \frac{1}{\rho(\hat{\mathbf{G}}\hat{\mathbf{W}})}.$$

Since $\hat{\mathbf{G}}$ and $\hat{\mathbf{W}}$ are block diagonal matrices with block element $\mathbf{G}$ and $\mathbf{W}$. From (2.22) we know the spectrum radius of $\hat{\mathbf{G}}$ is

$$\begin{aligned}\rho(\hat{\mathbf{G}}) = \rho(\mathbf{G}) &= \frac{h}{2}\frac{1}{\left(1-\cos(\frac{\pi}{N+1})\right)} \\ &= \frac{(b-a)}{2(pN+1)}\frac{1}{\left(1-\cos(\frac{\pi}{N+1})\right)}.\end{aligned}$$

Similarly,

$$\rho(\hat{\mathbf{W}}) = \rho(\mathbf{W}) = \frac{w}{N+1},$$

Thus the implementation converges if

$$\rho((\mathbf{K} + (\hat{\mathbf{G}}\hat{\mathbf{W}})^{-1}\hat{\mathbf{H}})) < \frac{2(pN+1)}{(b-a)}\frac{N+1}{w}\left(1-\cos\left(\frac{\pi}{N+1}\right)\right).$$

where p is the total number of subdomains, and N is number of discretization points on each subdomain. ■

Note that the bound (3.4) is dependent on the number of subdomains p, the number of discretization points on each subdomain N and the interval size $b-a$. In general, the larger the bound (3.4), the larger the class of problems for which the method converges. If $b-a$ is small, then the bound (3.4) is larger. Also we can increase the number of the subdomains p to obtain a larger bound in (3.4). In the biofilm example in Chapter 5, we found that by increasing the number of subdomains p, we can obtain the solution of the ODE on a larger spacial domain than previously possible.

3.3 Numerical Experiment

In this section, a numerical example is presented illustrating the effectiveness of the domain decomposition method with overlap. The approximate solution before and after applying the overlap domain decomposition technique is presented. The numerical results obtained are compared with the analytical solution.

Example 1. We consider the following second order ODE defined on the interval $[0, b]$:

$$\ddot{x}(z) = 100x(z) \tag{3.7}$$

with boundary value conditions

$$\begin{cases} x(0) = 1 \\ x(b) = e^{\sqrt{100}b}. \end{cases} \tag{3.8}$$

The true solution $x(z)$ is

$$x(z) = e^{\sqrt{100}z}. \tag{3.9}$$

We ran an implementation of the algorithm with domain decomposition and without domain decomposition, and the result is illustrated in the Table 3.1.

Example 2. We consider the following second order ODE defined on the interval $[0, 3]$:

$$\ddot{x}(z) = kx(z) \tag{3.10}$$

with boundary value conditions

$$\begin{cases} x(0) = 1 \\ x(3) = 0. \end{cases} \tag{3.11}$$

The true solution $x(z)$ is

$$x(z) = \frac{e^{\sqrt{k}z} - e^{6\sqrt{k}-\sqrt{k}z}}{1 - e^{6\sqrt{k}}}. \tag{3.12}$$

Table 3.1: Comparison of error in results of different methods

Interval size	Original method	Method with overlap domain decomposition
0.5	1.87e-07	1.24e-09
1	2.97e-06	7.50e-09
1.5	1.89e-05	2.18e-08
2	1.71e-02	2.47e-08
2.5	2.69e-02	3.47e-08
3	1.89e-01	5.78e-08

For $k = 100$ in ODE (3.10), the result obtained by method with overlap domain decomposition technique is shown in Figure 3.5, the result obtained by original method without the overlap domain decomposition technique is shown in Figure 3.4. The result shows that by applying the overlap domain decomposition technique, we can solve problem of more steep slope than previously possible, for example, the case when $k = 100$ as is illustrated in Figure 3.4 and Figure 3.5.

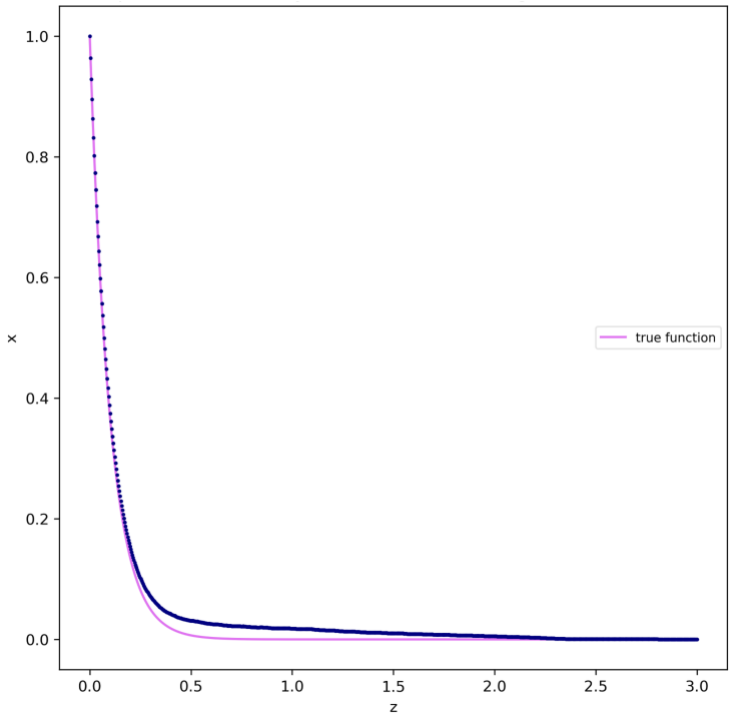

Figure 3.4: $k = 100$ in (3.10) solved without overlap domain decomposition

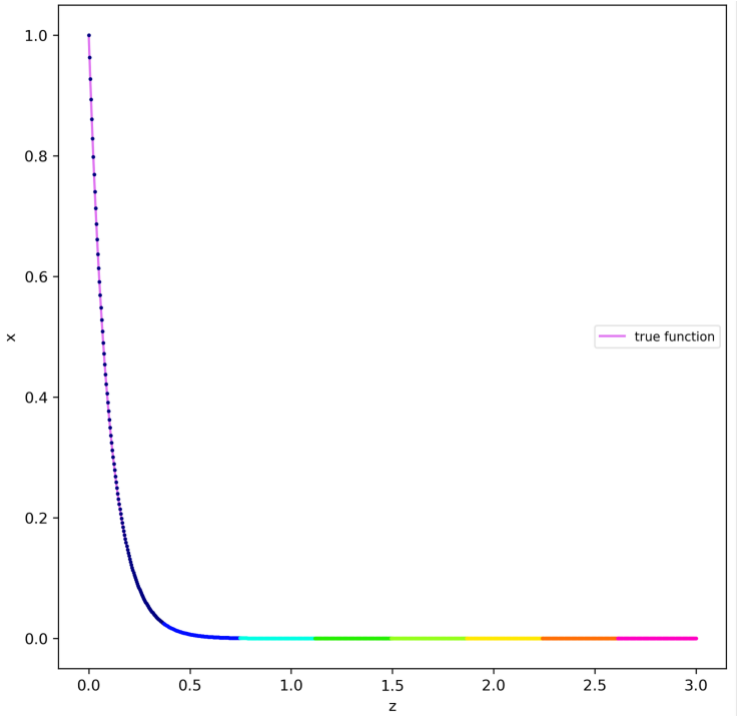

Figure 3.5: $k = 100$ in (3.10) solved with overlap domain decomposition

CHAPTER 4

ASYNCHRONOUS METHODS

4.1 Background and Motivation

With the advent of parallel computers, many new algorithms were devised or rediscovered for the new architectures. An important concept in the design of parallel algorithms is that of load balancing, which simply means that the work has to be approximately equally distributed among processors. Otherwise, some processors finish their task much earlier than others, and the waiting time (also called idle time) degrades the performance of the algorithm. This concept has been widely accepted as a requirement for efficient algorithms, and has dictated for example that when the geometric domain of a physical problem is divided into subdomains (to be processed by the different processors), each should be of approximately the same size. In contrast to load balancing, the idea of asynchronous methods is to avoid processor idle time by eliminating as much as possible synchronization points, i.e., points at which a processor must wait for information from other processors. In this way, problems which naturally would decompose into processes of very different size, e.g., those with unstructured meshes, can do so without difficulty. The price one pays for this freedom is that some processors will perform extra computations, and it is only when the load is not well balanced, or when communication between the processors is slow, that this approach is advantageous [29].

Since the publication of the pioneering paper in 1969 by Chazan and Miranker [18], the theory and application of asynchronous iterations has been studied and used by many authors. For early surveys of asynchronous iterative methods, see [2, 13, 14, 27] (see also the papers [59, 63]).

4.1.1 Computational and Mathematical Models

The computational and mathematical models of asynchronous methods in this section follows from [29]. Assume that we are given a product space $E = E_1 \times \ldots \times E_m$ and an application $H : E \to E$ whose components are denoted H_i, i.e., we have

$$H : E \to E, x = (x_1, \ldots, x_m) \to ((Hx)_1, \ldots, (HX)_m), \tag{4.1}$$

where $x_i, (Hx)_i = H_i(x) \in E_i, i = 1, \ldots, m$. The problem at hand is to find a fixed point of H. A standard procedure is to approximate such fixed point by variants of the the successive procedure

$$x^{k+1} = H(x^k), k = 0, 1, \ldots. \tag{4.2}$$

Computational Model 4.2.1:

while *not at convergence* **do**
  read x from common memory
  compute $x_i^{new} = H_i(x)$ for $i \in J_j$
  overwrite x_i in common memory with $x_i^{new}, i \in J_j$
  if *convergence* **then**
    break;
  end
end

Assume for now that we are working with a (shared memory) parallel computer with p processors $P_1, \ldots, P_p$ $(p \leq m)$, and associate a block of components $J_j \subseteq 1, \ldots, m$ with each processor P_j. Then a parallel variant

of the successive approximation procedure (4.2) can be implemented as in Computational Model 4.2.1 (pseudocode for processor P_j):

If processors would wait for each other to complete each run through the loop we would indeed get a (parallel synchronous) implementation of the successive approximation scheme (4.2). Since here processors do not wait, we actually get a much less structured iterative process where, due to different run times for each loop, processors get out of phase. At a given time point, different processors will have achieved different numbers of iterations (the iteration number k in (4.2) looses its meaning in this context). No idle time occur, since processors never wait for each other.

In order to mathematically analyze the Computational Model 4.2.1, we now step the iteration counter k by 1 each time x is read from the common memory by some processor $P_{j(k)}$. Then this x is made up of components each of which has been written back to memory as the result of the computation belonging to some earlier iteration. We therefore have $x = (x_1^{s_1(k)}, \ldots, x_m^{s_m(k)})$ with iteration counts $s_l(k) \in \mathbb{N}_0$, $l = 1, \ldots, m$, prior to k, indicating the iteration when the lth component just read was computed. A set I^k is defined indicating which components are computed at the kth iteration, i.e., $I^k = J_{j(k)}$. Using these sets, and under the very weak assumptions explained further below, the Computational Model 4.2.1 can be modeled mathematically according to the following definition; see, e.g., [64],

Definition 4.1 *For $k \in \mathbb{N}$, let $I^k \subseteq \{1, \ldots, m\}$ and $(s_1(k), \ldots, s_p(k)) \in \mathbb{N}_0^m$ such that*

$$\begin{aligned} &s_i(k) \leq k-1 \quad \textit{for } i \in \{1, \ldots, m\}, \quad k \in \mathbb{N}, \\ &\lim_{k\to\infty} s_i(k) = \infty \quad \textit{for } i \in \{1, \ldots, m\}, \\ &|\{k \in \mathbb{N} : i \in I^k\}| = \infty \quad \textit{for } i \in \{1, \ldots, m\}. \end{aligned} \tag{4.3}$$

Given an initial vector $x^0 \in E = E_1 \times \ldots \times E_m$, the iteration

$$x_i^k = \begin{cases} x_i^{k-1} & \textit{for } i \notin I^k \\ H_i(x_1^{s_1(k)}, \ldots, x_m^{s_m(k)}) & \textit{for } i \in I^k \end{cases} \tag{4.4}$$

is termed an asynchronous iteration (with strategy $I^k, k \in \mathbb{N}$ *and delays* $d_i(k) = k - s_i(k), i = 1, \ldots, n, k \in \mathbb{N}$).

The first hypothesis of (4.3) simply indicates that only components computed earlier (and not later ones) are used in current approximation. The second indicates that as the computation proceeds, eventually one reads newer information for each of the components. The third one indicates that no component fails to be updated as time goes on.

4.1.2 Convergence Theory

From [29], a general convergence theorem for the asynchronous iteration (4.4) is the following result of Betsekas [12].

Theorem 4.1 *Assume that there are sets* $E^k \subseteq E$ *which satisfy*

(a) $E^k = E_1^k \times \ldots \times E_m^k, k \in \mathbb{N}_0$, *(box condition)*

(b) $H(E^k) \subseteq E^{K+1} \subseteq E^k, k \in \mathbb{N}_0$, *(nested sets condition)*

(c) there exists x^* *such that*

$$y^k \in E^k, k \in \mathbb{N} \Rightarrow \lim_{k \to \infty} y^k = x^*$$

(synchronous convergence condition).

Then for an initial vector x^0 *the sequence of asynchronous iterates* x^k *from (4.4) converges to* x^**, the unique fixed point of* H*, provided assumptions (4.3) holds.*

From [29], there are several special cases of Theorem 4.1 which merit further discussion. First consider the case where each component space E_i is a normed linear space $(E_i, \|\cdot\|_i)$. Define $\|\cdot\|_w$ the weighted max-norm on E given as

$$\|x\|_w = \max_{i=1}^{m} \frac{\|x_i\|_i}{w_i}, \tag{4.5}$$

where $w = (w_1, \ldots, w_m)$ is a positive vector, i.e., $w_i > 0$ for $i = 1, \ldots, m$. From [28, 64], the following theorem holds by setting $E^k = \{x \in E : \|x - x^*\| \leq \gamma^k \cdot \|x^0 - x^*\|_w\}$ and apply Theorem 4.1. The following theorem is known as El Tarazi's theorem [3].

Theorem 4.2 *Assume that there exists $x^* \in E$ such that $H(x^*) = x^*$. Moreover, assume that there exists $\gamma \in [0,1]$ and $w \in \mathbb{R}^m$ positive, such that we have*

$$\|H(x) - x^*\|_w \leq \gamma \|x - x^*\|_2. \tag{4.6}$$

Then, for any initial vector x^0, the asynchronous iterates x^k converges to x^*, the unique common fixed point of H.

Corollary 4.1 *[9] Assume that H satisfies $\|Hx - x^*\|_w < \|x - x^*\|_w$ with respect to x^*. Then the asynchronous iterates x^k from (4.4) converge to x^*, the unique common fixed point of H.*

4.1.3 Results on Nonlinear Equations

Assume that we are given a nonlinear system of equations

$$F(x) = 0$$

where $F : D_F \subseteq \mathbb{R}^n \to \mathbb{R}^n$ Assume that this equation has exactly one solution x^* and let $H : D_H \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be an iteration function for this problem, i.e., x^* is the unique fixed point of H. Not too surprisingly, the following local version of Corollary 4.1 can be shown to hold [22, 29].

Lemma 4.1 *Assume that x^* lies in the interior of D_H and that H is Fréchet differentiable at x^*. If $\rho(|H'(x^*)|) < 1$ then there exists a neighborhood $\mathcal{N}$ of x^* such that the asynchronous iterates (4) converge to x^*; provided $x^0 \in \mathcal{N}$.*

4.2 Application of Asynchronous Methods

In Chapter 3 we applied the domain decomposition techniques, and we solved the system of nonlinear equations by an iteration method (3.3).

Now we would like to apply asynchronous iteration to solve the fixed point iteration (3.3). Assume $s_i(k), i = 1, 2, \ldots, p$, satisfies the Definition 4.1. Thus

we have

$$\begin{cases} X_1^k = \mathbf{GW} \cdot \mathbf{F}(X_1^{s_1(k)}) + \mathbf{H}_{\mathbf{A},\mathbf{x}_2^{s_2(k)}(z_{N+1})} \\ X_2^k = \mathbf{GW} \cdot \mathbf{F}(X_2^{s_2(k)}) + \mathbf{H}_{\mathbf{x}_1^{s_1(k)}(z_N),\mathbf{x}_3^{s_3(k)}(z_{2N+1})} \\ X_3^k = \mathbf{GW} \cdot \mathbf{F}(X_3^{s_3(k)}) + \mathbf{H}_{\mathbf{x_2}^{s_2(k)}(z_{2N}),\mathbf{x}_4^{s_4(k)}(z_{3N+1})} \\ \vdots \\ X_p^k = \mathbf{GW} \cdot \mathbf{F}(X_p^{s_p(k)}) + \mathbf{H}_{\mathbf{x}_{p-1}^{s_{p-1}(k)}(z_{pN}),\mathbf{B}}. \end{cases} \tag{4.7}$$

To apply the theory of asynchronous methods introduced in Section 4.1 on (3.6), define the operator H as

$$H(X) = \hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{F}(X) + \hat{\mathbf{H}} \cdot X + \mathbf{C}, \tag{4.8}$$

where $\hat{\mathbf{G}}$ and $\hat{\mathbf{W}}$ are defined as the block diagonal matrix with diagonal elements $\mathbf{G}$ and $\mathbf{W}$ and $\hat{\mathbf{G}}, \hat{\mathbf{W}}$ and $\hat{\mathbf{H}}$ are defined in the Chapter 3. Assume that we are given a product space $E = E_1 \times \cdots \times E_p$ and the application $H : E \to E$ whose components are denoted H_i, i.e., we have

$$H : E \to E, \quad X = (X_1, \ldots, X_p) \to ((HX)_1, \ldots, (HX)_p),$$

where $X_i, (HX)_i = H_i(X) \in E_i, i = 1, \ldots, p$.

Theorem 4.3 *Assume that* $\mathbf{F}$ *is Fréchet differentiable at a particular point* X^**, then* $H : E \to E$ *defined in* (4.8) *is also Fréchet differentiable at the particular point* X^**. Also the following relation holds,*

$$H'(X^*) = \hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{F}'(X^*) + \hat{\mathbf{H}}$$

Proof. From the problem assumption we know $\mathbf{F}$ is Fréchet differentiable at a particular point X, so that

$$\mathbf{F}(X + \Delta X) = \mathbf{F}(X) + \mathbf{F}'(X)\Delta X + e(\Delta X),$$

and the error term $e(\Delta X)$ satisfies

$$\lim_{\Delta X \to 0} \frac{\|e(\Delta X)\|}{\|\Delta X\|} = 0.$$

Notice that

$$\begin{aligned} H(X+\Delta X) &= \hat{\mathbf{G}}\hat{\mathbf{W}}\cdot\mathbf{F}(X+\Delta X)+\hat{\mathbf{H}}\cdot X+\hat{\mathbf{H}}\cdot\Delta X+\mathbf{C} \\ &= \hat{\mathbf{G}}\hat{\mathbf{W}}\left(\mathbf{F}(X)+\mathbf{F}'(X)\Delta X+e(\Delta X)\right)+\hat{\mathbf{H}}\cdot X+\hat{\mathbf{H}}\cdot\Delta X+\mathbf{C} \\ &= \underbrace{\hat{\mathbf{G}}\hat{\mathbf{W}}\cdot\mathbf{F}(X)+\hat{\mathbf{H}}\cdot X+\mathbf{C}}_{H(X)}+\underbrace{\hat{\mathbf{G}}\hat{\mathbf{W}}\cdot\mathbf{F}'(X)\Delta X+\hat{\mathbf{H}}\cdot\Delta X}_{H'(X)\Delta X} \\ &+\underbrace{\hat{\mathbf{G}}\hat{\mathbf{W}}\cdot e(\Delta X)}_{\text{small}}. \end{aligned}$$

Assume that there exists a fixed point X^* of H such that $H(X^*) = X^*$. This suggests that H is differentiable at X^* and that $H'(X)$ is the linear transformation defined by

$$H'(X^*) = \hat{\mathbf{G}}\hat{\mathbf{W}}\cdot\mathbf{F}'(X^*)+\hat{\mathbf{H}}.$$

To prove that this is true, we only need to show that

$$\lim_{\Delta X\to 0}\frac{\|\hat{\mathbf{G}}\hat{\mathbf{W}}\cdot e(\Delta X)\|}{\|\Delta X\|}=0$$

Then

$$\frac{\|\hat{\mathbf{G}}\hat{\mathbf{W}}\cdot e(\Delta X)\|}{\|\Delta X\|}\le\frac{\|\hat{\mathbf{G}}\hat{\mathbf{W}}\|\|e(\Delta X)\|}{\|\Delta X\|}$$

which approaches 0 as $\Delta X\to 0$. ■

Theorem 4.4 *The asynchronous method converges if*

$$\|\mathbf{F}'(X^*)\|\le\left(1-\|\hat{\mathbf{H}}\|\right)\frac{2(pN+1)}{(b-a)}\frac{N+1}{w}\left(1-\cos(\frac{\pi}{N+1})\right), \tag{4.9}$$

where p is the number of subdomains, N is the number of discretization points on each subdomain.

Proof. Lemma 4.1 is used in this proof. From Theorem 4.3, we have

$$H'(X^*) = \hat{\mathbf{G}}\hat{\mathbf{W}}\cdot\mathbf{F}'(X^*)+\hat{\mathbf{H}}.$$

From Lemma 4.1, the sufficient condition for convergence of asynchronous method is $\rho(|H'(X^*)|) < 1$. Thus the equivalent sufficient condition for convergence of asynchronous method is

$$\rho\left(\left|\hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{F}'(X^*) + \hat{\mathbf{H}}\right|\right) < 1.$$

Since for any vector induced matrix norm,

$$\begin{aligned}\rho\left(\left|\hat{\mathbf{G}}\hat{\mathbf{W}} \cdot \mathbf{F}'(X^*) + \hat{\mathbf{H}}\right|\right) &< \|\mathbf{F}'(X^*)\|\|\hat{\mathbf{G}}\hat{\mathbf{W}}\| + \|\hat{\mathbf{H}}\| \\ &= \|\mathbf{F}'(X^*)\|\|\hat{\mathbf{G}}\hat{\mathbf{W}}\| + \|\hat{\mathbf{H}}\|\end{aligned}$$

Thus it is sufficient to consider the condition

$$\|\mathbf{F}'(X^*)\|\|\hat{\mathbf{G}}\hat{\mathbf{W}}\| + \|\hat{\mathbf{H}}\| < 1.$$

This is equivalent to

$$\begin{aligned}\|\mathbf{F}'(X^*)\| &\leq \frac{1 - \|\hat{\mathbf{H}}\|}{\|\hat{\mathbf{G}}\hat{\mathbf{W}}\|} \\ &= \left(1 - \|\hat{\mathbf{H}}\|\right) \frac{2(pN+1)}{(b-a)} \frac{N+1}{w} \left(1 - \cos\left(\frac{\pi}{N+1}\right)\right).\end{aligned}$$

where p is the number of subdomains, N is the number of discretization points on each subdomain. ■

Note that the bound (4.9) is also dependent on the number of subdomains p, the number of discretization points on each subdomain N and the domain size $b - a$. In general, the larger the bound (3.4), the larger the class of problems for which the method converges. If $b - a$ is small, the bound (4.9) is larger. Also we can increase the number of the subdomains p to obtain a larger bound in (4.9). The bound (4.9) is similar to the bound (3.4) except for one extra term. The advantage of asynchronous method is that it can make efficient use of computation resources, avoid processor idle time and speed up the computation.

4.3 Numerical Experiments

In this section, a few numerical examples are presented illustrating the performance of the asynchronous methods. The numerical results obtained are compared with the analytical solution for each example. These are found to be in good agreement with each other.

Example 1. We will consider the following coupled second order ODE system of two equations defined on the interval $(0, b)$, where $b > 0$ is some arbitrary constant:

$$\begin{cases} \ddot{u}(z) = -u(z)v(z) - u^3(z) + 2 + z^2e^z + z^6 \\ \ddot{v}(z) = v(z) - u^2(z) + z^4 \end{cases} \tag{4.10}$$

with boundary value conditions

$$\begin{cases} u(0) = 0 \\ u(b) = b^2 \\ v(0) = 1 \\ v(b) = e^b. \end{cases} \tag{4.11}$$

The true solution $u(z), v(z)$ are

$$\begin{cases} u(z) = z^2 \\ v(z) = e^z. \end{cases} \tag{4.12}$$

We apply the shooting method, the nonlinear finite difference method and the integral method on this example with different values of b, i.e., different interval size, and compare the accuracy of each numerical solution.

In Table 4.1, we compare the error of the numerical results between four different methods. The error of the solutions for several different interval sizes from 5 to 35 are shown. In Table 4.2, we compare the residual of the numerical result between the four different methods. The residual of the solutions for several different interval sizes from 5 to 35 are shown. In Table 4.3, we compare the computational time of the four different methods. The computational time of the methods for several different interval sizes from 5 to 35 are shown.

Table 4.1: Comparison of error in results of different methods

Interval size	Asynchronous Finite-difference method	Shooting method	Integral method	Asynchronous Integral method
5	2.52e-07	5.17e-9	1.98e-07	2.62e-07
10	1.21e-06	2.75e-04	1.49e-06	2.11e-06
15	2.06e-06	3.66e-02	1.22e-06	2.06e-06
20	9.80e-06	7.87e-02	3.64e-06	6.75e-05
25	1.98e-05	8.64e-02	4.50e-06	9.07e-05
30	4.59e-05	4.56e-02	1.49e-05	3.20e-05
35	3.64e-05	6.85e-02	2.72e-05	6.40e-05

Table 4.2: Comparison of residual in results of different methods

Interval size	Asynchronous Finite-difference method	Shooting method	Integral method	Asynchronous Integral method
5	1.13e-10	1.56e-10	1.06e-10	1.39e-10
10	5.25e-09	3.96e-05	4.02e-09	5.43e-09
15	1.15e-09	9.45e-03	1.06e-09	1.58e-09
20	7.45e-08	5.24e-03	4.50e-08	7.64e-08
25	9.05e-08	3.56e-02	7.90e-08	1.46e-07
30	5.90e-07	8.43e-02	7.03e-08	1.24e-07
35	9.06e-07	9.84e-02	9.05e-07	2.35e-07

Table 4.3: Comparison of computational time in results of different methods

Interval size	Asynchronous Finite-difference method	Shooting method	Integral method	Asynchronous Integral method
5	125s	87s	164s	132s
10	327s	389s	379s	308s
15	2519s	4122s	3684s	3099s
20	6480s	8050s	7135s	6320s
25	15924s	23748s	19562s	17510s
30	31935s	58365s	40624s	31374s
35	69428s	97004s	82257s	69399s

This numerical example shows that the finite difference and integral method perform better than the shooting method, especially on large intervals. For example, when the interval size is 35, i.e., the interval is $(0, 35)$. The slope of the true solution is very large, approximately e^{35}. The shooting method performs very badly when the slope is large. In conclusion, finite difference methods and the integral methods are better in the extreme case when the slope is very large, which is also the case we are more likely to encounter in mathematical modelling of biofilm.

For asynchronous computations, we simulate how the algorithm runs on multiple processors or multiple computers by using MATLAB Parallel Computing Toolbox. You can run multiple MATLAB workers (MATLAB computational engines) on a single machine to execute applications in parallel, with Parallel Computing Toolbox. This approach allows more control over the parallelism than with built-in multithreading, and is often used for coarser grained problems such as running parameter sweeps in parallel.

In our example, each processor is responsible for updating the appointed subdomain. We simulate communications and random delays in the algorithm. For configuration of random delay, the algorithm adopts several ranges for different delay levels, and a discrete distribution indicating the percentage

of processors at each delay level. For example, we can input ranges $[0.6, 1]$, $[0.2, 0.6]$, $[0, 0.2]$ for high/medium/low delays, and a distribution $(0.2, 0.4, 0.4)$, then 20% of the processors will have a high random delay in range from 60% to 100% of the computation time, and 40% of the processors will have a random delay in range from 20% to 60% of the computation time, and 40% of the processors will have a random delay in range from 0 to 20% of the computation time. For results shown in the Table 4.1-4.3, seven threads are used with the following random delay setups: thread one has 60% random delay, thread two and five have 40% random delay, thread three and four have 15% random delay, thread six has 65% random delay and thread seven has 30% random delay.

Example 2 Consider the equation

$$\begin{cases} \ddot{u}(z) = -zu(z) - 2zv(z) - 4\cos(z) + (4z - 2z^2 - 2)\sin(z) \\ \ddot{v}(z) = -v(z) - z^2u(z) + 2(4 - z^2 + z^3)\sin(z) - (\pi^2 - 1)\sin(\pi z) \end{cases} \tag{4.13}$$

with boundary value conditions

$$\begin{cases} u(0) = 0 \\ u(1) = 0 \\ v(0) = 0 \\ v(1) = 0. \end{cases} \tag{4.14}$$

The true solution $u(z), v(z)$ are

$$\begin{cases} u(z) = 2\sin(z)(1 - z) \\ v(z) = \sin(\pi z). \end{cases} \tag{4.15}$$

In Table 4.4, the error, residual and computational time of the four methods are shown separately. From the experiment results in the Table 4.4, we can see the asynchronous finite difference method and asynchronous integral method is similar in terms of computational time. We can see that however, finite difference is slightly better in terms of error and residual attained. Note that applying the asynchronous technique on the integral method results in a improvement in terms of computational time compared to synchronous integral method.

Table 4.4: The numerical result of Example 2

Category	Asynchronous Finite-difference method	Shooting method	Integral method	Asynchronous Integral method
Error	2.6124e-04	2.2998e-02	5.2361e-03	5.9702e-03
Residual	6.8301e-09	4.8107e-04	1.7903e-09	4.8233e-09
Time	169s	575s	298s	173s

CHAPTER 5

OPTIMIZATION CONSTRAINED ODE: APPLICATION TO BIOFILM MODELING

5.1 Background and Motivation

Biofilms are a collective of one or more types of microorganisms that can grow on many different surfaces. Microorganisms that form biofilms include bacteria, fungi and protists. A biofilm is an assemblage of surface-associated microbial cells that is enclosed in an extracellular polymeric substance matrix. The biofilm consists of about 85–96% water, which means that only 2–5% of the total biofilm volume is detectable on dry surfaces [69].

Biofilms have been extensively studied over the past 20 years. Microbiological, physical, chemical, and microscopic methods have been applied to the study of biofilms. Mathematical models are powerful tools for understanding the function and evolution of biofilms in diverse communities, and early efforts on the mathematical modeling of biofilms can be traced back to the 1980s; see, e.g., [53, 54]. These studies are centered mostly on the steady-state biofilm

growth dynamics, and kinetic models are used to model them. The studies focused on modeling biofilms' thickness and spatial distribution of microbial species and substrate concentration. In [76], kinetics-free methods are used in modeling the biofilm where the classic kinetics functions are replaced by cell-level steady state metabolic pathway models.

Kinetic models have been developed in the past for modeling biofilms [73]. Kinetic models are based on the evaluation of the kinetic constants of the chemical reactions used to simulate the biological process. For example, in modeling the kinetics of the solid-state fermentation process, the parameters such as specific growth rate, process yield, process productivity, process control criteria, strategy for the production of a particular product, are involved [55].

For kinetic modeling approach, all reaction pathways can be decomposed into a series of elementary reaction steps, each of which is either unimolecular or bimolecular. By simulating each key step of a reaction, one can thus construct a model of the original pathway, independent of its complexity [70]. In kinetics modeling approach, most studies employ the reaction rate equations (RRE) to model biochemical systems. In the RRE, one simply defines the changes in the concentrations (or equivalently the number of molecules) as a function of time and location [52].

In [76], the authors propose a new approach, namely a systematic approach to model biofilm system in which a steady state kinetics dynamics is adopted. The advantage is that the model does not rely on information might be unavailable such as the kinetic rate parameters. Also, in [76] one stated goal is that of maximizing the biomass production, that is, an optimization approach is used. Additionaly, in [76] the environmental conditions can be imposed by setting constraints on the reaction fluxes.

Biofilm modeling is an important area of study. Microbial biofilms are ubiquitous in nature and clinical and public health microbiologists have recognized that biofilms should be studied to understand a number of infectious disease processes [40]. According to a study, the formation and persistence of surface-attached microbial communities, known as biofilms, are responsible for 75%

of human microbial infections (National Institutes of Health [43]). Biofilm formation is a complicated dynamical process governed by various physical, chemical principles and biological protocols. In conclusion, biofilm modeling is both an important topic and a complex system to study [75].

5.2 Optimization Constrained ODE: Biofilm System Example

We illustrate a model of biofilms by presenting a system of a single species biofilm taken from [76]. The biofilm is grown on a nutrient base, and it is considered to be glucose here. The biofilm is also exposed to oxygen from the top.

There are three kinds of internal (to the cells) metabolism in the biofilm growing process.

1). Glucose and oxygen together produced biomass through full respiration.

2). Lactate and oxygen together produced biomass through full respiration.

3). Glucose itself produced the lactate and biomass through fermentation without oxygen.

There are also three kinds of exchange between internal and external (to the cells) metabolites in the biofilm growing process: 1). Exchange of oxygen; 2). Exchange of glucose; 3). Exchange of lactate.

The model we consider is presented in the following. Denote the external (to cells) concentrations of oxygen, glucose and lactate by C_1, C_2, C_3 and denote the three fluxes of oxygen, glucose and lactate between interior and exterior of cells by e_1, e_2, e_3. We begin with the ODE part,

$$D_k \frac{d^2}{dz^2} C_k(z) = -e_k(z), k = 1, 2, 3, \tag{5.1}$$

with $z \in (0, L)$ and

$$\begin{aligned} e_1(z) &= -\beta_1(z) - \beta_3(z), \\ e_2(z) &= -\beta_1(z) - \beta_2(z), \\ e_3(z) &= \beta_2(z) - \beta_3(z), \end{aligned} \tag{5.2}$$

with boundary conditions

$$C_1|_{z=0} = 0, \quad C_2|_{z=0} = C_2^0, \quad C_3|_{z=0} = 0, \tag{5.3}$$

$$C_1|_{z=L} = C_1^L, \quad \frac{\partial C_2}{\partial z}|_{z=L} = 0, \quad \frac{\partial C_3}{\partial z}|_{z=L} = 0. \tag{5.4}$$

The boundary conditions are representations of the biofilm growing on top of an agar base made of glucose with exposure from above to oxygen. For the concentration of oxygen, C_1 at $z = 0$ being 0 means that the oxygen concentration at the bottom of the biofilm is zero, and C_1 at $z = L$ being C_1^L means that the oxygen concentration on the top of the biofilm is equal to the air concentration. For the concentration of glucose C_2 at $z = 0$ being C_2^L means that the glucose concentration at the bottom of the biofilm is equal to the glucose nutrient base concentration, and the derivative of C_2 at $z = L$ being 0 means that the glucose cannot be exchanged between the biofilm and the air. For the concentration of lactate C_3 at $z = 0$ being 0 means that all lactate is consumed at the bottom of biofilm, and derivative of C_3 at $z = L$ being 0 means that the lactate cannot enter from biofilm to air.

The optimization part can be summarized as below. We maximize for each of the z's as follows,

$$\max_{\beta_1(z),\beta_2(z),\beta_3(z)} Y_{\text{Glu}}\beta_1(z) + Y_{\text{Ferm}}\beta_2(z) + Y_{\text{Lac}}\beta_3(z). \tag{5.5}$$

with the constraints

$$\beta_j(z) \geq 0, \quad j = 1, 2, 3,$$

and $\beta' s$ satisfy the constraints

$$-e_k(z) \leq \frac{\eta_k C_k(z)}{K_k + C_k(z)}, \quad k = 1, 2, 3, \tag{5.6}$$

The inflow of the fluxes is bounded by the right hand side of equation (5.6). The β's are flux amplitudes produced by each of three metabolites, i.e., glucose full respiration, lactate full respiration and glucose fermentation. The optimization goal is to maximize the total biomass production rate of the three metabolites in the biofilm system with the inflow bounds and nonnegative flux amplitudes of each of the metabolisms.

The spatial domain $[0, L]$ is discretized into n sub-intervals of size L/n. Denote

$$\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \boldsymbol{\beta}_3^T)^T,$$
$$\mathbf{C} = (\mathbf{C}_1^T, \mathbf{C}_2^T, \mathbf{C}_3^T)^T.$$
$$\boldsymbol{\beta}_j = (\beta_{j,1}, \ldots, \beta_{j,n})^T, 1 \leq j \leq 3,$$
$$\boldsymbol{C}_k = (C_{k,1}, \ldots, C_{k,n})^T, 1 \leq k \leq 3.$$

A given flux $\boldsymbol{\beta}$ determines the right hand side of the differential equation (5.1) through (5.2), and the corresponding concentration profiles $\boldsymbol{C}$ are obtained by solving the differential equation (5.1). The mapping from $\boldsymbol{\beta}$ to $\boldsymbol{C}$ is denoted by a function $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{\beta})$.

We will use the notations of the following formulas. There are n objective functions, one for each interval. Denote

$$\vec{\beta_i} = (\beta_{1,i}, \beta_{2,i}, \beta_{3,i})^T,$$

which is the flux vector $\boldsymbol{\beta}$ on the i-th sub-interval.

The optimization problem can be equivalently described on the i-th subinterval by

$$\max_{\beta_{1,i}(z), \beta_{2,i}(z), \beta_{3,i}(z)} Y_{\text{Glu}}\beta_{1,i}(z) + Y_{\text{Ferm}}\beta_{2,i}(z) + Y_{\text{Lac}}\beta_{3,i}(z). \tag{5.7}$$

with the constraints

$$\beta_j(z) \geq 0, \quad j = 1, 2, 3,$$

and $\beta's$ satisfying the constraints

$$A \cdot \vec{\beta_i} \leq B_i(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})), \quad k = 1, 2, 3, \tag{5.8}$$

where

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix},$$

$$B_i(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})) = \begin{pmatrix} B_{1,i} \\ B_{2,i} \\ B_{3,i} \end{pmatrix} = \begin{pmatrix} \frac{\eta_1 C_{1,i}}{K_1 + C_{1,i}} \\ \frac{\eta_2 C_{2,i}}{K_2 + C_{2,i}} \\ \frac{\eta_3 C_{3,i}}{K_3 + C_{3,i}} \end{pmatrix},$$

The optimization step involves looping over all sub-intervals, and we used an alternating scheme to solve this, namely we solve for the ODE problem first and the optimization problem second, using the result to solve a new ODE and continue alternating between the ODE problem and the optimization problem:

Initialization:

Pick an initial $\boldsymbol{\beta}^{(0)}$

Let $\tilde{\boldsymbol{\beta}}^{(0)} = \boldsymbol{\beta}^{(k)}$

Solve for $\boldsymbol{C}(\boldsymbol{\beta}^{(k)})$ from the ODE problem

for $i = 1$ *to* n *(loop over subintervals)* **do**

Solve the optimization problem

$\max_{\beta_{1,i}, \beta_{2,i}, \beta_{3,i}} Y_{GR}\beta_{1,i} + Y_{GF}\beta_{2,i} + Y_{LR}\beta_{3,i}$

subject to $A \cdot \vec{\beta}_i \leq B_i(\boldsymbol{\beta}^{(k)}, \boldsymbol{C}(\boldsymbol{\beta}^{(k)})), \vec{\beta}_i \geq 0$

set $\tilde{\boldsymbol{\beta}}^{(i)}$ as $\tilde{\boldsymbol{\beta}}^{(i-1)}$ with i-th component replaced by $\vec{\beta}_i$.

end

$\boldsymbol{\beta}^{(k+1)} = \tilde{\boldsymbol{\beta}}^{(n)}$

Do until $\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\| < 10^{-6}\|\boldsymbol{\beta}^{(k)}\|$

Theorem 5.1 *(Contraction-Mapping Theorem) [46] Suppose that* $G : D \subset \mathbb{R}^n \to \mathbb{R}^n$ *maps a closed set* $D_0 \subset D$ *into itself and that*

$$\|Gx - Gy\| \leq \alpha\|x - y\|, x, y \in D_0,$$

for some $\alpha < 1$. *Then, for any* $x^0 \in D_0$, *the sequence* $x^{(k+1)} = Gx^k, k = 0, 1, \ldots,$ *converges to the unique fixed point* x^* *of* G *in* D_0 *and*

$$\|x^k - x^*\| \leq [\alpha/(1-\alpha)]\|x^k - x^{k-1}\|, k = 1, 2,$$

Proof. See 12.2.1 in [46].

Theorem 5.2 *If*

$$\frac{\eta_{\max} L^2}{8} < 1,$$

where $\eta_{\max} = \max\{\eta_1, \eta_2, \eta_3\}$ *and* L *is the right boundary of the domain* $[0, L]$, *consider the operator* $\vec{B} : \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ *in (5.9):*

$$\vec{B}^{(k)}(\boldsymbol{\beta}) = (B_1(\boldsymbol{\beta}^{(k)}, \boldsymbol{C}(\boldsymbol{\beta}^{(k)}))^T, \ldots, B_n(\boldsymbol{\beta}^{(k)}, \boldsymbol{C}(\boldsymbol{\beta}^{(k)}))^T)^T$$

with

$$B_i(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})) = (B_{1,i}(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})), B_{2,i}(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})), B_{3,i}(\boldsymbol{\beta}), \boldsymbol{C}(\boldsymbol{\beta}))^T.$$

Then

$$\|\vec{B}^{(k+1)} - \vec{B}^{(k)}\| < \|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\|$$

and the iteration converges as $k \to \infty$.

Proof. By (5.4), we know

$$\begin{aligned} B_i(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})) &= (B_{1,i}(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})), B_{2,i}(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta})), B_{3,i}(\boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{\beta}))^T \\ &= \left(\frac{\eta_1 [g(-e_1)]_i}{K_1 + [g(-e_1)]_i}, \frac{\eta_2 [g(-e_2)]_i}{K_2 + [g(-e_2)]_i}, \frac{\eta_3 [g(-e_3)]_i}{K_3 + [g(-e_3)]_i} \right)^T \end{aligned}$$

where

$$g(-e_k) = \int_0^L G_k(z, x)(-e_k(x))dx, k = 1, 2, 3$$

and $G_k(z, x)$ is the corresponding Green's function for the differential equation. Furthermore, the notation $[g(-e_k)]_i$ is the $g(-e_k)$ evaluated at the i-th discretization point $z_i = iL/n$. From (5.2), we know that

$$\begin{aligned} |e_1^{(k+1)} - e_1^{(k)}| &= | - \beta_1^{(k+1)}(z) - \beta_3^{(k+1)}(z) + \beta_1^{(k)}(z) + \beta_3^{(k)}(z)| \\ &\leq |\beta_1^{(k+1)} - \beta_1(k)| + |\beta_3^{(k+1)} - \beta_3^{(k)}| \end{aligned}$$

Similarly we can have

$$|e_2^{(k+1)} - e_2^{(k)}| \leq |\beta_1^{(k+1)} - \beta_1^{(k)}| + |\beta_2^{(k+1)} - \beta_3^{(k)}|$$

$$|e_3^{(k+1)} - e_3^{(k)}| \leq |\beta_2^{(k+1)} - \beta_1^{(k)}| + |\beta_3^{(k+1)} - \beta_3^{(k)}|$$

Thus

$$|e_j^{(k+1} - e_j^{(k)}| \leq 2\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\|. \tag{5.9}$$

We have under maximum norm,

$$\begin{aligned}
\|\vec{B}^{(k+1)} - \vec{B}^{(k)}\| &= \max_i \|B_i(\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)})\| \\
&\leq \max_{1\leq i\leq n} \max_{1\leq j\leq 3} \|B_{j,i}(\boldsymbol{\beta}^{(k+1)}) - B_{j,i}(\boldsymbol{\beta}^{(k)})\| \\
&= \max_{1\leq i\leq n} \max_{1\leq j\leq 3} \left(\frac{\eta_j [g(-e_j^{(k+1)})]_i}{K_j + [g(-e_j^{(k+1)})]_i} - \frac{\eta_j [g(-e_j^{(k)})]_i}{K_j + [g(-e_j^{(k)})]_i} \right) \\
&= \max_{1\leq i\leq n} \max_{1\leq j\leq 3} \frac{k_j \eta_j \left([g(-e_j^{(k+1)})]_i - [g(-e_j^{(k)})]_i \right)}{\left(K_j + [g(-e_j^{(k+1)})]_i \right) \left(K_j + [g(-e_j^{(k)})]_i \right)} \\
&\leq \max_{1\leq i\leq n} \max_{1\leq j\leq 3} \left(\frac{\eta_j}{2} \left([g(-e_j^{(k+1)})]_i - [g(-e_j^{(k)})]_i \right) \right) \\
&\leq \max_{z\in\mathbb{R}} \max_{1\leq j\leq 3} \frac{\eta_j}{2} \left(\int_0^L G_j(z,x)(-e_j^{(k+1)}(x))dx - \int_0^L G_k(z,x)(-e_j^{(k)}(x))dx \right) \\
&\leq \frac{\eta_{\max}}{2} \max_{z\in\mathbb{R}} \max_{1\leq j\leq 3} \int_0^L |G_j(z,x)| \left| -e_j^{(k+1)}(x) + e_j^{(k)}(x) \right| dx \\
&\leq \frac{\eta_{\max}}{2} \left(\max_{z\in\mathbb{R}} \max_{1\leq j\leq 3} \int_0^L |G_j(z,x)|\, dx \right) 2\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\| \quad \text{by(5.9)} \\
&= \alpha \|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\|
\end{aligned}$$

where

$$\alpha = \eta_{\max} \left(\max_{z\in\mathbb{R}} \max_{1\leq j\leq 3} \int_0^L |G_j(z,x)|\, dx \right)$$

From (2.12), we have

$$\alpha \leq \eta_{\max} \frac{L^2}{8}.$$

Thus if

$$\eta_{\max} \frac{L^2}{8} < 1,$$

then $\alpha < 1$ and the method converges. ■

Figure 5.1 below is the result for glucose, oxygen and lactate concentration density with respect to biofilm thickness. The numerical result in Figure 5.1

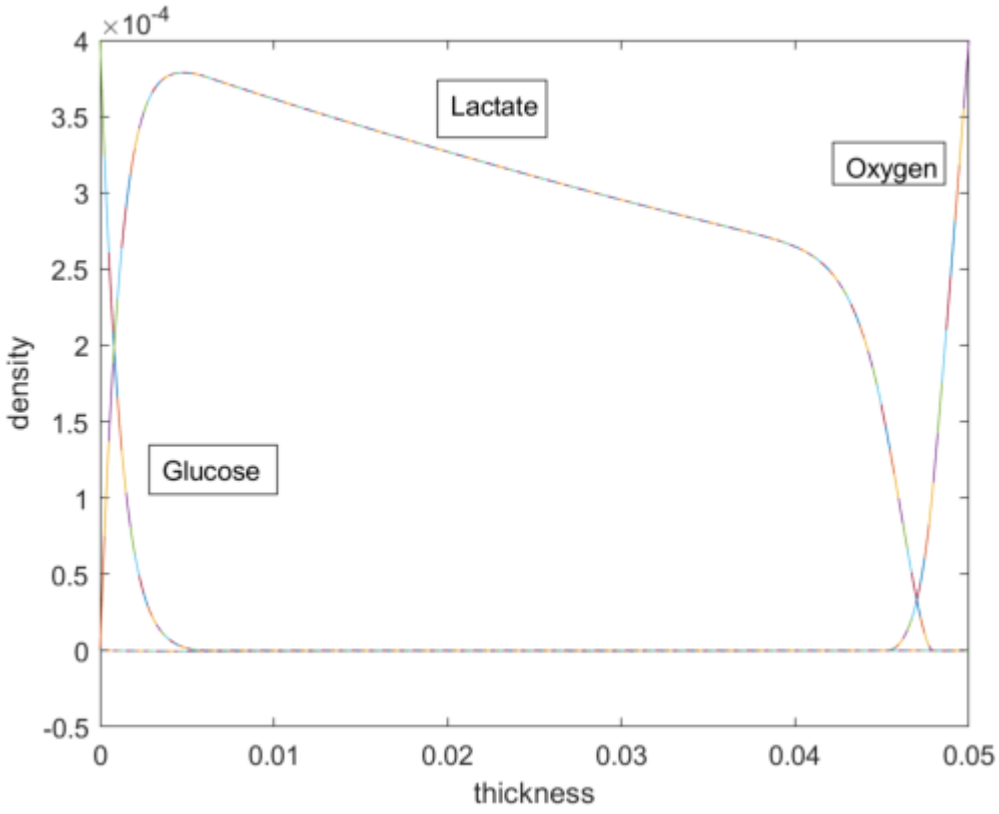

Figure 5.1: Glucose, Oxygen and Lactate concentration density as computed by 100 subdomains

is obtained with a MATLAB 2019a implementation run on a DELL XPS 15 9570 with Windows 10 64-bit, an Intel Core i7 processor, and 32GB of RAM. The number of subdomains $p = 100$, the number of discretization points on each subdomain $N = 10$. An initial approximation is used as follows

$$C_1(z) = C_1^L z/L, \quad C_2(z) = C_2^0, \quad C_3(z) = 0.$$

Note that we obtained different results when we did not apply the overlap domain decomposition technique, as shown in Figure 5.2, where we used domain size $p = 1$ and $N = 5000$ discretization points. One possible reason is that the system could have several different solutions satisfying the constraints. In other words, using the overlap domain decomposition technique

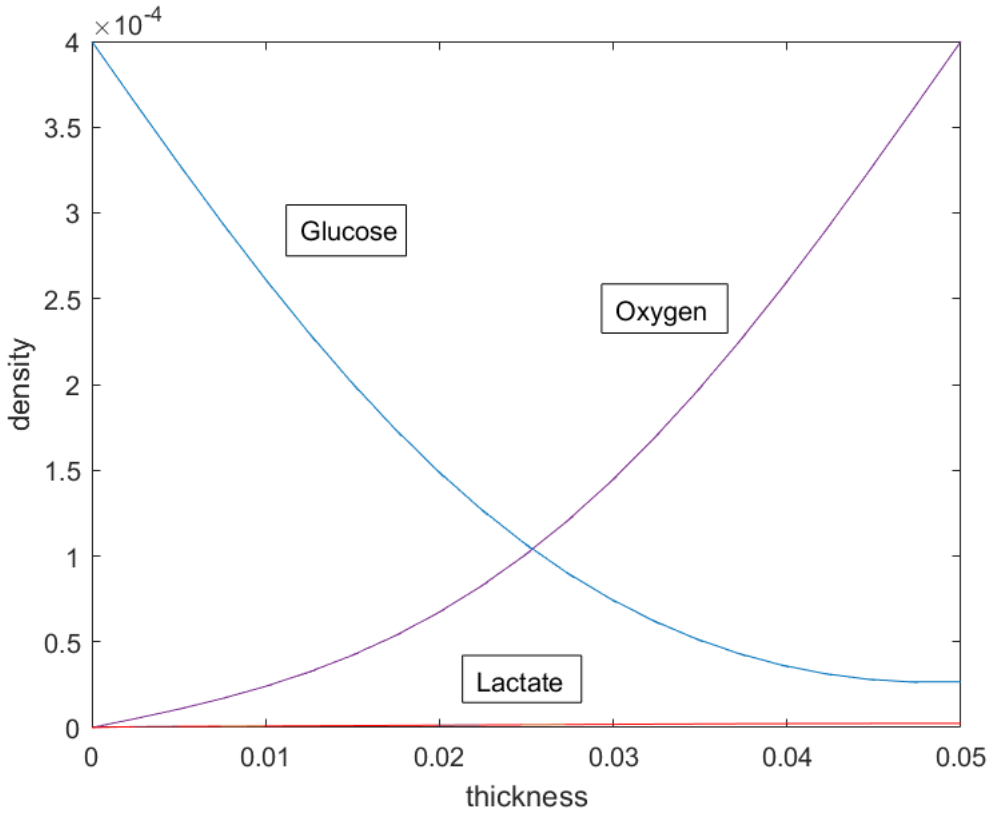

Figure 5.2: Glucose, Oxygen and Lactate concentration density without overlap domain decomposition

can find the solution where the slope of the particular solution is very steep on a large domain as is discussed in previous chapters.

The underlying biological meaning of Figure 5.1 is that at the bottom of the biofilm, when the oxygen level is low, glucose fermentation produces lactate and biomass without oxygen. Then the lactate produced by glucose fermentation diffuses to the upper part of the biofilm. As the oxygen concentration gradually increases in the upper part of the biofilm, the lactate respiration produces biomass with small amount of oxygen . At the top, when the oxygen level is high, all the lactate remaining are respired to produce biomass.

5.3 Computational Method

In this section, we discuss some computational methods for solving the problem of the just described biofilm model.

We will consider the problem in two parts, the first part is the ODE part and the second part is the optimization part.

ODE part:

The ODE in the biofilm optimization constrained ODE problem is described in the equation (5.1) and the boundary conditions (5.3), (5.4).

To solve this problem, first the spatial domain $[0, L]$ is divided into p overlapping subdomains with equal length as in Figure 3.2. Each one of the subdomains is solved with Dirichlet boundary data taken from the neighboring subdomain in the previous step.

Dirichlet boundary condition in two subdomains case

Suppose $\hat{C}_k, k = 1, 2, 3$, is the true solution for the ODE problem (5.5). Consider the error function $e_k = C_k - \hat{C}_k$. We have the domain $(0, L)$ is decomposed into two subdomains $\Omega_1 = (0, r_1)$ and $\Omega_2 = (l_2, L)$ with $l_2 < L$. Let $e^1_{k(n)}$ and $e^2_{k(n)}$ denote the error in the first and second subdomains at the n-th iteration for $k = 1, 2, 3$. The errors satisfy

$$\begin{aligned} \frac{d^2 e^1_{k(n+1)}}{dz^2} &= 0 \\ e^1_{k(n+1)}(0) &= 0 \\ e^1_{k(n+1)}(r_1) &= e^2_{k(n)}(r_1). \end{aligned}$$

and

$$\begin{aligned} \frac{d^2 e^2_{k(n+1)}}{dz^2} &= 0 \\ e^2_{k(n+1)}(l_2) &= e^1_{k(n)}(l_2) \\ e^2_{k(n+1)}(L) &= 0. \end{aligned}$$

Then we get

$$e^1_{k(n+1)}(z) = e^2_{k(n)}(r_1)\frac{z}{r_1}$$

$$e^2_{k(n+1)}(z) = e^1_{k(n)}(l_2)\frac{L-z}{L-l_2}$$

Thus we have

$$e^2_{k(n+1)}(r_1) = e^2_{k(n-1)}(r_1)\frac{L-r_1}{L-l_2}\frac{l_2}{r_1}$$

Let $\delta = r_1 - l_2$ denotes the size of the overlap, we have

$$e^2_{k(n+1)}(r_1) = e^2_{k(n-1)}(r_1)\frac{L-r_1}{L-l_2}\frac{l_2}{r_1} = \frac{1-\delta/(L-l_2)}{1+\delta/l_2}e^2_{k(n-1)}(r_1)$$

Thus if the overlap $\delta > 0$ then we have

$$\frac{1-\delta/(L-l_2)}{1+\delta/l_2} < 1.$$

Then the Dirichlet boundary condition in the neighboring subdomains makes the solution converge to the true solution, which is continuous and differentiable.

Now we consider p subdomains. Let C^j_k denotes the k-th concentration on the j-th subdomain, then we consider the following problems on each subdomain on p subdomains.

ODE for the first subdomain: Let C^1_k denotes the k-th concentration on the first subdomain and C^2_k denotes the k-th concentration on the second subdomain. The ODE in the biofilm optimization constrained ODE problem we are solving on the first subdomain is

$$D_k\frac{d^2}{dz^2}C^1_k(z) = -e^1_k(z), k = 1, 2, 3, \tag{5.10}$$

with boundary conditions

$$C^1_k|_{z=0} = 0,$$

$$C^1_k|_{z=\hat{N}+1} = C^2_k|_{z=\hat{N}+1},$$

$$k = 1, 2, 3.$$

ODE for the j-th subdomain with $1 < j < p$: Let C^j_k denotes the k-th concentration on the j-th subdomain where $1 < j < p$. The ODE in

the biofilm optimization constrained ODE problem we are solving on the j-th subdomain is

$$D_k \frac{d^2}{dz^2} C_k^j(z) = -e_k^j(z), k = 1, 2, 3, \tag{5.11}$$

with boundary conditions

$$C_k^j|_{z=(j-1)\hat{N}} = C_k^{j-1}|_{z=(j-1)\hat{N}},$$
$$C_k^j|_{z=j\hat{N}+1} = C_k^{j+1}|_{z=j\hat{N}+1},$$
$$k = 1, 2, 3,$$

i.e., each one of the subdomains is solved with Dirichlet boundary data taken from the neighboring subdomain in the previous step.

ODE for the p-th subdomain:

Let C_k^p denotes the k-th concentration on the p-th subdomain and C_k^{p-1} denotes the k-th concentration on the $(p-1)$-th subdomain. The ODE in the biofilm optimization constrained ODE problem we are solving on the last subdomain is

$$D_k \frac{d^2}{dz^2} C_k^p(z) = -e_k^p(z), k = 1, 2, 3, \tag{5.12}$$

with boundary conditions

$$C_k^p|_{z=(p-1)\hat{N}} = C_k^{p-1}|_{z=(p-1)\hat{N}}, k = 1, 2, 3,$$
$$C_1^p|_{z=p\hat{N}+1} = C_1^L, \frac{\partial C_2^p}{\partial z}|_{z=p\hat{N}+1} = 0, \frac{\partial C_3^p}{\partial z}|_{z=p\hat{N}+1} = 0.$$

Finite Difference Method for the ODE:

Below we discuss the finite difference method, note this is an approximations to the true ODE solution [4, 19, 23, 38, 49, 65, 67].

The spatial domain $[0, L]$ is divided into p overlapping subdomains with equal length and finite difference method is employed for the ODE problem on each spatial subdomain. To apply finite differences, each spatial subdomain is divided into $\hat{N}$ sub-intervals.

Recall that second-order center-difference approximations of $\frac{d^2}{dz^2}C_k(z)$ at grid point i is

$$\frac{d^2}{dz^2}C_k(z_i) = \frac{C_k(z_{i+1}) - 2C_k(z_i) + C_k(z_{i-1})}{\Delta z^2} + O(\Delta z^2).$$

Replacing $\frac{d^2}{dz^2}C_k(z)$ by the second-order centered-difference approximation and evaluating all terms at interior grid point i gives

$$\frac{C_k(z_{i+1}) - 2C_k(z_i) + C_k(z_{i-1})}{\Delta z^2} + O(\Delta z^2) = -e_k(z_i), \quad i = 2, \ldots, p-1.$$

We will use Dirichlet boundary conditions for the neighboring subdomains.

Finite difference method of the ODE part for the first subdomain:

For the approximation solution on the first subdomain, the interior finite difference is

$$\frac{C_k^1(z_{i+1}) - 2C_k^1(z_i) + C_k^1(z_{i-1})}{\Delta z^2} = -e_k^1(z_i)$$

where $1 \le i \le \hat{N}$. Since the Dirichlet boundary condition is given on the left end boundary is when $i = 1$

$$\frac{C_k^1(z_2) - 2C_k^1(z_1) + C_k^1(z_0)}{\Delta z^2} = -e_k^1(z_1)$$

and $C_k^1(z_0) = 0$ is given.

Similarly, for the right end boundary condition the Dirichlet boundary condition is also given, i.e., when $i = \hat{N}$ we have

$$\frac{C_k^1(z_{\hat{N}+1}) - 2C_k^1(z_{\hat{N}}) + C_k^1(z_{\hat{N}-1})}{\Delta z^2} = -e_k^1(z_1)$$

and $C_k^1(z_{\hat{N}+1}) = C_k^2(z_{\hat{N}+1})$ is given.

Finite difference method of the ODE part for the j-th subdomain with $1 < j < p$:

We discuss approximation solution on the j-th subdomain with $1 < j < p$. The interior finite difference is

$$\frac{C_k^j(z_{i+1}) - 2C_k^j(z_i) + C_k^j(z_{i-1})}{\Delta z^2} = -e_k^j(z_i)$$

where $(j-1)\hat{N}+1 \leq i \leq j\hat{N}$ for each subdomain. Since the Dirichlet boundary condition is given on the left end boundary is when $i = (j-1)\hat{N}+1$

$$\frac{C_k^j(z_{(j-1)\hat{N}+2}) - 2C_k^j(z_{(j-1)\hat{N}+1}) + C_k^j(z_{(j-1)\hat{N}})}{\Delta z^2} = -e_k^j(z_{(j-1)\hat{N}+1})$$

and $C_k^j(z_{(j-1)\hat{N}}) = C_k^{j-1}(z_{(j-1)\hat{N}})$ is given.

Similarly, for the right end boundary condition the Dirichlet boundary condition is also given, i.e., when $i = j\hat{N}$ we have

$$\frac{C_k^j(z_{j\hat{N}+1}) - 2C_k^j(z_{j\hat{N}}) + C_k^j(z_{j\hat{N}-1})}{\Delta z^2} = -e_k^j(z_{j\hat{N}})$$

and $C_k^j(z_{j\hat{N}+1}) = C_k^{j+1}(z_{j\hat{N}+1})$ is given.

Finite difference method of the ODE part for the p-th subdomain:

We discuss the approximation solution on the p-th subdomain, the interior finite difference is

$$\frac{C_k^p(z_{i+1}) - 2C_k^p(z_i) + C_k^p(z_{i-1})}{\Delta z^2} = -e_k^p(z_i)$$

where $(p-1)\hat{N}+1 \leq i \leq p\hat{N}+1$ for the p-th subdomain. Since the Dirichlet boundary condition is given on the left end boundary is when $i = (p-1)\hat{N}+1$

$$\frac{C_k^p(z_{(p-1)\hat{N}+2}) - 2C_k^p(z_{(p-1)\hat{N}+1}) + C_k^p(z_{(p-1)\hat{N}})}{\Delta z^2} = -e_k^p(z_{(p-1)\hat{N}+1})$$

and $C_k^p(z_{(p-1)\hat{N}}) = C_k^{p-1}(z_{(p-1)\hat{N}})$ is given.

For the Neumann boundary condition on the right hand side, consider the finite difference grid in the neighborhood of point $z_{p\hat{N}+1}$, which is illustrated in Figure 5.3. In Figure 5.3 the point $z_{p\hat{N}+2}$ is outside the domain, the point $z_{p\hat{N}+1}$ is on the right boundary of the domain and the point $z_{p\hat{N}}$ is inside the domain. To evaluate the second order centered-difference approximation at the right boundary point we need to use the point $z_{p\hat{N}+2}$ outside the domain.

The second order centered-difference approximation at boundary point $z_{p\hat{N}+1}$ is given by the following equation evaluated at $i = p\hat{N}+1$

$$\frac{C_k^p(z_{p\hat{N}+2}) - 2C_k^p(z_{p\hat{N}+1}) + C_k^p(z_{p\hat{N}})}{\Delta z^2} = -e_k^p(z_{p\hat{N}+1}),$$

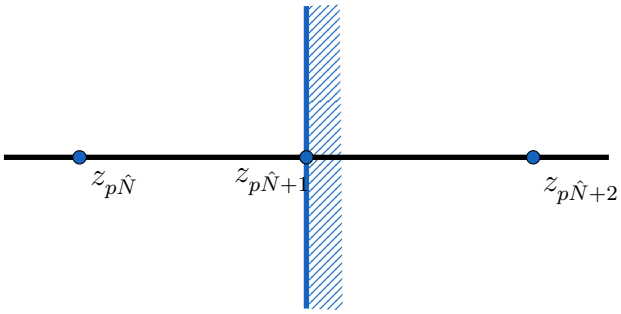

Figure 5.3: Right boundary condition discretization.

where the point $z_{p\hat{N}+2}$ is outside of the solution domain. The value of $z_{p\hat{N}+2}$ is unknown. It can be approximated by expressing the derivative boundary condition at point $z_{p\hat{N}+1}$ in finite difference form as follows:

$$\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1}) = \frac{C_k^p(z_{p\hat{N}+2}) - C_k^p(z_{p\hat{N}})}{2\Delta z} + O(\Delta z^2).$$

Solving for $C_k^p(z_{p\hat{N}+2})$ gives

$$C_k^p(z_{p\hat{N}+2}) = C_k^p(z_{p\hat{N}}) + 2\Delta z\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1}) + \Delta z O(\Delta z^2)$$

Truncating the remainder term yields an expression for $C_k^p(z_{p\hat{N}+2})$:

$$C_k^p(z_{p\hat{N}+2}) = C_k^p(z_{p\hat{N}}) + 2\Delta z\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1})$$

Substituting and simplifying yields the desired finite difference

$$\frac{C_k^p(z_{p\hat{N}}) + 2\Delta z\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1}) - 2C_k^p(z_{p\hat{N}+1}) + C_k^p(z_{p\hat{N}})}{\Delta z^2} = -e_k^p(z_{p\hat{N}+1}),$$

where $\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1})$ is already given, thus we have

$$2C_k^p(z_{p\hat{N}}) - 2C_k^p(z_{p\hat{N}+1}) = -\Delta z^2 e_k^p(z_{p\hat{N}+1}) - 2\Delta z\frac{d^2}{dz}C_k^p(z_{p\hat{N}+1}).$$

Optimization part:

Consider the following optimization problem for each z:

$$\max_{\beta_1(z),\beta_2(z),\beta_3(z)} Y_{GR}\beta_1(z) + Y_{GF}\beta_2(z) + Y_{LR}\beta_3(z).$$

such that

$$\begin{aligned} \beta_1(z) \geq 0, \beta_2(z) \geq 0, \beta_3(z) \geq 0 \\ \beta_1(z) + \beta_3(z) \leq \frac{\eta_1 C_1(z)}{K_1 + C_1(z)} \\ \beta_1(z) + \beta_2(z) \leq \frac{\eta_2 C_2(z)}{K_2 + C_2(z)} \\ -\beta_2(z) + \beta_3(z) \leq \frac{\eta_3 C_3(z)}{K_3 + C_3(z)} \cdot \end{aligned} \tag{5.13}$$

The optimization and ODE are related to each other by the variable $\beta_k, k = 1, 2, 3$, in the optimization and the right hand side of the ODE $e_k(z), k = 1, 2, 3$, as follows:

$$\begin{aligned} e_1(z) &= -\beta_1(z) - \beta_3(z), \\ e_2(z) &= -\beta_1(z) - \beta_2(z), \\ e_3(z) &= \beta_2(z) - \beta_3(z). \end{aligned} \tag{5.14}$$

Optimization part for j-th subdomain:

Let β_k^j denotes the parameters β_k for the j-th subdomain. Consider the following optimization problem for each z:

$$\max_{\beta_1^j(z), \beta_2^j(z), \beta_3^j(z)} Y_{GR}\beta_1^j(z) + Y_{GF}\beta_2^j(z) + Y_{LR}\beta_3^j(z).$$

such that

$$\begin{aligned} \beta_1^j(z) \geq 0, \beta_2^j(z) \geq 0, \beta_3^j(z) \geq 0 \\ \beta_1^j(z) + \beta_3^j(z) \leq \frac{\eta_1^j C_1^j(z)}{K_1 + C_1^j(z)} \\ \beta_1^j(z) + \beta_2^j(z) \leq \frac{\eta_2^j C_2^j(z)}{K_2 + C_2^j(z)} \\ -\beta_2^j(z) + \beta_3^j(z) \leq \frac{\eta_3^j C_3^j(z)}{K_3 + C_3^j(z)} \cdot \end{aligned} \tag{5.15}$$

Method for the optimization

The interior-point methods (also referred to as barrier methods) [50, 71, 72] can be used for the optimization (5.15). For simplicity, denote $f(x)$ by

$$f(x) = -\left(Y_{GR}x_1 + Y_{GF}x_2 + Y_{LR}x_3\right)$$

and denote the constraints $c_i, i = 1, \ldots, 6$ by

$$c_1 = x_1 \quad c_2 = x_2 \quad c_3 = x_3$$

$$c_4 = \frac{\eta_1^j C_1^j(z)}{K_1 + C_1^j(z)} - x_1 - x_3$$

$$c_5 = \frac{\eta_2^j C_2^j(z)}{K_2 + C_2^j(z)} - x_1 - x_2$$

$$c_6 = \frac{\eta_3^j C_3^j(z)}{K_3 + C_3^j(z)} + x_2 - x_3.$$

The optimization (5.15) is equivalent to the following optimization.

$$\min f(x) \quad \text{subject to} \quad c_i(x) \geq 0 \quad \text{for} \quad i = 1, \ldots, 6. \tag{5.16}$$

First the logarithmic barrier function associated with the optimization (5.16) is

$$B(x, \mu) = f(x) - \mu \sum_{i=1}^{6} \log(c_i(x)). \tag{5.17}$$

Here μ is a small positive scalar, sometimes called the "barrier parameter". As μ converges to zero the minimum of $B(z, \mu)$ should converge to a solution of (5.16). The barrier function gradient is

$$g_b = g - \mu \sum_{i=1}^{6} \frac{1}{c_i(z)} \nabla c_i(x) \tag{5.18}$$

where g is the gradient of $f(x)$, and ∇c_i is the gradient of c_i.

In addition to the original ("primal") variable z we introduce a Lagrange multiplier inspired dual variable λ

$$c_i(x)\lambda_i = \mu, \quad i = 1, \ldots, 6. \tag{5.19}$$

We try to find those (x_μ, λ_μ) for which the gradient of the barrier function is zero. Applying (5.19) to (5.18), we get an equation for the gradient:

$$g - A^T \lambda = 0, \tag{5.20}$$

where the matrix A is the Jacobian of the constraints $c(x)$.

The intuition behind (5.20) is that the gradient of $f(x)$ should lie in the subspace spanned by the constraints' gradients.

Applying Newton's method to (5.19) and (5.20), we get an equation for (x, λ) update (p_x, p_λ). Because of (5.16), (5.19) the condition $\lambda \geq 0$ should be enforced at each step. This is done by choosing appropriate α:

$$(x, \lambda) \to (x + \alpha p_x, \lambda + \alpha p_\lambda).$$

Relation between optimization and ODE:

On each of the subdomain, the optimization and ODE are related to each other by the variable $\beta_k^j, k = 1, 2, 3$ in the optimization and the right hand side of the ODE $e_k^j(z), k = 1, 2, 3$ as follows:

$$\begin{aligned} e_1^j(z) &= -\beta_1^j(z) - \beta_3^j(z), \\ e_2^j(z) &= -\beta_1^j(z) - \beta_2^j(z), \\ e_3^j(z) &= \beta_2^j(z) - \beta_3^j(z). \end{aligned} \tag{5.21}$$

5.4 A Second Biofilm Model Example

In this model, we consider two types of bacteria, *Staphylococcus aureus (S. aureus)* and *Staphylococcus epidermidis (S. epidermidis). S. epidermidis* is a Gram-positive bacterium, and one of over 40 species belonging to the genus Staphylococcus. It is part of the normal human flora, typically the skin flora, and less commonly the mucosal flora. It is a facultative anaerobic bacteria. *S. aureus* is a Gram-positive, round-shaped bacterium that is a member of the Firmicutes, and it is a usual member of the microbiota of the body, frequently found in the upper respiratory tract and on the skin.

Consider the following model

$$\frac{d^2 C_i(z,t)}{dz^2} = \frac{\eta C_i(z,t)}{K + C_i(z,t)} B_i(z,t), \quad i = 1, 2 \tag{5.22}$$

with boundary conditions

$$C_i(0,t) = 0, \quad i = 1, 2$$

$$C_i(10^{-3}, t) = 4 \times 10^{-4}, \quad i = 1, 2,$$

where C_1 represents the oxygen concentration for *S. aureus*, C_2 represents the oxygen concentration *S. epidermidis*, B_1 represents the population of *S. aureus* and B_2 represents the population of *S. epidermidis*.

Additionally, the bacteria population $B_i(z, t)$ satisfies

$$\frac{dB_i(z,t)}{dt} = r_i(z,t)B_i(z,t), \quad i = 1, 2 \tag{5.23}$$

with initial conditions

$$B_i(z, 0) = 1, \quad i = 1, 2$$

where r_1 is the growth rate of *S. aureus* and r_2 is the growth rate of *S. epidermidis*. The growth rates $r_1(z, t), r_2(z, t)$ are not fixed parameters like η, K, instead the growth rates r_1, r_2 are derived from the open source optimization software COnstraints Based Reconstruction and Analysis (COBRA)[1].

The openCOBRA project is an open preject which provides a repository for people to contribute directly to the open project. The openCOBRA project also makes it easier for people to access to the many core features of COBRA. The openCOBRA project can be used in Matlab, Python, Julia and some other languages.

For this example, we use flux balance analysis (FBA), which we briefly introduce next; and in fact, we use the FBA function of the COBRA software. FBA is a mathematical approach for analyzing the flow of metabolites through a metabolic network [47]. Kinetic parameters and concentration of the metabolites in the system are rarely required in flux balance analysis. FBA is based on two assumptions, one is that the system is in a steady state and the concentrations stay the same, and the other one is the system is optimized for a biological goal because of the evolution, for example, maximizing the growth rate or minimizing resource consumption rate. Concentration changes are represented as the product of the stoichiometric matrix S and the vector of unknown fluxes v. The product is set to be zero to represent that the

[1]The openCOBRA project, Available from: `https://opencobra.github.io/`

changes are zero and the system is steady. Linear programming is then used to calculate a solution of fluxes corresponding to the steady state [37].

The linear programming (LP) problem (5.24) below can be used to estimate the vector of unknown fluxes,

$$\begin{aligned} \min_{v} \quad & c^T v \\ \text{s.t.} \quad & Sv = 0 \\ & l \leq v \leq u \end{aligned} \tag{5.24}$$

where $c \in \mathbb{R}^n$ is a vector and $c^T v$ is optimized based on the opitmization assumption of FBA. $S \in \mathbb{R}^{m \times n}$ is a stoichiometric matrix, where m is the number of molecular species and n is the number of reactions. $Sv = 0$ means the system is in a balance state. The FBA consists of finding a v that optimizes the objective function while satisfying all the constraints. The u represents the upper bound and l represents the lower bound [47].

FBA can find an optimal flux vector v that both satisfy the optimization and the steady state assumptions we introduced earlier. We need to set a upper bound and lower bound for the flux vector in (5.24).

Calculating growth rate under aerobic conditions by COBRA:

The preparation of a metabolic network for a FBA process needs lots of work and can take months or years. The aerobic model metabolic network is provided by courtesy of Dr. Cristal Zuniga from the University of California San Diego.

First we need to rename the model to avoid confusion. Rename the model also makes it easier for future references. We are considering the aerobic model, so we rename the model as "modelaerobic".

```
modelaerobic = model;
```

The 'changeRxnBounds' function changes the flux constraints of the lower ('l'), upper ('u'), or both the bounds ('b'), of the specified reaction. Here, we will change the maximal uptake of oxygen to $\min\left(0, -\frac{\eta_i C_i(z,t)}{K_i + C_i(z,t)}\right)$ and denote it by 'flux'.

```
modelaerobic =
changeRxnBounds(modelaerobic,'EX_o2_[smp]',flux,'l');
```

The function 'optimizeCbModel' can be used to maximize or minimize for a objective reaction by specifying the optimization type parameter and it will return a solution, and the solution optimizes the system while satisfying the constraints.

```
FBAaerobic = optimizeCbModel(modelaerobic, 'max')
```

We then can get the growth rates of the two types of bacteria by using

```
Biomass = find(modelaerobic.c);
Growthrate = FBAaerobic.x(Biomass);
```

After we get the growth rate from COBRA by inputting the oxygen bound for each type of bacteria, we then solve for the population for each type of bacteria and then use it get a new oxygen bound and we iterate the above process until convergence. To summarize, we solved this system using the iteration steps.

Initialization:

Let $B_i^{(0)}(z,t) = ones(m,n), \epsilon = 1e-4$

for $k = 1$ *to* ∞ **do**

  Solve for $C_i^{(k)}(z,t)$ with parameter $B_i^{(k-1)}(z,t)$

  Solve for $r_i^{(k)}(z,t)$ with parameter $C_i^{(k)}(z,t)$ (*This step is obtained from COBRA*)

  Solve for $B_i^{(k)}(z,t)$ with parameter $r_i^{(k)}(z,t)$

  if $\|B_i^{(k)}(z,t) - B_i^{(k-1)}(z,t)\| < \epsilon$ **then**

    break;

  end

end

$C_1(z,t) = C_1^{(k)}(z,t), r_1(z,t) = r_1^{(k)}(z,t), B_1(z,t) = B_1^{(k)}(z,t)$

Figure 5.4, 5.5 and 5.6, show the oxygen, logarithm of oxygen and glucose concentration on the domain, we can see the oxygen concentration increases

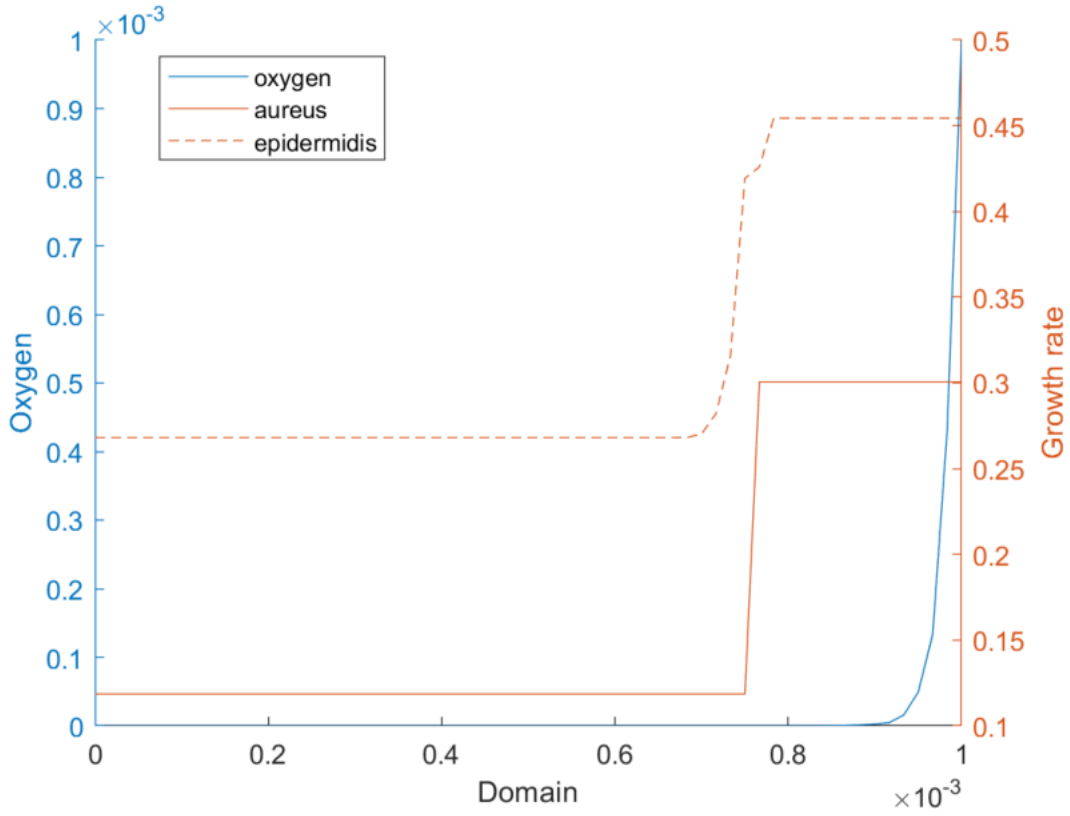

Figure 5.4: Oxygen concentration and *S.aureus* and *S. epidermidis* population

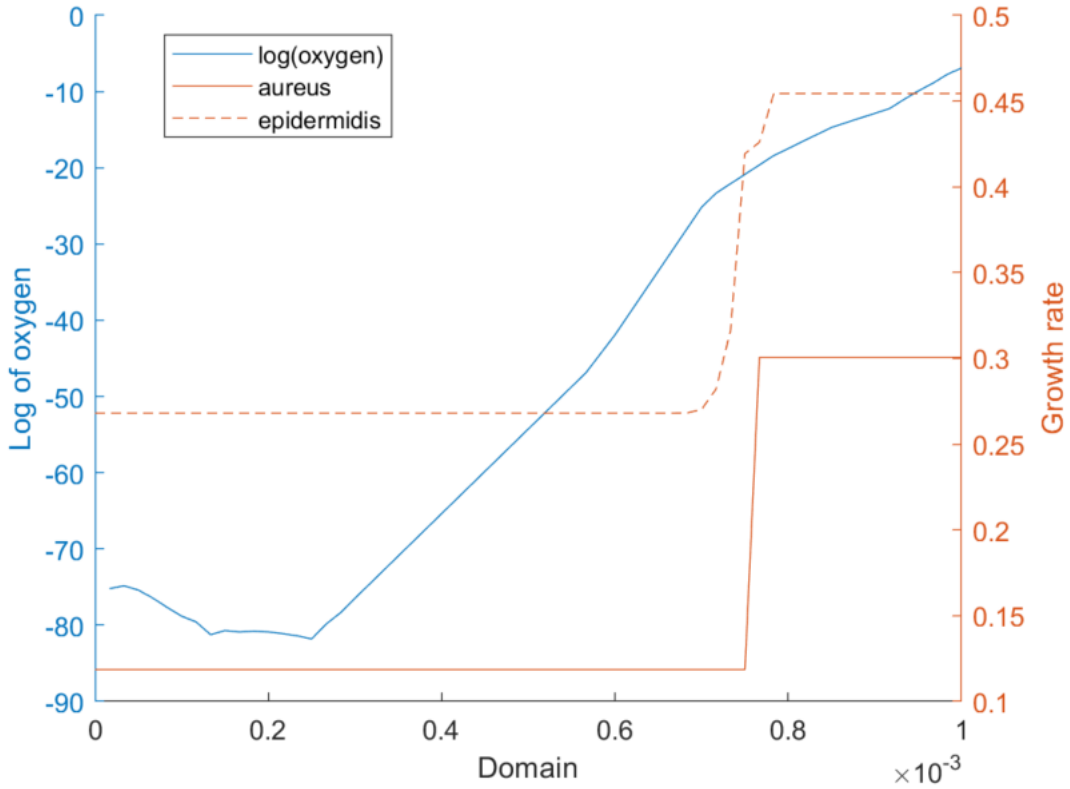

Figure 5.5: Logarithm of oxygen concentration and *S.aureus* and *S. epidermidis* population

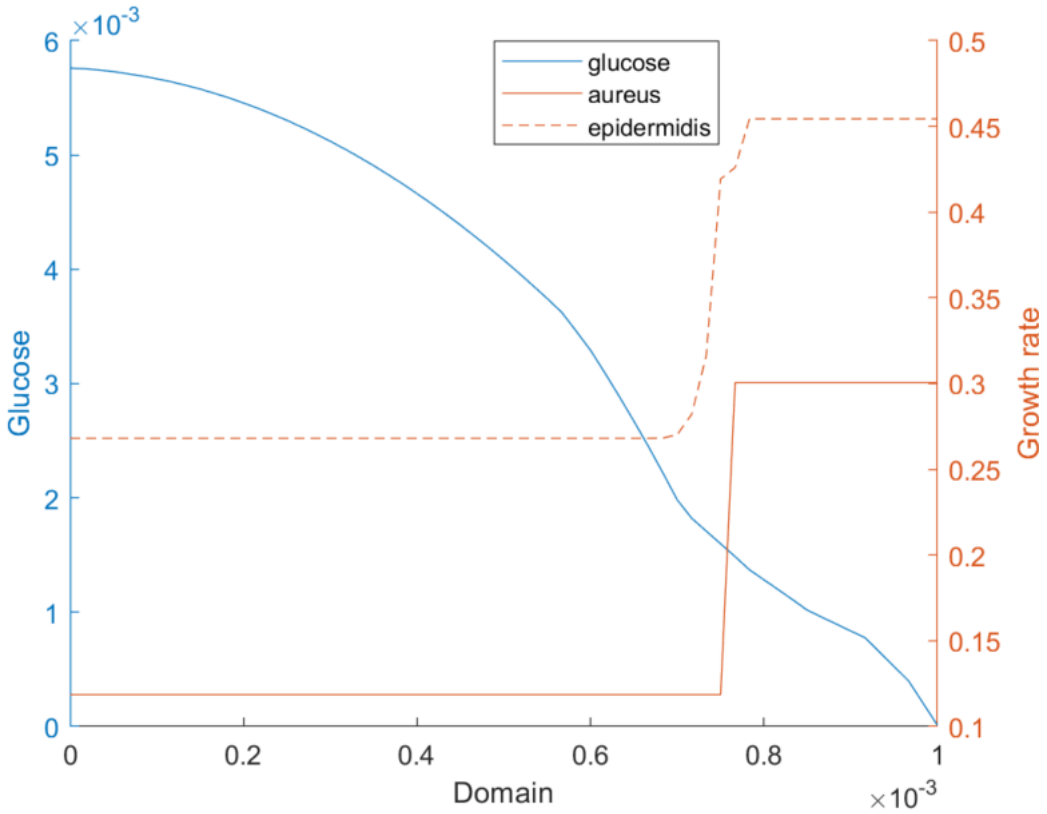

Figure 5.6: Glucose concentration and *S. aureus* and *S. epidermidis* population

and glucose decreases near the right boundary of the domain. The figures also show the population of *S. aureus* and *S. epidermidis*, we can see that the *S. epidermidis* has a larger population than *S. aureus*. We can also see there is a population increasing near the right boundary of the domain where has higher oxygen concentration and moderate glucose concentration.

5.5 Conclusion

From the numerical simulations, we can get some insight from the biofilm modeling problem. For the first model, we can see that when oxygen level is low, glucose fermentation is the major metabolic process going on. We can also see that when oxygen is high, lactate respiration is the major metabolic process

producing biomass. For the second model, we can see that the oxygen and glucose are both important factors in the biofilm growth and how the correlate with the biofilm growth. We can see from the result that the bacteria needs a proper concentration range of both oxygen and glucose for a large growth rate and oxygen is essential in the biofilm growth.

CHAPTER 6

CONCLUSION

In this thesis, we have considered the second order ODEs with optimization constraints arising in biofilm modeling. The major theoretical contribution of this thesis is that we propose a new numerical method with techniques such as domain decomposition and asynchronous iterations. We also presented theoretical conditions for the convergence of each of the techniques.

In addition to the theoretical results on the convergence of the method, we performed a variety of numerical experiments. These experiments were performed with a variety of parameters, for different domain sizes, as well as various slopes of the underlying solution, on which we ran the algorithm with and without each of the domain decomposition and asynchronous iterations techniques.

As expected from the proven theoretical results, we are able to solve bigger multi-scale problems with steeper slope than previously possible with the domain decomposition technique and solve the problem more efficiently with the asynchronous technique.

Moreover, in biofilm modeling, the ordinary differential equations are often coupled with an optimization problem, we present an alternating scheme in which we fix the value of the constraints, solve the ODE, then use this solution as data for the optimization problem, giving rise to new constraints and we repeat the process. Again, we propose conditions under which this process

converges.

As an application, we applied the algorithm to the biofilm modeling problems. From the numerical experiments, we can get biology insights like how metabolic process is affected by oxygen concentration, and how the growth of biofilm bacteria is affected by various factors such as glucose and oxygen.

We can see that, both in theory and in the considered numerical experiments, the numerical methods developed can be applied to this problem of solving second order ODEs with optimization constraints.

REFERENCES

[1] José Álvarez-Ramírez, Francisco J. Valdés-Parada, and Jesus Álvarez. A Green's function formulation for finite-differences schemes. *Chemical Engineering Science*, 62(12):3083–3091, 2007.

[2] Dganit Amitai, Amir Averbuch, Moshe Israeli, Samuel Itzikowitz, and Eli Turkel. A survey of asynchronous finite-difference methods for parabolic PDEs on multiprocessors. *Applied Numerical Mathematics*, 12(1-3):27–45, 1993.

[3] Mohamed-Naim Anwar and Mouhamed Nabih El Tarazi. Asynchronous algorithms for Poisson's equation with nonlinear boundary conditions. *Computing*, 34(2):155–168, 1985.

[4] Uri M. Ascher, Robert M. Mattheij, and Robert D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, Philadelphia, 1994.

[5] Mordecai Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[6] Erwin H. Bareiss. Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices. *Numerische Mathematik*, 13(5):404–424, 1969.

[7] Wayne W. Barrett. A theorem on inverse of tridiagonal matrices. *Linear Algebra and its Applications*, 27:211–217, 1979.

[8] György Barton. *Elements of Green's functions and propagation: potentials, diffusion, and waves*. Oxford University Press, Oxford, 1989.

[9] Gérard M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the ACM*, 25(2):226–244, 1978.

[10] Selçuk Ş. Bayin. *Mathematical Methods in Science and Engineering*. Wiley Online Library, 2006.

[11] James V. Beck, Kevin D. Cole, Abdolhossein S. Haji-Sheikh, and Bahman Litkouhl. *Heat conduction using Green's function*. Taylor & Francis, Philadelphia, 1992.

[12] Dimitri P. Bertsekas. Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27(1):107–120, 1983.

[13] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[14] Dimitri P. Bertsekas and John N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms—a survey. *Automatica*, 27(1):3–21, 1991.

[15] Seetharama M. Bhat and Debasish Ghose. Performance of parallel shooting method for closed loop guidance of an optimal launch vehicle trajectory. *Optimization and Engineering*, 1(4):399–435, 2000.

[16] Tuncer Cebeci and Herbert B. Keller. Shooting and parallel shooting methods for solving the falkner-skan boundary-layer equation. *Journal of Computational Physics*, 7(2):289–300, 1971.

[17] William G. Characklis. Fouling biofilm development: a process analysis. *Biotechnology and Bioengineering*, 102(2):310–347, 2009.

[18] Daniel Chazan and Willard Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2(2):199–222, 1969.

[19] Eusebius J. Doedel. Finite difference collocation methods for nonlinear two point boundary value problems. *SIAM Journal on Numerical Analysis*, 16(2):173–185, 1979.

[20] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An Introduction To Domain Decomposition Methods: Algorithms, Theory, And Parallel Implementation*. SIAM, Philadelphia, 2015.

[21] Dean G Duffy. *Green's functions with applications*. CRC Press, Boca Raton, FL, 2015.

[22] Mouhamed Nabih El Tarazi. Some convergence results for asynchronous algorithms. *Numerische Mathematik*, 39(3):325–340, 1982.

[23] Elsayed M. Elbarbary and Maher El-Kady. Chebyshev finite difference approximation for the boundary value problems. *Applied Mathematics and Computation*, 139(2-3):513–523, 2003.

[24] Mohamed Elouafi. On a relationship between Chebyshev polynomials and Toeplitz determinants. *Applied Mathematics and Computation*, 229:27–33, 2014.

[25] Charbel Farhat, Michel Lesoinne, Patrick LeTallec, Kendall Pierson, and Daniel Rixen. FETI-DP: a dual–primal unified FETI method—part I: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering*, 50(7):1523–1544, 2001.

[26] Christodoulos A. Floudas and Panos M. Pardalos. *Encyclopedia Of Optimization*, volume 1. Springer, New York, 2001.

[27] Andreas Frommer. Parallele asynchrone iterationen in: J. Herzberger (Ed.) Wissenschaftliches Rechnen, Akademie, Berlin, 1995, pp. 187-231 (Chapter 4).

[28] Andreas Frommer and Daniel B. Szyld. Asynchronous two-stage iterative methods. *Numerische Mathematik*, 69(2):141–153, 1994.

[29] Andreas Frommer and Daniel B. Szyld. On asynchronous iterations. *Journal Of Computational and Applied Mathematics*, 123:201–216, 2000.

[30] Michael J.C. Gover. The eigenproblem of a tridiagonal 2-Toeplitz matrix. *Linear Algebra and its Applications*, 197:63–78, 1994.

[31] Christian Grossmann, Hans-Görg Roos, and Martin Stynes. *Numerical Treatment Of Partial Differential Equations*. Springer, New York, 2007.

[32] Sung N. Ha. A nonlinear shooting method for two-point boundary value problems. *Computers & Mathematics with Applications*, 42(10-11):1411–1420, 2001.

[33] Eliseo Hernández-Martínez, Francisco J. Valdés-Parada, and José Álvarez-Ramírez. A Green's function formulation of nonlocal finite-difference schemes for reaction–diffusion equations. *Journal of Computational and Applied Mathematics*, 235(9):3096–3103, 2011.

[34] Joe D. Hoffman and Steven Frankel. *Numerical Methods For Engineers And Scientists*. CRC Press, Boca Raton, FL, 2018.

[35] Raymond W. Holsapple, Ram Venkataraman, and David Doman. New, fast numerical method for solving two-point boundary-value problems. *Journal of Guidance, Control, and Dynamics*, 27(2):301–304, 2004.

[36] Arieh Iserles. *A First Course In The Numerical Analysis Of Differential Equations*. Cambridge University Press, Cambridge, 2009.

[37] Kenneth J Kauffman, Purusharth Prakash, and Jeremy S Edwards. Advances in flux balance analysis. *Current opinion in biotechnology*, 14(5):491–496, 2003.

[38] Herbert B. Keller. *Numerical Methods For Two-point Boundary-value Problems*. Courier Dover Publications, New York, 2018.

[39] Addolorata Marasco and Antonio Romano. *Scientific Computing with Mathematica®: Mathematical Problems for Ordinary Differential Equations*. Springer, New York, 2001.

[40] Philip D Marsh. Dental plaque as a biofilm and a microbial community–implications for health and disease. In *BMC Oral health*, volume 6, page S14. BioMed Central, 2006.

[41] John C. Mason. Chebyshev polynomials of the second, third and fourth kinds in approximation, indefinite integration, and integral transforms. *Journal of Computational and Applied Mathematics*, 49(1-3):169–178, 1993.

[42] Yuri A. Melnikov. *Green's functions in applied mechanics*. Computational Mechanics Publications, Southampton, 1995.

[43] Sylvie Miquel, Rosyne Lagrafeuille, Bertrand Souweine, and Christiane Forestier. Anti-biofilm activity as a health issue. *Frontiers in microbiology*, 7:592, 2016.

[44] David D. Morrison, James D. Riley, and John F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Communications of the ACM*, 5(12):613–614, 1962.

[45] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, New York, 2006.

[46] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM, Philadelphia, 2000.

[47] Jeffrey D. Orth, Ines Thiele, and Bernhard Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245, 2010.

[48] Dan Erik Petersen. *Block tridiagonal matrices in electronic structure calculations*. PhD thesis, Dept. of Computer Science, Copenhagen University, 2008.

[49] Michael B. Porter and Edward L. Reiss. A note on the relationship between finite-difference and shooting methods for ode eigenvalue problems. *SIAM Journal on Numerical Analysis*, 23(5):1034–1039, 1986.

[50] Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000.

[51] Lothar Reichel and Lloyd N. Trefethen. Eigenvalues and pseudo-eigenvalues of Toeplitz matrices. *Linear Algebra and its Applications*, 162:153–185, 1992.

[52] Haluk Resat, Linda Petzold, and Michel F Pettigrew. Kinetic modeling of biological systems. In *Computational Systems Biology*, pages 311–335. Springer, 2009.

[53] Bruce E Rittmann and Perry L McCarty. Evaluation of steady-state-biofilm kinetics. *Biotechnology and Bioengineering*, 22(11):2359–2373, 1980.

[54] Bruce E Rittmann and Perry L McCarty. Model of steady-state-biofilm kinetics. *Biotechnology and bioengineering*, 22(11):2343–2357, 1980.

[55] Jose A Rodríguez-León, Júlio C de Carvalho, Ashok Pandey, Carlos R Soccol, and Daniel E Rodríguez-Fernández. Kinetics of the solid-state fermentation process. In *Current Developments in Biotechnology and Bioengineering*, pages 57–82. Elsevier, 2018.

[56] Andrzej P. Ruszczyński. *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, 2006.

[57] David A. Sánchez. An alternative to the shooting method for a certain class of boundary value problems. *The American Mathematical Monthly*, 108(6):552–555, 2001.

[58] Robert E. Schilson and Neal R Amundson. Intraparticle diffusion and conduction in porous catalysts—I: Single reactions. *Chemical Engineering Science*, 13(4):226–236, 1961.

[59] Jurij Silc, Borut Robic, and Theo Ungerer. Asynchrony in parallel computing: From dataflow to multithreading. *Parallel and Distributed Computing Practices*, 1(1):3–30, 1998.

[60] Gordon D. Smith. *Numerical Solution Of Partial Differential Equations: Finite Difference Methods*. Oxford University Press, Oxford, 1985.

[61] Daphne Soares and Webe J. Mansur. A time domain FEM approach based on implicit Green's functions for non-linear dynamic analysis. *International Journal for Numerical Methods in Engineering*, 62(5):664–681, 2005.

[62] Josef Stoer and Roland Bulirsch. *Introduction to Numerical Analysis*. Springer, New York, 2013.

[63] Yangfeng Su, Amit Bhaya, Eugenius Kaszkurewicz, and Victor S. Kozyakin. Further results on convergence of asynchronous linear iterations. *Linear Algebra and its Applications*, 281(1-3):11–24, 1998.

[64] Daniel B. Szyld. Different models of parallel asynchronous iterations with overlapping blocks. *Computational and Applied Mathematics*, 17:101–115, 1998.

[65] Ikram A. Tirmizi and Edward H. Twizell. Higher-order finite-difference methods for nonlinear second-order two-point boundary-value problems. *Applied Mathematics Letters*, 15(7):897–902, 2002.

[66] Andrea Toselli and Olof Widlund. *Domain Decomposition Methods-Algorithms And Theory*. Springer, New York, 2006.

[67] Lloyd N. Trefethen. *Finite difference and spectral methods for ordinary and partial differential equations*. Available at

http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html, unpublished text, 1996.

[68] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.

[69] Gun Wirtanen, ERNA STORGARDS, and Maria Saarela. 4.2 detection of biofilms in the food and beverage industry. 2000.

[70] Olaf Wolkenhauer, Peter Wellstead, Kwang-Hyun Cho, Ramon Grima, and Santiago Schnell. Modelling reaction kinetics inside cells. *Essays in biochemistry*, 45:41–56, 2008.

[71] Margaret Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56, 2005.

[72] Stephen J. Wright. Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77(1):161–187, 1993.

[73] Shang-Tian Yang, Xiaoguang Liu, and Yali Zhang. Metabolic engineering–applications, methods, and challenges. In *Bioprocessing for Value-Added Products from Renewable Resources*, pages 73–118. Elsevier, 2007.

[74] Valentin F. Zaitsev and Andrei D. Polyanin. *Handbook of Exact Solutions For Ordinary Differential Equations*. CRC Press, Boca Raton, FL, 2002.

[75] Tianyu Zhang, Breana Pabst, Isaac Klapper, and Philip S. Stewart. General theory for integrated analysis of growth, gene, and protein expression in biofilms. *PloS one*, 8(12):e83626, 2013.

[76] Tianyu Zhang, Albert Parker, Ross P. Carlson, Phil S. Stewart, and Isaac Klapper. Flux-balance based modeling of biofilm communities. *bioRxiv*, page 441311, 2018.