Fall 12-12-2020

Rapid Implicit Diagonalization of Variable-Coefficient Differential Operators Using the Uncertainty Principle

Carley Walker

RAPID IMPLICIT DIAGONALIZATION OF VARIABLE-COEFFICIENT DIFFERENTIAL OPERATORS USING THE UNCERTAINTY PRINCIPLE

by

Carley Rene Walker

A Thesis
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Mathematics and Natural Sciences
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved by:

Dr. James V. Lambers, Committee Chair
Dr. Haiyan Tian
Dr. Huiqing Zhu
Dr. Bernd Schroeder

December 2020

ABSTRACT

We propose to create a new numerical method for a class of time-dependent PDEs (second-order, one space dimension, Dirichlet boundary conditions) that can be used to obtain more accurate and reliable solutions than traditional methods. Previously, it was shown that conventional time-stepping methods could be avoided for time-dependent mathematical models featuring a finite number of homogeneous materials, thus assuming general piecewise constant coefficients. This proposed method will avoid the modeling shortcuts that are traditionally taken, and it will generalize the piecewise constant case of energy diffusion and wave propagation to work for an infinite number of smaller pieces, or a smoothly varying coefficient. We hypothesize that by treating a smoothly varying function as a piecewise constant function with infinitely many pieces, this potential method can be realized. Through the Uncertainty Principle, we will formulate highly accurate estimates of eigenvalues, and then through the QR algorithm, we will use these estimates to formulate highly accurate eigenvalues and eigenfunctions. Ultimately, we will produce a more efficient solution method that avoids traditional time-stepping.

ACKNOWLEDGMENTS

I am forever indebted to my advisor, professor, mentor, and friend, Dr. James Lambers, who has been an outstanding resource and vote of confidence for me throughout my entire college career. He has allowed me to truly reap the benefits and positive experience of my work, and his determined and proactive takes on challenges have brought to me a new perspective on academic life and life in general. Dr. Lambers will always be someone I look up to and I hope to receive the utmost respect from.

I am also thankful for the members of my Thesis committee, Dr. Haiyan Tian, Dr. Huiqing Zhu, and Dr. Bernd Schroeder, for their candor and ardor regarding my work, be it in the classroom or through my research.

Finally, I wish to thank my mom, my dad, and my sisters for their support and positivity throughout this process. I would not be where I am today without them.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

KSS - Krylov Subspace Spectral Methods
PDE - Partial Differential Equation
FFT - Fast Fourier Transform

Chapter 1

INTRODUCTION

1.1 The Problem

Modeling the diffusion of energy and the propagation of waves using the Uncertainty Principle has been an ongoing project by my professor, Dr. James Lambers, for the entirety of my college career. Originally, the ability to do so was proven with two homogenous materials [4] and was later proven again with piecewise constant coefficients [7]. We want to go another step further and prove that this same thing can be done through a more general, realistic material, or a material with smoothly varying properties, such as density or conductivity. This will generalize the algorithm found in previous works and make it more useful and efficient for anyone in the applied mathematics world to use.

1.2 The Purpose of a New Method

The purpose of this project is to unite the versatility and power of partial differential equations with reality. We are no longer assuming anything about coefficients in order to make the solution process easier, but we are rather attempting to create a single, omniscient method for a class of time-dependent PDEs (second-order, one space dimension, Dirichlet boundary conditions) that can actually be used in industry to obtain more accurate and reliable solutions to improve efficiency and productivity. Currently, researchers only have two options when solving complex partial differential equations: they can either take shortcuts and assume various quantities are constant in order to simplify the solution process [6], or they can solve the equation with its original coefficients using existing expensive numerical methods.

We hypothesize that by treating a smoothly varying function as a piecewise constant function with infinitely many pieces, as the latter is virtually indistinguishable from the former within a numerical method, this potential method can be realized. In Garon's project, it was shown that conventional time-stepping methods could be avoided for time-dependent mathematical models featuring a medium of two homogeneous materials, as time-stepping methods are problematic [4]. Long showed in her project that this same thing can be done through a medium of more than two heterogeneous materials, thus assuming

general piecewise constant coefficients [7]. This research project will also avoid treacherous time-stepping methods. Through the Uncertainty Principle [3], we will formulate highly accurate eigenvalue estimates, which will avoid such time-stepping methods. Then, using the QR algorithm and these estimates, we will formulate highly accurate eigenvalues and eigenfunctions.

In this paper we will introduce the PDE in question and consider its background, existing methods, and previous works related to the problem. Then, we will discuss a new, more efficient method for solving this PDE that avoids time-stepping and shortcuts. Next, we will provide numerical results and illustrate the benefits of this new method. Finally, we will give directions for future work with this method.

Chapter 2

BACKGROUND

2.1 The PDE

We consider the PDE

$$\frac{\partial u}{\partial t} = -Lu = \frac{\partial}{\partial x}\left(\alpha(x)^2 \frac{\partial u}{\partial x}\right), \quad 0 < x < 2\pi, \quad t > 0, \tag{2.1}$$

the initial condition

$$u(x,0) = f(x), \quad 0 < x < 2\pi, \tag{2.2}$$

and Dirichlet boundary conditions

$$u(0,t) = u(2\pi,t) = 0. \tag{2.3}$$

The operator L is given a minus sign in the PDE so that L itself is positive semidefinite; that is, L has nonnegative eigenvalues.

2.2 Existing Methods

Advances in numerical computing power have led to the simulation of physical phenomena at higher resolutions, but existing time-stepping methods, such as multistep [1] and Runge-Kutta methods [8] which were developed in the early 1900s before the age of electronic computers, were not designed for such high resolutions, and therefore have difficulties due to stiffness. Stiffness occurs when low- and high-frequency components of the solution cannot be evaluated independently of each other because separation is impossible, forcing the time step to be inefficiently small in numerical methods [2]. Small time steps mean more steps must be taken to achieve desired results, which can be extremely expensive [4]. This new proposed method will provide more reliable results by avoiding the mathematical shortcuts that are traditionally taken, and it will also provide a dramatically more efficient method for solving such PDEs. It will generalize previous projects and will ultimately generalize the piecewise constant case of energy diffusion and wave propagation to work for an infinite number of smaller pieces, or a smoothly varying coefficient.

Currently, researchers use the following methods to solve high resolution, time-dependent PDEs: they assume constant quantities, resulting in modeling error, or they solve the problem

with original coefficients using expensive numerical methods such as multi-step and Runge-Kutta. We aim to develop a new method for PDEs with smoothly varying coefficients that will provide more reliable and efficient results by avoiding short-cuts from oversimplifying the model and difficulties that arise from using traditional numerical methods.

2.3 Previous Works

In this section we will discuss the two other papers that presented the possibility of solving this PDE using the Uncertainty Principle. While these other two papers did not consider a smoothly varying coefficient, they did consider homogeneous mediums with either two pieces or more than two pieces. It is because of these papers that we believe treating a smoothly varying coefficient as an infinite number of small pieces will produce similar results.

2.3.1 Garon, 2017: The Case of Two Homogeneous Materials

Elyse Garon's 2017 paper considered a PDE with two homogeneous materials, or constant coefficients. Garon demonstrated that it was possible to avoid time-stepping and other inefficient methods for this specific PDE, as she produced eigenfunctions with varying frequencies split at the boundary between the two homogeneous materials. This demonstrates how a change in the coefficient results in a change in frequency at the interface between pieces. Garon's method computes the eigenfunctions and then computes the solution of the PDE using an eigenfunction expansion.

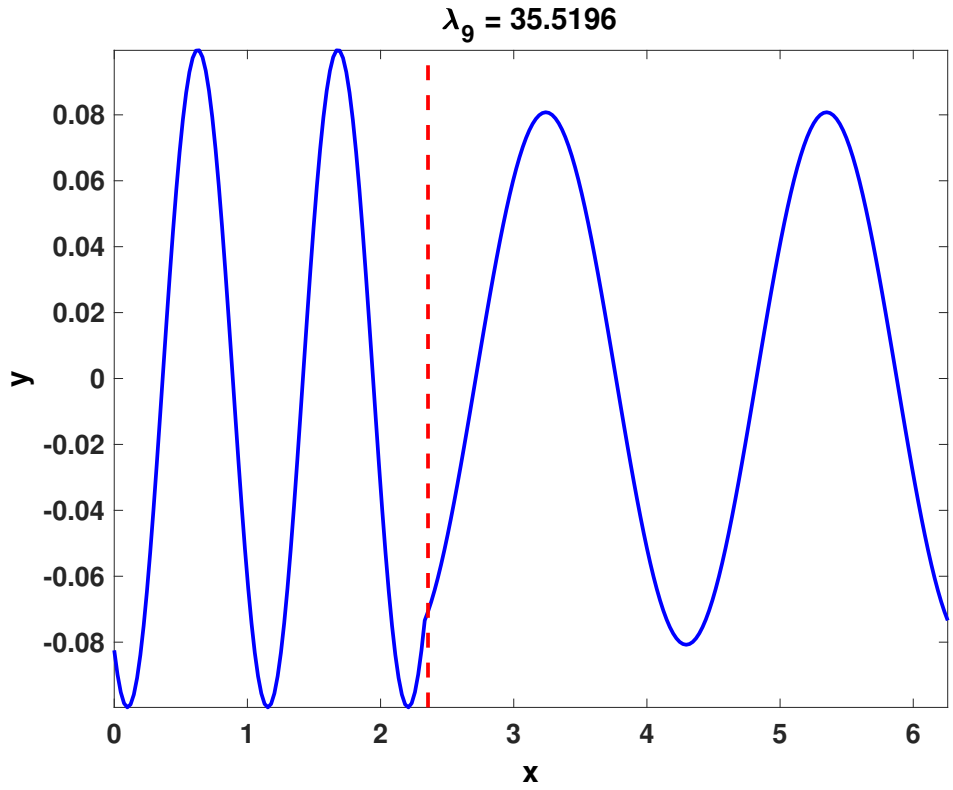

Figure 2.1: Eigenfunction of Constant Coefficient Case
An eigenfunction of a time-dependent model featuring a medium of two homogeneous materials, or where $n = 2$. The red dotted line illustrates the interface between the two pieces.

2.3.2 Long, 2019: The Case of More Than Two Homogeneous Materials

Two years later, Sarah Long was able to produce the same results and show they could be achieved for a medium of more than two homogeneous materials, or piecewise constant coefficients. Long demonstrates the solution of the PDE at a particular time and illustrates the changes in frequency that can be observed at the interfaces. Long's work demonstrates the solution of the heat equation through a medium of more than two homogeneous materials. Highly accurate eigenvalues and eigenfunctions were computed numerically, and then the solution of the PDE was represented using an eigenfunction expansion.

Long's method, labeled SAK, was used to obtain initial guesses of eigenvalues that could then be improved via iteraton. "SAK" comes from Fefferman's principle that the number of eigenvalues of $A(x,D)$ that are less than K is approximately equal to the number of "distorted unit cubes" packed in the set $S(A,K) = (x,\omega)|A(x,\omega) < K$ [Fefferman]. The red dotted line labeled "C-N" demonstrates results produced by Crank-Nicolson, the pink dashed line labeled "Krylov" demonstrates results produced by Lanczos approximating

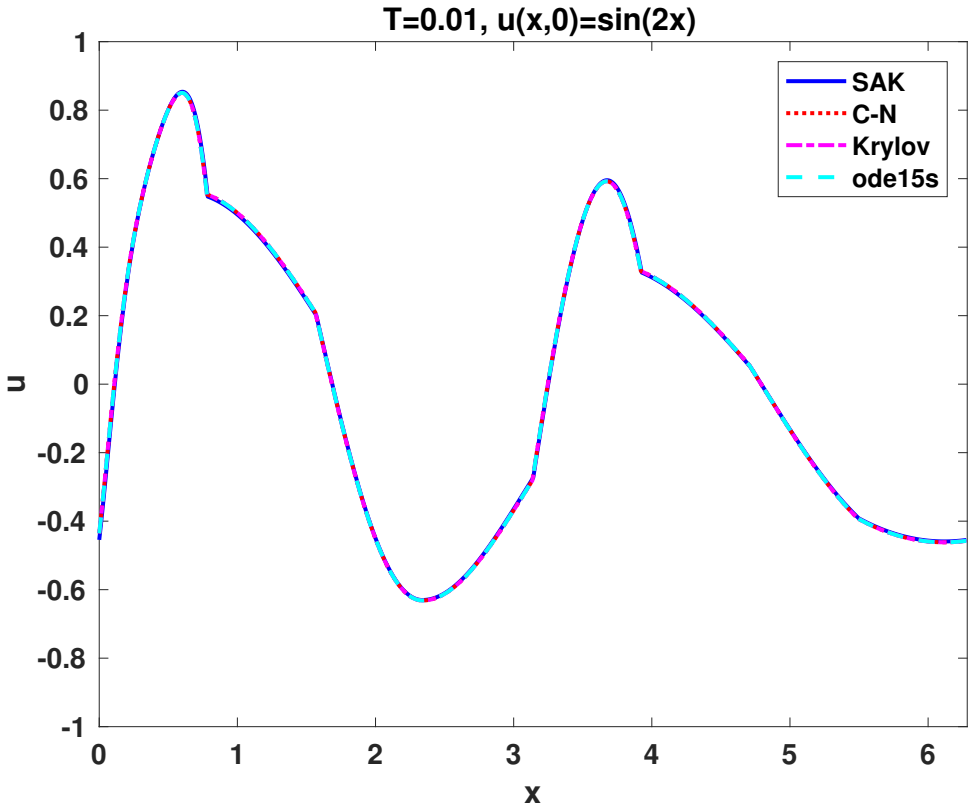

Figure 2.2: Eigenfunction of Piecewise Constant Coefficient Case
An eigenfunction at a time T featuring a medium of eight homogeneous materials, or $n > 2$, calculated by four different methods, including Long and Garon's method labeled SAK.

the matrix exponential, and the cyan dashed line labeled "ode15s" demonstrates results produced by a MATLAB solver for stiff systems of ODEs.

Chapter 3

A NEW METHOD - TRUNCATED EIGENFUNCTION EXPANSION VIA THE QR ALGORITHM

3.1 The Case of Infinitely Many Materials: Smoothly Varying Coefficients

Through simple numerical MATLAB experiments, we illustrate the similarities and differences between an eigenfunction of a smoothly-varying coefficient and a constant coefficient PDE. This demonstrates the idea that a smoothly-varying coefficient can be treated as a piecewise constant coefficient with virtually infinite pieces to produce an accurate, efficient solution.

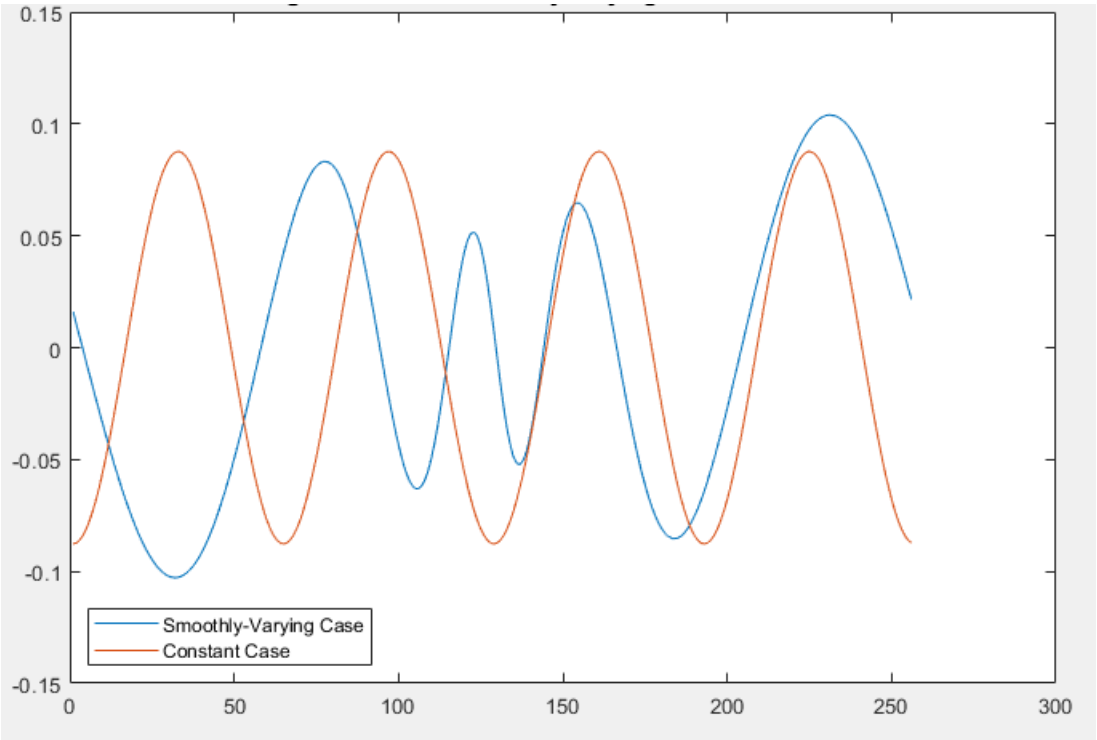

Figure 3.1: Eigenfunction Comparison
Example eigenfunctions of a smoothly-varying medium versus a constant medium.

Specifically, this graph illustrates that in the constant coefficient case, the eigenvector has uniform amplitude and frequency, but not in the smoothly varying case. The smoothly varying case has varying amplitude and frequency.

We aim to use a method similar to Garon and Long to solve a model featuring a smoothly varying medium, specifically where $n \to \infty$. We hypothesize that the smoothly varying coefficient $\alpha(x)$ can be treated as the $\lim_{n\to\infty}$ of the piecewise constant $\alpha(x)$.

3.2 What Happens as n goes to Infinity?

Spatially discretizing this problem on a uniform grid gives

$$x_j = j\Delta x, \quad j = 0, 1, \ldots, N-1, \quad \Delta x = \frac{2\pi}{N}. \tag{3.1}$$

We consider a *circulant* matrix

$$D_N = \frac{1}{\Delta x}\begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ & & & & -1 \end{bmatrix} \tag{3.2}$$

that implements a forward difference formula for the first derivative,

$$u_x(x_i) = \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} + O(\Delta x), \tag{3.3}$$

and A_N is a diagonal matrix, with diagonal entries given by the values of $\alpha(x)^2$ at the gridpoints. Then the PDE is approximated by the system of ODEs

$$\mathbf{u}' = -L_N\mathbf{u}, \quad \mathbf{u}(0) = \mathbf{f}, \tag{3.4}$$

where

$$L_N = D_N^T A_N D_N \tag{3.5}$$

is a $N \times N$ symmetric positive semidefinite matrix.

3.3 Form of the Solution

The solution of the original PDE initial value problem is

$$u(x,t) = \sum_{j=0}^{\infty} e^{-\lambda_j t} G_j(x) \langle G_j, f \rangle, \tag{3.6}$$

where each $G_j(x)$ is an eigenfunction of L with corresponding eigenvalue λ_j, and

$$\langle G_j, f \rangle = \int_0^{2\pi} G_j(x) f(x)\, dx \tag{3.7}$$

is the standard inner product of real-valued functions on $[0, 2\pi]$.

3.4 Solving the System of ODEs

We can perform the QR Algorithm on L_N to obtain its Schur Decomposition [5],

$$L_N = V_N \Lambda_N V_N^T, \tag{3.8}$$

where V_N is an orthogonal matrix whose columns are the eigenvectors of L_N. The solution of the system of ODEs is

$$\mathbf{u}(t) = \sum_{j=1}^{N} e^{-\lambda_j t} \mathbf{v}_j \mathbf{v}_j^T \mathbf{f}, \tag{3.9}$$

where now λ_j is an eigenvalue of the matrix L_N, and $\mathbf{v}_j$ is its corresponding eigenvector.

3.5 Partial Use of the QR Algorithm

While this summation (3.9) can be easily computed by applying the symmetric QR algorithm to (3.8) to get $\mathbf{v_j}$ and λ_j, it would also be wasteful for the following reasons:

- We don't need to explicitly compute V_N since it is a product of Givens rotations. Instead, $V_N^T \mathbf{f}$ can be computed by applying the Givens rotations to $\mathbf{f}$, and then after multiplying by $e^{-\Lambda_N t}$, we can multiply the vector $e^{-\Lambda_N t} V_N^T \mathbf{f}$ by V_N by applying the inverses of all rotations.

- Because L_N is already sparse, each such rotation requires $O(1)$ flops, and there are $O(N)$ such transformations, so the entire process would require only $O(N)$ flops. Similarly, after multiplying by $e^{-\Lambda_N t}$, which requires only $O(N)$ flops since Λ_N is a diagonal matrix, we can then multiply the vector $e^{-\Lambda_N t} V_N f$ by V_N by applying the inverses of all of these rotations, which would require exactly the same amount of work.

- Computing all of the eigenvalues and eigenvectors can be quite expensive; even for a matrix like L_N that is sparse and has a nice structure, the cost can be $O(N^3)$, when we would like to be able to compute the solution in $O(N)$ flops.

3.6 Computation of Eigenvalues and Eigenfunctions

To achieve rapid convergence, we need well-approximated eigenvalues for our QR shifts. Based on the Uncertainty and SAK principles used by Long and Garon, we can formulate an equation for our eigenvalue estimates [7]. We borrow Long's equation for ω_{jn}, the value of initial eigenvalue guesses, and for our case of smoothly varying coefficients, we take the

limit of n to infinity. In Long's case, $(\rho_i - \rho_{i-1})$ denotes the relative gaps between interfaces. Therefore, in our case, $(\rho_i - \rho_{i-1}) = \frac{\Delta x}{2\pi}$. Multiplying ω_{jn} by α_n and substituting in $\frac{\Delta x}{2\pi}$ gives us

$$\overline{\alpha} = \frac{1}{\frac{\Delta x}{2\pi}\sum_{i=1}^{n} \alpha(x_i)^{-1}}, \tag{3.10}$$

which can then be rewritten as

$$\overline{\alpha} = 2\pi \frac{1}{\sum_{i=1}^{n} \alpha(x_i)^{-1}\Delta x}. \tag{3.11}$$

Then, we take the limit of (3.11) as n goes to infinity and have

$$\overline{\alpha} = \lim_{n\to\infty} 2\pi \frac{1}{\sum_{i=1}^{n} \alpha(x_i)^{-1}\Delta x}, \tag{3.12}$$

and we denote this as the Riemann integral

$$\overline{\alpha} = 2\pi \Big(\int_0^{2\pi} \frac{1}{\alpha(x)} dx \Big)^{-1}. \tag{3.13}$$

Therefore, we choose our eigenvalue estimates of L to be

$$\lambda_j = \Big(\overline{\alpha}\frac{2j+1}{4}\Big)^2, \quad \overline{\alpha} = 2\pi \left(\int_0^{2\pi} \frac{1}{\alpha(x)}\, dx \right)^{-1}, \quad j \in \mathbb{Z} \tag{3.14}$$

where $\overline{\alpha}$ is the *harmonic average* of $\alpha(x)$ on $[0, 2\pi]$.

Through numerical MATLAB experiments, we believe to have found well-approximated eigenvalues to use as shifts in the QR algorithm, which will aid in rapid convergence. We show the accuracy of our estimated eigenvalues in Figure 3.2.

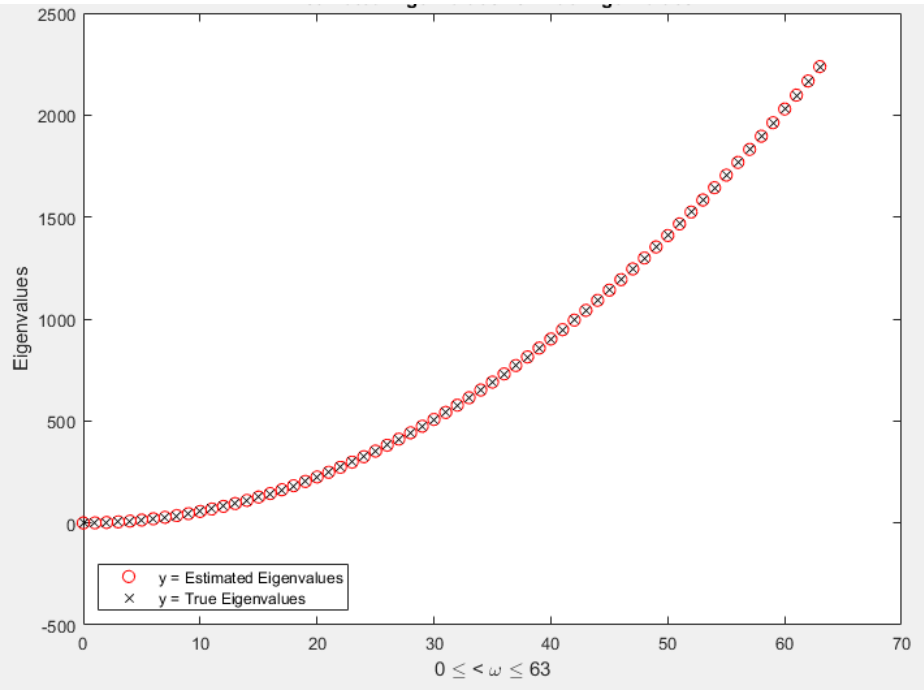

Figure 3.2: Estimated Eigenvalues vs. True Eigenvalues
Red circles show estimates of the eigenvalues using $\lambda_\omega = (\overline{\alpha}\frac{2\omega+1}{4})^2$ and $\overline{\alpha} = 2\pi\left(\int_0^{2\pi}\frac{1}{\alpha(x)}\,dx\right)^{-1}$, while black crosses show the true eigenvalues.

These true eigenvalues come from using the QR algorithm via MATLAB's `eig` function to obtain the eigenvalues from a matrix that discretizes the operator using the Fast Fourier Transform. Because it uses the FFT instead of finite difference, it is NOT a sparse matrix, therefore this would be too expensive to use in a method for solving PDEs, and therefore is only useful for validation of our estimated eigenvalues.

3.7 Applying the QR Algorithm

We will use these eigenvalues as the shift μ in each symmetric QR step in the QR algorithm. Our accurate initial shift μ will aid in quick, cubic convergence, and we will compute the Wilkinson shift as the iteration proceeds to use as μ instead. Figure 3.3 shows Gershgorin disks of our tridiagonal matrix and its estimated eigenvalues when we use Wilkinson shifts for every iteration of the algorithm. Figure 3.4 shows Gershgorin disks of our tridiagonal matrix and its estimated eigenvalues when we use our eigenvalues computed in 3.6 for the first 20 iterations, and then use Wilkinson shifts for the remaining iterations.

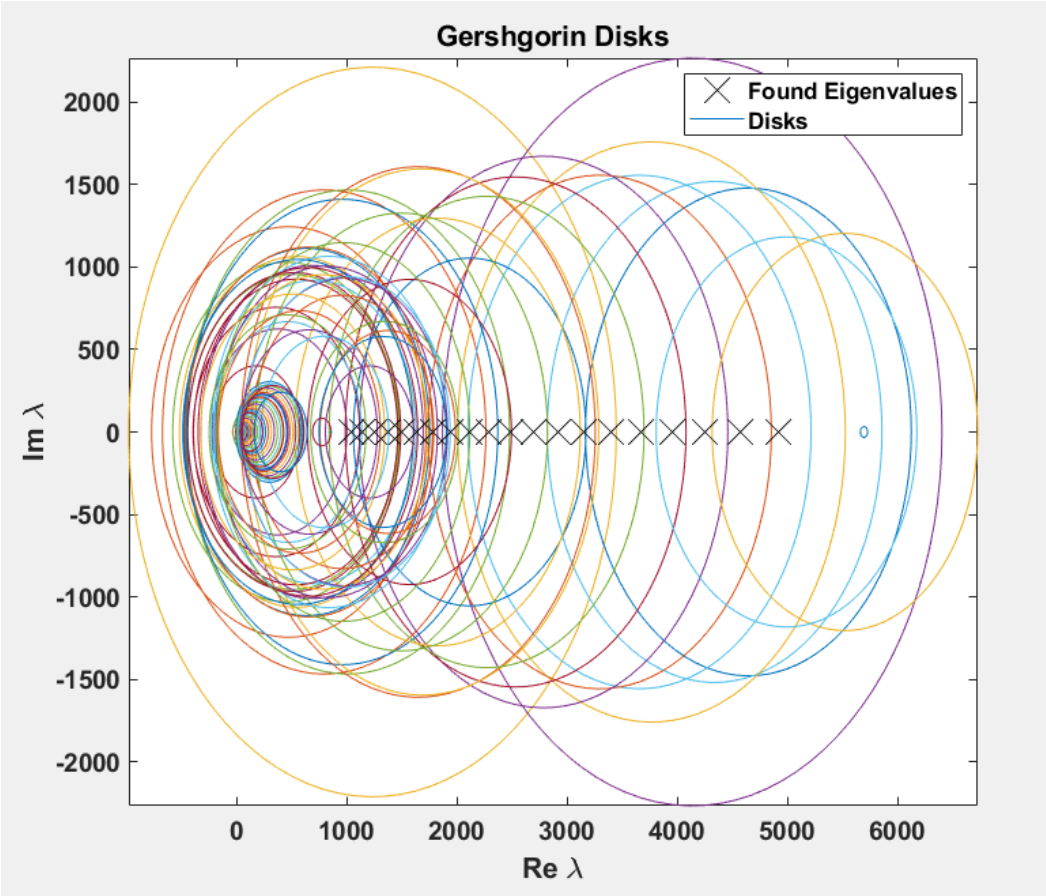

Figure 3.3: Gershgorin Disks for Wilkinson Shift as μ
Black x's show the estimated eigenvalues and multi-colored circles show the Gershgorin disks when we use Wilkinson shifts as μ for every iteration.

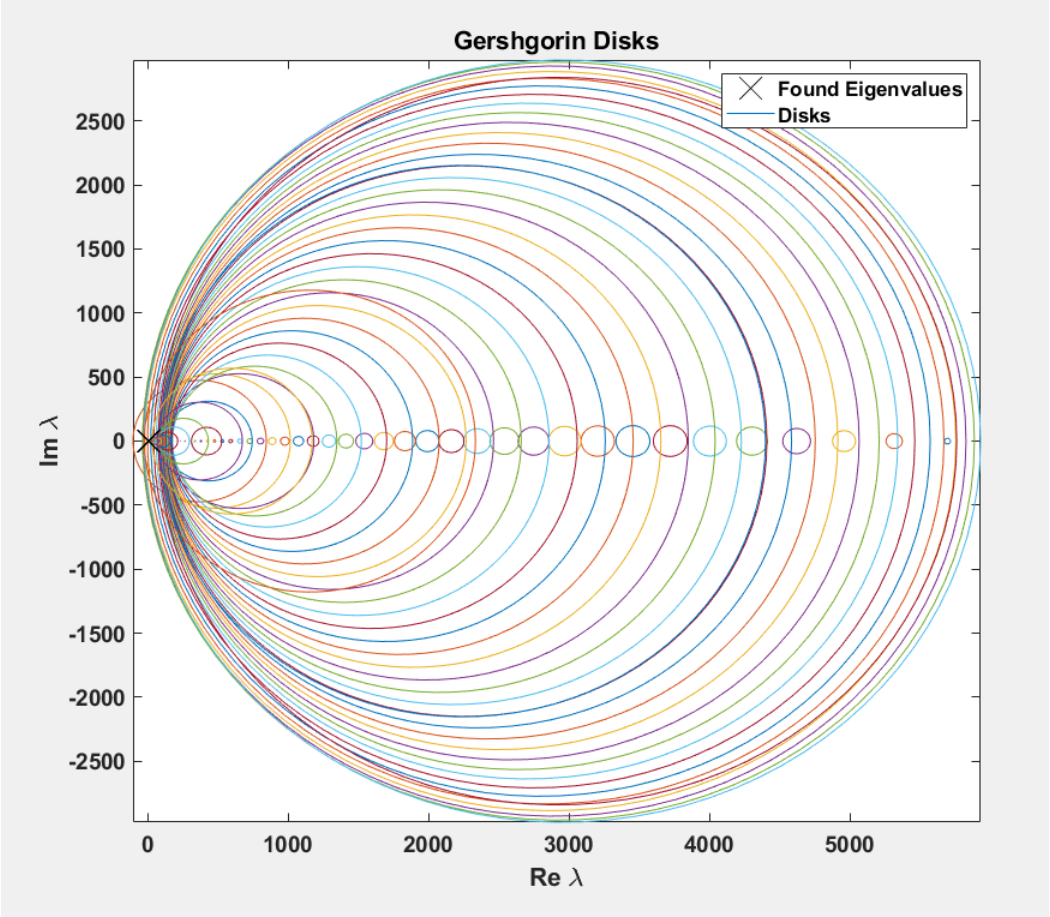

Figure 3.4: Gershgorin Disks for Wilkinson Shift as μ when Iterations > 20
Black x's show the estimated eigenvalues and multi-colored circles show the Gershgorin disks when we use Wilkinson shifts as μ for the twenty-first iteration and beyond.

We can see that the eigenvalues computed by the shifts in Figure 3.4 are much smaller, and will therefore give us a much more accurate solution. Therefore, we will use the Wilkinson shift after 20 iterations.

Chapter 4

NUMERICAL RESULTS

In this section we demonstrate the effectiveness and scalability of our method for computing eigenvalues and eigenfunctions of a smoothly varying coefficient PDE compared to Crank-Nicolson and MATLAB's ODE solver `ode15s`. We will compare these methods in tables where our method will be labeled "UC", for the Uncertainty Principle, Crank-Nicolson will be labeled "C-N", and MATLAB's solver will be labeled "`ode15s`". In Tables 4.1-4.5, we will be comparing these methods' speeds and scalability, and in Tables 4.6-4.9, we will compare these methods' accuracy.

We will apply these methods with three different grid sizes $N = 255,\ 511,\ 1023$ and two different coefficients

$$\alpha_1(x) = 1 + \frac{9}{10}\cos x \tag{4.1}$$

and

$$\alpha_2(x) = 2 + \cos x. \tag{4.2}$$

We will also be applying three different final times (in seconds) $t_f = 0.01,\ 0.1,\ 1$ and two different initial data of $u_1(x,0) = \sin 2x$ and a characteristic function $u_2(x,0)$ that equals 1 on the interval $[\frac{3\pi}{4}\ \ \pi]$ and 0 everywhere else.

Figures 4.1-4.4 demonstrate the accuracy of our solution compared to Crank-Nicolson and MATLAB's `ode15s`. We apply our method using our initial data of $u_1(x,0) = \sin 2x$ in Figures 4.1 and 4.2, and we use our initial data characterstic function $u_2(x,0)$ in Figures 4.3 and 4.4. We also apply our method using our coefficient $\alpha(x)$ from (4.1) in Figures 4.1 and 4.3, and we use our coefficient $\alpha(x)$ from (4.2) in Figures 4.2 and 4.4. We observe a jump from computing derivatives of discontinuous functions using finite differences in the Crank-Nicolson solution with the characteristic function at time $T = 0.1$ for both of our coefficients in Figures 4.3 and 4.4. This same jump can be observed in Table 4.4.

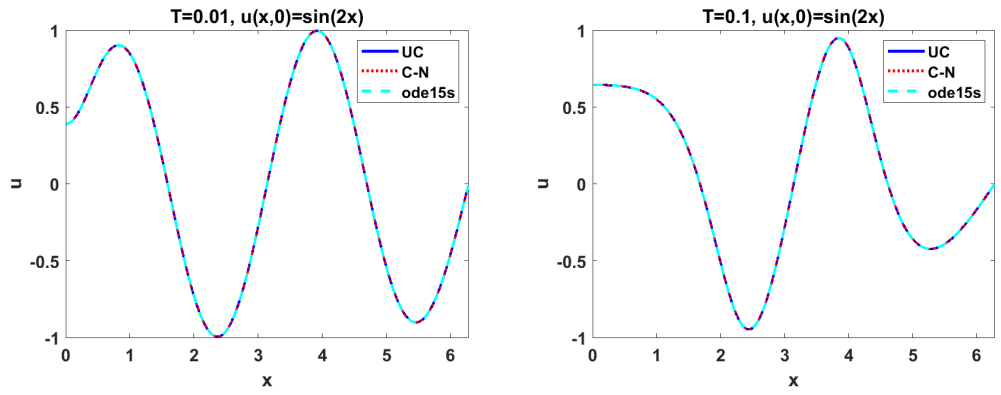

Figure 4.1: Solutions w/ $\alpha_1(x)$ and $u_1(x,0)$
Solutions of (2.1) with initial data $u_1(x,0) = \sin 2x$, computed at $t_f = 0.01, 0.1$ on a uniform grid with $N = 1,023$ points. The coefficient $\alpha(x)$ is defined by (4.1).

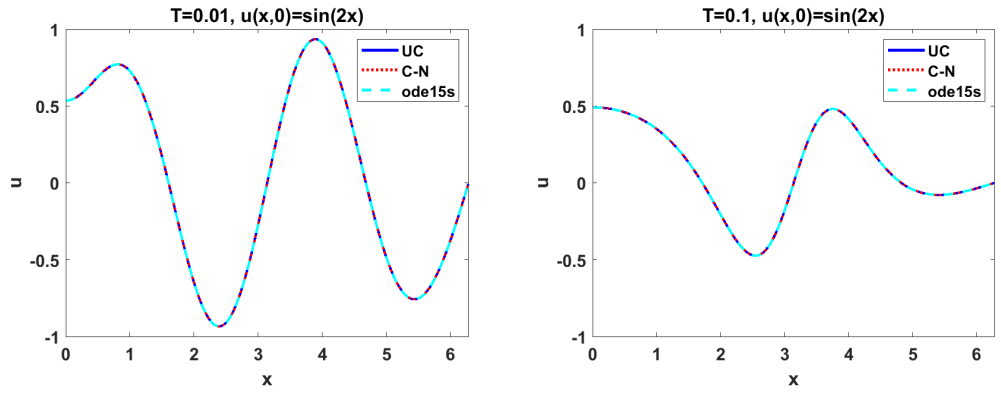

Figure 4.2: Solutions w/ $\alpha_2(x)$ and $u_1(x,0)$
Solutions of (2.1) with initial data $u_1(x,0) = \sin 2x$, computed at $t_f = 0.01, 0.1$ on a uniform grid with $N = 1,023$ points. The coefficient $\alpha(x)$ is defined by (4.2).

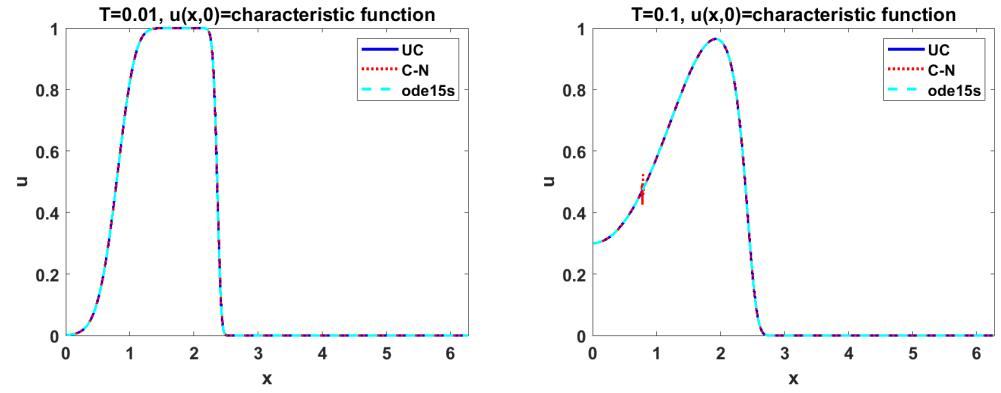

Figure 4.3: Solutions w/ $\alpha_1(x)$ and $u_2(x,0)$
Solutions of (2.1) with initial data $u_2(x,0)$, computed at $t_f = 0.01, 0.1$ on a uniform grid with $N = 1,023$ points. The coefficient $\alpha(x)$ is defined by (4.1).

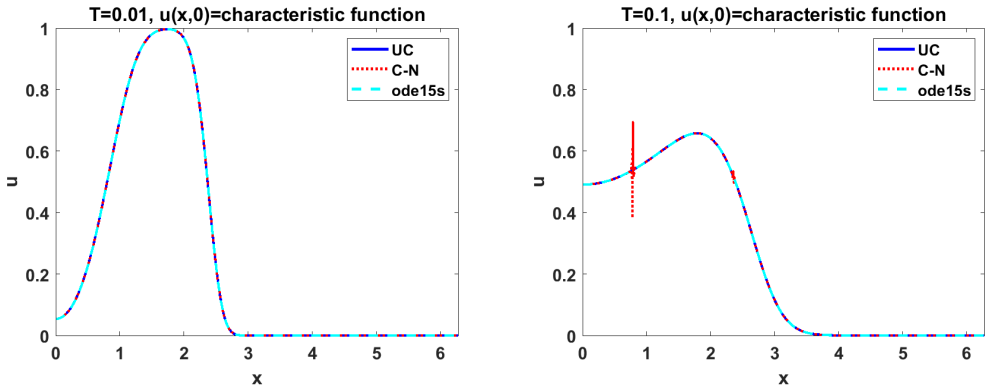

Figure 4.4: Solutions w/ $\alpha_2(x)$ and $u_2(x,0)$
Solutions of (2.1) with initial data $u_2(x,0)$, computed at $t_f = 0.01, 0.1$ on a uniform grid with $N = 1,023$ points. The coefficient $\alpha(x)$ is defined by (4.2).

4.1 Tables Comparing Scalability

Table 4.1: *Speed Comparison for* $\alpha_1(x)$ *and* $u_1(x,0)$.

		UC		C-N	`ode15s`	
t_f	N	time	terms	time	time	Δt_{avg}
0.01	127	0.414	125	0.035	0.024	3.846e-04
	255	1.257	210	0.135	0.079	2.857e-04
	511	3.991	209	0.360	0.285	2.273e-04
0.1	127	0.276	66	0.033	0.029	2.381e-03
	255	0.867	66	0.121	0.106	1.923e-03
	511	1.626	66	0.358	0.391	1.613e-03
1	127	0.153	21	0.021	0.048	1.282e-02
	255	0.336	21	0.098	0.179	1.111e-02
	511	0.636	21	0.366	0.610	1.010e-02

Table 4.2: *Speed Comparison for $\alpha_1(x)$ and $u_2(x,0)$.*

		UC		C-N	`ode15s`	
t_f	N	time	terms	time	time	Δt_{avg}
0.01	127	0.449	125	0.028	0.033	1.786e-04
	255	1.364	210	0.100	0.151	1.389e-04
	511	4.028	209	0.359	0.520	1.124e-04
0.1	127	0.311	66	0.024	0.047	1.235e-03
	255	0.673	66	0.109	0.197	9.901e-04
	511	1.589	66	0.358	0.721	8.403e-04
1	127	0.096	21	0.022	0.063	9.259e-03
	255	0.276	21	0.126	0.259	7.634e-03
	511	0.677	21	0.361	0.887	6.667e-03

Table 4.3: *Speed Comparison for $\alpha_2(x)$ and $u_1(x,0)$.*

		UC		C-N	`ode15s`	
t_f	N	time	terms	time	time	Δt_{avg}
0.01	127	0.280	53	0.036	0.020	3.125e-04
	255	0.580	53	0.099	0.086	2.439e-04
	511	1.357	53	0.358	0.335	1.961e-04
0.1	127	0.151	17	0.024	0.033	1.786e-03
	255	0.302	17	0.099	0.137	1.471e-03
	511	0.503	17	0.372	0.472	1.266e-03
1	127	0.033	5	0.036	0.055	1.087e-02
	255	0.086	5	0.099	0.203	9.615e-03
	511	0.192	5	0.358	0.655	8.621e-03

Table 4.4: *Speed Comparison for $\alpha_2(x)$ and $u_2(x,0)$.*

		UC		C-N	`ode15s`	
t_f	N	time	terms	time	time	Δt_{avg}
0.01	127	0.239	53	0.027	0.039	1.471e-04
	255	0.611	53	0.103	0.166	1.176e-04
	511	1.310	53	0.393	0.622	9.709e-05
0.1	127	0.121	17	0.027	0.055	1.042e-03
	255	0.239	17	0.110	0.219	8.696e-04
	511	0.500	17	0.392	0.858	7.576e-04
1	127	0.047	5	0.034	0.071	8.000e-03
	255	0.121	5	0.101	0.295	6.993e-03
	511	0.256	5	0.413	0.971	6.211e-03

We can see that although our method is slower under certain circumstances, its scalablity is better than other methods. Specifically for Tables 4.3 and 4.4, we will create new tables, Tables 4.5 and 4.6, respectively, with the ratios of new time t_n to old time t_o to more clearly see this improved scalability.

Table 4.5: *Scalability Comparison for* $\alpha_2(x)$ *and* $u_1(x,0)$.

	UC		C-N		`ode15s`	
N	time	$\frac{t_n}{t_o}$	time	$\frac{t_n}{t_o}$	time	$\frac{t_n}{t_o}$
0.01	0.280		0.036		0.020	
	0.580	**2.071**	0.099	**2.750**	0.086	**4.300**
	1.357	**2.340**	0.358	**3.616**	0.335	**3.900**
0.1	0.151		0.024		0.033	
	0.302	**2.000**	0.099	**4.125**	0.137	**4.152**
	0.503	**1.666**	0.372	**3.758**	0.472	**3.445**
1	0.033		0.036		0.055	
	0.086	**2.606**	0.099	**2.750**	0.203	**3.691**
	0.192	**2.233**	0.358	**3.616**	0.655	**3.227**

Table 4.6: *Scalability Comparison for* $\alpha_2(x)$ *and* $u_2(x,0)$.

	UC		C-N		`ode15s`	
N	time	$\frac{t_n}{t_o}$	time	$\frac{t_n}{t_o}$	time	$\frac{t_n}{t_o}$
0.01	0.239		0.027		0.039	
	0.611	**2.556**	0.103	**3.815**	0.166	**4.256**
	1.310	**2.441**	0.393	**3.816**	0.622	**3.747**
0.1	0.121		0.027		0.055	
	0.239	**1.975**	0.110	**4.074**	0.219	**3.982**
	0.500	**2.092**	0.392	**3.564**	0.858	**3.918**
1	0.047		0.034		0.071	
	0.121	**2.574**	0.101	**2.971**	0.295	**4.155**
	0.256	**2.116**	0.413	**4.089**	0.971	**3.292**

4.2 Tables Comparing Error

Table 4.7: *Error Comparison for* $\alpha_1(x)$ *and* $u_1(x,0)$.

		UC		C-N	`ode15s`	
t_f	N	error	terms	error	error	Δt_{avg}
0.01	127	3.4e-02	125	3.4e-02	3.4e-02	3.846e-04
	255	1.5e-02	210	1.5e-02	1.5e-02	2.857e-04
	511	5.2e-03	209	5.2e-03	5.2e-03	2.273e-04
0.1	127	1.4e-02	66	1.4e-02	1.4e-02	2.381e-03
	255	6.1e-03	66	6.1e-03	6.0e-03	1.923e-03
	511	2.1e-03	66	2.1e-03	2.0e-03	1.613e-03
1	127	4.0e-02	21	4.0e-02	4.0e-02	1.282e-02
	255	1.7e-02	21	1.7e-02	1.7e-02	1.111e-02
	511	5.7e-03	21	5.7e-03	5.7e-03	1.010e-02

Table 4.8: *Error Comparison for* $\alpha_1(x)$ *and* $u_2(x,0)$.

		UC		C-N	`ode15s`	
t_f	N	error	terms	error	error	Δt_{avg}
0.01	127	1.9e-01	125	1.9e-01	1.9e-01	1.786e-04
	255	7.3e-02	210	7.3e-02	7.3e-02	1.389e-04
	511	2.4e-02	209	2.4e-02	2.4e-02	1.124e-04
0.1	127	5.8e-02	66	5.8e-02	5.8e-02	1.235e-03
	255	2.5e-02	66	2.5e-02	2.5e-02	9.901e-04
	511	8.4e-03	66	8.4e-03	8.4e-03	8.403e-04
1	127	4.3e-02	21	4.3e-02	4.3e-02	9.259e-03
	255	1.9e-02	21	4.3e-02	1.9e-02	7.634e-03
	511	6.2e-03	21	2.5e-01	6.2e-03	6.667e-03

Table 4.9: *Error Comparison for $\alpha_2(x)$ and $u_1(x,0)$.*

		UC		C-N	`ode15s`	
t_f	N	error	terms	error	error	Δt_{avg}
0.01	127	3.1e-02	53	3.1e-02	3.1e-02	3.125e-04
	255	1.4e-02	53	1.4e-02	1.4e-02	2.439e-04
	511	4.6e-03	53	4.6e-03	4.6e-03	1.961e-04
0.1	127	1.6e-02	17	1.6e-02	1.6e-02	1.786e-03
	255	6.8e-03	17	6.8e-03	6.8e-03	1.471e-03
	511	2.3e-03	17	2.3e-03	2.3e-03	1.266e-03
1	127	2.1e-02	5	2.1e-02	2.1e-02	1.087e-02
	255	8.9e-03	5	2.4e-02	8.9e-03	9.615e-03
	511	3.0e-03	5	6.3e-02	3.0e-03	8.621e-03

Table 4.10: *Error Comparison for $\alpha_2(x)$ and $u_2(x,0)$.*

		UC		C-N	`ode15s`	
t_f	N	error	terms	error	error	Δt_{avg}
0.01	127	4.7e-02	53	4.7e-02	4.7e-02	1.471e-04
	255	2.0e-02	53	2.0e-02	2.0e-02	1.176e-04
	511	6.8e-03	53	6.8e-03	6.8e-03	9.709e-05
0.1	127	3.8e-02	17	3.8e-02	3.8e-02	1.042e-03
	255	1.6e-02	17	1.6e-02	1.6e-02	8.696e-04
	511	5.4e-03	17	3.7e-02	5.4e-03	7.576e-04
1	127	2.9e-02	5	4.1e-02	2.9e-02	8.000e-03
	255	1.3e-02	5	2.3e-01	1.3e-02	6.993e-03
	511	4.2e-03	5	5.6e-01	4.2e-03	6.211e-03

Chapter 5

CONCLUSION

We have developed an accurate, efficient, and scalable algorithm for computing the eigenvalues and eigenfunctions of a second-order, one-dimensional differential operator with a smoothly varying coefficient $\alpha^2(x)$ and Dirichlet boundary conditions by treating it as a piecewise constant coefficient with the limit as $n \to \infty$. We observe that smaller grid sizes result in our method being slower than other methods, but our method's scalability consistently proves to be better. At larger grid sizes, our method converges quicker than other methods, making it more efficient. We also observe that our method is as equally accurate as other methods.

Future work could consider generalization to other boundary conditions, such as periodic or Neumann boundary conditions. One could also consider a two-dimensional differential operator instead of one-dimensional, as we did in our case. Future work could also attempt to more accurately estimate eigenvalues for finite difference matrices, resulting in quicker convergence.

Another potential consideration would be to use Jacobi rotations instead of the QR algorithm for non-tridiagonal matrices that come from periodic boundary conditions or a two-dimensional case. The Jacobi method for estimating eigenvalues would be helpful, as the rotations can be applied in parallel. Modifying our method so that these Jacobi rotations would find the smallest eigenvalues first would make this method much more versatile for other problems.

Appendix A

MATLAB CODES

A.1 SYMEIG Code

The MATLAB code `SYMEIG` used to compute our diagonal matrix D is shown below.

```
function [D,Q,f,terms]=symeig(A,h,T,TOL,f)
[D,Q,f,terms]=symqr(A,h,T,TOL,f);
```

A.2 SYMQR Code

The MATLAB code `SYMQR` runs the QR Algorithm on our matrix D and is shown below.

```
function [D,Q,f,terms]=symqr(A,h,T,TOL,f)
D=A;
[~,n]=size(A);
Q=zeros(2*(n-1),4);
tol=10e-10;
t=linspace(0,2*pi);
Ddiag=diag(D);
where=1;
q0=ceil((2/h)*sqrt((-1/T)*log(TOL))-(1/2));
q0=max([q0 0]);
for q=0:min([q0 n-2])
    while abs(D(n-q,n-q-1))>tol
        X=(D(1:n-q,1:n-q));
        [T,G]=qrstep(X,q,h);
        f=rotate(G,f,1);
        D(1:n-q,1:n-q)=T;
        Q(where:(where+size(G,1)-1),:)=G;
        where=where+size(G,1);
    end
    if q==20
        Ddiag=diag(D);
        figure(2)
        plot(Ddiag(n-q+1:end),0*Ddiag(n-q+1:end),'xk','MarkerSize',20)
        hold on
        for i=1:n
            centerx(i)=D(i,i);
            centery(i)=0;
            radius(i)=norm(D(i,:),1)-abs(D(i,i));
            plot(centerx(i)+radius(i)*cos(t),centery(i)+radius(i)*sin(t))
            set(gca,'FontWeight','bold')
            set(gca,'FontSize',14)
```

```
            hold on
        end
        plot(Ddiag(n-q+1:end),0*Ddiag(n-q+1:end),'xk','MarkerSize',20)
        hold off
        axis tight
        xlabel('Re \lambda')
        ylabel('Im \lambda')
        title(['Gershgorin Disks' ])
        legend('Found Eigenvalues','Disks')
        saveas(gcf,savename,'fig')
        saveas(gcf,savename,'png')
        save(savename)
    end
end
terms=q;
Q=Q(1:where-1,:);
```

A.3 QRSTEP Code

The MATLAB code `QRSTEP` controls our shifts during the QR Algorithm and is shown below.

```
function [T,U]=qrstep(A,q,h)
[m,n]=size(A);
T=A;
U=zeros(n-1,4);
d=(T(n-1,n-1)-T(n,n))/2;
mu1=T(n,n)-((T(n,n-1)*T(n,n-1))/(d+sign(d)*sqrt(d^2+T(n,n-1)*T(n,n-1))));
g=zeros(n,2);
mu2=((h/4)+q*(h/2))^2;
mu3=((h/4)+(q+1)*(h/2))^2;
mu=mu1;
if mu1<(mu2+mu3)/2
    mu=mu1;
else
    mu=mu2;
end
x=T(1,1)-mu;
z=T(2,1);
for k=1:n-1
[c,s]=givens(x,z);
g(k,1)=c;
g(k,2)=s;
    G=[c -s; s c];
    col=max(1,k-1):min(n,k+2);
T(k:k+1,col)=G'*T(k:k+1,col);
    T(col,k:k+1)=T(col,k:k+1)*G;
    U(k,:)=[k,k+1,c,s];
    if (k<n-1)
        x=T(k+1,k);
        z=T(k+2,k);
    end
end
```

A.4 MAKEMAT Code

The MATLAB code MAKEMAT computes our tridiagonal matrix L and is shown below.

```
function [L,mat_time,h]=makemat(N,ALPHA)
tic
x=zeros(N,1);
deltax=(2*pi)/(N+1);
for j=1:N
    x(j)=j*deltax;
end
a=ALPHA(x);
h=1/(mean(1./a));
A=diag(a.^2);
m=N;
n=N;
d=ones(2,1);
D=zeros(N,2);
D(:,1)=-1;
d(1)=0;
D(:,2)=1;
d(2)=1;
D=spdiags(D,d,m,n);
D=(1/(deltax))*D;
L=D'*A*D;
mat_time=toc;
```

A.5 UNCERTAINTY Code

The MATLAB code UNCERTAINTY computes our solution u and is shown below.

```
function [u,terms]=uncertainty(f,T,L,h)
TOL=10e-10;
[Lamb,V,p,terms]=symeig(L,h,T,TOL,f);
lambda=diag(Lamb);
e=exp(-lambda*T);
u=rotate(V,(e.*p),2);
```

BIBLIOGRAPHY

[1] F. Bashforth and J. C. Adams. *An Attempt to test the Theories of Capillary Action by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops*. Cambridge University Press, Cambridge, UK, 1883.

[2] R. L. Burden and J. D. Faires. Numerical analysis. *Cengage Learning*, 9, 2010.

[3] C. Fefferman. The uncertainty principle. *Bulletin of the American Mathematical Society*, pages 129–206, 1983.

[4] E. M. Garon and J. V. Lambers. Modeling the diffusion of heat energy within composites of homogeneous materials using the uncertainty principle. *Computational and Applied Mathematics*, 37:2566–2587, 2017.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU Press, Baltimore, MD, 1983.

[6] R. Grimshaw, D. Pelinovsky, and E. Pelinovsky. Homogenization of the variable-speed wave equation. *Wave Motion*, 47:496–507, 2010.

[7] S. D. Long, S. Sheikholeslami, J. V. Lambers, and C. Walker. Diagonalization of 1-d differential operators with piecewise constant coefficients using the uncertainty principle. *Mathematics and Computers in Simulation*, 156:194–226, 2019.

[8] C. D. T. Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen, Springer*, 46:167–178, 1895.