Spectral Methods for Partial Differential Equations that Model Shallow Water Wave Phenomena

Maurice S. Fabien

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2014

Reading Committee:

Randall J. LeVeque, Chair

Bernard Deconinck, Member

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

Spectral Methods for Partial Differential Equations that Model Shallow Water Wave Phenomena

Maurice S. Fabien

Chair of the Supervisory Committee:
Randall J. LeVeque

Mathematical models for waves on shallow water surfaces has been of interest to researchers dating back to the 1800's. These models are governed by partial differential equations, and many of them have rich mathematical structure as well as real world applications. This thesis explores a class of numerical techniques for partial differential equations called spectral methods. One can use these spectral methods to approximate solutions to many partial differential equations that model wave type phenomena. Of particular interest are the KdV, BBM, Camassa-Holm, Boussinesq systems, Shallow Water, and Serre Green-Naghdi equations. For all examples presented MATLAB code is provided. These files will be uploaded to the GitHub page https://github.com/msfabien/.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables vi

Chapter 1: Introduction 1
1.1 The Purpose of This Thesis 2

Chapter 2: Fourier Spectral method for the Korteweg–de Vries (KdV) equation . . 3
2.1 KdV equation coefficients 4
2.2 Exact smooth solitary traveling wave solutions for the KdV equation 4
2.3 Fourier Spectral Methods for the KdV equation 7
2.4 Implementation of the Fourier Spectral Method 11
2.5 Small dispersion limit of the KdV equation 24

Chapter 3: Fourier Spectral method for the Benjamin–Bona–Mahony (BBM) equation 28
3.1 BBM equation 29
3.2 Fourier Spectral methods for the BBM equation 30
3.3 Implementation of the Fourier Spectral method 30

Chapter 4: Fourier Spectral method for the Camassa–Holm (CH) equation 37
4.1 CH equation 38
4.2 Fourier Spectral methods for the CH equation 39
4.3 Implementation of the Fourier Spectral method 39
4.4 Small dispersion limit of the CH equation 43

Chapter 5: Fourier Spectral method for the Shallow Water (SW) equations 47
5.1 1D Shallow Water equations 48
5.2 Fourier Spectral methods for the 1D SW equations 49
5.3 Implementation for the 1D SW equations 51
5.4 2D Shallow Water equations 61

5.5 Fourier Spectral methods for the 2D SW equations 61
5.6 Implementation for the 2D SW equations 62

Chapter 6: Radial basis function pseudospectral methods 64
6.1 Introduction to Radial basis functions 65
6.2 Radial basis function collocation and pseudospectral methods 77
6.3 Local radial basis function collocation methods 80
6.4 Radial basis function pseudospectral method for the KdV equation 84
6.5 Local radial basis function collocation method for the KdV equation 87
6.6 Radial basis function pseudospectral method for the BBM equation 91
6.7 Radial basis function pseudospectral method for a Boussinesq system 94
6.8 Radial basis function pseudospectral method for 1D/2D shallow water equations 98
6.9 Radial basis function pseudospectral method for the fully nonlinear 1D Serre–Green Nagdhi equations 101

Chapter 7: Conclusions and future directions 108
7.1 Conclusions 108
7.2 Future Directions 109

Bibliography 111

Appendix A: Where to find the files 118

Appendix B: Auxiliary files 119

LIST OF FIGURES

Figure Number — Page

2.1 (a) Single soliton solution (from equation (2.2)) traveling to the right with a speed $v = 1$. (b) Two solitons with nearly equal amplitudes interacting. . . . 6

2.2 Two solitons with nearly equal amplitudes interacting ($\epsilon = 0.3$ in equations (NE.1) and (NE.2)). . . . 6

2.3 (a) Pseudo-spectral method with Δt that satisfies (2.8). (b) Pseudo-spectral method with Δt that does not satisfy (2.8). . . . 15

2.4 Error comparison of IFRK4 and mETDRK4 schemes. The mETDRK4 scheme outperforms IFRK4. Spectral accuracy is observed in the spatial discretization, and 4th order accuracy in the temporal discretization. Amplitude v in equation (2.3.2) is $v = 25$, initial position is $x_0 = -2$, and final time simulation is $T_{\text{final}} = 0.001$. Errors are in the infinity norm. . . . 22

2.5 Error comparison of IFRK4 and mETDRK4 schemes for two solitons with similar amplitudes interacting. The mETDRK4 scheme outperforms IFRK4. Notice that there is a saturation of spectral accuracy in figure 2.5b. This is due to the two solitons interacting and leaving non–negligible residual errors. 23

2.6 Behavior of the small dispersion limit. . . . 25

2.7 Solution to the Hopf equation before t_c (the gradient catastrophe). . . . 26

3.1 (a) Single soliton solution (from equation (3.2)) traveling to the right with a speed $c_s = 2$. (b) Top-down view of the single soliton's evolution. . . . 29

3.2 Direct spectral approach breaking down. Relevant parameters: $\Delta t = 0.9$, $T_{\text{final}} = 15$, and $N = 2^7$. . . . 34

3.3 (a) Error due to temporal discretization. (b) Error due to spatial discretization. 36

4.1 (a) Single peakon given by $u(x,t) = e^{-|x-5-t|}$ traveling to the right. (b) Multipeakon given by $u(x,t) = e^{-|x-5-t|} + 2e^{-|x-8-t|}$. . . . 38

4.2 (a) Spatial discretization error ($\Delta t = 2 * 10^{-4}$, $2^5 \leq N \leq 2^{15}$). (b) Fourier spectral approximation ($N = 2^{11}, \Delta t = 2 * 10^{-4}$). . . . 42

4.3 (a) Fourier spectral approximation ($N = 2^8$, $\Delta t = 2 * 10^{-4}$). (b) A zoom in of the Fourier spectral approximation ($N = 2^{11}$, $\Delta t = 2 * 10^{-4}$). . . . 42

4.4 Behavior of the small dispersion limit for the CH equation. . . . 45

4.5 Behavior of the small dispersion limit for the CH equation with nonsmooth peakon initinal condition ($\kappa = 0$). . . . 46

5.1 Relevant variables for the SW equations. Similar figures can be found in [62]. 48

5.2 Simulation of 1D–shallow water equation. (Similar figures can be found in [61]) 48

5.3 Results of the Fourier pseudospectral method (this figure corresponds to Code Listing 5.3.4). Relevant parameters: $g = 9.81$, $\Delta t = 10^{-1}$, $N = 2^7$, $h(x,0) = 2 + \sin(x\pi/100)$, and $u(x,0) \equiv 0$. This is an example of smooth data that does not need hyperviscosity. 57

5.4 Results of the Fourier pseudospectral method without hyperviscosity (this figure corresponds to Code Listing 5.3.1). Relevant parameters: $g = 1$, $\Delta t = 10^{-3}$, $N = 2^8$, $h(x,0) = 1 + \exp(-8(x-5)^2)/4$, and $u(x,0) \equiv 0$. Although the initial condition is smooth, the nonlinearity of the SW equations generate less smooth data. "The front of the wave (relative to its direction of motion) steepens through a compression wave into a shock, while the back spreads out as a rarefaction wave." (R. J. LeVeque, [61]) 58

5.5 Results of the Fourier pseudospectral method with and without hyperviscosity. Relevant parameters: $g = 1$, $\Delta t = 10^{-3}$, $N = 2^7$. The effects of hyperviscosity are clear; and the Fourier power spectrum can be used to study how hyperviscosity can assist in improving stability and dealiasing. Figures 5.5a and 5.5c correspond to Code Listing 5.3.2. Figures 5.5b and 5.5d correspond to Code Listing 5.3.3. 59

5.6 Results of the Fourier pseudospectral method (corresponds to Code Listing 5.3.5). 60

5.7 Results of the Fourier pseudospectral method simulating the 2D shallow water equations. Relevant parameters: $g = 9.81$, $\Delta t = 10^{-2}$, $N = 40$, periodic boundary conditions. 63

6.1 RBF interpolation using different centers. 68

6.2 RBF differentiation example. Notice in figure 6.2c the error oscillates, this is due to N alternating between even and odd natural numbers. The RBF center layout changes depending on if N is even or odd. 72

6.3 Effect of shape parameter for RBF approximation. Function used is $f(x) = \text{sech}^2(x)$ on $[-1,1]$ with $N = 50$. The interval $[-1,1]$ divided into 25 evenly spaced points provides the RBF centers for this example. In figure 6.3b the evaluation matrices of first and second order are used (black and green lines respectively). 74

6.4 Shape parameters used are from the set $\{50, 10, 5, 2.5, 1, 10^{-8}\}$. 75

6.5 This figure shows that a small shape parameter is not always beneficial. . . . 76

6.6 Gaussian RBF approximation for the advection equation, $N = 100$, $\Delta t = 1/300$, $u(x,0) = e^{(-50(x+0.5-t)^2)}$, boundary condition $u(x,0) = 0$. 78

6.7 Stencil example: a domain with 8 grid points and a stencil size of $m = 4$. At the node x_2 (blue square), the $m - 1 = 3$ nearest neighbors (red circles) are used in calculations. 81

6.8 Sparsity patterns and the relationship between accuracy and stencil size. Local IMQ RBF was used to approximate the first derivative of the Runge function $1/(1 + 25x^2)$ on the interval $[-1, 1]$ (100 equally spaced points, and a blanket shape parameter of 5). 82

6.9 Spatial convergence for GA–RBF spectral method (single soliton initial condition). Relevant parameters: $\Delta t = 0.1, T_{\text{final}} = 10$. 86

6.10 Eigenvalues of the linearized differentiation operator $-\mathbf{W_x} - \mathbf{W_{xxx}}$. GA–RBF, $N = 400$, $\epsilon = 1$, and stencil size is 289. 87

6.11 Local RBF method. Figures 6.11c , and 6.11d have a large stencil size of 289, with $N = 400$. 88

6.12 Error vs stencil size for a local GA–RBF collocation method. 91

6.13 Spatial convergence for GA–RBF spectral method (single soliton initial condition). Notice in figure 6.13a the right boundary is providing dominant errors. 93

6.14 Plots of exact solution and MQ–RBF approximation. 95

6.15 Spatial convergence for MQ–RBF spectral method (single soliton initial condition). Notice in figure 6.15a the boundaries are providing dominant errors. The errors are comparing the function $u(x, t)$ ($\eta(x, t) \equiv 1$, and approximating a constant function in this case is trivial). 97

6.16 RBF spectral method for 2D SW equations (same water droplet in a bathtub initial condition used in figure 5.7). 98

6.17 Spatial convergence for Gaussian–RBF spectral method applied to the Dutykh et al. test cases [25]. Error is in the infinity norm, $\|\text{approx} - \text{exact}\|_\infty$. . 104

6.18 Gaussian–RBF spectral method simulation of a head on collision. Relevant parameters: $\epsilon = 2$, $N = 300$, $L = 30$, $T = 36$, $a = 0.15$, and $x_0 = \pm 20$. This test case can also be found in [25]. 104

6.19 Spatial convergence for Gaussian–RBF spectral method applied to the Bonneton et al. test cases [10]. Error is relative, $\|\text{approx} - \text{exact}\|_\infty / \|\text{exact}\|_\infty$. . 105

LIST OF TABLES

Table Number Page

2.1 Common coefficients for the KdV equation 4
2.2 Code Listing for some spectral methods (KdV equation) 11
2.3 Classification of the behavior of small dispersion KdV equation 27
3.1 Code Listing for some spectral methods (BBM equation) 31
3.2 Numerical experiment for the BBM equation. 35
4.1 Code Listing for a spectral method (CH equation) 40
5.1 Test problems for the 1D SW equations. 50
5.2 Code Listing for a pseudospectral method (SW equation) 51
6.1 Popular RBF functions (this table is adapted from [34]). 66
6.2 Test problem for the GA–RBF spectral method (KdV equation). 84
6.3 Test problem for the GA–RBF spectral method (BBM equation). 91
6.4 Test problem for the MQ–RBF spectral method (Boussinesq system). 94
6.5 Test problems for the 1D SGN equations. 103

ACKNOWLEDGMENTS

I must thank my thesis advisor, Professor Randall J. LeVeque. This work would not have been possible without him. He suggested the project, and I was able to complete it because of his patience, encouragement, and support. I would also like to thank K. David Prince, Professor Anne Greenbaum, and Professor James A. Morrow for helping develop and foster my interest in mathematics.

DEDICATION

to Merle, Elizabeth, Bert, and Sean

Chapter 1

INTRODUCTION

Spectral methods are a class of numerical methods that are often applied to approximating solutions to partial differential equations (PDEs). The basic idea is straight-forward: attempt to write the solution to the PDE as a linear combination of basis functions, and then select the coefficients so that the resulting linear combination approximates the solution decently. One main feature of spectral methods is that they provide so-called spectral accuracy, which means that the accuracy follows an exponential convergence rate. Other popular numerical techniques have accuracy constraints based on the order of the method being employed; say finite differences or finite elements are good contrasting examples. So, even if the function being approximated is analytic or C^∞, a finite difference scheme could for example obtain an accuracy of $O((\Delta x)^3)$ as the grid spacing Δx tends to zero.

Spectral methods do have a number of limitations. Time dependent PDEs introduce a handful of issues for spectral methods. One such issue is: "how do we discretize the spatial and temporal dimensions?" For time dependent PDEs it is common to evaluate the spatial derivatives spectrally, and use finite differences temporally. This puts an immediate constraint on the spectral method, since one will achieve spectral accuracy in space, but will only achieve $O((\Delta t)^p)$ as $\Delta t \to 0$ in time (it is also feasible to take $\Delta t \ll \Delta x$, which can result in comparable accuracy). The value of p varies, sometimes one would like to match the spatial discretization error, so p could be taken large. However, for computationally intensive problems, p could be taken to be smaller.

The theory dealing with convergence rates for spectral collocation and spectral Galerkin methods states that the speed of convergence depends on the smoothness of the function being approximated. For example, if a function being approximated is Lipshitz continuous or only C^0, then the rate of convergence will be significantly slower (than the cases where the function being approximated are much smoother).

Boundary conditions illustrate a particularly limiting issue for Fourier Spectral methods. The most natural boundary conditions for these spectral methods are periodic, however, many problems have boundary conditions other than periodic that must be enforced. Further complications arise when considering problems that require a system of PDEs. Of course, one can change the global basis. Chebyshev spectral methods are often used for nonperiodic problems.

There are a number of other numerical techniques for approximating time dependent PDEs that are more robust than spectral methods, but a numerical approximation based on spectral methods could be used to produce a high accuracy approximation reference.

1.1 The Purpose of This Thesis

An important portion of this thesis is dedicated to explicit code examples. Many authors in current (and past) research literature provide results from a numerical experiment but do not provide the code that generated said experiment. This is problematic for students or researchers who cannot recreate the results described. In other cases students and researchers would have to invest a large amount of time to recreate the results described. The programming language used in this thesis is MATLAB, version 8.1.0.604 (R2013a). It should be noted that some of the code segments presented in this thesis have been adapted from other sources:

- IFRK4 (KdV equation): adapted from Trefethen[1] [82].
- ETDRK4 (KdV equation): adapted from Kassam and Trefethen [59].
- RBF differentiation : adapted from Alfa R.H. Heryudono[2].

Chapters 2, 3, 4, and 5 will be dedicated to Fourier type spectral methods. So in these chapters "spectral methods" will refer to Fourier spectral methods, unless specified otherwise. Chapter 6 will focus on radial basis function spectral methods. Sample code files used in this thesis can be found at the GitHub page https://github.com/msfabien/.

[1] University of Oxford Mathematical Institute

[2] Department of Mathematics at the University of Massachusetts Dartmouth

Chapter 2

FOURIER SPECTRAL METHOD FOR THE KORTEWEG–DE VRIES (KDV) EQUATION

The Korteweg-de Vries equation (KdV equation) is a particularly nice PDE to consider for first studying wave type behavior. A very useful feature of the KdV equation is that it is nonlinear but has exact smooth solitary traveling wave solutions. One is introduced to issues of non-linearity, but also able to exploit exact solutions in the process. Interestingly enough, interaction between different solitary waves yields particle-like behavior. See [43] and [20] for more detail about the KdV equation, applications, and history.

Section 2.1 gives a quick note about the coefficients of the KdV equation. Different authors use different coefficients, however these choices can be made arbitrary. The most frequent selections are also provided in this section.

Section 2.2 has an overview of exact smooth traveling wave solutions that the KdV equations admits. Most literature involving error analysis use the single soliton solution for simplicity. Exact solutions for the interaction of multiple solitons are also available.

Section 2.3 explains the algorithm (loosely called *spectral methods*) used to approximate solutions to the KdV equation. Some concepts that are needed for the algorithm are presented, but only at the level of working knowledge. References to more concise coverage of these concepts are provided. The algorithm is walked through step by step explaining the general procedure. Stability and convergence are briefly mentioned, and the reader can visit more exhaustive explanations if necessary.

In section 2.4 there are explicit code examples implementing the algorithm described in section 2.3. In addition, key steps of the implementation are described in detail. These key steps are the integrating factor spectral method with a 4th order Runge-Kutta time-stepping (IFRK4), and the exponential time differencing spectral method with a 4th order Runge-Kutta time-stepping (ETDRK4/mETDRK4). Three spectral methods are compared, the pseudo-spectral method, IFRK4, and mETDRK4, and the mETDRK4 method outperforms the other two.

Section 2.5 explores a variation of the KdV equation (small dispersion limit) which adds a small parameter ϵ^2 to the dispersive term u_{xxx}. The focus in current literature for the small dispersion KdV equation has been on its asymptotic solutions and approximations. Even though this is the case, spectral methods can still provide accurate references for asymptotic solutions and approximations.

2.1 KdV equation coefficients

The KdV equation is a scalar equation, given by

$$\frac{\partial u}{\partial t} + \beta u \frac{\partial u}{\partial t} + \alpha \frac{\partial^3 u}{\partial x^3} = 0, \tag{2.1}$$

where $u = u(x,t)$ and α, β are non-zero real constants (see [43] and [82]). There seems to be no common convention for the choice of α and β, however, popular choices for these constants are:

α	β
1	1
6	1
6	-1

Table 2.1: Common coefficients for the KdV equation.

As long as α and β are non-zero, the solutions of interest do not change their global behavior. By a simple substitution one can select α and β arbitrarily with the restriction that neither constant can be zero. To see this, let $x = aX$, $t = bT$, and $u = cU(X,T)$. From the chain rule we have that

$$\frac{\partial u}{\partial T} = \frac{\partial u}{\partial t}\frac{\partial t}{\partial T} = bu_t, \quad \text{and} \quad \frac{\partial u}{\partial X} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial X} = au_x.$$

Rewriting equation (2.1) in terms of U, X, and T gives us

$$c\frac{U_T}{b} + c^2\frac{\beta U U_X}{a} + \frac{\alpha c U_{XXX}}{a^3} = 0.$$

Upon simplification it follows that

$$U_T + bc\frac{\beta U U_X}{a} + \frac{\alpha b U_{XXX}}{a^3} = 0.$$

As a specific example, if we wanted coefficients of one in the KdV equation we could select $a = 1$, $b = 1/\alpha$, and $c = \alpha/\beta$.

2.2 Exact smooth solitary traveling wave solutions for the KdV equation

The KdV equation admits simple traveling wave solutions ($u(x,t) = f(\eta)$, $\eta = x - ct$). For the normalized KdV equation ($\alpha = \beta = 1$) the "single soliton" solution is given by

$$u(x,t) = 3v^2 \text{sech}^2[v((x - x_0 - v^2 t)/2], \tag{2.2}$$

where v^2 is the wave speed and x_0 is the initial location of the wave profile. If $\alpha = 1$ and $\beta = 6$ the single soliton solution is given by

$$u(x,t) = (v/2)\text{sech}^2[\sqrt{v}(x - x_0 - vt)/2],$$

where v is the wave speed and x_0 is the initial location of the wave profile. One can easily verify that these are indeed solutions to the KdV equation with the correct corresponding coefficients. Haberman in [43] has a derivation of this single soliton solution using phase plane analysis. Cnoidal waves (naturally periodic traveling wave) are another type of exact solution to the KdV which should be better for Fourier spectral methods, see Bottman and Deconinck in [81].

A celebrated behavior of these soliton solutions is how they interact with each other. In particular, an N soliton interaction results in the solitons "passing through" one another; as if they were unaltered in shape and speed by a collision with other solitons. Of course in real world applications "true" soliton behavior is not observed – in many cases small oscilatory residuals are left over, and small amplitude changes occur (small changes in propagation directions are common in higher dimensional cases). Formulas for exact solutions representing the interaction of N solitons can be found in [91].

Below we present an exact soliton solution for $N = 2$ interactions (due to Whitham in [91]):

$$u(x,t) = \frac{12[\alpha_1^2 f_1 + \alpha_2^2 f_2 + 2(\alpha_1 - \alpha_2)^2 f_1 f_2 + \epsilon^2(\alpha_2^2 f_1^2 f_2 + \alpha_1^2 f_1 f_2^2)]}{(1 + f_1 + f_2 + \epsilon f_1 f_2)^2}, \tag{NE.1}$$

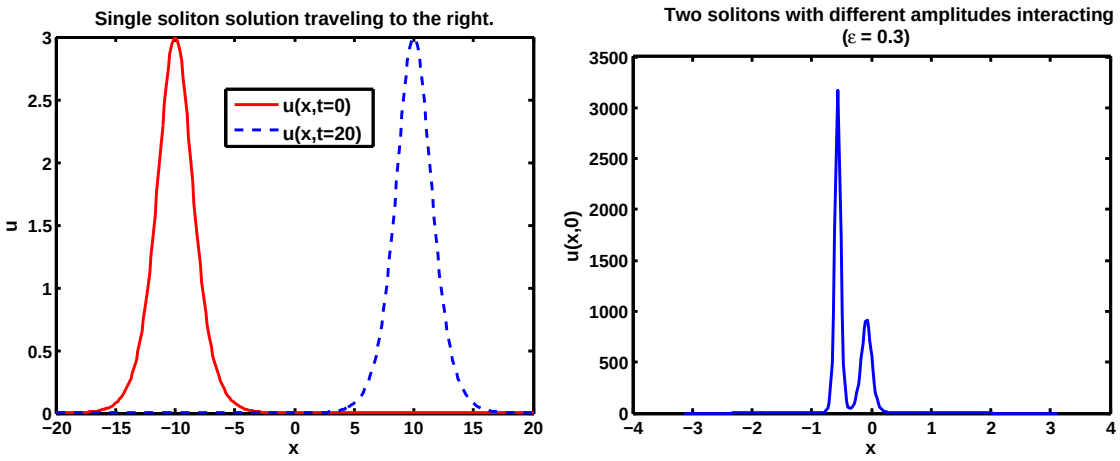
where

$$f_j = \exp\left[-\alpha_j(x - s_j) + \alpha_j^3 t\right], \quad 1 \le j \le N, \text{ and} \quad \epsilon = \frac{\alpha_2 - \alpha_1}{\alpha_2 + \alpha_1}. \tag{NE.2}$$

The parameter ϵ in this case allows the user to specify how much the two solitons interact at $t = 0$. If the user wants the soliton's peaks to be further apart, then one can simply use a transformation in t or just use negative t values. Figure 2.1 visualizes this interaction for two solitons of different amplitudes and $\epsilon = 0.2$. A noteworthy feature of these soliton solutions (for any $N \in \mathbb{N}$) is that the solutions decay to zero as $x \to \pm\infty$. In addition to their rapid decay for large x, the soliton solutions are smooth. This smoothness accelerates the convergence of the spectral method. Trefethen in [83] and [82] has more detail on how the smoothness of a function being approximated effects the performance of spectral-type approximations. Spectral accuracy refers to convergence at the rate $\mathcal{O}(C^{-N})$ for $C > 1$.

In equation (2.2) has the form of a squared hyperbolic secant function, which is really the product of exponential functions. Hence, we can expect to obtain spectral accuracy (in the spatial dimensions) since $u(x,\cdot)$ from equation (2.2) is a C^∞ function.

Since the KdV equation admits explicit exact smooth solutions, we can easily implement these solutions and then compare them to numerical approximations of the KdV equation.



(a) (b) $0 \leq t \leq 5 \cdot 10^{-4}$

Figure 2.1: (a) Single soliton solution (from equation (2.2)) traveling to the right with a speed $v = 1$. (b) Two solitons with nearly equal amplitudes interacting.

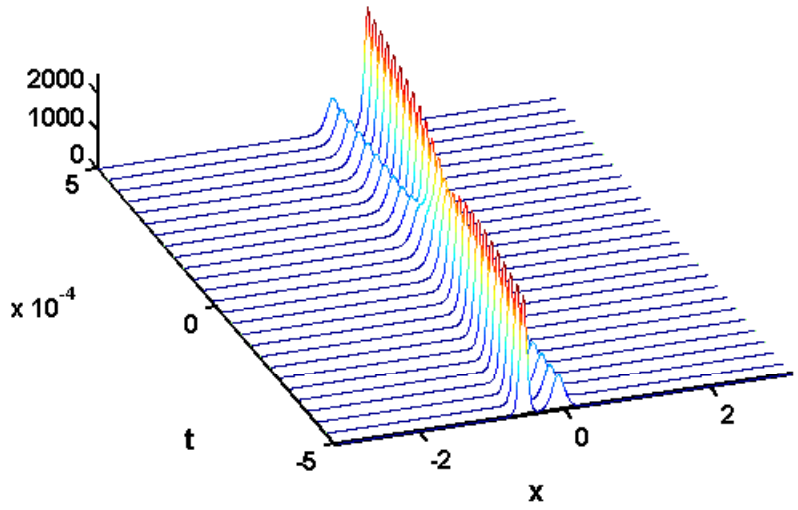


Figure 2.2: Two solitons with nearly equal amplitudes interacting ($\epsilon = 0.3$ in equations (NE.1) and (NE.2)).

2.3 Fourier Spectral Methods for the KdV equation

In this section a general spectral method will be presented, and in addition, this method will be applied/implemented throughout this thesis. The general idea is very straight forward:

Algorithm for Fourier spectral method

- Step 1: Select a computational domain (say $[-\pi L, \pi L]$ or $[0, 2\pi L]$) and scale the PDE accordingly.
- Step 2: Take the Fourier transform of the scaled equation.
- Step 3: Use an integrating factor or a exponential integrator to solve the linear term exactly (if applicable).
- Step 4: Use the method of lines to rewrite the PDE as a system of ODE's in time. The exponential integrator alleviates the often stiff system of ODE's.
- Step 5: Apply a time stepping technique to solve the resulting system of ODE's.

The set of steps above give a general idea of the flow of the algorithm. Other advanced steps can be included, for instance, preconditioning, filtering, dealiasing, smoothing, and other post-processing techniques.

To start we state the general abstract problem. We would like to solve the KdV equation[1] given an initial condition and enforcing periodic boundary conditions:

$$\begin{aligned} u_t + uu_x + u_{xxx} = 0, \quad & x \in [-L, L],\ t > 0 \\ u(x, 0) = f(x), & \\ u(-L, t) = u(L, t), \quad & t \geq 0. \end{aligned}$$

2.3.1 Step 1

The first step is to use a change a variables to scale the domain of $[-L, L]$ to a computational domain of $[-\pi, \pi]$ (This is done because eventually we will discretize the Fourier transform and employ the Fast Fourier Transform). To do this, make a change of variables: let $x = Xb$, and $u(x, t) = v(x/b, t) = v(X, t)$ where $b = \pi/L$. Then the KdV equation becomes

$$v_t + (1/b)vv_X + (1/b^3)v_{XXX} = 0. \tag{2.3}$$

Notice that if $L = \pi$, then $b = 1$ (in other words, the natural domain is equal to the computational domain).

[1] Without a loss of generality we use the normalized KdV equation ($\alpha = \beta = 1$ in equation (2.1)).

2.3.2 Step 2

Let the continuous univariate Fourier transform be denoted by the symbol $\mathcal{F}(\cdot)$.

Definition 2.3.1. Given a function $f \in C^0$, the Fourier transform of f is

$$\mathcal{F}(f(t)) = \hat{f}(k) = \int_{-\infty}^{\infty} f(t)e^{-2\pi ikt}\,dt.$$

Definition 2.3.1 requests that f is continuous, however, this condition can be weakened. In order for spectral accuracy f will have to be smoother than just continuous. Also, the reader may be aware that there are many different (but equivalent) Fourier transform definitions. The integral form in definition 2.3.1 was adopted for simplicity. For more information see the comprehensive Fourier analysis book [51].

There are two properties of the Fourier transform that we are mainly interested in for the spectral method. The first property is linearity.

Fourier Transform property 1. Given functions $f, g \in C^0$ and scalars $a, b \in \mathbb{R}$, we have $\mathcal{F}(af + bg) = a\mathcal{F}(f) + b\mathcal{F}(g)$

The second property deals with differentiation identities:

Fourier Transform property 2. If $f \in C^n$, then

$$\mathcal{F}(f^{(n)}(t)) = (ik)^n \mathcal{F}(f(t)),$$

where $f^{(n)}$ is the nth derivative of f and i is the imaginary unit.

Taking the Fourier transform of both sides of equation (2.3) yields

$$\hat{v}_t + (ik/b)\mathcal{F}(v^2) - (ik^3/b^3)\hat{v} = 0, \tag{2.4}$$

where $\hat{v} = \mathcal{F}(v)$. The nonlinear term $(1/b)vv_x$ in equation (2.3) is handled by noticing that $vv_x = (1/2)(v^2)_x$.

2.3.3 Step 3

In equation (2.4) we can isolate linear and non-linear terms in the form of $L(\hat{v}) + N(\hat{v}) = \hat{v}_t$:

$$\left(\frac{\partial}{\partial t} - \frac{ik^3}{b^3}\right)\hat{v} - (ik/b)\mathcal{F}(v^2) = 0.$$

To reduce numerical stiffness it is commonplace to use an integrating factor or exponential integrator to exactly solve the linear portion of equation (2.4). Since integrating factors are a standard technique in the theory of linear differential equations, they will be introduced first (for an introductory discussion of integrating factors, see chapter 8 of [12]).

2.3.4 Step 4

The method of integrating factors seeks a function $\mu(t)$ (called an integrating factor) such that the linear portion of a differential equation multiplied by μ can be written as the derivative of a product. In our case:

$$\mu(t)\left(\hat{v}_t - \frac{ik^3}{b^3}\hat{v}\right) = (\mu(t)\hat{v})_t.$$

In order for this equality to hold, the function $\mu(t)$ must take the form

$$\mu(t) = \exp\left[\int^t (-ik^3/b^3)\, d\tau\right] = \exp\left(-ik^3 t/b^3\right).$$

Hence, multiplying equation (2.4) by μ we have

$$(\mu(t)\hat{v})_t = -(ik/b)\mathcal{F}(v^2)\mu(t). \tag{2.5}$$

To complete the integrating factor method, we make a change of variables in equation (2.5), namely: $w(X,t) = \mu(t)v$. We then find that equation (2.5) becomes

$$(\hat{w})_t = -(ik/b)\mathcal{F}((w\mu^{-1})^2)\mu(t). \tag{2.6}$$

Equation (2.6) has the distinct advantage of recovering the linear portion of the differential equation exactly when one applies an ODE solver. In practice one replaces t with Δt in the function $\mu(t)$, however, this occurs naturally in the discretization process for the PDE's considered in this thesis.

2.3.5 Step 5

To approximate equation (2.6) numerically, we can use the well known method of lines. If we discretize the spatial dimension in equation (2.6), this results in a system of coupled ODEs that we can approximate using an ODE solver. A concise text on the method of lines can be found in [72].

Truncating to a finite domain, we divide the interval $[-L, L]$ into N evenly spaced grid points defined by $x_j = -L + 2Lj/N$, for $j = 0, \ldots, N-1$. Then, one can truncate the Fourier transform of $\hat{w}(k,t)$, let $\hat{w}(k,t) = (1/N)\sum_{j=0}^{N} w(x_j,t)e^{-ikx_j}$, for $k = -N/2+1, \ldots, N/2$. In addition, the inverse Fourier transform is truncated in a similar fashion:

$$w(x_j,t) = \sum_{k=-N/2+1}^{N/2} \hat{w}(k,t)e^{ikx_j}.$$

Finally, the Fourier transform differential identity has a straight forward evaluation on the grid:

$$\mathcal{F}\left(\frac{\partial^n}{\partial x^n}w(x_j,t)\right) = (ik)^n\mathcal{F}(w(x_j,t))$$

With this spatial discretization a system of ordinary differential equations arises from equation (2.6):

$$\frac{d\widehat{w}(k,t)}{dt} = -(ik/b)\mathcal{F}((w(x_j,t)/\mu(t))^2)\mu(t), \quad k = 0, \pm 1, \ldots \pm N/2. \tag{2.7}$$

As mentioned above the Fourier transform has many variations of its definition, and this is also true of the discrete Fourier transform. For more details on the Fourier transform discretization we refer the reader to [51]. The derivation above of the discretization was brief; for a slower analysis with examples see [4]. The work in [4] also explores numerical methods for stiff systems more thoroughly.

2.3.6 Step 6

To complete the approximation of the KdV equation one needs to apply an ODE solver to the system of ODEs found in equation 2.7. By applying an ODE solver, we are no longer achieving spectral accuracy. In practice if one uses a time stepping technique of order 2 or higher the global accuracy is still reasonable[2] , and in many cases competitive with other numerical methods. In many situations a high order time stepping technique is used to complement the spectral accuracy received in the spatial discretization.

The choice of a time stepping technique is largely up to the user, however, ensuring that the method of lines is stable may influence this choice. In chapter 10 of [82], Trefethen gives a "rule of thumb":

> The method of lines is stable if the eigenvalues of the (linearized) spatial discretization operatior, scaled by Δt, lie in the stability region of the time-discretization operator.

The theory for many numerical methods for PDEs is well behind that of its practice, so the "rule of thumb" is just a general guide and might not apply to every problem. Since time stepping techniques often have different stability regions; this rule of thumb could play an important role in selecting a time stepping techniques. There is a wealth of information (including numerical examples) about time stepping techniques and stability regions in [82]. In the case of the KdV equation, Trefethen in [82] gives a stability requirement of $\Delta t \leq 0.4/N^2$ (the time stepping used is the so-called Runge Kutta 4th order method[3]). This inequality can be found by discretizing the continuous operator $\partial^3/\partial x^3 + \partial/\partial x$ and studying its eigenvalues.

---

[2] According toTrefethen in [82]

[3] See [53] for an in-depth overview of the Runge-Kutta methods

Remark 2.3.1. The integrating factor method applied in subsection 2.3.4 (which is also applied in [82]) is no longer state of the art. Methods for stiff ordinary differential equations called *exponential integrators* have been investigated as early as the 1960's (see [66] and [14]). Cox and Matthews in [69] provide a fourth order exponential time differencing (ETDRK4) method which improves upon the integrating factor method. However, the algorithm they presented suffers from numerical instability. Trefethen and Kassam in [59] explore the numerical instability of the ETDRK4 method, and give a modified ETDRK4 that is numerically stable and is claimed to be the first fully practical ETD method for general use. Their paper includes MATLAB sample codes that implement this modified ETDRK4 method. Implementing the modified ETDRK4 method (mETDRK4) for the KdV equation does give an improvement over the IFRK4 method from [82].

In subsection 2.3.2, the nonlinear term in the KdV equation (vv_x) was evaluated spectrally by using the identity $vv_x = (1/2)(v^2)_x$. For more complicated nonlinear terms, it might be useful to "pseudo-spectrally" evaluate the nonlinear term. This is done by noting that $v_x = \Re[\mathcal{F}^{-1}(ik\hat{v})]$, and then approximating vv_x by the term $v\Re[\mathcal{F}^{-1}(ik\hat{v})]$. This pseudo-spectral approach has the advantage of being able to address "severe" nonlinearities. A monograph on pseudo-spectral methods by Fornberg can be found in [31]. For many PDEs the pseudo-spectral approach is restricted to much smaller time steps (see subsection 2.4.1).

The "rule of thumb" (and stability analysis) described in subsection 2.3.6 is not restricted to the KdV equation, it applies to other time dependent PDEs.

2.4 Implementation of the Fourier Spectral Method

In this section explicit code examples are given for approximating the KdV equation using spectral methods. After the code has been presented comments, figures, and analysis will follow.

Method	Code Listing
Pseudo-Spectral	2.4.1 and 2.4.2
IFRK4	2.4.3
mETDRK4	2.4.4

Table 2.2: Organization of spectral method and their respective code listings (KdV equation).

Code Listing 2.4.1

Main Psuedo-Spectral file

```
% Approximate the solution to the KdV equation u_t + uu_x + u_xxx = 0 on
% the domain [-pi,pi] with periodic boundary conditions via a
% pseudo-spectral method with RK4 time-stepping scheme.

clear all;  close all;  clc;
N = 2^7;                                % Mesh refinement
L = 2*pi;                               % Domain length
dt = 0.05*(L/N)^3;
x  = (2*pi/N)*(-N/2:N/2-1)';
x0 = -2;                                % Soliton initial position
A  = 16;                                % Soliton amplitude
U = @(x,t) 3*A^2*sech(A*(x - x0)/2 - A^3*t/2).^2;
u = U(x,0.0);
u0 = u;
tf = 0.006;                             % Final time
nt = round(tf/dt);                      % Total timesteps
for index = 1:nt
a = -dt*RHS_KdV(u);                     % March forward du/dt = -RHS_KdV(u)
b = -dt*RHS_KdV(u + 0.5*a);             % using the RK4 scheme
c = -dt*RHS_KdV(u + 0.5*b);
d = -dt*RHS_KdV(u + c);
u = u + (a + 2*b + 2*c + d)/6;          % Update solution u
end
plot(x,u,x,u0,x,U(x,tf),'.')
norm(U(x,tf) - u,inf)
```

Code Listing 2.4.2

Psuedo-Spectral function

```
function U = RHS_KdV(u)
% This function pseudospectrally calculates the spatial derivatives in the
% KdV equation u_t = - ( uu_x + u_xxx ).  We can deal with the nonlinear
% term uu_x easily this way.
  N = length(u);
  uhat = fft(u);
  k = [0:N/2-1 0 -N/2+1:-1]';
  ux = real(ifft(1i*k.*uhat));
  uxxx = real(ifft(-1i*k.^3.*uhat));
  U = u.*ux + uxxx;
```

Code Listing 2.4.3

Main IFRK4 file

```
% Approximate the solution to the KdV equation u_t + uu_x + u_xxx = 0 on
% the domain [-pi,pi] with periodic boundary conditions via a spectral
% integrating factor method with a RK4 (IFR4) time-stepping scheme.

clear all;  close all;  clc;

% Set up grid and single-soliton initial data:
N = 2^7;                                          % Mesh refinement
L = 2*pi;                                         % Domain length
dt = 0.4/N^2;
x  = (L/N)*(-N/2:N/2-1)';
x0 = -2;                                          % Soliton initial position
A  = 5;                                           % Soliton amplitude
tf = 0.1;                                         % Final time

U   = @(x,t) 3*A^2*sech(A*(x - x0)/2 - A^3*t/2).^2;
u   = U(x,0);                                     % Initial condition
UF  = U(x,tf);                                    % Final solution
v   = fft(u);
k   = [0:N/2-1 0 -N/2+1:-1]';
ik3 = 1i*k.^3;

nmax = round(tf/dt);
for index = 1:nmax
        g = -.5*1i*dt*k;
    E = exp(dt*ik3/2);                            % Integrating Factor
    E2 = E.^2;
    a = g.*fft(real( ifft(     v    ) ).^2);
    b = g.*fft(real( ifft(E.*(v+a/2)) ).^2);      % 4th-order Runge-Kutta
    c = g.*fft(real( ifft(E.*v + b/2) ).^2);      % with integrating factor
    d = g.*fft(real( ifft(E2.*v+E.*c) ).^2);
    v = E2.*v + (E2.*a + 2*E.*(b+c) + d)/6;       % Update approximation
    u = real(ifft(v));
end
plot(x,u,x,UF,'r.')
norm(u - UF,inf)
```

Code Listing 2.4.4

Main mETDRK4 file

```
% Approximate the solution to the KdV equation u_t + uu_x + u_xxx = 0 on
% the domain [-pi,pi] with periodic boundary conditions via a spectral
% modified exponential time differencing method with a RK4 (mETDR4)
% time-stepping scheme.
clear all;  close all;  clc;
N = 2^7;                                          % Mesh refinement
L = 2*pi;                                         % Domain length
dt = 0.4/N^2;  h = dt;                       % Time step
x  = (L/N)*(-N/2:N/2-1)';
x0 = -2;                                          % Soliton initial position
A  = 5;                                           % Soliton amplitude
tf = 0.1;                                         % Final time
U   = @(x,t) 3*A^2*sech(A*(x - x0)/2 - A^3*t/2).^2;
u   = U(x,0);                                     % Initial condition
UF  = U(x,tf);                                    % Final solution
% Precompute mETDRK4 scalar quantities (due to Kassam-Trefethen):
k = [0:N/2-1 0 -N/2+1:-1]';             % Wave numbers
L = 1i*k.^3;                            % Fourier multipliers
E = exp(h*L); E2 = exp(h*L/2);
M = 64;                                 % Number of points for complex
                                        % means
r = exp(2i*pi*((1:M)-0.5)/M);           % Roots of unity
LR = h*L(:,ones(M,1))+r(ones(N,1),:);
Q  = h*mean(                  (exp(LR/2)-1)./LR   ,2);
f1 = h*mean((-4-LR+exp(LR).*(4-3*LR+LR.^2))./LR.^3,2);
f2 = h*mean(    (4+2*LR+exp(LR).*(-4+2*LR))./LR.^3,2);
f3 = h*mean((-4-3*LR-LR.^2+exp(LR).*(4-LR))./LR.^3,2);
g = -.5i*k;
% March forward in time via ETDRK4 formula (due to Cox-Matthews):
v = fft(u);
nmax = round(tf/dt);                    % Total timesteps
for index = 1:nmax
    Nv = g.*fft(real(ifft(v)).^2);
    a = E2.*v+Q.*Nv;        Na = g.*fft(real(ifft(a)).^2);
    b = E2.*v+Q.*Na;        Nb = g.*fft(real(ifft(b)).^2);
    c = E2.*a+Q.*(2*Nb-Nv); Nc = g.*fft(real(ifft(c)).^2);
    v = E.*v+(Nv.*f1+(Na+Nb).*f2+Nc.*f3);
    u = real(ifft(v));
end
plot(x,u,x,UF,'r.'),  norm(u - UF,inf)
```

2.4.1 Pseudo-spectral method

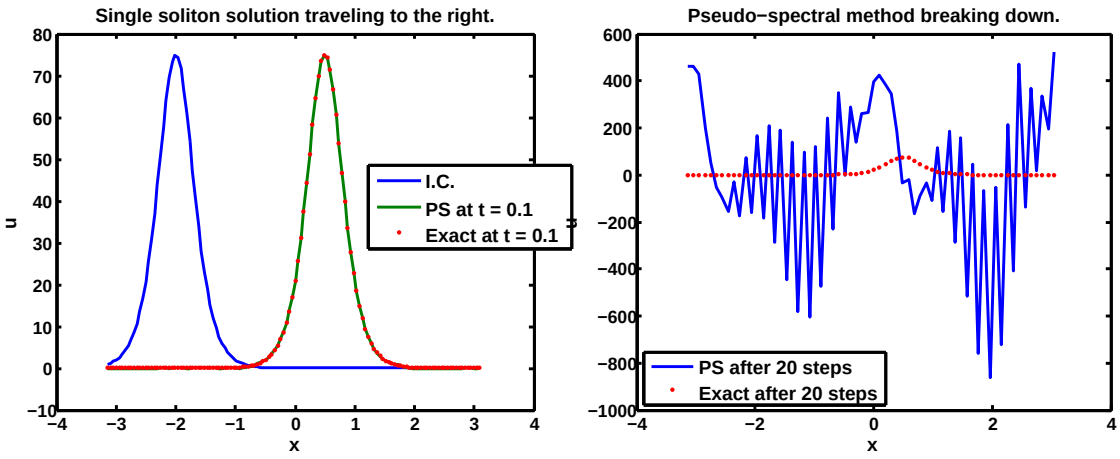
To select a time step for the pseudo-spectral method for the KdV equation, we can use the so-called *oscillation equation*: $d\psi(t)/dt = i\omega\psi(t)$. If the value of $|\omega\Delta t|$ is too large, the solution ψ will amplify. For the RK4 method it turns out that the maximum value of $|\omega\Delta t|$ before ψ will amplify is around 2.82 (see [24] for more information about the oscillation equation). Notice that in the KdV equation the term that has the largest wave numbers is u_{xxx}; this term provides a factor of $(ik)^3$. Then, to ensure the RK4 method produces non-amplifying ψ, we must have $2.82 > \max_n |\omega_n \Delta t|$, where $\omega_n = -k_n^3$, and k_n are the discretized wave numbers. But then we have

$$2.82 > \max_n |\omega_n \Delta t| = \max_n \left| \left(\frac{\pi N}{L} \right)^3 \Delta t \right|, \tag{2.8}$$

and it follows that $\Delta t < \Delta t_c = (2.82/\pi^3)(L/N)^3 = (2.82/\pi^3)(\Delta x)^3 \approx 0.09(\Delta x)^3$. The variable Δt_c is called the 'critical' time-step. In Code Listing 2.4.1, notice the time step selected is

$$\Delta t = 0.05(\Delta x)^3 < 0.09(\Delta x)^3 = \Delta t_c.$$

This time-step is much more restrictive than that of the IFRK4 time-step $(0.4/N^2)$ discussed in subsection (2.3.6). For the pseudo-spectral method, the inequality in (2.8) really does need to hold. See figure 2.3 (b), which plots an approximation that violates the RK4 stability limit given in (2.8). After only 20 steps (there are a total of around 1000 steps until the final time is reached) the method is promptly diverging.



(a) $\Delta t = 0.05(\Delta x)^3$ (b) $\Delta t = 0.11(\Delta x)^3$

Figure 2.3: (a) Pseudo-spectral method with Δt that satisfies (2.8). (b) Pseudo-spectral method with Δt that does not satisfy (2.8).

The pseudo-spectral method is not a particularly robust numerical method. Since the restriction on the time-step is harsh, reducing it any further will add considerable computational time to the method. The combination of stiffness and nonlinearity causes this method to be inefficient in many cases. However, what makes the pseudo-spectral method attractive is that it can evaluate complicated nonlinear terms by evaluating the spatial derivatives spectrally.

2.4.2 Spectral integrating factor method (IFRK4)

The IFRK4 offers an improvement over the pseudo-spectral approach. In section 2.3 the IFRK4 method was discussed step by step in detail. The RK4 update is a little different than the description in section 2.3, so in this section we will derive the RK4 update given in Code Listing 2.4.3.

Note that the variable $w(x,t) = mu(t)v(x,t)$ from section 2.3 is never defined in Code Listing 2.4.3. This is because it is easier to evaluate the nonlinear term $\mathcal{F}(v^2)$ using v and not w. Next we derive the RK4 forumlas used in Code Listing 2.4.3. Recall that the RK4 scheme for the problem $u' = f(t,u)$, $u(t_0) = u_0$ is given by

$$\begin{aligned} u_{n+1} &= u_n + (h/6)(a + 2b + 2c + d), \\ t_{n+1} &= t_n + h, \end{aligned}$$

where $h = \Delta t$ and

$$\begin{aligned} a &= f(t_n, u_n), \\ b &= f(t_n + h/2, u_n + ha/2), \\ c &= f(t_n + h/2, u_n + hb/2), \\ d &= f(t_n + h, u_n + hc). \end{aligned}$$

Let $w = ue^{-C(t-t_n)}$, where $C = ik^3$ in the case of the KdV equation ($e^{-C(t-t_n)}$ is the integrating factor found in section 2.3). For the KdV equation, the system of ODEs we want to update is given by

$$(\hat{w})_t = -(ik/B)\mathcal{F}((w\mu^{-1})^2)\mu(t),$$

Where B is the coefficient of b in equation (2.7). So, in terms of the variable w, the right hand side function $f = F(we^{Ct}, t)c^{-Ct}$, where $F(\cdot, t) = -(ik/b)\mathcal{F}((\cdot)^2)$. Then notice that the first RK4 increment a can be written as

$$a = f(w_n, t_n) = F(w_n e^{Ct}, t_n)e^{-Ct} = F(u_n, t_n)e^{-Ct_n}.$$

Let $F_n = F(u_n, t_n)$, $t_{n+1} = t_n + h$, and $t_{n+1/2} = t_n + h/2$. For the second RK4 increment b we have

$$\begin{aligned}
b &= f(w_n + ah/2, t_{n+1/2}) \\
&= F(e^{C(t_n+h/2)}[w_n + ah/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(e^{C(t_n+h/2)}[u_n e^{-Ct_n} + ah/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + ahe^{C(t_n+h/2)}/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + (F(u_n, t_n)e^{-Ct_n})he^{C(t_n+h/2)}/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + F_n he^{Ch/2}/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(e^{Ch/2}[u_n + ha/2], t_{n+1/2})e^{-C(t_n+h/2)}.
\end{aligned}$$

Following the same procedure as above, the third RK4 increment c we have

$$\begin{aligned}
c &= f(w_n + bh/2, t_{n+1/2}) \\
&= F(e^{C(t_n+h/2)}[w_n + bh/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(e^{C(t_n+h/2)}[u_n e^{-Ct_n} + bh/2], t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + be^{C(t_n+h/2)}h/2, t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + [F(e^{Ch/2}[u_n + ha/2], t_{n+1/2})e^{-C(t_n+h/2)}]e^{C(t_n+h/2)}h/2, t_{n+1/2})e^{-C(t_n+h/2)} \\
&= F(u_n e^{Ch/2} + 0.5h[F(e^{Ch/2}[u_n + ha/2], t_{n+1/2})], t_{n+1/2})e^{-C(t_n+h/2)}.
\end{aligned}$$

For the fourth RK4 increment d, observe that

$$\begin{aligned}
d &= f(w_n + ch, t_n + h) \\
&= F(e^{C(t_n+h)}(w_n + hc), t_n + h)e^{-C(t_n+h)} \\
&= F(e^{Ch}u_n + e^{C(t_n+h)}ch), t_n + h)e^{-C(t_n+h)} \\
&= F(e^{Ch}u_n + he^{\frac{Ch}{2}}F(u_n e^{\frac{Ch}{2}} + 0.5hF(e^{\frac{Ch}{2}}[u_n + ha/2], t_{n+1/2}), t_{n+1/2}), t_{n+1})e^{-C(t_{n+1})}.
\end{aligned}$$

The update is then

$$w_{n+1} = w_n + (h/6)(a + 2b + 2c + d), \tag{2.9}$$

but $w_n = u_n e^{-Ct_n}$ and $w_{n+1} = u_{n+1}e^{-C(t_n+h)}$, hence

$$u_{n+1}e^{-C(t_n+h))} = u_n e^{-Ct_n} + (h/6)(a + 2b + 2c + d).$$

The RK4 update is then given by

$$u_{n+1} = e^{Ch}u_n + e^{C(t_n+h)}(h/6)(a + 2b + 2c + d). \tag{2.10}$$

Even though a, b, c, and d have been derived, these are not the same as the variables used in Code Listing 2.4.3. In equation (2.10), notice the term $e^{C(t_n+h)}$ is still present. It

is inefficient to evaluate it as is, since the terms a, b, c, and d have factors of e^{-Ct_n}. In particular, notice that

$$e^{C(t_n+h)}a = F(u_n, t_n)e^{Ch},$$
$$e^{C(t_n+h)}b = e^{Ch/2}F(e^{Ch/2}[u_n + ha/2], t_{n+1/2}]),$$
$$e^{C(t_n+h)}c = e^{Ch/2}F(u_n e^{Ch/2} + 0.5h[F(e^{Ch/2}[u_n + ha/2], t_{n+1/2})], t_n + h/2),$$
$$e^{C(t_n+h)}d = F(e^{Ch}u_n + he^{\frac{Ch}{2}}F(u_n e^{\frac{Ch}{2}} + 0.5hF(e^{\frac{Ch}{2}}[u_n + ha/2], t_{n+1/2}), t_{n+1/2})), t_{n+1}).$$

Let a_c, b_c, c_c, and d_c be the RK4 increment used in Code Listing 2.4.3. Define a_c as

$$a_c = hF(u_n, t_n). \tag{2.11}$$

Then, define b_c as

$$b_c = hF(e^{Ch/2}[u_n + ha/2], t_{n+1/2}),$$

and notice that

$$b_c = hF(e^{Ch/2}[u_n + a_c/2], t_{n+1/2}). \tag{2.12}$$

Similarly, if we define c_c as

$$c_c = hF(u_n e^{Ch/2} + 0.5h[F(e^{Ch/2}[u_n + ha/2], t_{n+1/2})], t_{n+1/2}),$$

it follows that

$$c_c = hF(u_n e^{Ch/2} + 0.5b_c, t_{n+1/2}). \tag{2.13}$$

Finally if we define d_c as

$$d_c = hF(e^{Ch}u_n + he^{Ch/2}F(u_n e^{Ch/2} + 0.5hF(e^{Ch/2}[u_n + ha/2], t_{n+1/2}), t_{n+1/2})), t_n + h),$$

one arrives at

$$d_c = hF(e^{Ch}u_n + e^{Ch/2}c_c, t_n + h). \tag{2.14}$$

Rewriting equation (2.10) with respect to a_c, b_c, c_c and d_c we find that

$$u_{n+1} = e^{Ch}u_n + (1/6)(e^{Ch}a_c + 2e^{Ch/2}b_c + 2e^{Ch/2}c_c + d_c). \tag{2.15}$$

It is clear from equation (2.15) that one can simply update u_n, and that the integrating factor is constant throughout the time-stepping. Equation (2.15) is much more suitable for computational purposes. If one used a direct implementation using equation (2.10), this may cause inaccuracies since the integrating factor $\mu(t) = e^{-k^3(t-t_n)}$ depends on t. So as one updates from t_n to $t_{n+1/2}$ (or t_{n+1}), $\mu(t)$ is a non-constant function. However, the form of equation (2.15) does not have this issue since it occurs naturally that t is replaced with $h = \Delta t$ or $0.5\Delta t$ in $\mu(t)$.

Note that equation (2.15) is the update used in Code Listing (2.4.3). In addition, in the analytic derivation of the integrating factor method, there was a change of variables, namely $w(t) = u(t)\mu^{-1}(t)$. Equations (2.10) and (2.15) do not use $w(t)$, for the same reasoning as above: $\mu(t)$ would need to be updated. Equation (2.9) is equivalent to equation (2.10), but if we initialize our code with a u_0, then there is no need to use $w(t)$ which has a varying exponential that needs to be updated.

2.4.3 Spectral modified exponential time differenecing method (ETDRK4)

In this section we will provide a short introduction to exponential time differenecing methods, then compare the spectral mETDRK4 and spectral IFRK4 methods. For the KdV equation, mETDRK4 is observed to be more accurate.
As mentioned above, the integrating factor method is no longer state of the art; a technique called *Exponential Time Differencing* (ETD) has been shown to be more accurate, especially in the case of dissipative and dispersive partial differential equations (forced ordinary differential equations with stiff linear parts). In the case of KdV equation and IFRK4, Cox and Matthews state (in [69]):

> In addition to these multistep ETD methods, we have derived new RungeKutta forms of the ETD method, of second, third, and fourth order. These are easier to use than the high-order multistep forms, since they do not require initialization, and are more accurate. [Cox and Matthews in [69]]

To get an idea for where the ETD methods come from, we consider the model first order ODE:

$$\dot{u} = cu + F(u,t), \tag{2.16}$$

where c is constant and F represents nonlinear/forcing terms. Notice equation (2.16) has the form of $\dot{u} = L(u) + N(u)$, where L is linear function, and N is a non-linear function. Using the integrating factor of e^{-ct} (from the linear part of this ODE), we are able to rewrite the equation as

$$\frac{d}{dt}(e^{-ct}u) = e^{-ct}F.$$

To recover the original function we can integrate both sides. In terms of numerics, we can just integrate over the interval from $t = t_n$ to $t = t_n + h = t_{n+1}$. Let $u(t_n + h) = u_{n+1}$ and $u(t_n) = u_n$. Integrating over one time step yields

$$\int_{t_n}^{t_{n+1}} \left[\frac{d}{dt}(e^{-ct}u)\right] dt = \int_{t_n}^{t_{n+1}} e^{-ct}F\,dt \implies e^{-c(t_{n+1})}u_{n+1} - e^{-c(t_n)}u_n = \int_{t_n}^{t_{n+1}} e^{-ct}F\,dt.$$

Rewriting the numerical scheme gives

$$u_{n+1} = e^{ch}u_n + e^{c(t_n+h)}\int_{t_n}^{t_n+h} e^{-ct}F(u,t)\,dt.$$

By making a substitution of $\tau = t - t_n$, the scheme becomes

$$u_{n+1} = e^{ch}u_n + e^{c(t_n+h)}\int_0^h e^{-ct}F(u(t_n+\tau), \tau + t_n)\,dt. \tag{2.17}$$

At this point, the approach is identical to the IF method. The main difference between the IF and ETD schemes is the change of variables for IF is complete. The way one approximates the integral in (2.17) leads to different ETD methods. The simplest of which is the so-called

ETD1 scheme, which supposes that F is a constant, $F = F_n + \mathcal{O}(h)$ over the interval $t = t_n$ to $t = t_n + h$. As such, the ETD1 scheme is:

$$u_{n+1} = u_n e^{ch} + F_n(e^{ch} - 1)/c.$$

This scheme ends up having a local truncation error of $h^2\dot{F}/2$. One can improve on the local truncation error of these ETD schemes by selecting higher order approximations for the integral in (2.17). Cox and Matthews give ETD scheme using Runge-Kutta methods, denoted ETDRK (see [69]). Of particular interest is ETDRK4, the 4th order Runge-Kutta ETD scheme. It turns out that ETDRK methods of higher than order 2 suffer from numerical instability (see [59]). The main issue is that the expression

$$g(z) = \frac{e^z - 1}{z},$$

suffers from catastrophic cancellation.

Kassam and Trefethen in [59] explore the numerical instability of the ETDRK4 scheme, and even give a modified ETDRK4 that is numerically stable and is claimed to be the first fully practical ETD method for general use:

> In section 2 we propose a modification of the ETD schemes that solves these numerical problems. The key idea is to make use of complex analysis and evaluate certain coefficient matrices or scalars via contour integrals in the complex plane. Other modifications would very possibly also achieve the same purpose, but so far as we know, this is the first fully practical ETD method for general use. [Kassam and Trefethen in [59]]

Their paper includes MATLAB sample codes that implement this modified ETDRK4 method. Implementing the modified ETDRK4 method for the KdV equation does give an improvement over the IFRK4 method. Equation (2.17) can be generalized to a system of ODEs:

$$u_t = \mathbf{L}u + \mathbf{N}(u, t), \tag{2.18}$$

where $\mathbf{L}$ and $\mathbf{N}$ are the respective linear and non-linear discretization operators. The ETDRK4 scheme applied to the prototype problem in equation (2.18) proposed by Cox and Matthews is

$$a_n = u_n e^{\mathbf{L}h/2} + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - 1)F(u_n, t_n), \tag{2.19}$$

$$b_n = u_n e^{\mathbf{L}h/2} + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - 1)F(a_n, t_{n+1/2}), \tag{2.20}$$

$$c_n = a_n e^{\mathbf{L}h/2} + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - 1)(2F(b_n, t_{n+1/2}) - F(u_n, t_n)), \tag{2.21}$$

$$u_{n+1} = u_n e^{\mathbf{L}h} + h^{-2}\mathbf{L}^{-3}\{[-4 - \mathbf{L}h + e^{\mathbf{L}h}(4 - 3\mathbf{L}h + (\mathbf{L}h)^2)]F(u_n, t_n) \tag{2.22}$$

$$+ 2[2 + \mathbf{L}h + e^{\mathbf{L}h}(-2 + \mathbf{L}h)](F(a_n, t_{n+1/2}) + F(b_n, t_{n+1/2}) \tag{2.23}$$

$$+ [-4 - 3\mathbf{L}h - (\mathbf{L}h)^2 + e^{\mathbf{L}h}(4 - \mathbf{L}h)]F(c_n, t_{n+1})\}. \tag{2.24}$$

The above equations (2.19) – (2.24) suffer from catastrophic cancellation if implemented directly in this format. Expressions of the form $g(z) = \frac{e^z-1}{z}$ can cause rounding errors rapidly. Higham in [47] investigates phenomena similar to this in more detail. Equations (2.19) – (2.21) have an expression similar to $g(z)$. Moreover, equations (2.22) – (2.23) contain expressions that are higher order analogs of $g(z)$ (see [69] and [59]).

So, to efficiently implement the ETDRK4 scheme in general, Kassam and Trefethen in [59] propose to evaluate these computationally sensitive expressions with high accuracy.[4] In particular, they use an approximation based on contour integration from complex analysis. More specifically, using the Cauchy integral representation on a circle radius 1 centered at z for $|z| < 0.5$, one has

$$\phi_l(z) = \frac{1}{2\pi i}\int \frac{\phi_l(s)}{s-z}\,ds \approx \frac{1}{M}\sum_{k=1}^{M} \phi_l(z + e^{2\pi ik/M}), \quad l \geq 0 \tag{2.25}$$

where $\phi_l(z) = \sum_{k=l}^{\infty} (1/k!) z^{k-l}$. Notice ϕ_l for the first few values of l takes the form

$$\phi_1(z) = \frac{e^z - 1}{z}, \quad \phi_2(z) = \frac{e^z - z - 1}{z^2}, \quad \phi_3(z) = \frac{e^z - z^2/2 - z - 1}{z^3}.$$

The approximation in equation (2.25) is intended for scalar or diagonal matrices $\mathbf{L}$. For non-diagonal cases there is a generalization of equation (2.25) that can be found in [59] and [78]. For more information about equation (2.25)[5], see [85].

To implement the modified ETDRK4 scheme, we remedy the numerical instabilities by applying equation (2.25). Below is a list of the expressions that cause numerical difficulties:

$$\begin{aligned}
Q &= \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - 1) \\
f_1 &= h^{-2}\mathbf{L}^{-3}[-4 - \mathbf{L}h + e^{\mathbf{L}h}(4 - 3\mathbf{L}h + (\mathbf{L}h)^2)] \\
f_2 &= h^{-2}\mathbf{L}^{-3}[4 + 2\mathbf{L}h + e^{\mathbf{L}h}(-4 + 2\mathbf{L}h)] \\
f_3 &= h^{-2}\mathbf{L}^{-3}[-4 - 3\mathbf{L}h - (\mathbf{L}h)^2 + e^{\mathbf{L}h}(4 - \mathbf{L}h)]
\end{aligned}$$

The functions Q, f_1, f_2, and f_3 appear in the ETDRK4 scheme, and we can precompute

[4] There are other techniques to avoid these numerical issues. Berland et al in [6] showcase a Padé approximant approach to resolve this numerical instability.

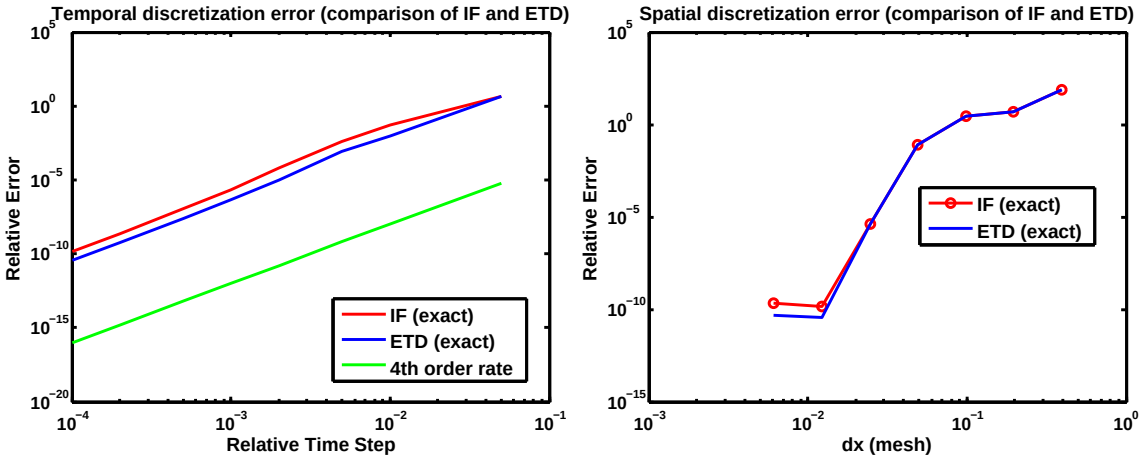
[5] The trapezoidal rule for integration is efficient (geometric convergence) for analytic functions that tend to zero quickly.

this quantities via equation (2.25). Let $q = \mathbf{L}h$, and the function Q becomes

$$\begin{aligned} Q &= \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - 1) \\ &= \frac{e^{q/2} - 1}{q/h} = h\frac{e^{q/2} - 1}{q} \\ &\approx \frac{1}{M}\sum_{k=1}^{M} h\frac{e^{(q+z_k)/2} - 1}{q + z_k} \\ &= \frac{h}{M}\sum_{k=1}^{M} \frac{e^{(q+z_k)/2} - 1}{q + z_k} \\ &= \frac{h}{M}\sum_{k=1}^{M} \frac{e^{(\mathbf{L}h+z_k)/2} - 1}{\mathbf{L}h + z_k}, \end{aligned}$$

where $z_k = e^{2\pi ik/M}$ (roots of unity). Equation (2.25) is a mean of f multiplied by h, which is straight forward to implement. Using the same substitution ($q = \mathbf{L}h$) one can arrive at the expressions for Q, f_1, f_2, and f_3 used in Code Listing 2.4.4.

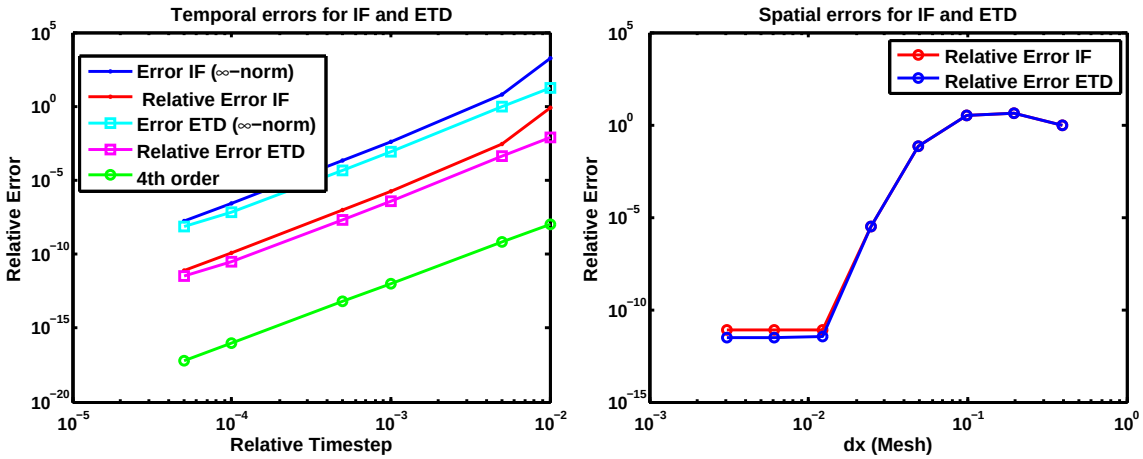
To complete this section, figure 2.4 (errors are in the infinity norm) confirms that the mETDRK4 scheme performs better than the IFRK4 scheme. The test problem used to compare these methods is the KdV equation with a single soliton initial condition.



(a) $10^{-7} \leq \Delta t \leq 5 \cdot 10^{-5}$, $N = 512$ (b) $16 \leq N \leq 512$, $\Delta t = 10^{-7}$

Figure 2.4: Error comparison of IFRK4 and mETDRK4 schemes. The mETDRK4 scheme outperforms IFRK4. Spectral accuracy is observed in the spatial discretization, and 4th order accuracy in the temporal discretization. Amplitude v in equation (2.3.2) is $v = 25$, initial position is $x_0 = -2$, and final time simulation is $T_{\text{final}} = 0.001$. Errors are in the infinity norm.

Figure 2.5 has an error analysis for a two solition interaction test problem. The relevant parameters from equations (NE.1) and (NE.2) are $\alpha_1 = 25(1 + 0.1)$, $\alpha_2 = 25(1 - 0.1)$, $s_1 = -\log(0.1)/\alpha_1$, $s_2 = -\log(0.1)/\alpha_2$, and the final simulation time is $T_{\text{fina}} = 0.001$. The spatial error stagnates due to the residuals left over from the soliton interation.



(a) $10^{-7} \leq \Delta t \leq 5 \cdot 10^{-5}$, $N = 512$

(b) $16 \leq N \leq 2048$, $\Delta t = 10^{-8}$

Figure 2.5: Error comparison of IFRK4 and mETDRK4 schemes for two solitons with similar amplitudes interacting. The mETDRK4 scheme outperforms IFRK4. Notice that there is a saturation of spectral accuracy in figure 2.5b. This is due to the two solitons interacting and leaving non–negligible residual errors.

2.5 Small dispersion limit of the KdV equation

A variation of the KdV equation introduces a parameter $\epsilon^2 \ll 1$ on the dispersive term:

$$u_t + 6uu_x + \epsilon^2 u_{xxx} = 0. \tag{2.26}$$

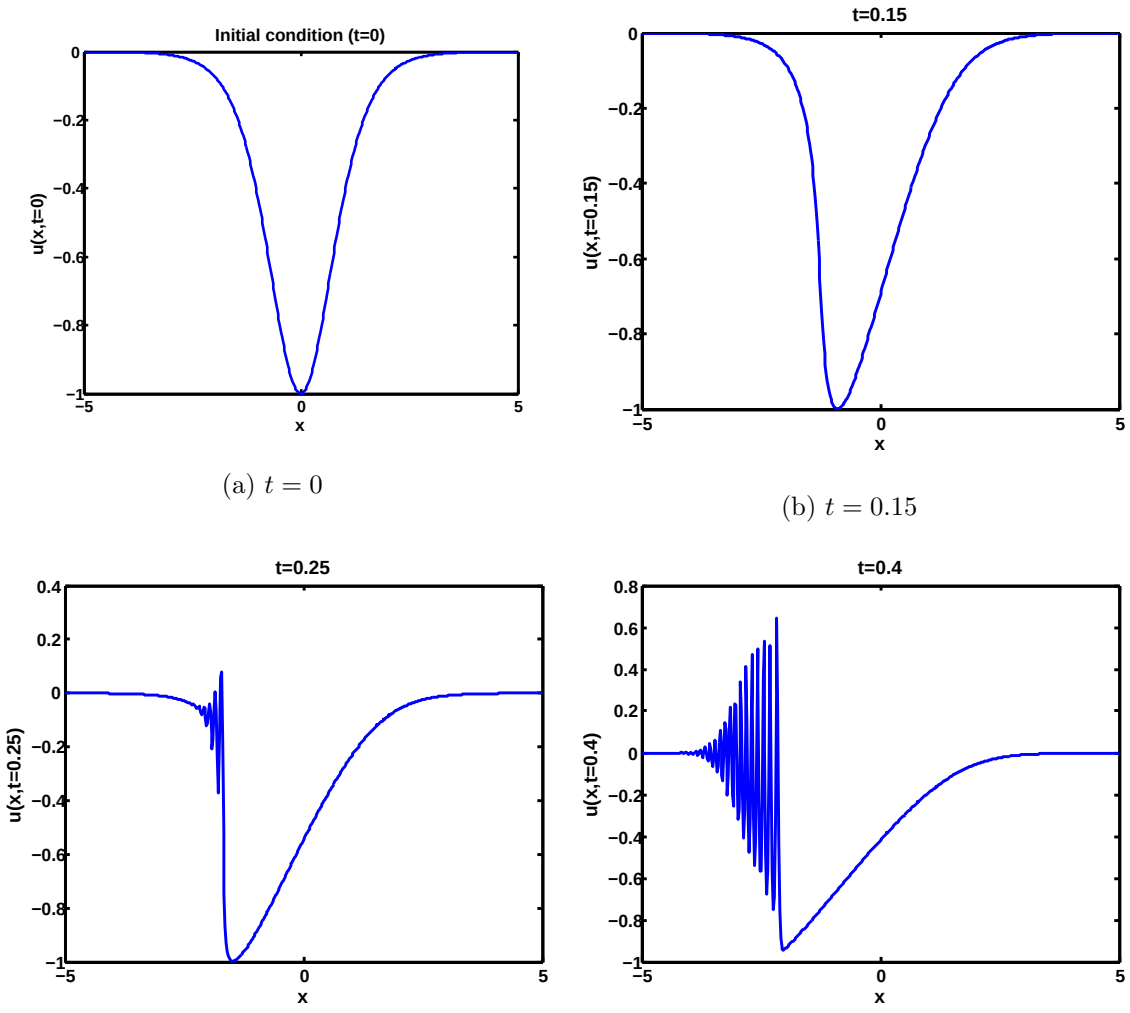
Many researchers are interested in the asymptotic behavior of equation (2.26); the asymptotics for this equation change depending on what region of the (x,t) plane the solution is in. Lax and Levermore in [65], Venakides in [89], and Grava and Klein in [39] are standard references for this work. Spectral methods for the KdV equation can provide high accuracy reference approximations; even in the case of the small dispersion limit. Grava and Klein in [39] use a mETDRK4 scheme to compare with various asymptotic formulas.

The KdV equation admits simple traveling wave solutions, however, the small parameter in equation (2.26) can change this behavior quite drastically. In fact, rapid oscillations occur after a time t_c (called the time of the *gradient catastrophe* or *shock formation*). Figure 2.6 has the evolution of equation (2.26) for $\epsilon = 10^{-1.5}$, $u(x,0) = -\text{sech}^2(x)$, $0 \le t \le 0.4$, and $x \in [-5,5]$. This test problem was taken from [39]. For explicit code[6], see Code Listing 2.5.1.

Code Listing 2.5.1 Main Small Dispersion Limit file

```
% Approximate the solution to the Small Dispersion Limit KdV equation
% u_t + 6uu_x + ep^2u_xxx = 0 on the domain [-5,5] with periodic boundary
% conditions via a IFRK4 scheme.  Initial condition of u_0 = -sec^2(x).
clear all;  close all;  clc;
N = 2^(9);  tmax = 0.4;  dt = 1e-3;
ep = 10^(-1.5);
L = 10;  b = L/(2*(pi));  B = b;                    % Accounting for domain
x = (L/N)*(-N/2:N/2-1)';                            % length change
u = - 1./cosh(x).^2;          v = fft(u);
k = [0:N/2-1 0 -N/2+1:-1]';  ik3 = 1i*k.^3/b^3;
for n = 1:round(tmax/dt)
    g = -3*1i*dt*k/B;                               % Accounting for domain
    E = exp(ep^2*dt*ik3/2);                         % length change
    E2 = E.^2;
    a = g.*fft(real( ifft(     v    ) ).^2);
    b = g.*fft(real( ifft(E.*(v+a/2)) ).^2);        % 4th-order
    c = g.*fft(real( ifft(E.*v + b/2) ).^2);        % Runge-Kutta
    d = g.*fft(real( ifft(E2.*v+E.*c) ).^2);
    v = E2.*v + (E2.*a + 2*E.*(b+c) + d)/6;
    u = real(ifft(v));
end
```

[6] Here the IFRK4 scheme was used. One can also use the ETDRK4 scheme. Something noteworthy about this code is that it handles a domain change, and we are no longer using the normalized KdV equation.



(a) $t = 0$

(b) $t = 0.15$

(c) $t = 0.25$

(d) $t = 0.4$

Figure 2.6: Behavior of the small dispersion limit.

The KdV equation can be thought of as a singular perturbation of the Hopf equation (also known as the inviscid Burgers' equation):

$$u_t + 6uu_x = 0. \tag{2.27}$$

We can gather information from this dispersionless KdV equation in (2.27). The time of the gradient catastrophe, t_c, can be found by using the method of characteristics on (2.27). The characteristic equations are given by the following system of ODEs:

$$\frac{du}{dt} = 0, \quad \frac{dx}{dt} = u, \tag{2.28}$$

which has a solution of $u(x,t) = u_0(\xi)$, $x = 6tu_0(\xi) + \xi$, for an initial condition u_0. The solution travels to the left, however, its minimum travels much faster. This results in a gradient catastrophe phenomena. The spatial derivative of $u(x,t)$ is

$$u_x = u_0'(\xi)\xi_x. \tag{2.29}$$

The derivative of u in equation (2.29) will grow without bound provided that $\xi_x \to \infty$. Differentiating $x = 6tu_0(\xi) + \xi$ with respect to x yields

$$1 = 6tu_0'(\xi)\xi_x + \xi_x \implies \xi_x = \frac{1}{1 + 6tu_0'(\xi)}. \tag{2.30}$$

From equation (2.30), we find that ξ_x will blow up at

$$t_c = \min_{\xi \in \mathbb{R}} \frac{-1}{6u_0'(\xi)}. \tag{2.31}$$

The solution to equation (2.27) after t_c (the gradient catastrophe) will not exist in the classical sense. Figure 2.7 plots solutions to the Hopf equation before t_c. It turns out that

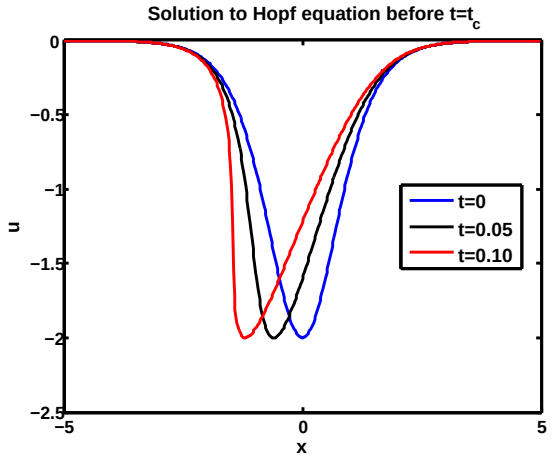


Figure 2.7: Solution to the Hopf equation before t_c (the gradient catastrophe).

the small dispersion KdV equation inherits complications around t_c. Table 2.3 has some heuristics describing how the small dispersion KdV equation evolves in the (x,t) plane.

In [39], [41], [74], [77], [76], and [75] a deeper study of the small dispersion KdV equation is given, with an emphasis on asymptotics. The asymptotic nature of the small dispersion KdV equation is complicated; the asymptotic formulas described in the articles above are implicit. In addition, after t_c the small dispersion KdV equation solution exhibits rapid oscillations; for which exact solutions are not known. Because of this, the spectral methods are valuable in that they can provide reference solutions - they can be used to compare against the various asymptotic approximations.

For $t \ll t_c$	For $t \approx t_c$	For $t > t_c$
The dispersive term $\epsilon^2 u_{xxx}$ can be neglected, since the Hopf solution and derivatives remain small (see figures 2.6a and 2.6b).	The dispersive term $\epsilon^2 u_{xxx}$ will now contribute nontrivial behavior, even for small ϵ. The large spatial derivatives drive this behavior (see figure 2.6c).	An oscillatory zone forms; a region in the (x, t) plane where the solution exhibits rapid oscillations (see figure 2.6d).

Table 2.3: Classification of the behavior of small dispersion KdV equation.

Trogdon et al. in [86] also utilizes a Fourier spectral method to verify approximations to the KdV equation based on numerical inverse scattering. The Fourier spectral method is applied only for short time spans, but is much easier to implement. Numerical inverse scattering and its analysis is much more involved, but can provide uniformly valid approximations.

Chapter 3

FOURIER SPECTRAL METHOD FOR THE BENJAMIN–BONA–MAHONY (BBM) EQUATION

The Benjamin–Bona–Mahony (BBM) equation is a partial differential equation that models the unidirectional propagation of long waves with small amplitudes. In [5], the BBM equation is shown to be an improvement of the KdV equation in multiple ways. One improvement that is directly related to Fourier spectral methods is its dispersion relation. The dispersion relation for the linearized KdV equation is a cubic function in wavenumbers k, and this leads to a restriction on time steps (see equation (2.8)). The dispersion relation for the BBM equation is bounded for all wavenumbers k, which allows for much larger time steps. Also, the dispersion relation can act as an aliasing filter for large wavenumbers k (compare equations (2.4) and (3.7)). In the original article by Benjamin, Bona, and Mahony ([5]) more theoretical details are given.

The main purpose of this chapter is to apply Fourier spectral methods to the BBM equation so that we can approximate solutions.

In section 3.1 the KdV-BBM equation is presented, along with exact soliton solutions that are smooth and nonperiodic.

Section 3.2 derives the discretizations that will be used to implement the Fourier spectral methods to the BBM equation. Among the Fourier spectral methods studied are: a direct spectral method, an integrating factor spectral method, and an exponential time differencing spectral method. The direct spectral method is slower, less accurate, but more general, and easier to implement. The integrating factor and exponential time differencing methods exploit linearity that is present and results in numerical schemes that are faster, more accurate, and have better stability properties.

Explicit code examples are given in section 3.3. These examples codes are the implementation of the various Fourier spectral methods described in section 3.2. Stability and error analyses are also presented.

3.1 BBM equation

The KdV-BBM equation[1] is the partial differential equation

$$u_t + \alpha u_x + \beta u u_x - \gamma u_{xxt} + \delta u_{xxx} = 0, \tag{3.1}$$

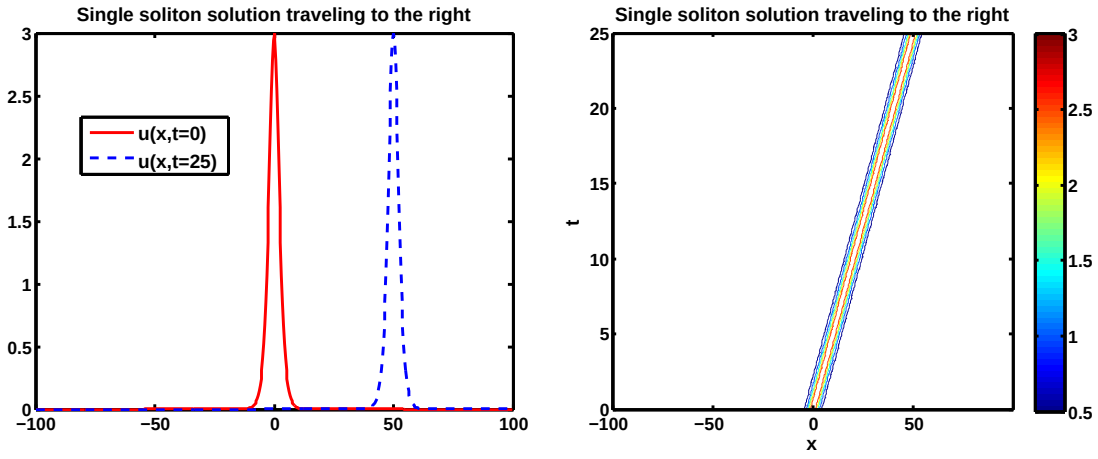
for $x \in \mathbb{R}$, $t > 0$, u represents the free surface elevation above the still water level $u = 0$ and $\alpha, \beta, \gamma, \delta$ are positive constants. This equation has a more general form than we need. If $\delta = 0$, then (3.1) reduces to the BBM equation, and if $\gamma = 0$, then (3.1) becomes the KdV equation.

It is straightforward to verify that the following function is a solution to (3.1)

$$u(x,t) = 3\left(\frac{c_s - \alpha}{\beta}\right)\operatorname{sech}^2\left(0.5\sqrt{\frac{c_s - \alpha}{\gamma c_s + \delta}}(x - c_s t)\right). \tag{3.2}$$

This solitary solution is smooth, nonperiodic, and travels to the right with an arbitrary speed c_s (equation (3.2) can be found in [26]).

The BBM equation has also been studied with non-constant coefficients, and this type of equation models the propagation of solitary waves as they travel over an uneven bottom (see [73]). In this thesis we will use a normalized BBM equation ($\alpha = \beta = \gamma = 1, \delta = 0$).



(a) (b)

Figure 3.1: (a) Single soliton solution (from equation (3.2)) traveling to the right with a speed $c_s = 2$. (b) Top-down view of the single soliton's evolution.

[1] In [26] there is more information about the KdV-BBM equation, as well as approximations to it based on finite difference and finite volume methods.

3.2 Fourier Spectral methods for the BBM equation

First, we begin with accounting for domain length change. Let L be the length of the interval that the BBM equation will be approximated on. To map this domain to a computational domain of $[-\pi, \pi]$, we set $B = L/2\pi$, and $u(Bx, t) = v(x, t)$. With this change of variables the BBM equation is then given by

$$v_t + (1/B)v_x + (1/B)vv_x - (1/B)^2 v_{xxt} = 0, \quad x \in [-\pi, \pi], t > 0 \tag{3.3}$$

$$v(x, 0) = u(B, 0), \tag{3.4}$$

$$v(-\pi, t) = v(\pi, t), \quad t \geq 0 \tag{3.5}$$

Following the general algorithm for the spectral method given in section 2.3, we take the Fourier transform of both sides of (3.3):

$$\widehat{v}_t + (ik/B)\widehat{v} + 0.5(ik/B)\mathcal{F}(v^2) + (k/B)^2\widehat{v}_t = 0.$$

Isolating the time derivative terms yields

$$(B^2 + k^2)\widehat{v}_t + (ikB)\widehat{v} + 0.5(ikB)\mathcal{F}(v^2) = 0. \tag{3.6}$$

After simplification we have

$$\widehat{v}_t + \left(\frac{ikB}{B^2 + k^2}\right)\widehat{v} + \left(\frac{0.5ikB}{B^2 + k^2}\right)\mathcal{F}(v^2) = 0. \tag{3.7}$$

Notice equation (3.7) can be written in the form $\widehat{v}_t + L(\widehat{v}) = N(\widehat{v})$, where $L(\cdot)$ and $N(\cdot)$ are linear and non-linear functions, respectively.

The various spectral methods will essentially be based off of equation (3.7). As done with the KdV equation, we will examine various methods; a direct spectral approach, and the IF and ETD approaches. The IF and ETD approaches will solve for the linear part of equation (3.7) exactly:

$$\frac{d(\mu(t)\widehat{v})}{dt} = -\mu(t)\left(\frac{0.5ikB}{B^2 + k^2}\right)\mathcal{F}(v^2) = 0, \tag{3.8}$$

where

$$\mu(t) = \exp\left(\frac{ikBt}{B^2 + k^2}\right).$$

Equation (3.8) has a very similar form to that of equation (2.5). So we can expect that the IF/ETD schemes will be competitive with each other, and they should also outperform the direct spectral approach. The direct spectral approach approximates the systems of ODEs that arises from equation (3.7), but this system is stiff due to the linear term that is present.

3.3 Implementation of the Fourier Spectral method

Just as with the KdV equation, in this section explicit code examples are given for approximating the BBM equation using Fourier spectral methods.

Method	Code Listing
Direct–Spectral	3.3.1
IFRK4	3.3.2
mETDRK4	3.3.3

Table 3.1: Organization of spectral method and their respective code listings (BBM equation).

Code Listing 3.3.1

Main Direct-Spectral file

```
% Approximate the solution to the BBM equation, which is given by
% u_t + u_x + uu_x - u_{xxt} = 0.  The spatial domain is [-100,100] and
% periodic boundary conditions are assumed.  The approximation is done by a
% direct-spectral method with RK4 time-stepping.
clear all;  close all;  clc
%Set spatial and temporal information
N = 2^(8);
L = 200;                  %spatial domain length
b = L/(2*(pi));           %Rescaling paramter
x = (L/N)*(-N/2:N/2-1)';
T = 15;                   %Final time value
dt = 1e-1;                %Large time step
nmax = round(T/dt);
%Parameters for exact solution to the BBM equation
c_s = 2;    %Profile speed
U = @(x,t) 3*(c_s-1)*(sech( 0.5*sqrt((c_s-1)/c_s)*(x - c_s*t) )).^2;
%Set up right hand side function for the problem dw/dt = RHS(w)
k = [0:N/2-1 0 -N/2+1:-1]'; %Wave numbers
g1 = -0.5*1i* b*k./(b^2 + k.^2);  g2 = 2*g1;
RHS = @(v) g1.*fft(real(ifft(v)).^2) + g2.*fft(real(ifft(v)));
u = U(x,0.0);  v = fft(u);  %Initial condition
for n = 1:nmax
    a = RHS(v);
    b = RHS(v + a*(dt/2));
    c = RHS(v + b*(dt/2));
    d = RHS(v + c*(dt));
    v = v + (dt/6)*(a + 2*(b + c) + d);
end
u = real(ifft(v));
Err = norm(u - U(x,T),inf)
plot(x,u,'r.',x,U(x,T),x,U(x,0),'g')
```

Code Listing 3.3.2

Main IFRK4 file

```
% Approximate the solution to the BBM equation, which is given by
% u_t + u_x + uu_x - u_{xxt} = 0.  The spatial domain is [-100,100] and
% periodic boundary conditions are assumed.  The approximation is done by a
% integrating factor spectral method with RK4 time-stepping.

clear all;  close all;  clc;
%Set spatial and temporal information
N = 2^(8);
L = 200;                 %spatial domain length
b = L/(2*(pi));
x = (L/N)*(-N/2:N/2-1)';
T = 15;                  %Final time value
dt = 1e-1;               %Large time step
nmax = round(T/dt);

%Parameters for exact solution to the BBM equation
c_s = 2;    %Profile speed
U = @(x,t) 3*(c_s-1)*(sech( 0.5*sqrt((c_s-1)/c_s)*(x - c_s*t) )).^2;
k = [0:N/2-1 0 -N/2+1:-1]';
ik = 1i*b* k./(b^2 + k.^2);
g = -0.5*dt*ik;
E = exp(-dt*ik/2);  E2 = E.^2;  %Integrating factor exponentials

u = U(x,0.0);  v = fft(u);  %Initial condition
for n = 1:nmax
    a = g.*fft(real( ifft(     v    ) ).^2);
    b = g.*fft(real( ifft(E.*(v+a/2)) ).^2);     % IFRK4
    c = g.*fft(real( ifft(E.*v + b/2) ).^2);
    d = g.*fft(real( ifft(E2.*v+E.*c) ).^2);
    v = E2.*v + (E2.*a + 2*E.*(b +c) + d)/6;
end
u = real(ifft(v));

Err = norm(u - U(x,T),inf)
plot(x,u,'r.',x,U(x,T),x,U(x,0),'g')
```

Code Listing 3.3.3

Main ETDK4 file

```
% Approximate the solution to the BBM equation, which is given by
% u_t + u_x + uu_x - u_{xxt} = 0.  The spatial domain is [-100,100] and
% periodic boundary conditions are assumed.  The approximation is done by a
% exponential time differencing spectral method with RK4 time-stepping.
clear all;  close all;  clc;
%Set spatial and temporal information
N = 2^(8);
L = 200;  b = L/(2*(pi));  %spatial domain length and rescaling
x = (L/N)*(-N/2:N/2-1)';
T = 15;                  %Final time value
dt = 1e-1;  h = dt;      %Large time step
nmax = round(T/dt);
%Parameters for exact solution to the BBM equation
c_s = 2;    %Profile speed
U = @(x,t) 3*(c_s-1)*(sech( 0.5*sqrt((c_s-1)/c_s)*(x - c_s*t) )).^2;
% Precompute ETDRK4 scalar quantities (due to Kassam-Trefethen):
k = [0:N/2-1 0 -N/2+1:-1]';           % Wave numbers
L = -1i*b*k./(b^2+k.^2);              % Fourier multipliers
E = exp(h*L); E2 = exp(h*L/2);
M = 2^6;                                % Number of points for complex
                                        % means
r = exp(2i*pi*((1:M)-0.5)/M);           % Roots of unity
LR = h*L(:,ones(M,1))+r(ones(N,1),:);
Q  = h*mean(                  (exp(LR/2)-1)./LR   ,2);
f1 = h*mean((-4-LR+exp(LR).*(4-3*LR+LR.^2))./LR.^3,2);
f2 = h*mean(    (4+2*LR+exp(LR).*(-4+2*LR))./LR.^3,2);
f3 = h*mean((-4-3*LR-LR.^2+exp(LR).*(4-LR))./LR.^3,2);
g = 0.5*L;
u = U(x,0.0);  v = fft(u);  %Initial conditon
% March forward in time via ETDRK4 formula (due to Cox-Matthews):
for index = 1:nmax
    Nv = g.*fft(real(ifft(v)).^2);  a = E2.*v+Q.*Nv;
    Na = g.*fft(real(ifft(a)).^2);  b = E2.*v+Q.*Na;
    Nb = g.*fft(real(ifft(b)).^2);  c = E2.*a+Q.*(2*Nb-Nv);
    Nc = g.*fft(real(ifft(c)).^2);
    v = E.*v+(Nv.*f1+(Na+Nb).*f2+Nc.*f3);
end
u = real(ifft(v));  Err = norm(u - U(x,T),inf)
plot(x,u,'r.',x,U(x,T),x,U(x,0),'g')
```

3.3.1 Fourier spectral stability

The dispersion relation for the transformed BBM equation in (3.8) is different than the transformed KdV equation in (2.4). For the KdV equation, the dominate wave numbers are k^3, but from linearized form of equation (3.7) we can see that the dominate wave numbers are $1.5(kB)/(B^2+k^2)$. This means that for the BBM equation we can select similar time steps for the direct spectral, and IF/ETD schemes. For the KdV equation a much smaller time step was needed for the direct spectral scheme, relative to the IF/ETD schemes.

If we use the classic Runge-Kutta time stepping, then the dispersion relation for the BBM equation is bounded by

$$2.82 > \max_n |\omega_n \Delta t| = \max_n \left| \frac{k_n B}{B^2 + k_n^2} \right| \Delta t,$$

where k_n are the discretized wave numbers, and ω_n is the dispersion relation for the BBM equation. The extremea for $\omega_n(k)$ occur at $k = \pm B$. In the case where $L = 200$, an estimate for the stability criteria is $\Delta t < 0.49967$.

In the case of the BBM equation we can use much larger time steps for any of the Fourier spectral schemes, due to the dispersion relation and associated group velocity being bounded. For the KdV equation, the dispersion relation ($\omega = -k^3$) and associated group velocity ($\partial\omega/\partial k = -3k^2$) are both unbounded functions of k. Figure 3.2 has a plot of the direct spectral approach breaking down; this is due to selecting a time step too large.

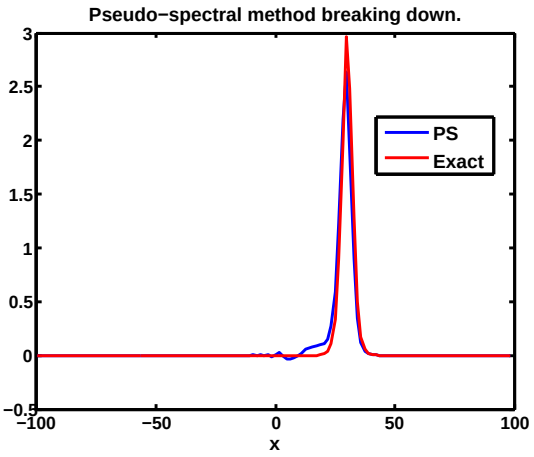


Figure 3.2: Direct spectral approach breaking down. Relevant parameters: $\Delta t = 0.9$, $T_{\text{final}} = 15$, and $N = 2^7$.

Sample code for the direct spectral scheme is given in Code Listing 3.3.1.

3.3.2 Spectral integrating factor method method (IFRK4)

Similar to the IFRK4 method for the KdV equation, the IFRK4 method for the BBM equation performs better than the direct spectral approach. The time stepping loop in Code Listing 3.3.2 is identical to Code Listing 2.4.3 (IFRK4 applied to the KdV equation). This is because the BBM equation and KdV equation are generalized by the KdV-BBM equation; and the KdV-BBM equation shares the same nonlinear term. The Fourier multipliers are different, however (see equation (3.8) and equation (2.5)). Subsection 2.4.2 has a derivation of the terms in the RK4 time stepping scheme found in Code Listing 3.3.2.

The time stepping loop seen in Code Listing 3.3.2 is a better way to implement the IFRK4 scheme (rather than implementing IFRK4 as is in equation (3.8)). It simplifies the integrating factor $\mu(t)$ found in equation (3.8), so that $\mu(t)$ is constant. Another computational consideration is the domain length. A large domain of $[-100, 100]$ was selected simply so that the user can watch the single soliton propagate as time increases.

Sample code for the IFRK4 spectral method is given in Code Listing 3.3.2.

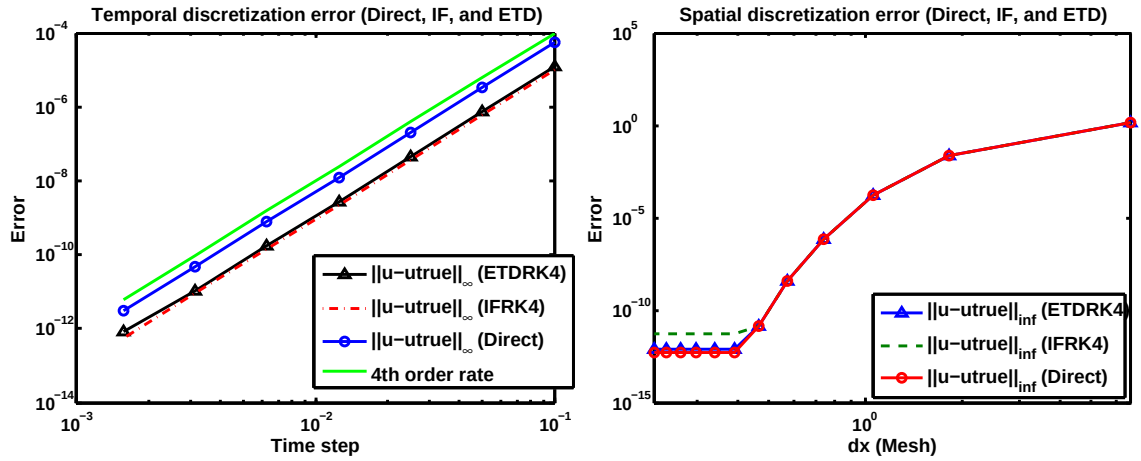
3.3.3 Spectral modified exponential time differencing method (ETDRK4)

The ETD method used in Code Listing 3.3.3 is the modified exponential time differencing presented in subsection 2.4.3. Since IFRK4 and mETDRK4 methods exactly solve for the linear part of the system in equation (3.8), they relieve some of the numerical stiffness that is present due to the dispersive term u_{xxt} from the BBM equation.

Figure 3.3 exhibits an error analysis study for a test problem for the BBM equation. In particular we observe spectral convergence in the spatial discretization and a 4th order convergence rate in the temporal discretization (from RK4).

	Temporal discretization	Spatial discretization
Δt	$(0.1)/2^m$, where $0 \leq m \in \mathbb{N} \leq 6$	10^{-3}
N	2^9	$N = (880/11)m + 30$, where $0 \leq m \in \mathbb{N} \leq 11$
c_s	2	2
T_{final}	15	15
Domain	$[-100, 100]$	$[-100, 100]$

Table 3.2: Various parameters for the error analysis shown in figure 3.3.



(a) $N = 2^9$, $\Delta t = (0.1)/2^j$, where $0 \leq j \in \mathbb{N} \leq 6$. (b) $\Delta t = 10^{-3}$, $30 \leq N \in \mathbb{N} \leq 910$.

Figure 3.3: (a) Error due to temporal discretization. (b) Error due to spatial discretization.

In figure 3.3 we can see that the mETDRK4 scheme is competitive with the IFRK4 scheme for the temporal discretization. For the spatial discretization the mETDRK4 scheme performs better than the IFRK4. The mETDRK4 scheme is also slightly faster than the IFRK4 scheme. The direct spectral method does work, but since we can solve for the linear term exactly in the BBM equation, the IF and ETD schemes are faster, more accurate, and have better stability properties.

Chapter 4

FOURIER SPECTRAL METHOD FOR THE CAMASSA–HOLM (CH) EQUATION

The Camassa-Holm equation is a nonlinear partial differential equation that models waves in shallow water. Camassa and Holm introduced this PDE in [13], and it has been shown to have a strong mathematical structure. A noteworthy feature of this PDE is that it has the ability to admit smooth or non-smooth solitary wave solutions that are solitons. By tweaking a parameter in the CH equation one can force smooth or non–smooth solutions. The non-smooth solutions are called "peakons", they are solitons with sharp peaks (cusps); this results in a discontinuous derivative of the soliton. As such, these peakons are only solutions in the weak or distributional sense. For more details of the physical and mathematical background of the CH equation, see [13] and [67].

As done with the KdV and BBM equation, this section seeks to apply Fourier spectral methods to approximate solutions of the CH equation.

Section 4.1 has a brief introduction to the CH equation as well as some of the complications that arise from using Fourier spectral methods to approximates solutions to it.

In section 4.2 the general Fourier spectral scheme is described. Comparisons are drawn to the KdV and BBM equations; since the CH equation is the most advanced scalar equation examined in this thesis. The main complications of using a Fourier spectral method are addressed (non–smooth solutions, IF/ETD schemes are not applicable, and severe nonlinearities).

A code example for the implementation of the Fourier spectral method (from section 4.2) can be found in section 4.3. Also, an analysis of convergence and alternate Fourier spectral schemes are discussed.

Section 4.3.1 showcases a Fourier spectral method for the small dispersion limit CH equation. Further work and applications of the small dispersion limit CH equation can be found in [40] and [2].

4.1 CH equation

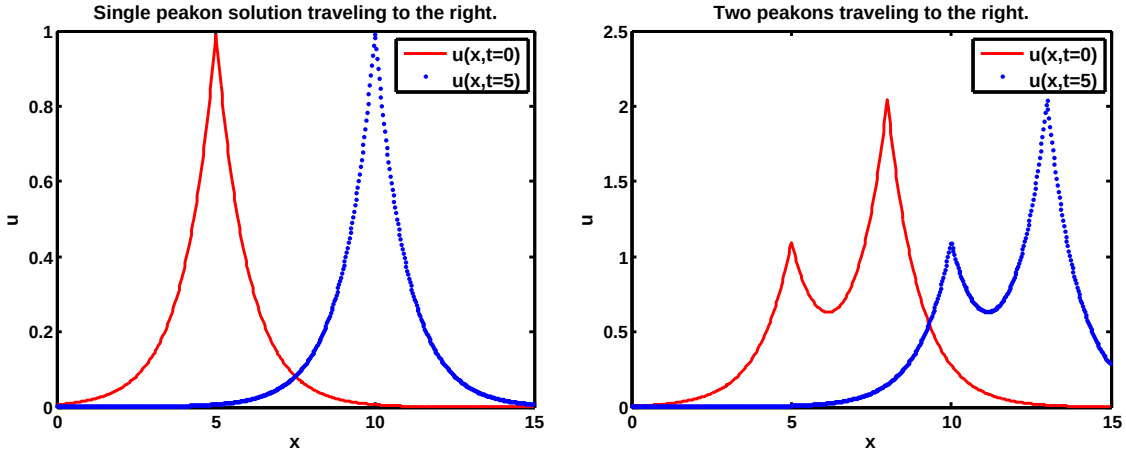
The Camassa-Holm equation is given by the following partial differential equation

$$u_t + 2\kappa u_x - u_{xxt} + 3uu_x = 2u_x u_{xx} + uu_{xxx}, \tag{4.1}$$

where κ is a nonneagative constant. If κ is equal to zero, the CH equation admits the so–called peakon solutions. These peakons take the form

$$u(x,t) = ce^{-|x-x_0-ct|}, \quad x_0, c \in \mathbb{R}.$$

Figure 4.1 has some example plots of peakons. In the case where κ is positive, the CH equation admits smooth traveling wave solutions. These smooth solutions can only be described implicitly. See [55], [48], [93], and [54] for a further study of smooth and non–smooth solutions for the CH equation. It should be noted that when $\kappa = 0$, the CH equation loses its physical significance.



(a) (b)

Figure 4.1: (a) Single peakon given by $u(x,t) = e^{-|x-5-t|}$ traveling to the right. (b) Multipeakon given by $u(x,t) = e^{-|x-5-t|} + 2e^{-|x-8-t|}$.

In terms of nonlinearities, the CH equation is more complicated than the KdV and BBM equations. The KdV and BBM equations share the same simple nonlinearity $uu_x = 0.5(u^2)_x$, but the CH equation has much more severe nonlinear terms. This can complicate matters in the search for a Fourier spectral method for the CH equation. In addition, the non–smooth solutions will slow the rate of convergence down for spectral methods. These issues help illustrate that the Fourier spectral methods have limitations.

4.2 Fourier Spectral methods for the CH equation

Kalisch and Raynaud in [42] have found that writing equation (4.1) (with $\kappa = 0$) as

$$(1 - \partial_x^2)u_t = (1 - \partial_x^2)(0.5(u^2)_x) + (u^2 + 0.5(u_x)^2)_x = 0, \tag{4.2}$$

allows the CH equation to be approximated by a Fourier spectral method somewhat easily. Taking the Fourier transform of both sides of equation (4.2), we have

$$\hat{u}_t = -0.5ik\mathcal{F}(u^2) - \frac{ik}{(1+k^2)}(\mathcal{F}(u^2) + \mathcal{F}(u_x^2)). \tag{4.3}$$

From equation (4.3) one can apply the method of lines and a time stepping technique. The term $\mathcal{F}(u_x^2)$ in equation (4.3) will require us to evaluate u_x spectrally. When we march forward in time using a finite difference, this update is for u (in Fourier space) and not u_x. However, we can approximate u_x at time step t_n spectrally by $u_x(\cdot, t_n) \approx \Re\{\mathcal{F}^{-1}[ik\mathcal{F}(u^2)]\}$. The severe nonlinearities and non–smooth solutions will require us to use dealiasing in order to avoid aliasing errors caused by the FFT. A very popular dealiasing is the "Orszag Two-Thirds Rule", which zeroes out the largest third of the frequencies in the nonlinear terms of equation (4.3). These frequencies often can correspond to data that is undesirable. Purging one third of the original data (corresponding to the largest frequencies) allows for two thirds of the original data to be kept intact. This purging is an irreversible loss of data, and zeroing out more frequencies can cause a large loss in data. More details about this dealiasing rule can be found in [11].

Another difference between the KdV/BBM equations and the CH equation is that the former have a linear differential operator that is straightforward to exploit. However, in the case of the CH equation (even with $\kappa = 0$) there does not appear to be a natural integrating factor or exponential integrator. The Fourier spectral methods for the KdV/BBM equation were successful because the IF/ETD schemes could resolve numerical stiffness in the system of ODEs that resulted from the method of lines. The CH equation cannot enjoy this feature, however. This issue coupled with harsh nonlinearities, and non–smooth solutions results in a volatile spectral method. A large number of grid points (around 2^{11}) and a small timestep (around 10^{-4}, for stability reasons) are needed to obtain reasonable progress. And even then (with $\kappa = 0$), the rate of convergence will be much lower due to the non–smooth solutions.

Despite all of these complications, Fourier spectral methods can still be of use when applied to the CH equation. For instance, the authors in [2] study the small dispersion limit of the CH equation, and to analyze various asymptotic and analytic formulas, they use a Fourier spectral method to approximate solutions to the CH equation (with $\kappa > 0$).

4.3 Implementation of the Fourier Spectral method

In this section explicit code examples are given for the CH equation. A domain length change parameter is included in the wave numbers, and the two thirds rule is utilized in Code Listing 4.3.1.

Method	Code Listing
Fourier–Spectral	4.3.1
Fourier–Spectral (small dispersion limit)	4.4.1

Table 4.1: Organization of spectral method and their respective code listings (CH equation).

Code Listing 4.3.1

Main Fourier-Spectral file

```
% Approximates the solution to the CH equation, which is given by
% u_t - u_{xxt} + 3uu_x = 2u_xu_{xx}+uu_{xxx}. The spatial domain is [0,50]
% and periodic boundary conditions are assumed.  The approximation is done
% by a Fourier spectral method with RK4 time-stepping.
clear all;  close all;  clc;
N = 2^11;  L = 50;  T = 3.2;  c = 1;  dt = 2e-4;
x = L*(0:(N-1))/N;  nmax = round(T/dt);
M = [0:(N/2-1) (-N/2):(-1)];
k = 2*pi*M/L;  %Domain length incorporated into wavenumbers
V = @(x,t) c*exp(-abs(x-5-c*t));  %Exact solution

c_1 = -1i*k/2;  c_2 = -1i*k./(1+k.^2);
RHS1 = @(v) c_1.*fft(real(ifft(v)).^2);
RHS2 = @(v) c_2.*fft(real(ifft(v)).^2 + 0.5*real(ifft(1i*k.*v)).^2 );
RHS  = @(v) RHS1(v) +  RHS2(v);  %Right hand side for v_t = RHS(v)
u = V(x,0.0);  v = fft(u);  %Initial condition
for n = 1:nmax   %Time stepping loop
    for jj = 1 : length(k)  % 2/3's dealiasing rule
        if ( abs(M(jj)) > (2*pi/L)*2*N/3 )
            v(jj) = 0;
        end
    end
    a = dt*RHS(v);    %RK4 update
    b = dt*RHS(v + a/2);
    c = dt*RHS(v + b/2);
    d = dt*RHS(v + c);
    v = v + (a + 2*b + 2*c + d)/6;
end
u = real(ifft(v));  norm(u - V(x,T),inf)
plot(x,u,'r.',x,V(x,T),x,V(x,0.0),'g')
```

4.3.1 Fourier spectral schemes

Figure 4.2a has a plot of the spatial discretization error from the Fourier spectral method. Notice that there is convergence, but it is much slower. This is to be expected since the function we are approximating is not smooth. In figure 4.2b the approximation looks to be reasonable at first glance, however, even with dealiasing, the approximation has artificial oscillations (see Figure 4.3b). Figure 4.3a has a plot of the Fourier spectral method for $N = 2^8$. Here the approximation has noticeable artificial oscillations, and also doesn't capture the general shape of the exact peakon data.

There are other Fourier spectral schemes than the one found in equation (4.3). A more direct Fourier spectral scheme can be obtained by simply taking the Fourier transform of equation (4.1),

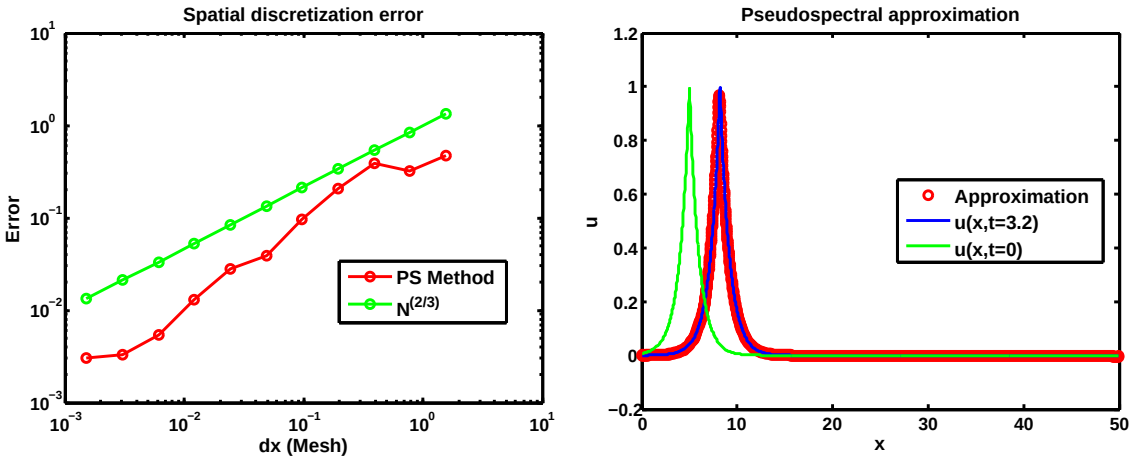
$$\hat{u}_t = \left(\frac{1}{1+k^2}\right)(-2\kappa ik\hat{u} - (3/2)ik\mathcal{F}(u^2) + \mathcal{F}(2u_x u_{xx}) + \mathcal{F}(uu_{xxx})). \tag{4.4}$$

Equation (4.4) has a dominant Fourier multiplier of $1/(1+k^2)$. The transformed BBM equation has a similar dominant Fourier multiplier (see equation (3.7)). The transformed KdV equation (see equation (2.4)) has a dominant Fourier multiplier of k^3. These Fourier multipliers explain why the CH equation provides a better approximation to one dimensional wave phenomena than the KdV equation for large spatial frequencies (see [13] and [49] for deeper discussions).

The dominant Fourier multiplier of $1/(1+k^2)$ in equation (4.4) has some useful features in the application of the small dispersion limit of the CH equation. Abenda, Grava, and Klein in [2] state

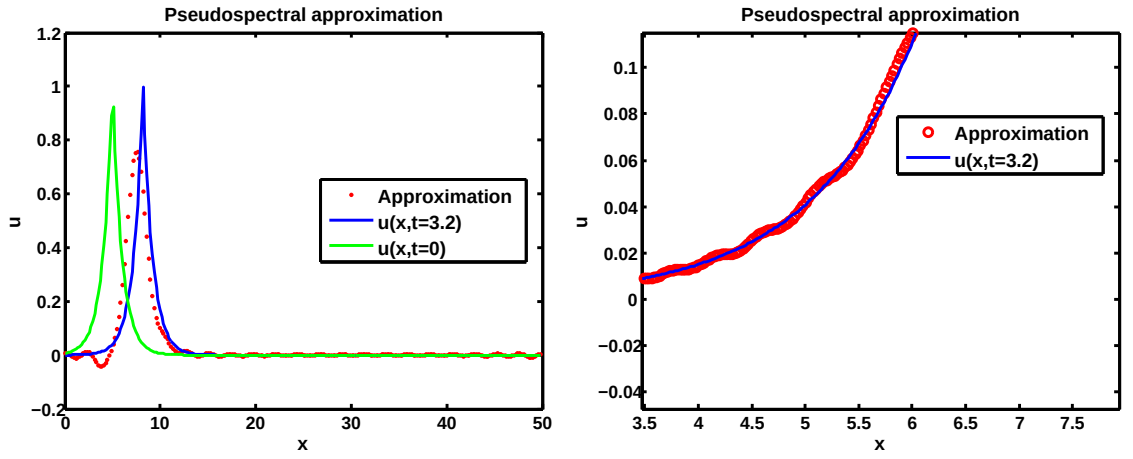
> First it provides a high frequency filtering which allows in practice for larger time steps in the computation. Secondly it suppresses the rapid modulated oscillations in the shock region of the dispersionless equation.

There is an overview of the small dispersion limit of the CH equation in section 4.4. Of course, one doesn't need to use a spectral method to approximate solutions to the CH equation. In [18], there is a nice comparison between multiple numerical methods: finite difference, spectral, multisymplectic, lie group integrator, and multipeakon methods. Dahlby (the author of [18]) found that spectral methods worked well for short time spans, but that for non–smooth initial conditions the solutions blew up for long time spans. The former behavior agrees with the behavior seen in figures 4.2 and 4.3. The latter behavior can be remedied by selecting a larger grid and a smaller time step, however, doing this results in a much longer computational time.



(a) (b)

Figure 4.2: (a) Spatial discretization error ($\Delta t = 2 * 10^{-4}$, $2^5 \leq N \leq 2^{15}$). (b) Fourier spectral approximation ($N = 2^{11}, \Delta t = 2 * 10^{-4}$).



(a) (b)

Figure 4.3: (a) Fourier spectral approximation ($N = 2^8$, $\Delta t = 2 * 10^{-4}$). (b) A zoom in of the Fourier spectral approximation ($N = 2^{11}$, $\Delta t = 2 * 10^{-4}$).

4.4 Small dispersion limit of the CH equation

One can study a variation of the CH equation where a small parameter ϵ^2 is attached to the dispersive terms:

$$u_t + 2\kappa u_x + 3uu_x - \epsilon^2(u_{xxt} + 2u_x u_{xx} + uu_{xxx}) = 0, \quad x \in \mathbb{R}, \quad t > 0. \tag{4.5}$$

The solutions of the small dispersion limit to the CH equation have noticably different behavior depending on what region of the (x, t) plane the solution is in. In particular, the emergence of a region of rapid oscillations occurs, this is very similar to the solutions of the small dispersion limit KdV equation. As with the small dispersion limit KdV equation, most researchers are interested in obtaining asymptotic formulas for the small dispersion limit to the CH equation. These asymptotic formulas are fairly involved, and different formulas are needed for different regions in the (x, t) plane. In [2] Abenda, Grava, and Klein have a thorough discussion of this research. In addition, they use a Fourier spectral method to verify the validity their various asymptotic formulas.

Using a Fourier spectral method we can obtain an approximation that is valid for the critical regions of the (x, t) plane where rapid oscillations occur. The method will be based off of equation (4.3):

$$\hat{u}_t = \left(\frac{-0.5ik(\epsilon^2 + \epsilon^2 k^2)}{(1 + \epsilon^2 k^2)}\right)\mathcal{F}(u^2) - \frac{ik}{(1 + \epsilon^2 k^2)}(\mathcal{F}(u^2) + \epsilon^2 k^2 \mathcal{F}(u_x^2)) - \left(\frac{2\kappa ik}{(1 + \epsilon^2 k^2)}\right)\hat{u}. \tag{4.6}$$

The last member in equation (4.6) is due to the assumption that $\kappa \neq 0$ (allows smooth solutions for the CH equation). We use a test case found in [2], namely, $\epsilon = 10^{-1.5}$, $u(x, 0) = -\text{sech}^2(x)$, $0 \leq t \leq 1$, and $x \in [-5, 5]$. Figure 4.4 has plots of the implementation of equation (4.6). For further comparisons between the small dispersion KdV and CH equations see [40].

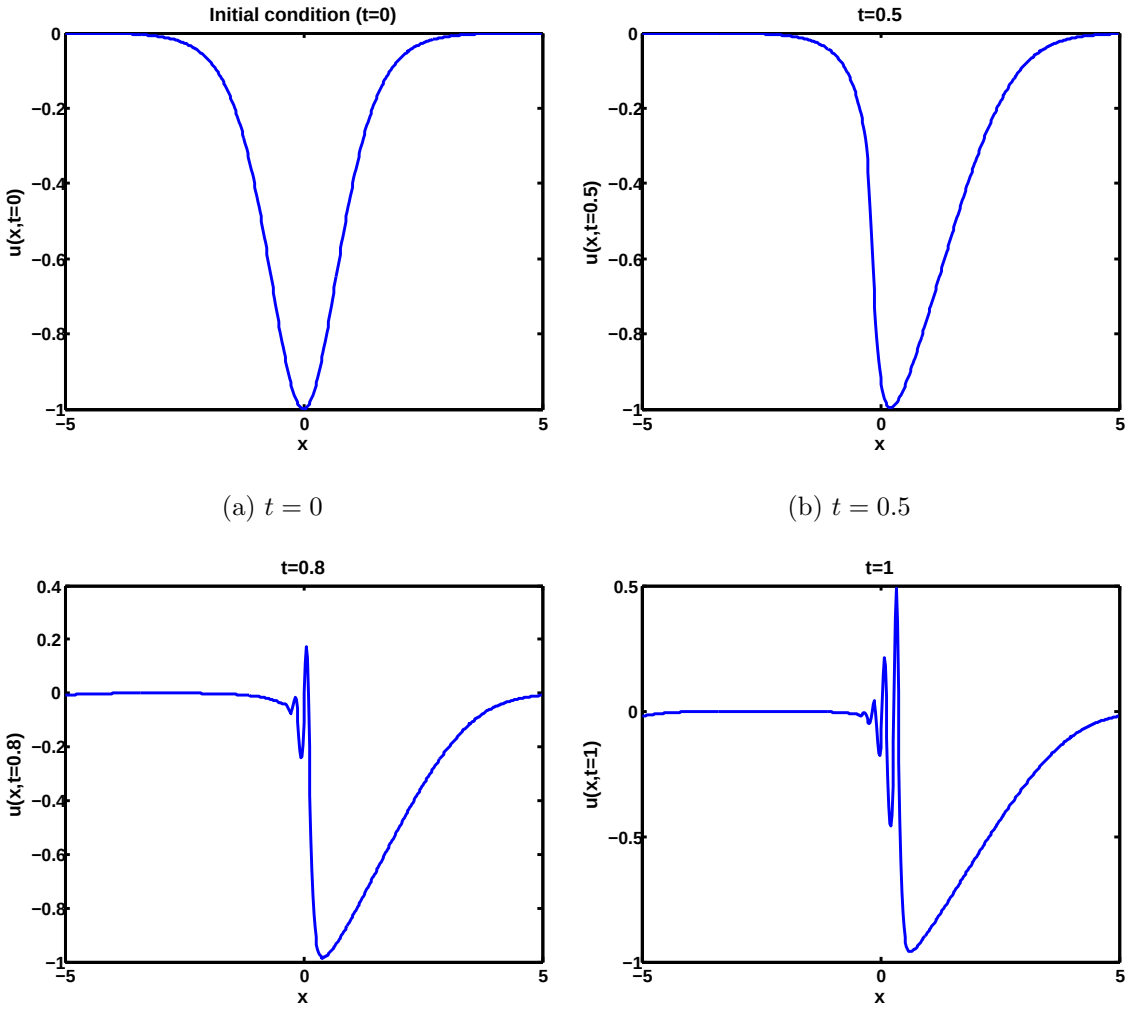
Figure 4.5 has the results of a test case for a nonsmooth initial condition ($\kappa = 0$). The oscillations are not as prominent as in the smooth case for the same simulation run time ($t = 1$). The nonsmooth test case is: $\epsilon = 10^{-1.5}$, $u(x, 0) = e^{-|x-2-t|}$, $0 \leq t \leq 1$, and $x \in [-5, 5]$.

A code example of approximating the small dispersion limit CH equation for the given test problem can be found in Code Listing 4.4.1.

Code Listing 4.4.1

Main Small Dispersion Limit file

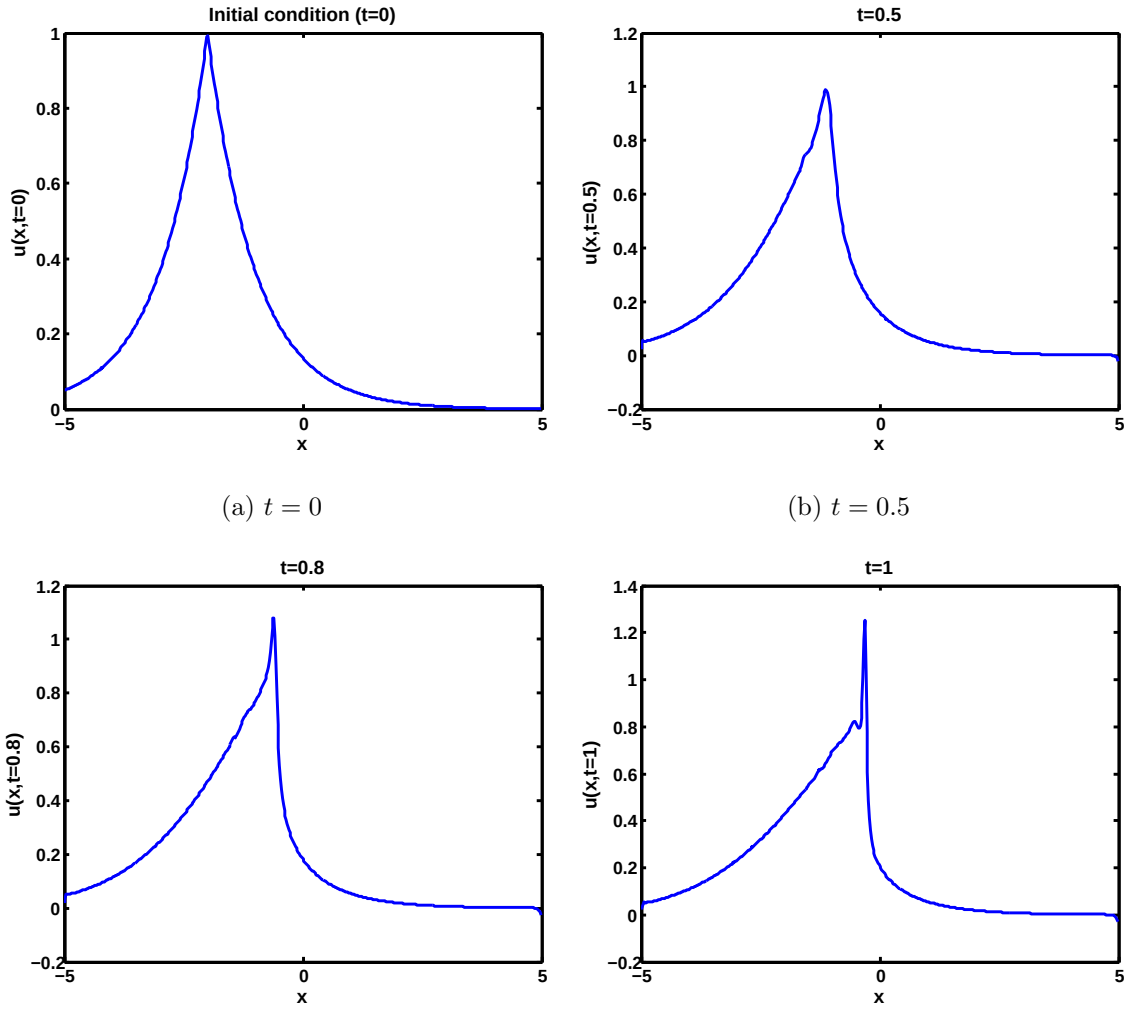
```
% Approximates the solution to the small dispersion CH equation, which is
% given by u_t +2*Ku_x+ 3uu_x - ep^2(u_{xxt}+2u_xu_{xx}+uu_{xxx})= 0. The
% spatial domain is [-5,5] and periodic boundary conditions are assumed.
% The approximation is done by a Fourier spectral method with RK4 time-
% stepping.  (K = 1.2).
clear all;  close all;  clc

N = 2^(9); T = 1.0; dt = 1e-3;  nmax = round(T/dt);
L = 10;  ep = 10^(-1.5);  kappa = 1.2;
x = (L/N)*(-N/2:N/2-1);

M = [0:N/2-1 0 -N/2+1:-1];
k = 2*pi*M/L; %Domain length incorporated into wavenumbers

V = @(x,t)  -sech(x).^2;
c_1 = (-1i*k/2).*(ep^2+ep^2*k.^2)./(1 + ep^2*k.^2);
c_2 = -1i*k./(1 + ep^2*k.^2);
RHS1 = @(v) c_1.*fft(real(ifft(v)).^2);
RHS2 = @(v) c_2.*fft(real(ifft(v)).^2 + ep^2*0.5*real(ifft(1i*k.*v)).^2 );
RHS3 = @(v) 2*kappa*c_2.*v;
RHS  = @(v) RHS1(v) +  RHS2(v) +  RHS3(v);

u0 = V(x,0.0);
v = fft(u0);

for n = 1:nmax
    a = dt*RHS(v);
    b = dt*RHS(v + a/2);
    c = dt*RHS(v + b/2);
    d = dt*RHS(v + c);
    v = v + (a + 2*b + 2*c + d)/6;
end
u = real(ifft(v));  plot(x,u)
```



(a) $t = 0$ (b) $t = 0.5$

(c) $t = 0.8$ (d) $t = 1$

Figure 4.4: Behavior of the small dispersion limit for the CH equation.



(a) $t = 0$ (b) $t = 0.5$

(c) $t = 0.8$ (d) $t = 1$

Figure 4.5: Behavior of the small dispersion limit for the CH equation with nonsmooth peakon initinal condition ($\kappa = 0$).

Chapter 5

FOURIER SPECTRAL METHOD FOR THE SHALLOW WATER (SW) EQUATIONS

The shallow water (SW) equations are a set of PDEs that model situations in fluid dynamics where the horizontal wave length is much larger than the depth of the fluid. Even though the SW equations are a simplification of the Navier–Stokes equations, they have a wide range of applicability, and also have a plethora of computational problems. Some common applications of the SW equations are: estuary, river, channel, lake, and coastal flows, ocean tides, storm surges, and tsunamis. For an introduction to the numerics and applications of the SW equations see, [90]. For work relating to tsunami modeling, see [62].

This chapter is focused on simulation and not high accuracy. The one and two–dimensional SW equations are considered. Simple topography is also examined.

5.1 1D Shallow Water equations

The one–dimensional shallow water equations are given by

$$\begin{bmatrix} h \\ uh \end{bmatrix}_t + \begin{bmatrix} uh \\ hu^2 + gh^2/2 \end{bmatrix}_x = \begin{bmatrix} 0 \\ -ghB_x \end{bmatrix}, \tag{5.1}$$

where $h = h(x,t)$ is the fluid height, $u = u(x,t)$ is the fluid horizontal velocity, g is the acceleration due to gravity, and B is the bathymetry/topography. See figures 5.1 and 5.2 for illustrations. The system in (5.1) is nonlinear and hyperbolic; a basic coverage of system (5.1) can be found in [61].

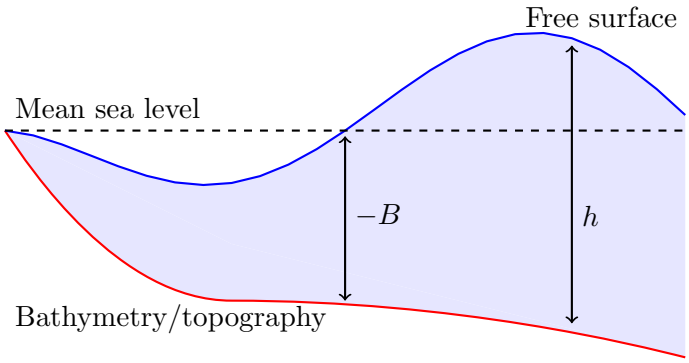
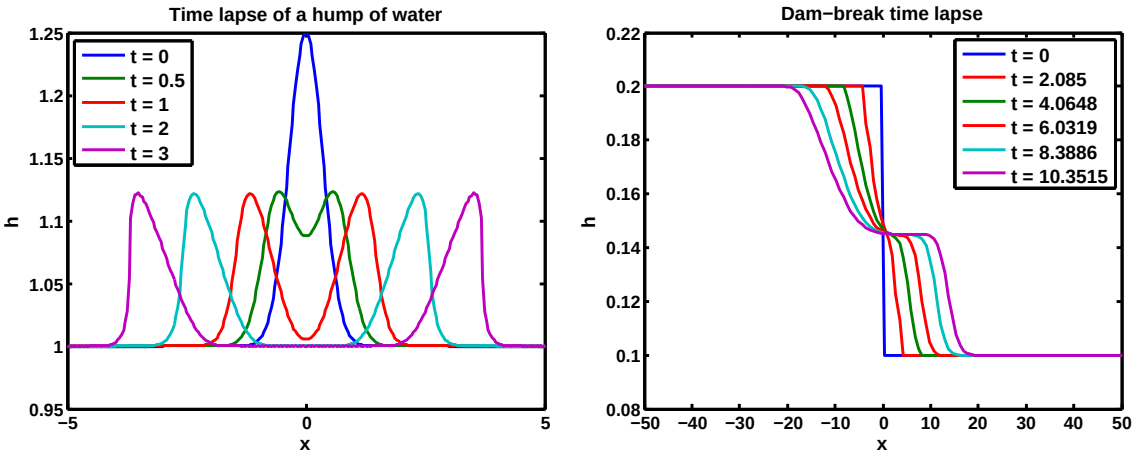


Figure 5.1: Relevant variables for the SW equations. Similar figures can be found in [62].



(a) Hump of water (showcases nonlinearity)

(b) Dam–break

Figure 5.2: Simulation of 1D–shallow water equation. (Similar figures can be found in [61])

When using a Fourier spectral method to discretize the spatial dimension of (5.1), a main concern is nonsmooth data, or smooth data that evolves into nonsmooth data. So, interesting phenomena similar to shock waves and the like might not be approximated with reasonable accuracy. Even when given smooth data, long time spans, errors due to aliasing, and establishing stability are main concerns. Also, when implementing the Fourier pseudospectral scheme we need to use periodic boundary conditions (if we want to utilize the Fast Fourier transform). These constraints give a limited numerical method, but as with the CH equation, select test cases may be able to provide useful data.

5.2 Fourier Spectral methods for the 1D SW equations

Although (5.1) is a system, we can still employ a Fourier pseudospectral algorithm. To do this, we evaluate the spatial derivatives of each component from (5.1) spectrally. The time stepping is then carried out in the original (x,t) space (as opposed to Fourier space – compare Code Listings 2.4.1 and 2.4.3). Psuedospectrally evaluating the spatial derivatives one has

$$\begin{bmatrix} h \\ uh \end{bmatrix}_t + \begin{bmatrix} \Re(\mathcal{F}^{-1}[ik\mathcal{F}(uh)]) \\ \Re(\mathcal{F}^{-1}[ik\mathcal{F}(hu^2 + gh^2/2)]) \end{bmatrix} = \begin{bmatrix} 0 \\ -ghB_x(x,t) \end{bmatrix}. \tag{5.2}$$

Applying the method of lines, we can then update the first component of (5.2), and then naively update the second component of (5.2). In the second component of (5.2) the quantity uh is updated, but the term hu^2 is required. In order to avoid explicitly updating u, we can calculate hu^2 by the expression $(uh)^2/h$ (we can select problems such that $h(x,t) \neq 0$).

To clarify the time stepping procedure, consider the following example with a forward euler method.

Algorithm for 1D SWE

- Step 1: Update $h(x,t)$ by computing $h_{n+1} = h_n + (\Delta t)F_1(u_n, h_n)$, where $h_n = h(x,t_n)$, $u_n = u(x,t_n)$, and $F_1(u,h) = -(uh)_x$.
- Step 2: Update uh by computing $(uh)_{n+1} = (uh)_n + (\Delta t)F_2((uh)_n, h_{n+1})$, where $(uh)_n = u(x,t_n)\cdot h(x,t_n)$, and $F_2 = -(hu^2 + gh^2/2)_x$.

In step 2 we are treating uh as a variable (not explicitly calculating $u_n \cdot h_n$). It should also be noted that using this scheme we never explicitly update u_n. In addition, notice that h_{n+1} is an input into the function F_2, and not h_n. When using this scheme for the Fourier pseudospectral approach, larger time steps (as well as longer time spans) can be used without instabilities. Aliasing still afflicts this scheme, however, a smoother (discussed below) can be employed to combat aliasing. Code Listings 5.3.1 through 5.3.5 have the described scheme implemented. One can of course use a built in ODE solver instead.

Fourier spectral and pseudospectral methods can suffer from artificial aliasing [1]. In most cases, Orszag's two thirds rule will not be enough to drastically reduce undesired aliasing. A concept called *spectral viscosity* for scalar nonlinear conservation laws can be found in [80], [79], and [1]. The basic idea is to add a diffusive term to the diffusion free problem; in hopes that the high frequencies that cause aliasing will be controlled appropriately. A similar concept called *hyperviscosity* has also been used with success (for instance see [64] and [7]). Figures 5.4b, 5.4d , 5.5a , and 5.5b have plots of the Fourier spectral method with and without hyperviscosity. A comparison of Fourier power spectrum for two test cases is displayed in figures 5.5c and 5.5d.

The four test cases used are as follows (the hump of water and dam–break test cases were adapted from [61]):

	Hump of water	Dam–break	Sine wave	Sine wave
Initial depth $h(x,0)$	$1+0.4e^{-5(x-5)^2}$	$\begin{cases} 3, & \text{if } x \leq 5 \\ 1, & \text{if } x > 5 \end{cases}$	$2+\sin\left(\frac{x\pi}{100}\right)$	$1+\sin\left(\frac{x\pi}{10}\right)$
Initial velocity $u(x,0)$	0	0	0	0
N	2^7	2^7	2^7	2^7
Δt	10^{-3}	10^{-3}	10^{-1}	10^{-3}
Final simulation time	3	8	15	1.5
Spatial domain length	$[0,10]$	$[0,100]$	$[0,200]$	$[0,10]$
Gravity (g)	1	1	9.81	9.81
Topography	0	0	0	$1-\frac{x}{10}$
Boundary conditions	Periodic	Periodic	Periodic	Periodic
Associated figure	5.4 and 5.5a	5.5b	5.3	5.6

Table 5.1: Test problems for the 1D SW equations.

See table 5.2 for an organization of the Code Listings.

[1] In fact, many spectral methods based on differentiation matrices also have this issue. For instance, it is common for Chebyshev and Radial Basis spectral methods to exhibit this behavior for diffusion free problems. Multiple eigenvalues of these interpolation and differentiation operators tend to lie in the right half of the complex plane. This becomes problematic when using explicit time stepping schemes.

5.3 Implementation for the 1D SW equations

Example codes for the test problems shown in table 5.1 are organized in table 5.2.

Method	Code Listing
Fourier–Pseudospectral (Water Hump)	5.3.1
Fourier–Pseudospectral (Water Hump, Hyperviscosity)	5.3.2
Fourier–Pseudospectral (Dam-Break, Hyperviscosity)	5.3.3
Fourier–Pseudospectral (Sine wave)	5.3.4
Fourier–Pseudospectral (Sine wave, Hyperviscosity, topography)	5.3.5

Table 5.2: Organization of spectral method and their respective code listings (SW equation).

Hyperviscosity is a term of the form $\gamma\Delta^{2s}g(x,t)$, where $\gamma \in \mathbb{R}$, and $s \in \mathbb{N}$ is the order of the Laplacian. The natural number s satisfies $s \geq 1$. If we wanted to attach hyperviscosity to the first component of equation (5.1), we would replace $h_t + (uh)_x = 0$ with $h_t + (uh)_x = \gamma\Delta^{2s}h$. In Code Listings 5.3.2 and 5.3.3, γ was selected through trial and error, and $s = 1$. In those two cases $\gamma \approx -n^{-0.68}$ and $\gamma \approx -n^{-0.65}$, respectively, give decent results. Hyperviscosity in a related context can be found in [30].

For nonsmooth initial conditions, or smooth initial conditions that propagate into nonsmooth data, the effects of hyperviscosity are very visible. This can be seen in figures 5.4b, 5.4d , 5.5a , and 5.5b.

If the initial data is sufficiently smooth (and remains sufficiently smooth throughout the simulation), then hyperviscosity might not add any benefits. Code Listing 5.3.4 has an example of this situation. Figure 5.3 visualizes the implementation of Code Listing 5.3.4.

The pseudospectral methods presented are not incredibly robust, and significantly tweaking the parameters found in table 5.1 will require the user to modify the code as needed. See [24] for a reference on stability of finite differences applied to systems of PDEs (emphasis on fluid dynamics).

Code Listing 5.3.1

Main Pseudospectral (no hyperviscosity) file

```
% Approximates a solution to the 1D SW equations given by the system
% (h)_t  + (uh)_x             = 0                (1)
% (uh)_t + (u^2h + 0.5gh^2)_x = 0,               (2)
% with an initial hump h(x,0) = 1 + 0.4*exp(-5(x-5)^2) and initial velocity
% of u(x,0) = 0.  Periodic boundary conditions on [0,10] are assumed, and
% the approximation is done by a Fourier pseudospectral method, with
% a RK4 time stepping.  No hyperviscosity is used.
clear all;  close all;  clc;
N = 2^(8);   g = 1.0;
L = 10.0;    B = L/(2*pi); %Domain rescale factor
Tfin = 3.0;  dt = 1e-3;    nmax = floor(Tfin/dt);
x = linspace( 0, L, N )';

h = 1.0 + 0.4*exp(-5*(x-L/2).^2);  h0 = h;  %Initial depth
u = zeros(size(h));                         %Initial velocity
uh = u.*h;                                  %Initial discharge

k = [0:N/2-1 0 -N/2+1:-1]';

%Pseudospectrally evaluate spatial derivatives.
RHS_h  = @(h,uh) -real(ifft(1i*k.*fft(uh)));                  %RHS for (1)
RHS_uh = @(h,uh) -real(ifft(1i*k.*fft(uh.^2. h+0.5*g*h.^2))); %RHS for (2)

for jj = 1 : nmax
    k1 = (dt/B)*RHS_h(h       ,uh);  %Update h via RK4
    k2 = (dt/B)*RHS_h(h + k1/2,uh);
    k3 = (dt/B)*RHS_h(h + k2/2,uh);
    k4 = (dt/B)*RHS_h(h + k3  ,uh);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = (dt/B)*RHS_uh(h,uh);        %Update uh via RK4
    k2 = (dt/B)*RHS_uh(h,uh + k1/2);
    k3 = (dt/B)*RHS_uh(h,uh + k2/2);
    k4 = (dt/B)*RHS_uh(h,uh + k3);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
end
plot(x,h0,'r',x,h)
```

Code Listing 5.3.2

Main Pseudospectral (hyperviscosity) file

```
% Approximates a solution to the 1D SW equations given by the system
% (h)_t  + (uh)_x             = 0                (1)
% (uh)_t + (u^2h + 0.5gh^2)_x = 0,               (2)
% with an initial hump h(x,0) = 1 + 0.4*exp(-5(x-5)^2) and initial velocity
% of u(x,0) = 0.  Periodic boundary conditions on [0,10] are assumed, and
% the approximation is done by a Fourier pseudospectral method, with
% a RK4 time stepping.  Hyperviscosity is used.
clear all;  close all;  clc;
N = 2^(8);   g = 1.0;
L = 10.0;    B = L/(2*pi); %Domain rescale factor
Tfin = 3.0;  dt = 1e-3;    nmax = floor(Tfin/dt);
x = linspace( 0, L, N )';

h = 1.0 + 0.4*exp(-5*(x-L/2).^2);  h0 = h;  %Initial depth
u = zeros(size(h));                         %Initial velocity
uh = u.*h;                                  %Initial discharge

k = [0:N/2-1 0 -N/2+1:-1]';

%Pseudospectrally evaluate spatial derivatives.
RHS_h  = @(h,uh) -real(ifft(1i*k.*fft(uh))) - ...             %RHS for (1)
n^(-0.68)*real(ifft(k.^2.*fft(h)));             %(hyperviscosity term)
RHS_uh = @(h,uh) -real(ifft(1i*k.*fft(uh.^2. h+0.5*g*h.^2))); %RHS for (2)

for jj = 1 : nmax
    k1 = (dt/B)*RHS_h(h       ,uh);  %Update h via RK4
    k2 = (dt/B)*RHS_h(h + k1/2,uh);
    k3 = (dt/B)*RHS_h(h + k2/2,uh);
    k4 = (dt/B)*RHS_h(h + k3  ,uh);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = (dt/B)*RHS_uh(h,uh);        %Update uh via RK4
    k2 = (dt/B)*RHS_uh(h,uh + k1/2);
    k3 = (dt/B)*RHS_uh(h,uh + k2/2);
    k4 = (dt/B)*RHS_uh(h,uh + k3);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
end
plot(x,h0,'r',x,h)
```

Code Listing 5.3.3

Main Pseudospectral (hyperviscosity) file

```
% Approximates a solution to the 1D SW equations given by the system
% (h)_t  + (uh)_x             = 0                (1)
% (uh)_t + (u^2h + 0.5gh^2)_x = 0,               (2)
% dam-break I.C. h(x >= L/2) = 1;  h(x <= L/2) = 3; and initial velocity
% of u(x,0) = 0.  Periodic boundary conditions on [0,100] are assumed, and
% the approximation is done by a Fourier pseudospectral method, with
% a RK4 time stepping.  Hyperviscosity is used.
clear all;  close all;  clc;
n = 2^(8);   g = 1.0;
L = 100.0;    B = L/(2*pi); %Domain rescale factor
Tfin = 8.0;  dt = 1e-3;    nmax = floor(Tfin/dt);
x = linspace( 0, L, n )';

h(x >= L/2) = 1;  h(x <= L/2) = 3;  h = h';  %Initial depth
h0 = h;
u = zeros(size(h));                          %Initial velocity
uh = u.*h;                                   %Initial discharge

k = [0:n/2-1 0 -n/2+1:-1]';

%Pseudospectrally evaluate spatial derivatives.
RHS_h  = @(h,uh) -real(ifft(1i*k.*fft(uh))) - ...             %RHS for (1)
n^(-0.65)*real(ifft(k.^2.*fft(h)));             %(hyperviscosity term)
RHS_uh = @(h,uh) -real(ifft(1i*k.*fft(uh.^2. h+0.5*g*h.^2))); %RHS for (2)

for jj = 1 : nmax
    k1 = (dt/B)*RHS_h(h       ,uh);  %Update h via RK4
    k2 = (dt/B)*RHS_h(h + k1/2,uh);
    k3 = (dt/B)*RHS_h(h + k2/2,uh);
    k4 = (dt/B)*RHS_h(h + k3  ,uh);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = (dt/B)*RHS_uh(h,uh);        %Update uh via RK4
    k2 = (dt/B)*RHS_uh(h,uh + k1/2);
    k3 = (dt/B)*RHS_uh(h,uh + k2/2);
    k4 = (dt/B)*RHS_uh(h,uh + k3);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
end
plot(x,h0,'r',x,h)
```

Code Listing 5.3.4

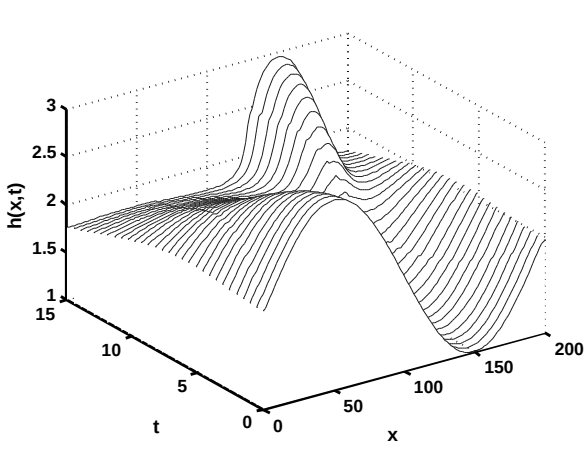
Main Pseudospectral (no hyperviscosity) file

```
% Approximates a solution to the 1D SW equations given by the system
% (h)_t  + (uh)_x             = 0                (1)
% (uh)_t + (u^2h + 0.5gh^2)_x = 0,               (2)
% sine wave initial depth h(x,0) = 2 + sin((pi/100)x) initial velocity
% of u(x,0) = 0.  Periodic boundary conditions on [0,200] are assumed, and
% the approximation is done by a Fourier pseudospectral method, with
% a RK4 time stepping.  No hyperviscosity is used (solution stays smooth).
clear all;  close all;  clc;
N = 2^(7);   g = 9.81;
L = 200.0;    B = L/(2*pi); %Domain rescale factor
Tfin = 15.0;  dt = 1e-1;    nmax = floor(Tfin/dt);
x = linspace( 0, L, N )';

h = 2.0 + sin((2*pi/L) * x);              %Initial depth
u = zeros(size(h));                       %Initial velocity
uh = u.*h;                                %Initial discharge

k = [0:N/2-1 0 -N/2+1:-1]';

%Pseudospectrally evaluate spatial derivatives.
RHS_h  = @(h,uh) -real(ifft(1i*k.*fft(uh)));                  %RHS for (1)
RHS_uh = @(h,uh) -real(ifft(1i*k.*fft(uh.^2. h+0.5*g*h.^2))); %RHS for (2)

for jj = 1 : nmax
    k1 = (dt/B)*RHS_h(h       ,uh);  %Update h via RK4
    k2 = (dt/B)*RHS_h(h + k1/2,uh);
    k3 = (dt/B)*RHS_h(h + k2/2,uh);
    k4 = (dt/B)*RHS_h(h + k3  ,uh);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = (dt/B)*RHS_uh(h,uh);        %Update uh via RK4
    k2 = (dt/B)*RHS_uh(h,uh + k1/2);
    k3 = (dt/B)*RHS_uh(h,uh + k2/2);
    k4 = (dt/B)*RHS_uh(h,uh + k3);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
    H(:,jj) = h;
end
[X,T] = meshgrid(x,linspace(0,Tfin,size(H,2)));
figure(1),  surf(X,T,H'),  shading interp
figure(2),  plot(x,h)
```

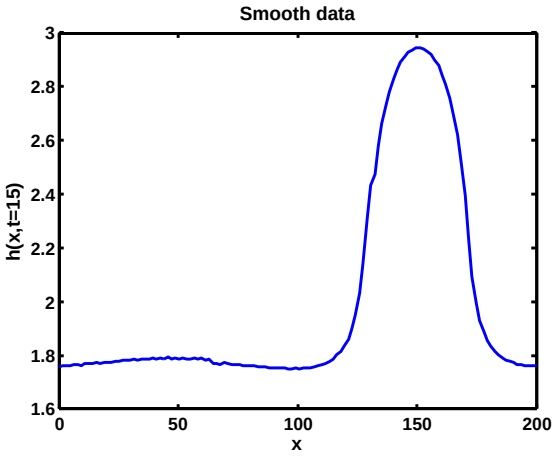
Code Listing 5.3.5

Main Pseudospectral (hyperviscosity and topography) file

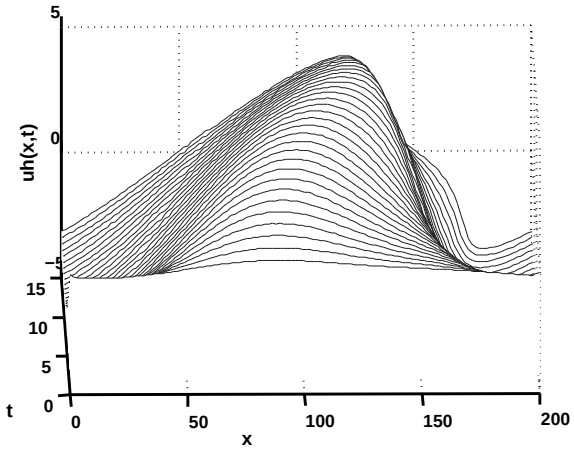
```
% Approximates a solution to the 1D SW equations given by the system
% (h)_t  + (uh)_x             = 0                (1)
% (uh)_t + (u^2h + 0.5gh^2)_x = -ghB_x,          (2)
% sine wave initial depth h(x,0) = 1 + sin((pi/10)x) initial velocity
% of u(x,0) = 0.  Periodic boundary conditions on [0,10] are assumed, and
% the approximation is done by a Fourier pseudospectral method, with
% a RK4 time stepping.  Hyperviscosity is used, and nonzero topography is
% imposed.
clear all;  close all;  clc;
N = 2^(7); g = 9.81; L = 10; b = L/(2*pi);
Tfin = 1.5; dt = 1e-3; nmax = floor(Tfin/dt);
x = linspace(0,L,N)'; h = 1 + sin((1*pi/L)*x);
u = zeros(size(h)); uh = u.*h;
k = [0:N/2-1 0 -N/2+1:-1]';

% Linear topography 1-x/10
p = polyfit([0 10],[1 0.0],1);
B    = @(x,t) (-1/10)*x + 1;
B_x  = @(x,t) (-1/10);

RHS_h  = @(h,uh,t) -real(ifft(1i*k.*fft(uh))) - ...
(5e-1*N^(-0.5))*real(ifft(-(1i*k).^2.*fft(h))); % Hyperviscosity
RHS_uh = @(h,uh,t) -real(ifft(1i*k.*fft(uh.^2 ./ h + 0.5*g*h.^2))) - ...
g*h.*B_x(x,t);  % Linear topography


for jj = 1 : nmax
    t = jj*dt;
    k1 = (dt/b)*RHS_h(h       ,uh,t);  %       RK4
    k2 = (dt/b)*RHS_h(h + k1/2,uh,t);
    k3 = (dt/b)*RHS_h(h + k2/2,uh,t);
    k4 = (dt/b)*RHS_h(h + k3  ,uh,t);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = (dt/b)*RHS_uh(h,uh,t);
    k2 = (dt/b)*RHS_uh(h,uh + k1/2,t);
    k3 = (dt/b)*RHS_uh(h,uh + k2/2,t);
    k4 = (dt/b)*RHS_uh(h,uh + k3,t);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
end
plot(x,h)
```



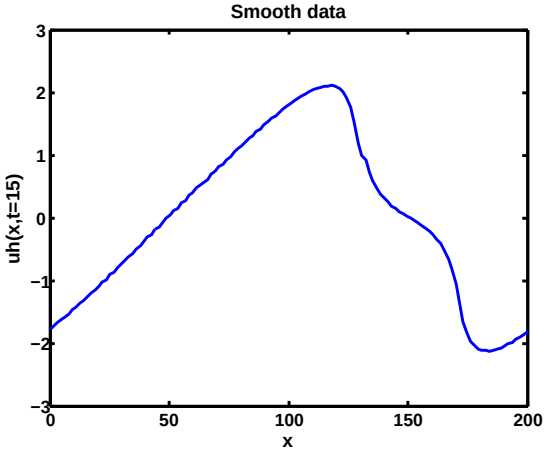
(a) Depth h



(b) $h(x, t = 15)$

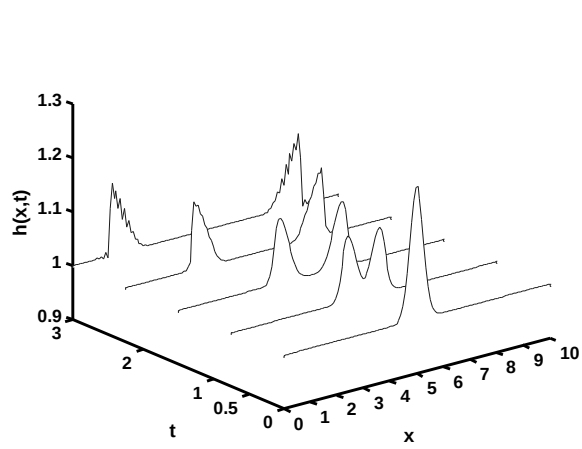


(c) Momentum uh

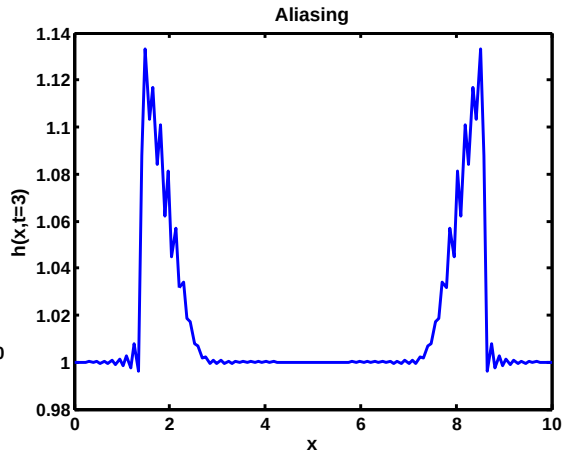


(d) $uh(x, t = 15)$

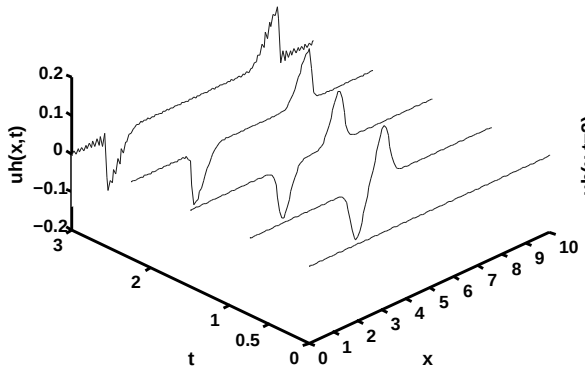
Figure 5.3: Results of the Fourier pseudospectral method (this figure corresponds to Code Listing 5.3.4). Relevant parameters: $g = 9.81$, $\Delta t = 10^{-1}$, $N = 2^7$, $h(x, 0) = 2 + \sin(x\pi/100)$, and $u(x, 0) \equiv 0$. This is an example of smooth data that does not need hyperviscosity.



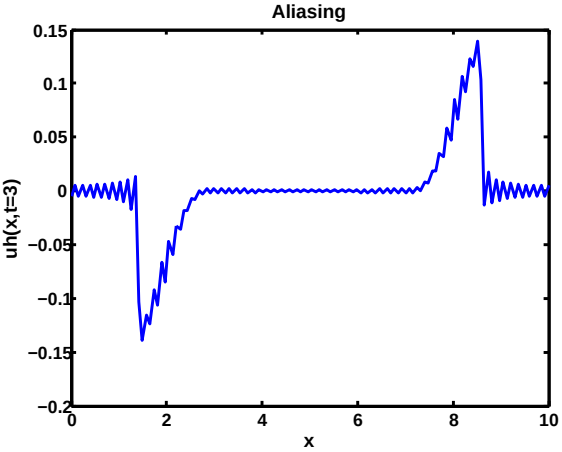
(a) Depth h



(b) $h(x, t = 3)$

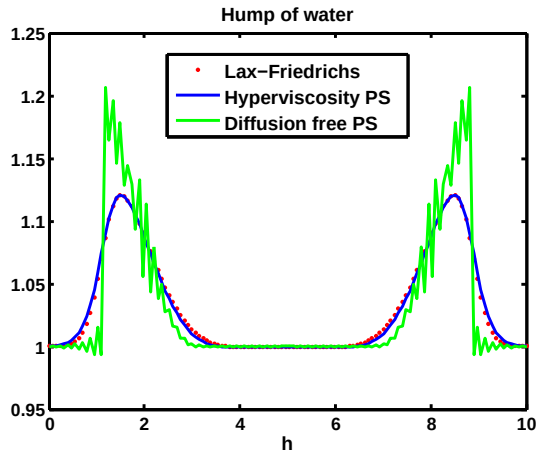


(c) Momentum uh

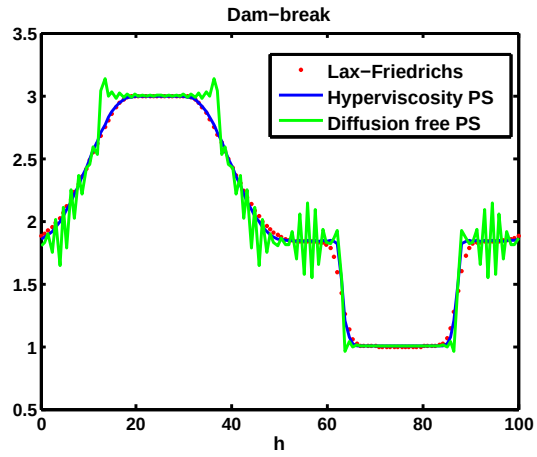


(d) $uh(x, t = 3)$

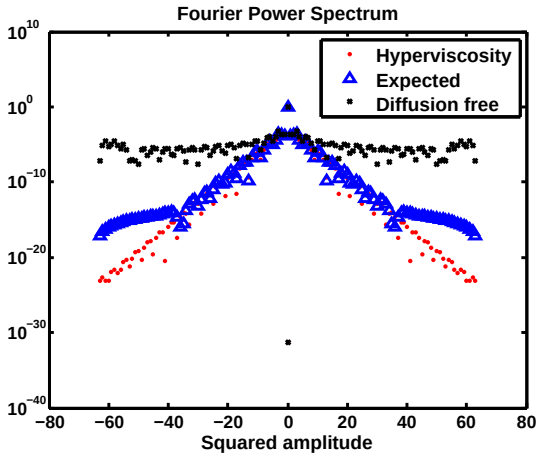
Figure 5.4: Results of the Fourier pseudospectral method without hyperviscosity (this figure corresponds to Code Listing 5.3.1). Relevant parameters: $g = 1$, $\Delta t = 10^{-3}$, $N = 2^8$, $h(x, 0) = 1 + \exp(-8(x - 5)^2)/4$, and $u(x, 0) \equiv 0$. Although the initial condition is smooth, the nonlinearity of the SW equations generate less smooth data. "The front of the wave (relative to its direction of motion) steepens through a compression wave into a shock, while the back spreads out as a rarefaction wave." (R. J. LeVeque, [61])



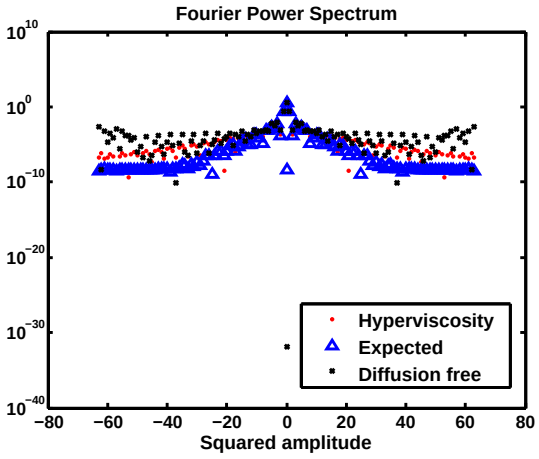
(a) Hump of water ($t = 3$)



(b) Dam–break ($t = 8$)

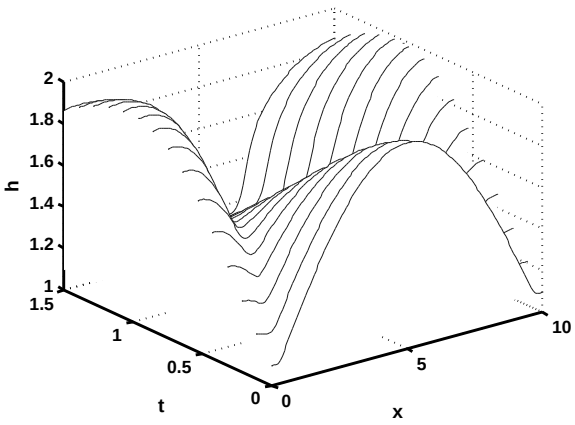


(c) Spectrum for the water hump

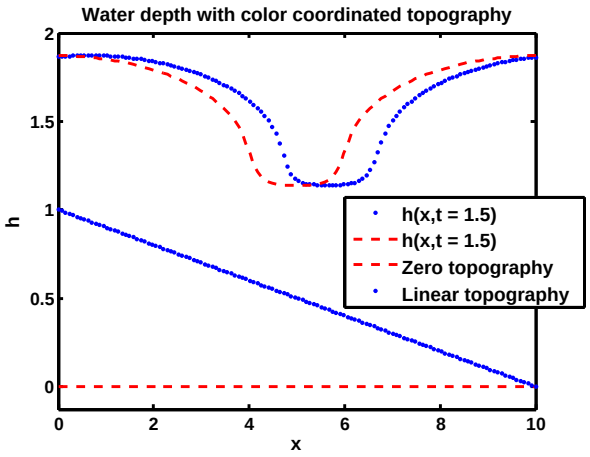


(d) Spectrum for the dam-break

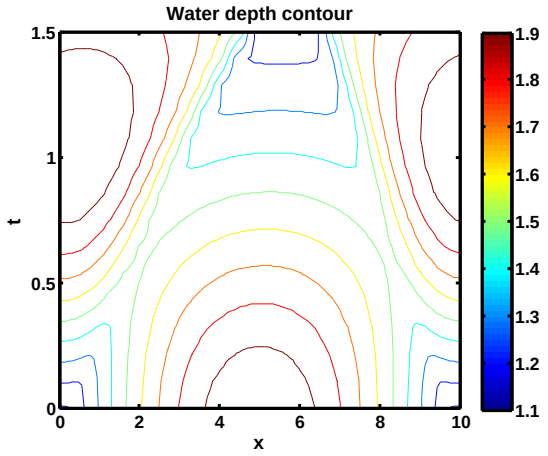
Figure 5.5: Results of the Fourier pseudospectral method with and without hyperviscosity. Relevant parameters: $g = 1$, $\Delta t = 10^{-3}$, $N = 2^7$. The effects of hyperviscosity are clear; and the Fourier power spectrum can be used to study how hyperviscosity can assist in improving stability and dealiasing. Figures 5.5a and 5.5c correspond to Code Listing 5.3.2. Figures 5.5b and 5.5d correspond to Code Listing 5.3.3.



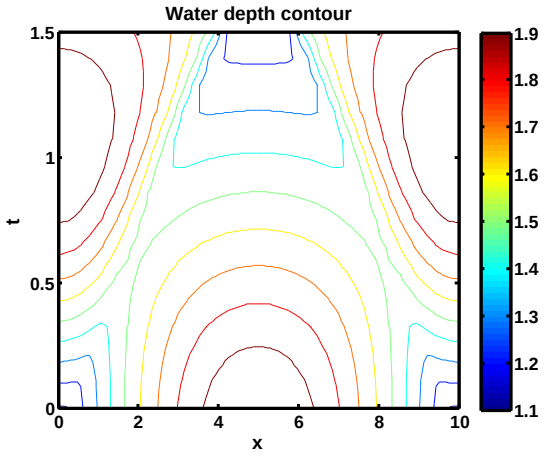
(a) Sine wave



(b) Topography/ No topography



(c) Water depth contour (linear topography)



(d) Water depth contour (zero topography)

Figure 5.6: Results of the Fourier pseudospectral method (corresponds to Code Listing 5.3.5).

5.4 2D Shallow Water equations

The two–dimensional depth averaged shallow water equations are given by

$$h_t + (uh)_x + (vh)_y = 0, \tag{5.3}$$
$$(uh)_t + (u^2h + 0.5gh^2)_x + (uvh)_y = -ghB_x, \tag{5.4}$$
$$(vh)_t + (uvh)_x + (v^2h + 0.5gh^2)_y = -ghB_y, \tag{5.5}$$
$$\tag{5.6}$$

where $h = h(x, y, t)$, $u = u(x, y, t)$, $v = v(x, y, t)$ are water depth, depth averaged velocity components in the two horizontal directions. The function B represents the bathymetry or topography, and g is the acceleration due to gravity (see [61] and [62]). These equations are a very powerful tool for modeling fluid flow or wave propagation problems. See [62] for a detailed discussion of tsunami modeling.

For the 2D shallow water equations, a simple simulation will be featured, just to present a Fourier spectral method. As with the 1D SW equations, there are a number of difficulties that make the Fourier spectral method not an attractive numerical method for approximating these equations. However, it is feasible to run simple simulation under some assumptions. The simulation ran for the 2D SW equations is a water droplet in a bathtub (rectangular parallelepiped) with periodic boundary conditions. The water droplet collapses and generates outward momentum (see figure 5.7).

5.5 Fourier Spectral methods for the 2D SW equations

The same naive procedure is used as was done with the 1D shallow water equations. That is psuedospectrally evaluate the spatial derivatives, and use the method of lines to create a system of ODEs; which are approximated by a finite difference.

$$\begin{bmatrix} h \\ uh \\ vh \end{bmatrix}_t + \begin{bmatrix} \Re(\mathcal{F}^{-1}[ik_x\mathcal{F}(uh)]) + \mathcal{F}^{-1}([ik_y\mathcal{F}(vh)]) \\ \Re(\mathcal{F}^{-1}[ik_x\mathcal{F}(hu^2 + gh^2/2)]) + \Re(\mathcal{F}^{-1}[ik_y\mathcal{F}(uvh)]) \\ \Re(\mathcal{F}^{-1}[ik_x\mathcal{F}(uvh)]) + \Re(\mathcal{F}^{-1}[ik_y\mathcal{F}(hv^2 + gh^2/2)]) \end{bmatrix} = \begin{bmatrix} 0 \\ -ghB_x(x, y, t) \\ -ghB_y(x, y, t) \end{bmatrix}. \tag{5.7}$$

In equation 5.7, $\mathcal{F}(g(x, y))$ is the 2–dimensional Fourier transform, and k_x and k_y are the respective wave numbers in the x and y variables. For simplicity, a forward euler scheme is used to march forward in time. Similar to the 1D SW equations, we use a implicit–explicit type method. One can of course use a built in ODE solver instead.

Algorithm for 2D SWE

- Step 1: Update $h(x,y,t)$ by computing $h_{n+1} = h_n + (\Delta t)F_1(u_n, v_n, h_n)$, where $u_n = u(x,y,t_n)$, $v_n = v(x,y,t_n)$, $h_n = h(x,y,t_n)$, and $F_1(u,v,h) = -(uh)_x - (vh)_y$.
- Step 2: Update uh by computing $(uh)_{n+1} = (uh)_n + (\Delta t)F_2((uh)_n, (vh)_n, h_{n+1})$, where $(uh)_n = u(x,y,t_n) \cdot h(x,y,t_n)$, $(vh)_n = v(x,y,t_n) \cdot h(x,y,t_n)$, and $F_2(u,v,h) = -(u^2h + 0.5gh^2)_x - (uvh)_y$.
- Step 3: Update vh by computing $(vh)_{n+1} = (vh)_n + (\Delta t)F_3((uh)_{n+1}, (vh)_n, h_{n+1})$, and $F_3(u,v,h) = -(v^2h + 0.5gh^2)_y - (uvh)_x$.

5.6 Implementation for the 2D SW equations

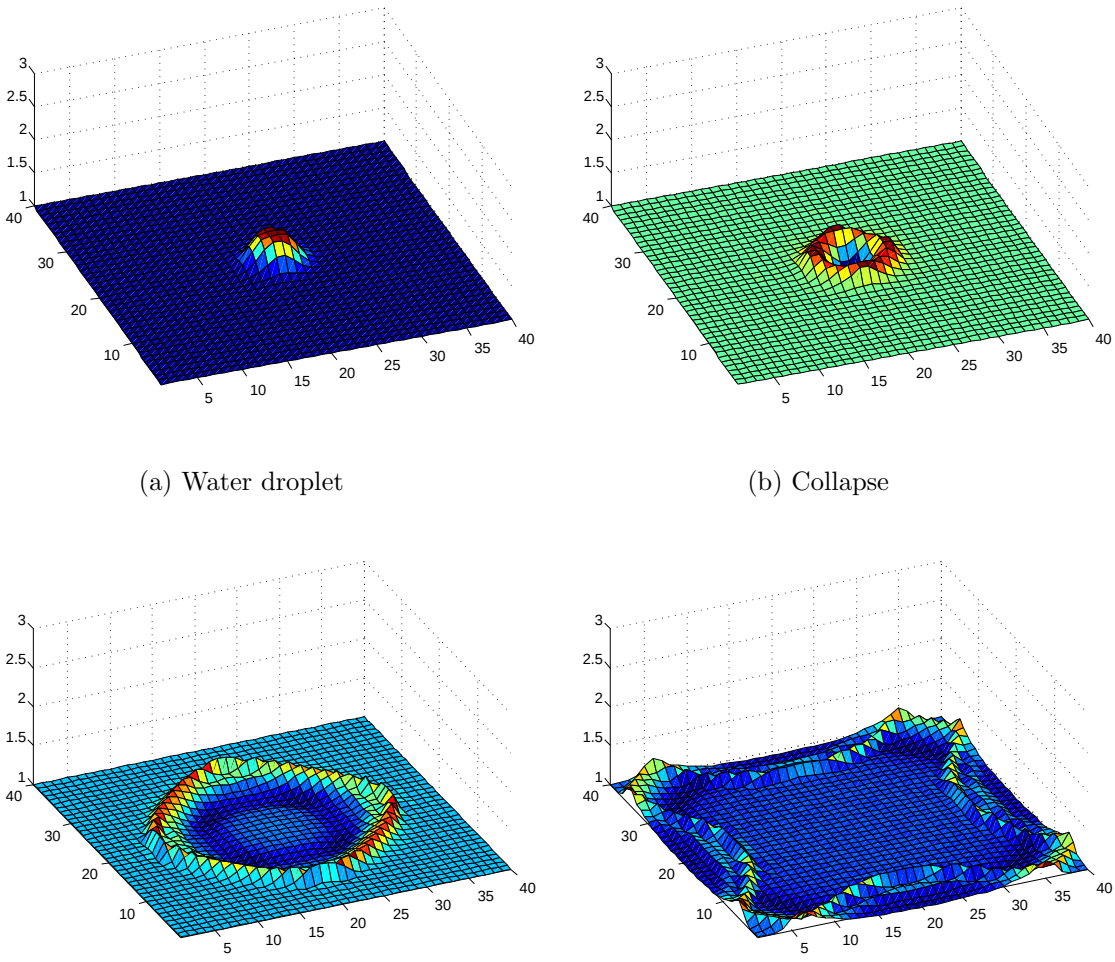
The implementation for the Fourier pseudospectral method described above can be found in Code Listing 5.6.1. See figure 5.7 for plots of the simulation. See [3] and [19] for examples of more robust numerical methods not based on a spectral approach.

Code Listing 5.6.1

Main RBF Pseudospectral file

```
% A 2D SWE simulation.  Water droplet in a rectangular paralelipiped.  The
% boundary conditions are periodic.  Domain is [-100,100]x[-100,100].
clear all;  close all;  clc;
N = 2^(6);  L = 200;  x = linspace(-L/2,L/2,N)';  y = x;  g = 9.8;
Tfin = 45;  dt = 5e-2;  nmax = floor(Tfin/dt);
B = L/(2*pi);  %Domain scaling parameter
k = [0:N/2-1 0 -N/2+1:-1]';  %1D wave numbers
[kx,ky] = meshgrid(k);       %2D wave numbers
[xx, yy] = meshgrid(x,y);
%Initial condition
h  = 1.0 + exp(-(xx.^2+yy.^2)/(L/2));  uh = zeros(size(h));  vh = uh;
RHS_h  = @(h,uh,vh) -real(ifft2(1i*kx.*fft2(uh))) - ...
    real(ifft2(1i*ky.*fft2(vh)));
RHS_uh = @(h,uh,vh) -real(ifft2(1i*kx.*fft2(uh.^2 ./ h + 0.5*g*h.^2))) -...
    -real(ifft2(1i*ky.*fft2(((vh)./h).*uh)));
RHS_vh = @(h,uh,vh) -real(ifft2(1i*ky.*fft2(vh.^2 ./ h + 0.5*g*h.^2))) -...
    -real(ifft2(1i*kx.*fft2(((uh)./h).*vh)));

grid = surf(h);                                  %plot simulation
axis([1 N 1 N 1 3]); view(-20,55);  hold all;  %plot simulation
for jj = 1 : nmax
    set(grid ,'zdata', h);  drawnow            %plot simulation
```

```
    h  = h  + (dt/B)*RHS_h(h ,uh,vh);          %naive forward euler
    uh = uh + (dt/B)*RHS_uh(h,uh,vh);
    vh = vh + (dt/B)*RHS_vh(h,uh,vh);
end
```



(a) Water droplet

(b) Collapse

(c) Outward momentum

(d) Periodic boundary

Figure 5.7: Results of the Fourier pseudospectral method simulating the 2D shallow water equations. Relevant parameters: $g = 9.81$, $\Delta t = 10^{-2}$, $N = 40$, periodic boundary conditions.

Chapter 6

RADIAL BASIS FUNCTION PSEUDOSPECTRAL METHODS

In chapters 1 through 5 we were using a very specific function space method; where complex exponentials were taken as basis functions. The general idea is straight forward, attempt to exhibit the function you are approximating by a linear combination of predetermined basis functions, then attempt to recover the respective linear combination weights. The Fourier spectral method, often just called spectral method is so popular due to its connection to the fast Fourier transform. Changing the basis to Chebyshev polynomials leads to a Chebyshev spectral method, which is also popular. See [11], [83], and [82] for more details on Fourier and Chebyshev spectral methods.

A spectral method that has gained attention recently is based on the so–called *radial basis functions* (RBFs). Traditionally, RBFs have been used in scattered data modeling as well as computer graphics, however, they have seen a wide range of applications including differential equations. RBFs offer a number of appealing properties, for instance they are a type of meshfree method, they naturally capable of producing multivariate approximations, and there is flexibility with the choice of a family of basis functions. RBFs can avoid costly mesh generation, scale to high dimensions, and has a diverse selection of basis functions with varying smoothness. For instance, common RBF choices are: compactly supported and finitely smooth, global and finitely smooth, and global, infinitely differentiable (comes with a free parameter).

6.1 Introduction to Radial basis functions

A brief introduction to radial basis functions will be provided in this section.

Definition 6.1.1. A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is said to be **radial** if there exists a function $\varphi : [0, \infty) \to \mathbb{R}$, such that

$$\Phi(\vec{x}) = \varphi(r),$$

where $r = \|\vec{x}\|$, and $\|\cdot\|$ is a norm on $\mathbb{R}^d$ (The norm is typically taken to be the Euclidean norm).

Definition 6.1.2. The function $\varphi(r)$ in definition 6.1.1 is called a **radial basis function**. That is, a radial basis function is a real–valued function whose value depends only on the distance from some point $\vec{y}$ called a *center*, so that $\varphi(\vec{x}, \vec{y}) = \varphi(\|\vec{x} - \vec{y}\|)$. In some cases radial basis functions contain a free parameter ϵ, called a *shape parameter*.

Radial basis functions were traditionally used for *scattered node interpolation problems*: given a set of data $\{(\vec{x}_j, f_j) : \vec{x}_j \in \mathbb{R}^d, f_j \in \mathbb{R} \text{ for } 1 \leq j \in \mathbb{N} \leq N\}$, we seek a smooth function g such that $g(x_j) = f_j$ for $1 \leq j \in \mathbb{N} \leq N$ (assume that f is some function, so $f_j = f(\vec{x}_j)$). Given a set of N centers $\{\vec{y}_j : \vec{y}_j \in \mathbb{R}^d \text{ for } 1 \leq j \in \mathbb{N} \leq N\}$, a radial basis function interpolant is a linear combination of said radial basis functions:

$$g(\vec{x}) = \sum_{j=1}^{N} \alpha_j \phi(\|\vec{x} - \vec{y}_j\|_2). \tag{6.1}$$

By enforcing the condition $g(\vec{y}_i) = f(\vec{y}_i)$ ($\vec{y}_i$ belong to the set of N centers), we can obtain the linear combination weights α_j. This is done by constructing the $N \times N$ linear system

$$\mathbf{A}\vec{\alpha} = \vec{f}, \quad \vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T, \quad \vec{f} = [f_1, f_2, \ldots, f_N]^T. \tag{6.2}$$

The matrix $\mathbf{A}$ is called the *RBF interpolation matrix* or the *system matrix*. It has entries of the form

$$\mathbf{A}_{ij} = \phi(\|\vec{y}_i - \vec{y}_j\|_2), \quad \text{for } i, j = 1, 2, \ldots N. \tag{6.3}$$

For special choices of radial basis function, it can be shown that the interpolation matrix $\mathbf{A}$ is always nonsingular. Table 6.1 has some of the most popular RBFs used in the literature. Inverse quadratic, inverse multiquadric, multiquadratic, and Gaussian RBFs have been shown to generate nonsingular interpolation matrices. The critical details are as follows:

Definition 6.1.3. A function ϕ is **completely monotone** on $[0, \infty)$ if

1. $\phi \in C[0, \infty)$,
2. $\phi \in C^\infty(0, \infty)$,
3. $(-1)^m \phi^{(m)}(r) \geq 0$ where $r > 0$ and $m = 0, 1, 2, \ldots$, and $\phi^{(m)}$ is the mth derivative of ϕ.

Theorem 6.1.1. Suppose that $\phi(r) = \varphi(\sqrt{r})$, where $\phi \in C[0, \infty)$ and $\phi(r) > 0$ for $r > 0$. In addition, suppose that $\phi'(r)$ is completely monotone and nonconstant on $(0, \infty)$. Then, for any set of N distinct centers $\{\vec{y}_j\}_{j=1}^N$, the $N \times N$ matrix $\mathbf{A}$ with entries $\mathbf{A}_{ij} = \varphi(\|\vec{y}_i - \vec{y}_j\|_2)$ is nonsingular.

See [63] for more detail and proofs. Multiquadratic RBFs satisfy the assumptions of theorem 6.1.1. Inverse quadratic, inverse multiquadric, and Gaussian RBFs are actually *strictly positive definitive* functions, and this leads to positive definite interpolation matrices.

Name of RBF	Abbreviation	Definition
Smooth, global		
Multiquadratic	MQ	$\sqrt{1 + (\epsilon r)^2}$
Inverse multiquadratic	IMQ	$(1 + (\epsilon r)^2)^{-1/2}$
Inverse quadratic	IQ	$\frac{1}{1+(\epsilon r)^2}$
Gaussian	GA	$e^{-(\epsilon r)^2}$
Piecewise smooth, global		
Cubic	CU	$\|r\|^3$
Quartic	QUA	$\|r\|^4$
Quintic	QUI	$\|r\|^5$
Thin plate spline type, order k	TPS	$\|r\|^{2k} \log \|r\|$
Piecewise smooth, compact		
Wendland type, order 2	W2	$(1 - \epsilon r)^4(4\epsilon r + 1)$
Order 4	W4	$(1 - \epsilon r)^6((35/3)(\epsilon r)^2 + 6\epsilon r + 1)$
Order 6	W6	$(1 - \epsilon r)^8(32(\epsilon r)^3 + 25(\epsilon r)^2 + 8\epsilon r + 1)$

Table 6.1: Popular RBF functions.

6.1.1 RBF interpolation example

To get an idea of how RBF interpolation works, we consider a simple example. Suppose that we want to approximate the Runge function $f(x) = (1+25x^2)^{-1}$ on the domain $[-1, 1]$ with a multiquadratic RBF (shape parameter is taken to be $\epsilon = 1.25$). To discretize the domain, we use 5 evenly spaced points, $[-1, -0.5, 0, 0.5, 1]$. We analyze three different sets of RBF centers. Let $\vec{w} = [-1, 0, 1]$, $\vec{u} = [-1, -0.5, 0, 0.5, 1]$, and $\vec{v} = [-1, -\sqrt{2}/2, 0, \sqrt{2}/2, 1]$.

The vector $\vec{w}$ is three evenly spaced points, $\vec{u}$ is 5 evenly spaced points, and $\vec{v}$ is 5 Chebyshev points on the interval $[-1, 1]$. For the 3 evenly spaced RBF centers given in $\vec{w}$, the interpolation matrix is

$$\mathbf{A} = \begin{bmatrix} \phi(\|\vec{w}_1 - \vec{w}_1\|_2) & \phi(\|\vec{w}_1 - \vec{w}_2\|_2) & \phi(\|\vec{w}_1 - \vec{w}_3\|_2) \\ \phi(\|\vec{w}_2 - \vec{w}_1\|_2) & \phi(\|\vec{w}_2 - \vec{w}_2\|_2) & \phi(\|\vec{w}_2 - \vec{w}_3\|_2) \\ \phi(\|\vec{w}_3 - \vec{w}_1\|_2) & \phi(\|\vec{w}_3 - \vec{w}_2\|_2) & \phi(\|\vec{w}_3 - \vec{w}_3\|_2) \end{bmatrix}.$$

The linear system $\mathbf{A}\vec{\alpha} = [f(\vec{w}_1), f(\vec{w}_2), f(\vec{w}_3)]^T$ can then be solved for $\vec{\alpha}$. This only approximates the function at the given centers $\vec{w}$, and not the 5 evenly spaced points $\vec{x} = [-1, -0.5, 0, 0.5, 1]^T$. To complete the interpolation we construct the 5×3 evaluation matrix $\mathbf{H}$

$$\mathbf{H} = \begin{bmatrix} \phi(\|\vec{x}_1 - \vec{w}_1\|_2) & \phi(\|\vec{x}_1 - \vec{w}_2\|_2) & \phi(\|\vec{x}_1 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_2 - \vec{w}_1\|_2) & \phi(\|\vec{x}_2 - \vec{w}_2\|_2) & \phi(\|\vec{x}_2 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_3 - \vec{w}_1\|_2) & \phi(\|\vec{x}_3 - \vec{w}_2\|_2) & \phi(\|\vec{x}_3 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_4 - \vec{w}_1\|_2) & \phi(\|\vec{x}_4 - \vec{w}_2\|_2) & \phi(\|\vec{x}_4 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_5 - \vec{w}_1\|_2) & \phi(\|\vec{x}_5 - \vec{w}_2\|_2) & \phi(\|\vec{x}_5 - \vec{w}_3\|_2) \end{bmatrix},$$

and multiply it by $\vec{\alpha}$. Thus, $[f(\vec{x}_1), f(\vec{x}_2), f(\vec{x}_3), f(\vec{x}_4), f(\vec{x}_5)]^T \approx \mathbf{H}\alpha$. The maximum error in this case is around 0.5274.

If we select the vector $\vec{u}$ (5 evenly spaced points) as our RBF centers, the evaluation matrix is equal to the interpolation matrix, and

$$\mathbf{A} = \begin{bmatrix} \phi(\|\vec{x}_1 - \vec{x}_1\|_2) & \phi(\|\vec{x}_1 - \vec{x}_2\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_5\|_2) \\ \phi(\|\vec{x}_2 - \vec{x}_1\|_2) & \phi(\|\vec{x}_2 - \vec{x}_2\|_2) & \phi(\|\vec{x}_2 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_5\|_2) \\ \phi(\|\vec{x}_3 - \vec{x}_1\|_2) & \phi(\|\vec{x}_3 - \vec{x}_2\|_2) & \phi(\|\vec{x}_3 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_5\|_2) \\ \phi(\|\vec{x}_4 - \vec{x}_1\|_2) & \phi(\|\vec{x}_4 - \vec{x}_2\|_2) & \phi(\|\vec{x}_4 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_5\|_2) \\ \phi(\|\vec{x}_5 - \vec{x}_1\|_2) & \phi(\|\vec{x}_5 - \vec{x}_2\|_2) & \phi(\|\vec{x}_5 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_3\|_2) & \phi(\|\vec{x}_1 - \vec{x}_5\|_2) \end{bmatrix}.$$

Then if we set $[f(\vec{x}_1), f(\vec{x}_2), f(\vec{x}_3), f(\vec{x}_4), f(\vec{x}_5)]^T = \mathbf{A}\alpha$, can we can solve this linear system for α. In this case the maximum error is $1.0797 \cdot 10^{-14}$.

If we select the vector $\vec{v}$ (5 Chebyshev points on $[-1, 1]$) as our RBF centers, the evaluation matrix is equal to the interpolation matrix, and

$$\mathbf{A} = \begin{bmatrix} \phi(\|\vec{x}_1 - \vec{v}_1\|_2) & \phi(\|\vec{x}_1 - \vec{v}_2\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_5\|_2) \\ \phi(\|\vec{x}_2 - \vec{v}_1\|_2) & \phi(\|\vec{x}_2 - \vec{v}_2\|_2) & \phi(\|\vec{x}_2 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_5\|_2) \\ \phi(\|\vec{x}_3 - \vec{v}_1\|_2) & \phi(\|\vec{x}_3 - \vec{v}_2\|_2) & \phi(\|\vec{x}_3 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_5\|_2) \\ \phi(\|\vec{x}_4 - \vec{v}_1\|_2) & \phi(\|\vec{x}_4 - \vec{v}_2\|_2) & \phi(\|\vec{x}_4 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_5\|_2) \\ \phi(\|\vec{x}_5 - \vec{v}_1\|_2) & \phi(\|\vec{x}_5 - \vec{v}_2\|_2) & \phi(\|\vec{x}_5 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_3\|_2) & \phi(\|\vec{x}_1 - \vec{v}_5\|_2) \end{bmatrix}.$$

Then if we set $[f(\vec{x}_1), f(\vec{x}_2), f(\vec{x}_3), f(\vec{x}_4), f(\vec{x}_5)]^T = \mathbf{A}\alpha$, can we can solve this linear system for α. In this case the maximum error is $1.9151 \cdot 10^{-15}$.

In figure 6.1 a plot of the Runge function, and the RBF interpolation results can be found. In the numerical experiments that follow in this chapter, best results were found by taking

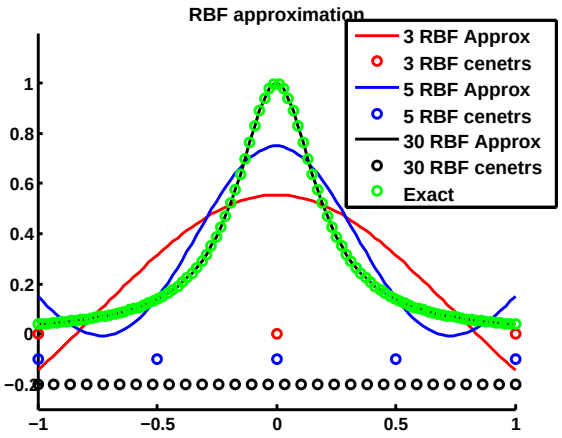


Figure 6.1: RBF interpolation using different centers.

the RBF centers to be the same as the computational domain discretization. Sample code for this implementation can be found in Code Listing 6.1.1.

Code Listing 6.1.1

Main RBF Interpolation

```
% Approximates the Runge function 1/(1+25x^2) on [-1,1] using 3 different
% RBF centers.  The RBF chosen is a multiquadratic with a shape parameter
% of 1.25.  Needs the function file mq.m.
clear all;  close all; clc;
f  = @(x) 1./(1+25*x.^2);%Runge function
a = -1;  b = 1;
MM = 100;
xp = linspace(a,b,MM)'; %fine grid to smooth out RBF approximations
INDEX = [3,5,30]; %number of RBF centers
shape = 1.25;     %RBF blanket shape parameter
%Variables for plots
IdX = [0,-0.1,-.2];  ccc    = {'r','b','k'};  cc2    = {'ro','bo','ko'};
hold on,
for ii = 1 : length(INDEX)
    nmm = INDEX(ii);
    uu = linspace(a,b,nmm)';  cx = shape*ones(nmm,1);
    n = INDEX(ii);
    [Ap] = deal(zeros(MM,n));
    for j = 1 : nmm
        [Ap(:,j)] = mq(xp,uu(j),cx(j));
    end
    alpha = Ap \ f(xp);
```

```
    y4 = Ap*alpha;
    plot(xp,y4,ccc{ii}),
    plot(uu,ones(size(uu))*IdX(ii),cc2{ii})
end
plot(xp,f(xp),'go')
hold off
legend('3 RBF Approx','3 RBF cenetrs','5 RBF Approx','5 RBF cenetrs',...
    '30 RBF Approx','30 RBF cenetrs','Exact')
title('RBF approximation'),  ylim([-0.3 1.2])
```

6.1.2 RBF differentation

In the previous section we attempted to approximate a function $g(\vec{x})$ as a linear combination of RBFs:

$$g(\vec{x}) = \sum_{j=1}^{N} \vec{\alpha}_j \phi(\|\vec{x} - \vec{y}_j\|_2),$$

for N distinct centers that are stored in the vector $\vec{y}$. To approximate a derivative, we simply differentiate both sides to obtain

$$\frac{\partial}{\partial \vec{x}_i}(g(\vec{x})) = \sum_{j=1}^{N} \alpha_j \frac{\partial}{\partial \vec{x}_i}\phi(\|\vec{x} - \vec{y}_j\|_2) = \left[\frac{\partial}{\partial \vec{x}_i}\mathbf{A}\right]\vec{\alpha}. \tag{6.4}$$

We define the first order evaluation matrix $\mathbf{D_1}$ such that

$$(\mathbf{D_1})_{ij} = \frac{\partial}{\partial \vec{x}_i}\phi(\|\vec{x} - \vec{y}_j\|_2), \quad i, j = 1, 2, \ldots N.$$

If we select a RBF that gives rise to a nonsingular interpolation matrix, equation (6.2) implies that $\vec{\alpha} = \mathbf{A}^{-1}\vec{f}$. Then, from equation (6.4), we have

$$\frac{\partial}{\partial \vec{x}_i}(g(\vec{x})) = \left[\frac{\partial}{\partial \vec{x}_i}\mathbf{A}\right]\vec{\alpha}, \quad \text{for } i = 1, 2, \ldots, N \implies [\mathbf{D_1}]\mathbf{A}^{-1}\vec{f},$$

and we define the *differentation matrix* $\mathbf{D_x}$ to be

$$\mathbf{D_x} = \mathbf{D_1}\mathbf{A^{-1}}. \tag{6.5}$$

Following this process one can easily construct differentiation matrices of arbitrary order:

$$\frac{\partial^{(m)}}{\partial \vec{x}_i^{(m)}}(g(\vec{x})) = \left[\frac{\partial^{(m)}}{\partial \vec{x}_i^{(m)}}\mathbf{A}\right]\vec{\alpha} = \left[\frac{\partial^{(m)}}{\partial \vec{x}_i^{(m)}}\mathbf{A}\right]\mathbf{A}^{-1}\vec{f}. \tag{6.6}$$

In practice the matrix $\mathbf{A}^{-1}$ is never actually formed, a matrix system solver is used instead. In MATLAB this can be done by the forward slash operator, or for singular or near singular

problems the pseudoinverse. The differentiation matrices derived above are called *global*, since they use information from every center.

To illustrate the RBF differentiation matrices, we consider an example. Let $f(x) = (1 + 25x^2)^{-1}$, on the interval $[-1, 1]$. We partition $[-1, 1]$ into 3 equally spaced points. In addition, we select an inverse quadratic RBF with a shape parameter of 5 (which should match the Runge function to machine precision). Let $\vec{x} = [-1, 0, 1]^T$. The interpolation matrix is given by

$$\mathbf{A} = \begin{bmatrix} \phi(\|\vec{x}_1 - \vec{x}_1\|_2) & \phi(\|\vec{x}_1 - \vec{x}_2\|_2) & \phi(\|\vec{x}_1 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_2 - \vec{x}_1\|_2) & \phi(\|\vec{x}_2 - \vec{x}_2\|_2) & \phi(\|\vec{x}_2 - \vec{w}_3\|_2) \\ \phi(\|\vec{x}_3 - \vec{x}_1\|_2) & \phi(\|\vec{x}_3 - \vec{x}_2\|_2) & \phi(\|\vec{x}_3 - \vec{x}_3\|_2) \end{bmatrix}$$

The first and second order evaluation matrices are computed as

$$\mathbf{D_1} = \begin{bmatrix} \phi_x(\|\vec{x}_1 - \vec{x}_1\|_2) & \phi_x(\|\vec{x}_1 - \vec{x}_2\|_2) & \phi_x(\|\vec{x}_1 - \vec{w}_3\|_2) \\ \phi_x(\|\vec{x}_2 - \vec{x}_1\|_2) & \phi_x(\|\vec{x}_2 - \vec{x}_2\|_2) & \phi_x(\|\vec{x}_2 - \vec{w}_3\|_2) \\ \phi_x(\|\vec{x}_3 - \vec{x}_1\|_2) & \phi_x(\|\vec{x}_3 - \vec{x}_2\|_2) & \phi_x(\|\vec{x}_3 - \vec{x}_3\|_2) \end{bmatrix}$$

$$\mathbf{D_2} = \begin{bmatrix} \phi_{xx}(\|\vec{x}_1 - \vec{x}_1\|_2) & \phi_{xx}(\|\vec{x}_1 - \vec{x}_2\|_2) & \phi_{xx}(\|\vec{x}_1 - \vec{w}_3\|_2) \\ \phi_{xx}(\|\vec{x}_2 - \vec{x}_1\|_2) & \phi_{xx}(\|\vec{x}_2 - \vec{x}_2\|_2) & \phi_{xx}(\|\vec{x}_2 - \vec{w}_3\|_2) \\ \phi_{xx}(\|\vec{x}_3 - \vec{x}_1\|_2) & \phi_{xx}(\|\vec{x}_3 - \vec{x}_2\|_2) & \phi_{xx}(\|\vec{x}_3 - \vec{x}_3\|_2) \end{bmatrix}$$

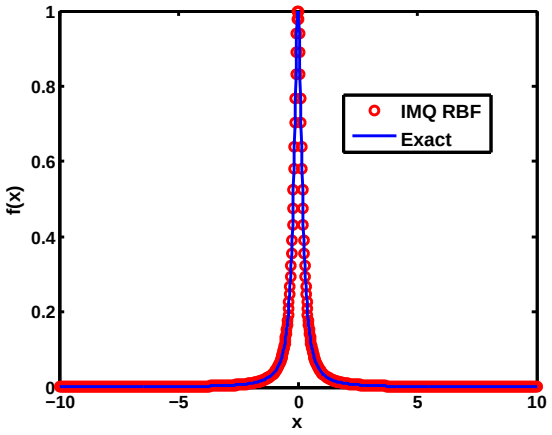
To construct the first and second order differentiation matrices, one needs to solve the linear systems $\mathbf{D_x A} = \mathbf{D_1}$ and $\mathbf{D_{xx} A} = \mathbf{D_2}$. Figure 6.2 illustrates RBF differentiation. Example code for the example problem discussed can be found in Code Listing 6.1.2. In figure 6.2d one can see that the choice of RBF can change the approximations accuracy. Even when the function being approximated is smooth, it might take very large N to see the desired accuracy in the derivatives (see figure 6.2d). For the Runge function an IQ–RBF method might provide a better approximation (see table 6.1).

Code Listing 6.1.2

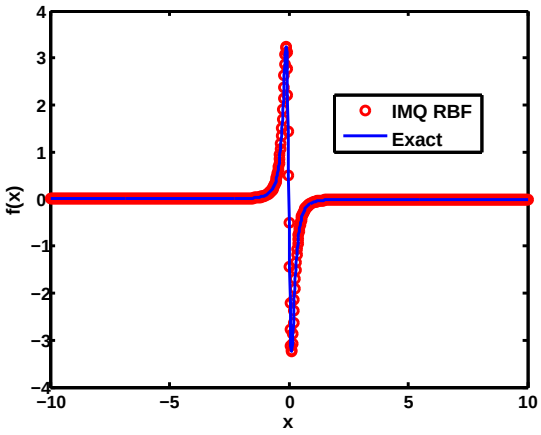
Main RBF Derivative file

```
% Approximates the Runge function 1/(1+25x^2) on [-1,1] divided into three
% equally spaced points.  The RBF is a inverse multiquadratic with a shape
% parameter equal to 5.  Needs the file imq.m.
clear all;  close all;  clc;
%Exact
f  = @(x) 1./(1+25*x.^2);
fp = @(x) (-50*x).*(f(x)).^2;
fpp = @(x)  50*(75*x.^2-1) .* (f(x)).^3;
a = -1;  b = -a;
n = 31;
x = linspace(a,b,n)';  shape = 5; %shape parameter
cx = shape*ones(n,1);  [A,D1,D2] = deal(zeros(n));
for j = 1 : n
    [A(:,j),D1(:,j),D2(:,j)] = imq(x,x(j),cx(j)); %construct RBF matrices
```

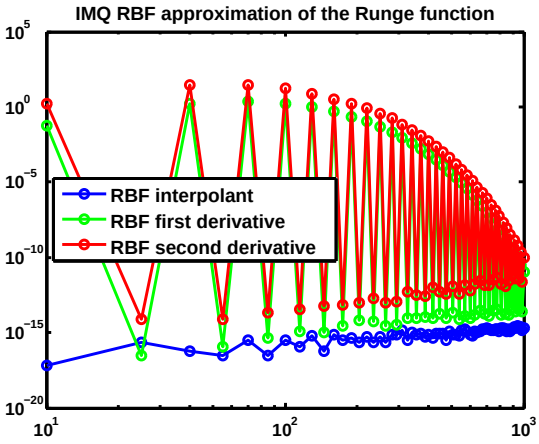
```
end
D1x = D1 / ( A );  D2x = D2 / ( A ); %solve for RBF derivative matrices

lam = A \ f(x);
Err1 = norm(A*lam - f(x),inf)         %Interpolation error
Err2 = norm(D1x*f(x) - fp(x),inf)     %1st derivative error
Err3 = norm(D2x*f(x) - fpp(x),inf)    %2nd derivative error
plot(xx,f(xx)),  hold on,  plot(x,A*lam,'ro'),  hold off
```



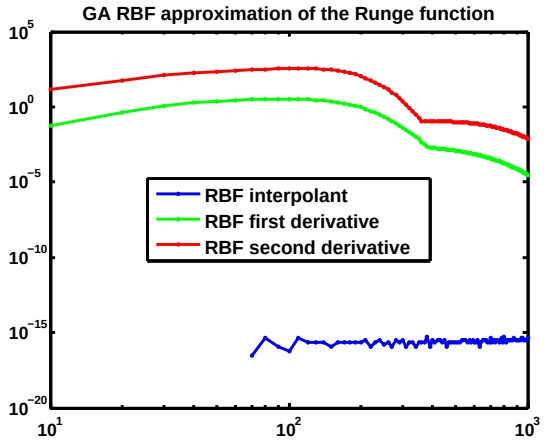
(a) Approximate $f(x)$



(b) Approximate $f'(x)$



(c) Approximation errors



(d) Effect of different RBF ($\epsilon = 15$)

Figure 6.2: RBF differentiation example. Notice in figure 6.2c the error oscillates, this is due to N alternating between even and odd natural numbers. The RBF center layout changes depending on if N is even or odd.

Remark 6.1.1. A difficulty with RBFs, is the very selection of a RBF, as well as selecting a shape parameter (typically denoted by ϵ). In the current literature trial and error is the most frequently used technique. Some general ideas on shape parameter selection can be found in [70] and [27]. To make matters more interesting, there is a connection between the shape parameter ϵ and the condition number of the interpolation and differentiation matrices. As $\epsilon \to 0$, $\kappa(\mathbf{A}) \approx \mathcal{O}(\epsilon^{-M})$ for large $M > 0$. This creates an issue of balancing a trade off. Either you have high accuracy but poor conditioning (ϵ small and $\kappa(\mathbf{A})$ large), or you have better conditioning but lower accuracy (ϵ large and $\kappa(\mathbf{A})$ small). Figure 6.3 illustrates this trade off. Since the differentiation matrices are constructed using the interpolation matrix $\mathbf{A}$, they inherit the same liner dependency on the shape parameter (see figure 6.3).

The choice of shape parameter can 'make or break' the approximation, figure 6.3 shows this. In many cases a small shape parameter $\epsilon \ll 1$ will not provide an optimal approximation. The advection equation and Serre-Green Nagdhi equation test problems (see Code Listings 6.2.1 and 6.9.1) show that shape parameters $\epsilon \ll 1$ are not always necessary.

In the RBF literature a *blanket* shape parameter is most commonly implemented. Shape parameters have a fine degree of control – each RBF center is allocated its own shape parameter constant. A blanket shape parameter assigns the same shape parameter to each RBF center. More advanced shape parameter selections have been suggested. Perhaps the simplest are due to Hardy [44] and Franke [36]. Hardy proposed a shape parameter

$$\vec{c} = \frac{0.815}{N} \sum_{j=1}^{N} d_j,$$

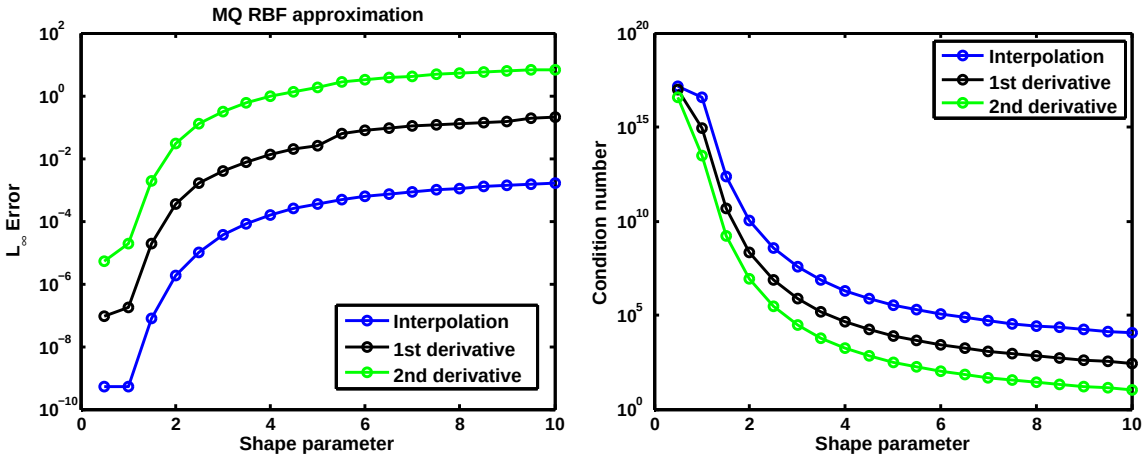
where d_j is the distance from the ith center to its nearest neighbor. A similar recommendation from Franke is

$$\vec{c} = \frac{1.25D}{\sqrt{N}},$$

where D is the diameter of the smallest circle that contains all of the centers. The recommendations from Franke and Hardy were from the perspective of scattered node interpolation.

Another approach from statistics has been applied to the selection RBF centers. In [68] the technique *leave–one–out cross validation* (LOOCV) has been implemented with success. See [28] for an implementation of LOOCV from the perspective of applying the technique to RBF collocation/pseudospectral methods.

Variable shape parameter techniques have also been studied, for instance, see [56], [57], and [71]. Adaptive shape parameter selection falls into the variable class, see [23] for an example. In [23] the adaptive shape parameter selection is paired with an adaptive RBF center selection, which seems to hold promise for time dependent PDEs with nonsmooth solutions. A survey of future directions and current/past ideas related to shape parameter selection are discussed in [70] and [27].



(a) Approximation vs shape parameter

(b) Shape parameter vs condition number

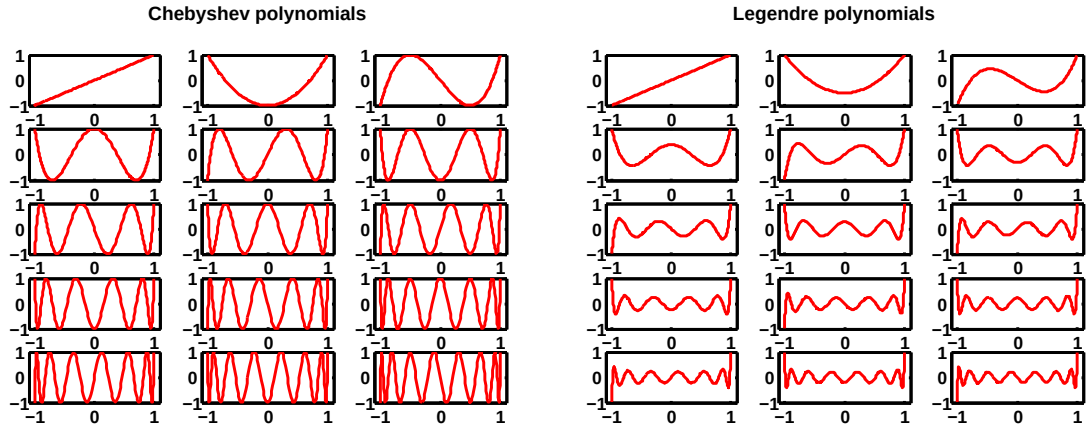
Figure 6.3: Effect of shape parameter for RBF approximation. Function used is $f(x) = \text{sech}^2(x)$ on $[-1, 1]$ with $N = 50$. The interval $[-1, 1]$ divided into 25 evenly spaced points provides the RBF centers for this example. In figure 6.3b the evaluation matrices of first and second order are used (black and green lines respectively).

The amount of flexibility in RBFs can be seen as an advantage or a disadvantage. There is no agreed upon consensus to select a RBF, RBF centers, or RBF shape parameters. However this generality is not always detrimental to approximation. For classic approximations like the various pseudospectral methods (Fourier, Legendre, Chebyshev, etc.), there is very little flexibility in the choice of basis and grid points. From table 6.1, one can see that RBFs can have varying degrees of smoothness, and in some cases the RBF does not come with a shape parameter.

RBFs are quite general. Discretized RBFs are independent of coordinate system, dimension and even domain geometry. For instance, if node layouts are in spherical coordinates, the form of the interpolant says the same (equation (6.1)). This is because distances are formed straight through the sphere, and not via arcs (see [29]).

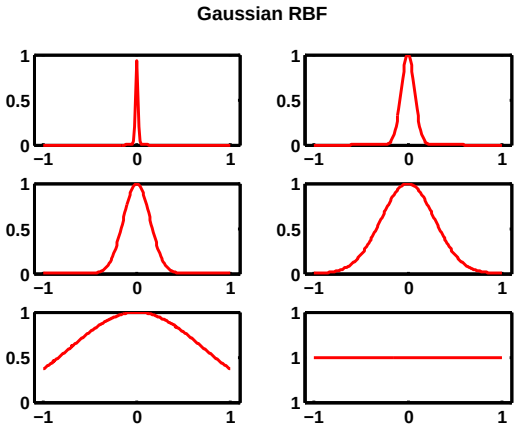
RBFs can also be seen as generalizations of the classical pseudospectral approximations in the limit as $\epsilon \to 0$. In the case of scattered node interpolation, one–dimensional Gaussian RBFs have been shown to converge to the Lagrange interpolation polynomial interpolants in the limit as $\epsilon \to 0$ (see [22]). Other classical pseudospectral approximations follow as well. For instance, RBF approximations converging to Fourier pseudospectral approximations in the limit as $\epsilon \to 0$ has been studied in [32]. The multivariate case is more difficult to analyze, but progress has been make (see [35]). The meaning of these results is that RBF approximations (global, infinitely differentiable) generalize all the classical pseudospectral methods (Fourier, Chebyshev, Legendre, etc.); as along as the RBF centers are taken to be the associated pseudospectral grid.

Often classical pseudospectral approximations have an orthogonality condition, so higher degree pseudospectral interpolants become increasingly oscillatory. On the other hand, global infinitely differentiable RBFs have controllable "flatness" – as $\epsilon \to 0$ the RBF approximants become increasingly "flat". Figure 6.4 has visualizations of this behavior. The classical pseudospectral approximations do not have the fine degree control that RBFs have; each Chebyshev polynomial is fixed, only the linear combination weights can change. RBFs can change bases, RBF centers, and each RBF center[1] has its own shape parameter (which controls the "flatness" of the RBF approximant at a given RBF center).

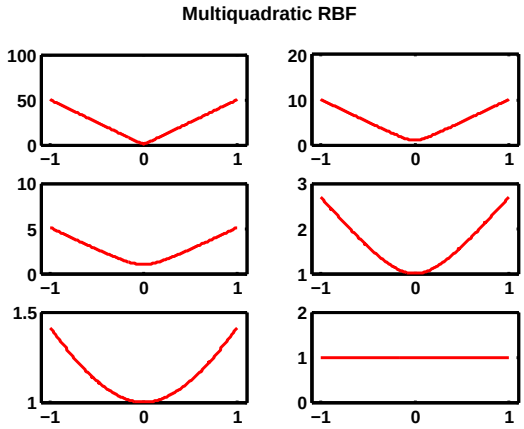


(a) Chebyshev polynomials

(b) Legendre polynomials



(c) Gaussian RBF



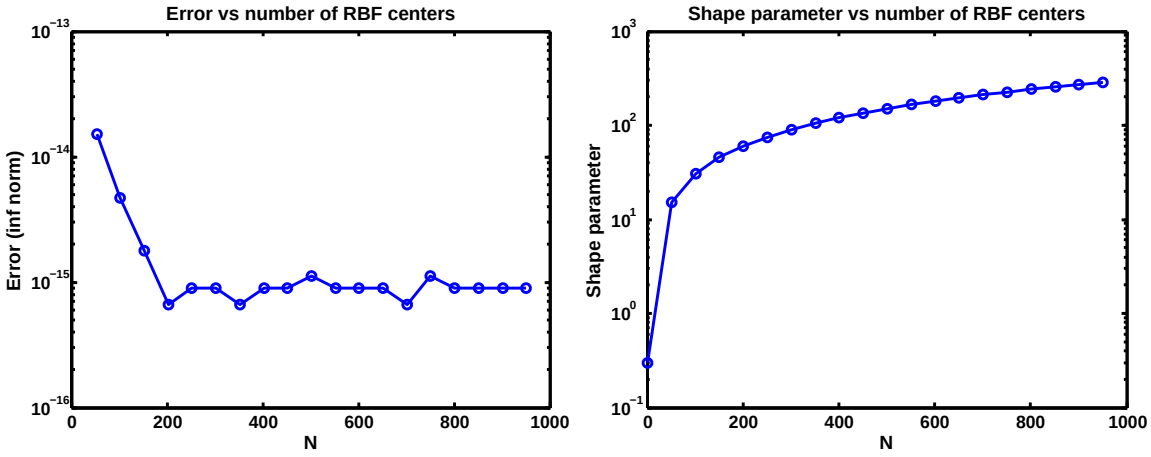
(d) Multiquadratic RBF

Figure 6.4: Shape parameters used are from the set $\{50, 10, 5, 2.5, 1, 10^{-8}\}$.

[1]Not all RBFs have a shape parameter.

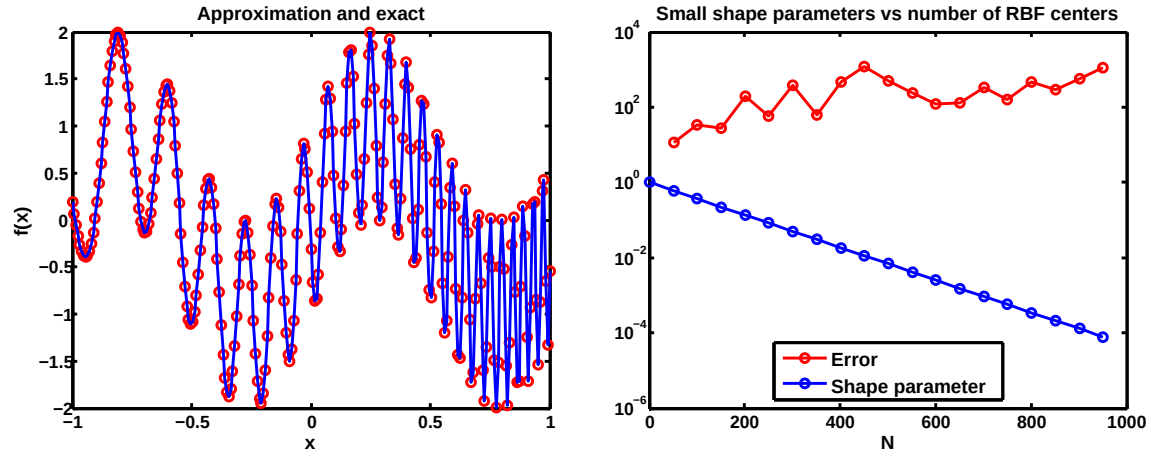
In figure 6.5 an analysis of the effect of large and small blanket shape parameters is given. The test function is $f(x) = \sin(6x) + \sin(60e^x)$ on the interval $[-1, 1]$ (see chapter 3 of [83]).

Solving the linear system 6.2 directly in practice (often called *RBF–Direct*) is an unstable algorithm (see [33]). This is due to the trade off (uncertainty) principle; as $\epsilon \to 0$, the system matrix becomes more ill conditioned. Two stable algorithms currently exist for small shape parameters, Contour–Padé and RBF–QR methods (see [27] and [33]).



(a) Error vs large shape parameters

(b) Large shape parameters



(c) Approximation (large shape parameters)

(d) Error vs small shape parameters

Figure 6.5: This figure shows that a small shape parameter is not always beneficial.

6.2 Radial basis function collocation and pseudospectral methods

A natural transition after examining RBF interpolation and differentiation, is to apply RBFs to differential equations. Hyperbolic PDEs have not seen as much attention as elliptic and parabolic PDEs when considering RBFs. Nonsmooth data and stability are the major deterrents. Nevertheless, there are some successful applications of RBFs to hyperbolic problems. When studying numerical methods for hyperbolic problems, it is standard to examine the advection equation $u_t + c(x)u_x = 0$. To illustrate a RBF collocation/pseudospectral method, we will use the advection equation as an example.

As with the Fourier spectral methods, one discretizes the spatial domain, and then applies the method of lines to arrive at a fully discrete system. For the advection equation, there is only the first spatial differentiation operator that needs to be resolved. Suppose that $c(x) \equiv 1$, and we are interested in the solution to the advection equation at $t = 1$ on a domain of $[-1, 1]$ and $u(0, t) = 0$ is a boundary condition. Partition the domain into N grid points $x_1, x_2, \ldots, x_N$. Then, after selecting a RBF and N distinct centers $y_1, y_2, \ldots, y_N$, one can construct the associated RBF interpolation matrix ($\mathbf{A}$, see equation (6.3)) and differentiation matrix ($\mathbf{D_x}$, see equation (6.5)). A semidiscrete system in time is formed as

$$(\vec{u})_t = -\mathbf{D_x}\vec{u}, \quad \text{where } \vec{u} = [u(x_1, t), u(x_2, t), \ldots, u(x_N, t)]^T.$$

From here we can employ the method of lines to fully discretize the system. For stability reasons, we will use a Crank–Nicolson method:

$$\vec{u}^{n+1} - \vec{u}^n = 0.5(\Delta t)[-\mathbf{D_x}\vec{u}^{n+1} - \mathbf{D_x}\vec{u}^n],$$

where $\vec{u}^n = [u(x_1, t_n), u(x_2, t_n), \ldots, u(x_N, t_n)]^T$ (with $t_n = n\Delta t$), and upon isolating $\vec{u}^{n+1}$ it follows that

$$(\mathbf{I} + (0.5\Delta t)\mathbf{D_x})\vec{u}^{n+1} = \vec{u}^n - 0.5(\Delta t)[\mathbf{D_x}\vec{u}^n], \tag{6.7}$$

where $\mathbf{I}$ is the $(N-1) \times (N-1)$ identity matrix. At each time step, a linear system needs to be solved in equation (6.7). To impose the boundary condition $u(0, t) = 0$, we can simply remove the first row of the matrix $\mathbf{D_x}$ (there is no need to do the multiplication if it results in zero for all t). The discussion on boundary conditions in [82] (chapter 13) applies to the case of RBFs, since they also deal with interpolation and differentiation matrices. An example implementation of the RBF advection equation is given in Code Listing 6.2.1[2], with an initial condition of $u(x, 0) = e^{(-50(x+0.5-t)^2)}$. A plot from Code Listing 6.2.1 is displayed in figure 6.6.

CODE LISTING 6.2.1

Main RBF Advection file

```
% Approximates the advection equation u_t=-u_x on [-1,1].  The RBF is a
% Gaussian RBF with a shape parameter equal to 15.  Needs the file gau.m.
```

[2]Instead of a manual time stepping, one can use an ODE solver from MATLAB using the "Mass matrix" option.

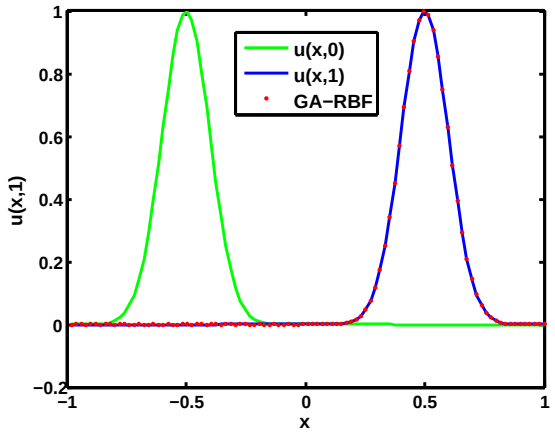


Figure 6.6: Gaussian RBF approximation for the advection equation, $N = 100$, $\Delta t = 1/300$, $u(x,0) = e^{(-50(x+0.5-t)^2)}$, boundary condition $u(x,0) = 0$.

```
clear all;  close all;  clc;
%Exact
f  = @(x,t) exp(-50*(x+0.5-t).^2);
a = -1;  b = -a;  %plotting domain

n = 100;  x = linspace(a,b,n)';
shape = 15;             %shape parameter
cx = shape*ones(n,1);
[A,D1] = deal(zeros(n));
for j = 1 : n
    [A(:,j),D1(:,j)] = gau(x,x(j),cx(j)); %construct RBF matrices
end
D1x = D1 / ( A );  %solve for RBF derivative matrices

Tfin = 1;  dt = 1/300;  nmax = floor(Tfin/dt);

D1x = D1x(2:end,2:end); %this handles the BC u(x,0)=0
x = x(2:end);           %BC

u = f(x,0);  I = eye(n-1);
for jj = 1 : nmax
    u = (I + 0.5*dt*D1x) \ (u - 0.5*dt*(D1x*u)); %Crank-Nicolson
end
norm(u-f(x,Tfin),inf),  plot(x,f(x,0),'g',x,f(x,Tfin),'b',x,u,'r.')
```

The method of lines is our main discretization tool in the temporal domain. Thus, the main stability verification will be to analyze the location of the eigenvalues of the linearized differentiation matrices. Even then, one might have to study pseudospectra, since these eigenvalues can sometimes be very sensitive to perturbations (RBF interpolation and differentiation matrices are highly nonnormal and ill conditioned). Hyperviscosity has been used in the RBF collocation/psuedospectral methods for hyperbolic problems. Currently, hyperviscosity is the most widely used technique to guarantee stability. However, including hyperviscosity is mostly a trial and error endeavour.

For a more detailed discussion of RBF collocation/pseudospectral methods (especially for elliptic problems) see [27]. The reference [27] has a similar format to Trefethen's *Spectral Methods in* MATLAB ([82]). A general algorithm for applying RBFs to time dependent PDEs is given below. Many authors choose to explicitly update the RBF linear combination weights (α_j's in equation 6.1). For examples of this equivalent approach in a time dependent PDE setting (long wave small amplitude phenomena), see [87], [88], and [17].

RBF methods are sometimes called *meshfree*, since RBF approximations are independent of a mesh. This is in contrast to the main approximation techniques for PDEs: finite difference (FD), finite volume (FV), and finite element (FE) methods. FD, FV, and FE, are defined on meshes of data points, and mesh generation can be very costly in terms of computation time.

Algorithm for RBF spectral method

- Step 1: Select a RBF, RBF centers, collocation points, final simulation time, and a time step.
- Step 2: Construct the required RBF interpolation and differentation matrices outside of the main time stepping loop.
- Step 3: Discretize the spatial domain of the PDE(s) using the RBF interpolation and differentation matrices.
- Step 4: Fully discretize PDE(s) using the method of lines.
- Step 5: Use an ode solver or time stepping to advance the approximation forward in time.

6.3 Local radial basis function collocation methods

The RBF spectral method presented in section 6.2 is sometimes called a *global* radial basis function method. This distinction is important since the global approach uses 'global' data to approximate function values or derivatives. That is, to approximate function values or derivatives at a point, data from every center in the computational domain is used. Since differentiation is a local property, it might seem more natural to use only a subset of the available centers. Finite differences follow this rule. For instance a forward difference approximation with second order error given by $2(\Delta x)F'(x) \approx -F(x+2\Delta x)+4F(x+\Delta x)-3F(x)$, uses data from three grid points x, $x+\Delta x$ and $x+2\Delta x$.

This idea of local differentiation has been applied to RBFs – and it is very popular in the RBF literature, especially when concerning time dependent PDEs. Local RBFs, also known as RBF–FD (radial basis function finite differences) have produced a lot of interest due to their interpolation and differentiation matrix structure. The interpolation and differentiation matrices generated by local RBFs have a controllable degree of sparsity. This sparsity can allow for much larger problems and make use of parallelism. In some cases local RBFs have been used as a form of *model order reduction*. It has been found in some situations that the local RBF can match or provide accuracy similar to that of the global RBF method, for smaller grid sizes (see [16]). Using smaller grid sizes gives rise to a smaller system of ODEs, but the overall accuracy remains controlled. For example applications of local RBFs to problems in the geosciences see [7] and [30].

The main drawback of local RBFs is that spectral accuracy is no longer obtained. In fact, the accuracy for local RBFs is dictated by the stencil used (the situation is similar for finite differences). The literature for RBFs is currently leaning towards local RBFs, since global RBFs produce dense, highly nonnormal, ill conditioned matrices. This drastically limits the scalability of global RBFs. In this section we will examine the simplest of local RBFs[3], however, more advanced local RBFs can be found in [92].

A RBF–FD stencil of size m requires the $m-1$ nearest neighbors (see figure 6.7). The local RBF interpolant takes the form

$$I_m g(\vec{x}) = \sum_{k \in \mathcal{I}_i} \vec{\alpha}_k \phi(\|\vec{x} - \vec{y}_k\|_2),$$

where $\vec{y}$ contains the N RBF centers, $\mathcal{I}_i$ is a set associated with RBF center i and whose elements are RBF center i's $m-1$ nearest neighbors. The vector $\vec{\alpha}$ is the unknown weights for the linear combination of RBFs. These weights can be calculated by enforcing

$$I_m g(\vec{x}_k) = g(\vec{x}_k),$$

for each $k \in \mathcal{I}_i$. This results in N linear systems of size $m \times m$, $\mathbf{B}\vec{\alpha} = \vec{g}$. The vector $\vec{g}$ is given by $\vec{g} = [g(\vec{x}_1), g(\vec{x}_2), \dots g(\vec{x}_N)]^T$. The entries of $\mathbf{B}$ have a familiar form

$$\mathbf{B}_{jk} = \phi(\|\vec{x}_j - \vec{y}_k\|_2), \quad j,k \in \mathcal{I}_i.$$

[3]The local RBF derivation given in this section is adapted from [16]

By selecting global infinitely differentiable RBFs (GA, MQ, IMQ, and IQ, etc.), the matrix $\mathbf{B}$ is guaranteed to be nonsingular. This implies that the coefficients on each stencil are uniquely defined. Local RBF derivatives are formed by evaluating

$$\mathcal{L}g(\vec{x}) = \sum_{j=1}^{N} \vec{\alpha}_j \mathcal{L}\phi(\|\vec{x} - \vec{y}_j\|_2),$$

at a RBF center where the stencil is based, and $\mathcal{L}$ is a linear operator. This equation can be simplified to

$$\mathcal{L}g(\vec{x}_i) = \vec{h}^T \vec{\alpha},$$

where $\vec{h}$ (a $m \times 1$ vector and $\vec{\alpha}$ contains the contains the RBF linear combination weights for the centers in $\mathcal{I}_i$) has components of the form

$$(\vec{h})_i = \phi(\|\vec{x} - \vec{y}_i\|_2), \quad i = 1, 2, \ldots, m.$$

It then follows that

$$\mathcal{L}g(\vec{x}_i) = (\vec{h}^T \mathbf{B}^{-1}) g\big|_{\mathcal{I}_i} = (\vec{w}_i) g\big|_{\mathcal{I}_i},$$

and $\vec{w}_i = \vec{h}^T \mathbf{B}^{-1}$ are the stencil weights at RBF center i ($\vec{w}_i$ multiplied by the function values provides the derivative approximation). The ith row of the mth order local RBF differentiation matrix is then given by $(\mathbf{W_m})_i = [\vec{w}_i]$.

Code Listing 6.3.1 has an example of an implementation of this local RBF for odd stencil sizes. Figure 6.3.1 has plots showing sparsity patterns for various stencil sizes, and well as the relationship between accuracy and stencil size. In figure 6.8d a local IMQ RBF was used to approximate the first derivative of the Runge function $1/(1 + 25x^2)$ on the interval $[-1, 1]$ (100 equally spaced points, and a blanket shape parameter of 5). Figure 6.8d also demonstrates that smaller stencil sizes result in lower accuracy. The sparsity patterns in figures 6.8a, 6.8b , and 6.8c show another flexibility of local RBF approximants. With local RBFs, one has the option of selecting bases, RBF centers, shape parameters, and stencil sizes. Section 6.5 has an example of a local RBF collocation/pseudospectral method applied to the KdV equation.

For large problems the introduction of sparsity is welcomed, since the global RBF approximants are dense, nonnormal and highly ill conditioned. Moreover, local RBFs are amenable to parallel computing. For some references on local RBFs see [7], [16], [92], and [30]. Unfortunately, none of these references have any simple explicit working code examples.

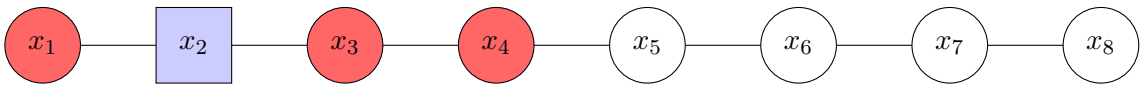
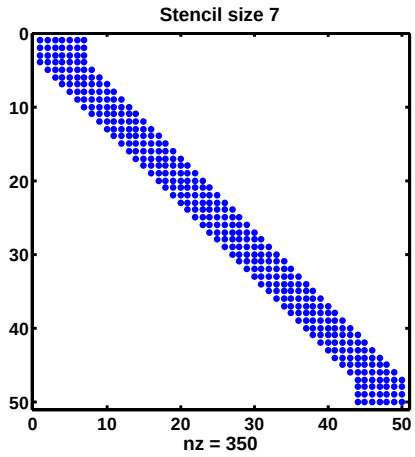
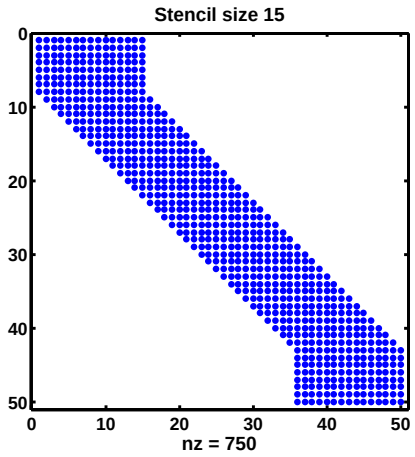


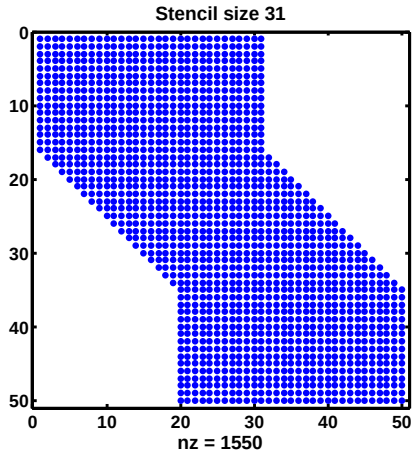
Figure 6.7: Stencil example: a domain with 8 grid points and a stencil size of $m = 4$. At the node x_2 (blue square), the $m - 1 = 3$ nearest neighbors (red circles) are used in calculations.



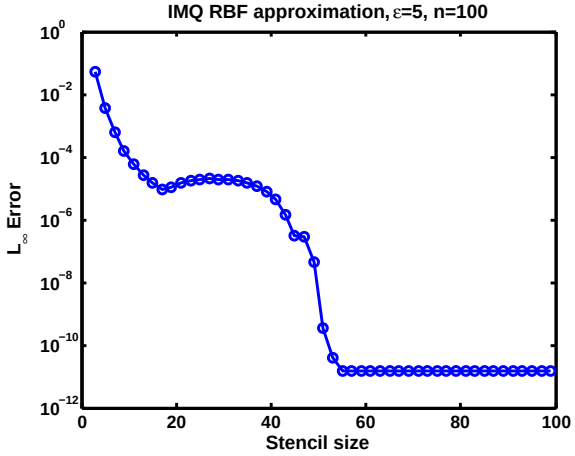
(a) Stencil size 7 (50×50 matrix)



(b) Stencil size 15 (50×50 matrix)



(c) Stencil size 31 (50×50 matrix)



(d) Effect of stencil size

Figure 6.8: Sparsity patterns and the relationship between accuracy and stencil size. Local IMQ RBF was used to approximate the first derivative of the Runge function $1/(1+25x^2)$ on the interval $[-1,1]$ (100 equally spaced points, and a blanket shape parameter of 5).

Code Listing 6.3.1

Main Local RBF Example file

```
% Approximates the first derivative of the Runge function 1/(1+25x^2) on
% the domain [-1,1] via a local Multiquadratic RBF with a shape parameter
% of 2.  The stencil size can be any odd number less than or equal to n.
clear all;  close all;  clc;
f  = @(x) 1./(1+25*x.^2);     %Exact Runge function
fp = @(x) (-50*x).*(f(x)).^2; %Exact Runge function derivative
n = 50;  x = linspace(-1,1,n)';  Hh = zeros(n,sten);
sten = 7;  %Stencil size must be odd, and less than or equal to n.
cx1 = (2)*ones(sten,1);   x1 = x(1:sten);  [A,D1] = deal(zeros(sten));
for j=1:sten
    [A(:,j),D1(:,j)] = mq(x1,x1(j),cx1(j)); %RBF interpolant pattern
end                                         %repeats for uniform grids
for ii = 1 : n              %construct stencil weights at center ii
    if ( ii <= 0.5*(sten-3)+1 )
        D1para = ii;  h = D1(D1para,:);  Hh(ii,:) = h;
    elseif ( ii >=  n - (0.5*(sten-3)) )
        D1para = ii - n + sten;  h = D1(D1para,:);  Hh(ii,:) = h;
    else
        D1para = (sten+1)/2;  h = D1(D1para,:);  Hh(ii,:) = h;
    end
end
WW = Hh / (A);   %construct stencil weights at every center
W = deal(zeros(n));  index1 = 1;  index2 = sten;
for ii = 1 : n               %construct sparse differentiation matrix
    if ( ii <= 0.5*(sten-3)+1 )
        aa      = 1;      bb = sten;
        W(ii,aa:bb) = WW(ii,:);
    elseif ( ii >=  n - (0.5*(sten-3)) )
        aa      = n - sten+1;      bb = n;
        W(ii,aa:bb) = WW(ii,:);
    else
        aa = index1;  bb = index2; index1 = index1 + 1; index2 = index2 + 1;
        W(ii,aa:bb) = WW(ii,:);
    end
end
norm(W*f(x)-fp(x),inf),  figure(1), plot(x,W*f(x),'r.',x,fp(x)),
figure(2),  spy(W)
```

6.4 Radial basis function pseudospectral method for the KdV equation

In this section we approximate solutions to the KdV equation using a RBF spectral method. We partition the spatial domain (an interval) as $x_1, x_2, \ldots, x_N$, and suppose that N centers $y_1, y_2, \ldots y_N$ have been selected. Let the RBF interpolation matrix be given by $\mathbf{A}$, and the first and third order evaluation matrices be denoted by $\mathbf{D_1}$ and $\mathbf{D_3}$, respectively. Then the first and third RBF differentiation matrices denoted by $\mathbf{D_x}$ and $\mathbf{D_{xxx}}$, respectively, are defined as $\mathbf{D_x} = \mathbf{D_1}\mathbf{A}^{-1}$ and $\mathbf{D_{xxx}} = \mathbf{D_3}\mathbf{A}^{-1}$. Let

$$\vec{u} = \begin{bmatrix} u(x_1, t) \\ u(x_2, t) \\ \vdots \\ u(x_N, t) \end{bmatrix},$$

where $u(x, t)$ solves the normalized KdV equation $u_t + uu_x + u_{xxx} = 0$. We can write the KdV equation as a semidiscrete system

$$(\vec{u})_t = -\vec{u}. * (\mathbf{D_x}\vec{u}) - \mathbf{D_{xxx}}\vec{u},$$

where the $.*$ operator is elementwise multiplication. With a differentiation matrix the nonlinearity uu_x is handled easily. The method of lines can be used to fully discretize the system. An example implementation of this discretization can be found in Code Listing 6.4.1. In table 6.2 a test case is examined. The results of this test case are visualized in figure 6.9. For an alternative numerical method (based on updating RBF coefficients via an implicit ODE solver) see [21] and [17].

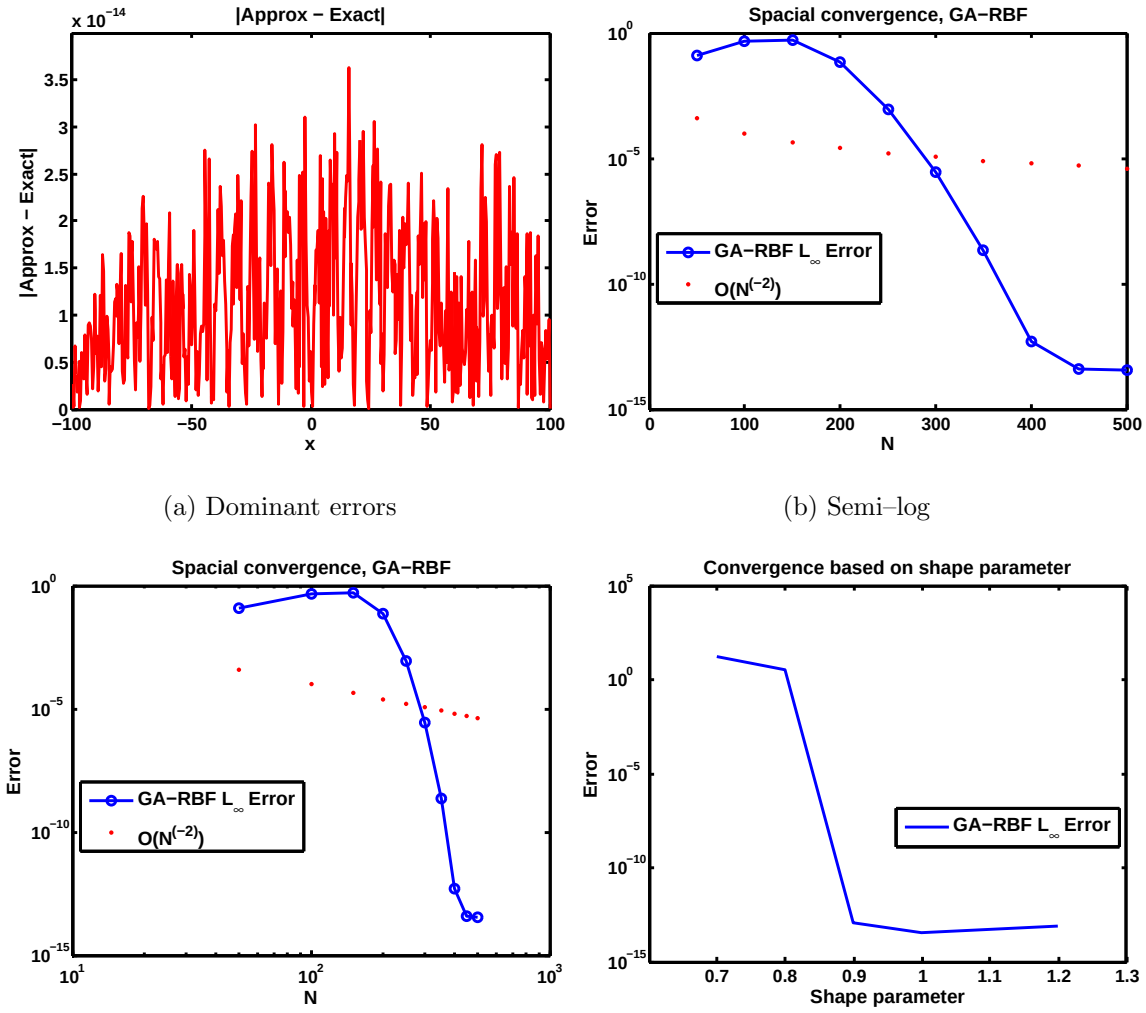
	One Soliton
Amplitude (a)	0.5
Speed (c)	0.25
Final time (T)	10
Spatial domain length	$[-100, 100]$
Radial basis function	Gaussian (GA)
RBF shape parameter	1
Associated figure	6.9
Boundary conditions	Zero flux
N	450

Table 6.2: Test problem for the GA–RBF spectral method (KdV equation).

Code Listing 6.4.1

Main RBF Pseudospectral file (KdV)

```
% Approximates the solution to the KdV equation (u_t + uu_x + u_xxx = 0).
% The spatial domain is [-100,100] with zero flux boundary conditions.  The
% approximation is done by a Gaussian radial basis function spectral method
% with shape parameter 1.  A uniform grid is used for the RBF centers.
clear all;  close all;  clc;
n = 350;  L1  = 100;  x = linspace ( -L1, L1, n )';
dt = 1e-1;  Tfin = 10;  nmax = floor(Tfin/dt);  tspan = 0 : dt: Tfin;

shape = 1:  %Shape parameter
cx = (shape)*ones(n,1);

%Set up RBF interpolation and differentiation matrices.
[Ax,D1x,D3x] = deal(zeros(n));
for j=1:n
    [Ax(:,j),D1x(:,j),~,D3x(:,j)] = gau(x,x(j),cx(j));
end
D1x = D1x /( Ax );  D3x = D3x /( Ax );
%Zero flux boundaries
D1x(1,:) = zeros(size(D1x(1,:)));  D1x(end,:) = zeros(size(D1x(1,:)));
D3x(1,:) = zeros(size(D1x(1,:)));  D3x(end,:) = zeros(size(D1x(1,:)));

%Exact solution
A = 1/sqrt(6);  L = 1;  x0 = 0;
U = @(x,t) 3*A^2*sech(A*L*(x - x0/L)/2 - A^3*t/2).^2;

%Right hand side of u_t=-u_x-u_{xxx}
RHS_u = @(t,u) -u.*(D1x*u) - D3x*u;

init = U(x,0.0);
options = odeset('RelTol',2.3e-14,'AbsTol',1e-16);
[t,w] = ode113(@(t,u) RHS_u(t,u),tspan,init,options);
W1 = w(end,:);  W1 = W1';

Error = norm(W1 - U(x,Tfin),inf)
plot(x,W1,'r.',x,U(x,Tfin),x,init,'go')
```



(a) Dominant errors

(b) Semi–log

(c) Log–log

(d) Effect of shape parameter

Figure 6.9: Spatial convergence for GA–RBF spectral method (single soliton initial condition). Relevant parameters: $\Delta t = 0.1, T_{\text{final}} = 10$.

6.5 Local radial basis function collocation method for the KdV equation

Figure 6.11 has information relating to a local RBF collocation method for the KdV equation. Figure 6.11a shows that for a spatial domain (of N grid points), increasing the stencil size improves the accuracy. For a stencil size of around 300, one can nearly match the accuracy given by a global RBF spectral method (see figures 6.9b and 6.9c). The sparsity patterns (stencil size 289, and grid size $N = 400$) for the interpolation and differentiation matrices are given in figures 6.11b and 6.11c. Sparsity ratios can be found in figure 6.11d.

The local RBF collocation/pseudospectral method remains largely the same, except that the matrices $\mathbf{D_x}$ and $\mathbf{D_{xxx}}$ are replaced with their local counterparts $\mathbf{W_x}$ and $\mathbf{W_{xxx}}$ (see section 6.3 and replace $\mathcal{L}$ with the appropriate differential operator). After discretizing the domain and selecting a RBF, RBF centers, shape parameters, and a stencil size, the semidiscrete system takes the form

$$(\vec{u})_t = -\vec{u}. * (\mathbf{W_x}\vec{u}) - \mathbf{W_{xxx}}\vec{u},$$

where $\vec{u}$ is defined in section 6.4.

Code Listing 6.5.1 provides a MATLAB function file that will calculate a local RBF differentiation matrix. In Code Listing 6.5.2 a local RBF collocation/pseudospectral method is applied to the KdV equation (same test problem parameters from table 6.2). A plot of the eigenvalues of the linearized differentiation operator $-\mathbf{W_x} - \mathbf{W_{xxx}}$ scaled by Δt can be found in figure 6.10. The eigenvalues are outside of the stability range for the Adams–Bashforth–Moulton PECE solver (`ode113`), but the horizontal scale is so small that the scaled eigenvalues are close enough to see convergence. Hyperviscosity and pseudospecta analysis could further add to the stability of this local RBF spectral method.

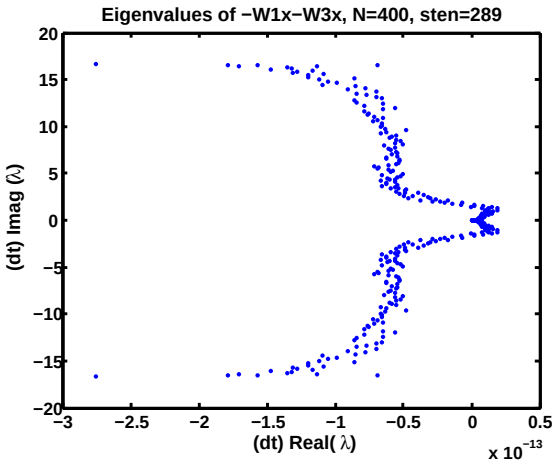
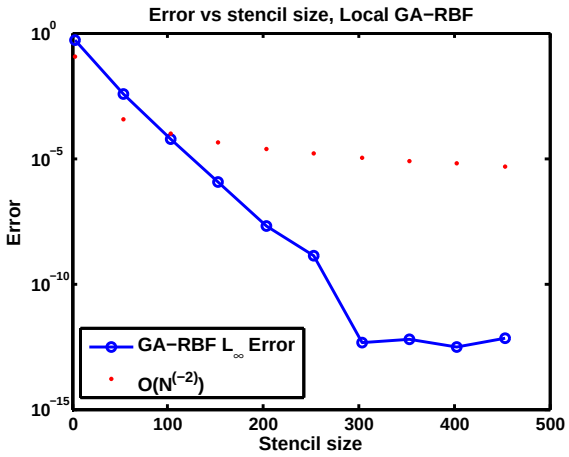
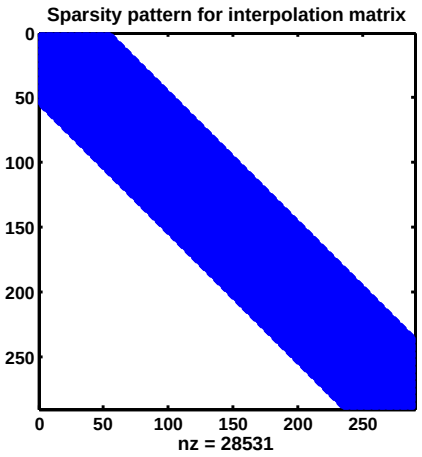


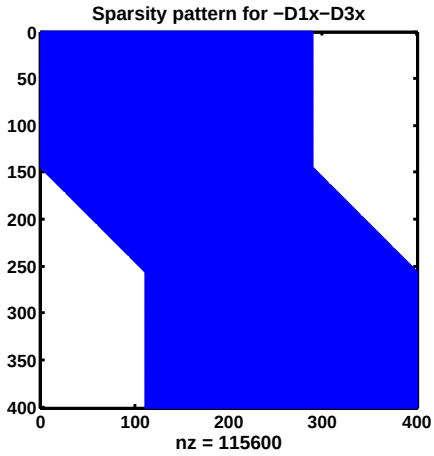
Figure 6.10: Eigenvalues of the linearized differentiation operator $-\mathbf{W_x} - \mathbf{W_{xxx}}$. GA–RBF, $N = 400$, $\epsilon = 1$, and stencil size is 289.



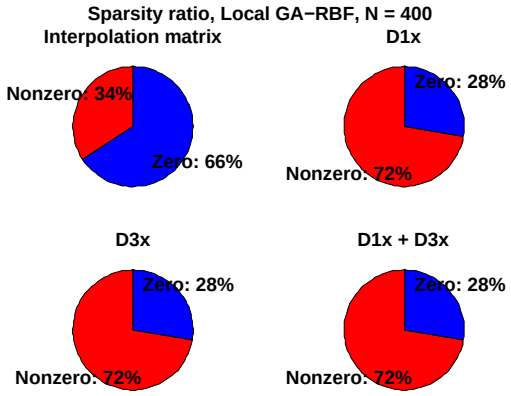
(a) Effect of stencil size



(b) Interpolation matrix sparsity



(c) Differentiation matrix sparsity



(d) Sparsity pie chart

Figure 6.11: Local RBF method. Figures 6.11c , and 6.11d have a large stencil size of 289, with $N = 400$.

Code Listing 6.5.1

Main Local RBF Differentiation file

```
% This function constructs the local RBF differentiation matrices.
% INPUT VARIABLES:
% **A       , RBF interpolation matrix of size (sten x sten)
% **L       , RBF differentiation matrix of size (sten x sten)
% **sten    , Stencil size, must be odd and less than or equal to n
% **n       , number of RBF centers
function [W] = constructW(A,L,sten,n)
Hh = zeros(n,sten);
for ii = 1 : n   %Construct RBF differentiation weights at center ii
    if ( ii <= 0.5*(sten-3)+1 )
        Lpara = ii;  h = L(Lpara,:);  Hh(ii,:) = h;
    elseif ( ii >=  n - (0.5*(sten-3)) )
        Lpara = ii - n + sten;  h = L(Lpara,:);  Hh(ii,:) = h;
    else
        Lpara = (sten+1)/2;   h = L(Lpara,:);  Hh(ii,:) = h;
    end
end
WW = Hh / (A); %Construct all RBF differentiation weights
W = deal(zeros(n));  index1 = 1;  index2 = sten;
for ii = 1 : n  %Construct RBF differentiation matrix W
    if ( ii <= 0.5*(sten-3)+1 )
        aa      = 1;  bb = sten;  W(ii,aa:bb) = WW(ii,:);
    elseif ( ii >=  n - (0.5*(sten-3)) )
        aa      = n - sten+1;  bb = n;  W(ii,aa:bb) = WW(ii,:);
    else
        aa = index1; bb = index2; index1 = index1 + 1; index2 = index2 + 1;
        W(ii,aa:bb) = WW(ii,:);
    end
end
```

Code Listing 6.5.2

Main Local RBF Pseudospectral file (KdV)

```
% Approximates a solution to the KdV equation on the domain [-100,100] with
% zero flux boundary conditions.  A local Gaussian RBF spectral method is
% implemented with a shape parameter of 1.  The stencil size must be odd.
% Needs the files gau.m and constructW.m.
clear all;  close all;  clc;
dt = 1e-1;  Tfin = 10;  nmax = floor(Tfin/dt);
L1  = 100;  tspan = 0 : dt: Tfin;
n = 350;  x = linspace ( -L1, L1, n )';

sten = 253;  x1 = x(1:sten);  shape = 1;
cx1 = (shape)*ones(sten,1);  [A,D1,D3] = deal(zeros(sten));
for j=1:sten
    [A(:,j),D1(:,j),~,D3(:,j)] = gau(x1,x1(j),cx1(j));
end
%Construct local RBF differentiation matrices
[W1] = constructW(A,D1,sten,n);  [W2] = constructW(A,D3,sten,n);
W1x = W1;  W3x = W2;

%Zero flux boundary conditions
W1x(1,:) = zeros(size(W1x(1,:)));  W1x(end,:) = zeros(size(W1x(1,:)));
W3x(1,:) = zeros(size(W1x(1,:)));  W3x(end,:) = zeros(size(W1x(1,:)));

A = sqrt(1)/sqrt(6);  L = 1;  x0 = 0;
U = @(x,t) 3*A^2*sech(A*L*(x - x0/L)/2 - A^3*t/2).^2; %Exact solution
RHS_u = @(t,u) -u.*(W1x*u) - W3x*u;  %Right hand side, u_t=-uu_x-u_{xxx}

init = U(x,0.0);
options = odeset('RelTol',2.3e-14,'AbsTol',1e-16);
[t,w] = ode113(@(t,u) RHS_u(t,u),tspan,init,options);
w1 = w(end,:);  w1 = w1';

norm(w1 - U(x,Tfin),inf)
```

6.6 Radial basis function pseudospectral method for the BBM equation

The semidiscrete system for the BBM equation $u_t + u_x + uu_x - u_{xxt} = 0$ takes the form

$$(\mathbf{I} - \mathbf{D_{xx}})(\vec{u})_t = -\vec{u}. * (\mathbf{D_x}\vec{u}) - \mathbf{D_x}\vec{u}, \tag{6.8}$$

where $\vec{u}$ is defined in section 6.4, and $\mathbf{I}$ is the identity matrix of appropriate size. The left hand side of equation 6.8 has a matrix that needs to be inverted. At each step a linear system is to be solved. All of MATLAB's ODE routines come equipped with a "Mass" matrix option that can handle ODE systems of the form $\mathbf{M}(t)\vec{u}' = \vec{F}(t, \vec{u})$, where $\mathbf{M}(t)$ is a matrix (that is not necessarily constant). Code Listing 6.6.1 has a sample implementation of a global GA–RBF spectral method. Figure 6.12 has an analysis of the method for a test case described in table 6.3. The results of a local RBF method applied to the same test case is shown in figure 6.13.

	One Soliton ($N = 350$)
Amplitude (a)	0.5
Speed (c)	7/6
Final time (T)	10
Spatial domain length	$[-100, 100]$
Radial basis function	Gaussian (GA)
RBF shape parameter	0.8
Associated figure	6.13

Table 6.3: Test problem for the GA–RBF spectral method (BBM equation).

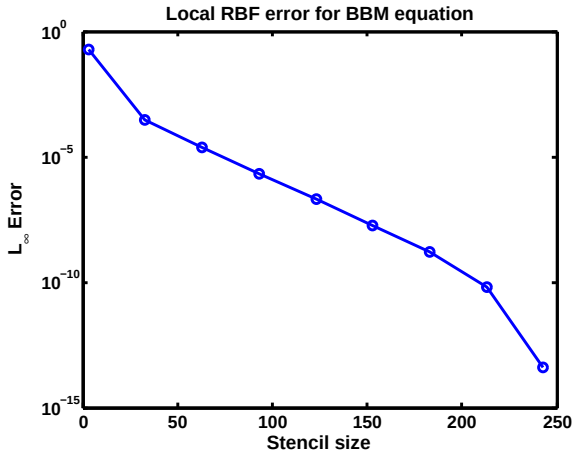
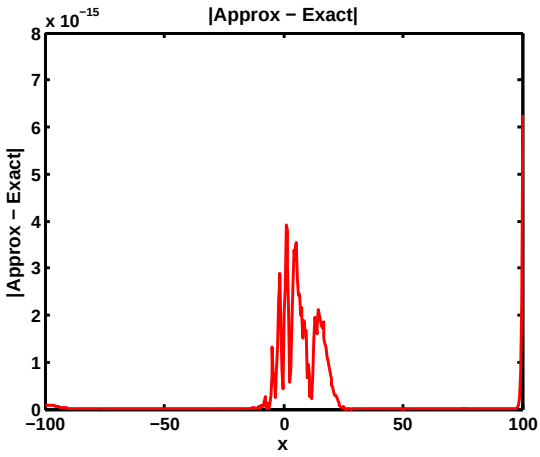


Figure 6.12: Error vs stencil size for a local GA–RBF collocation method.

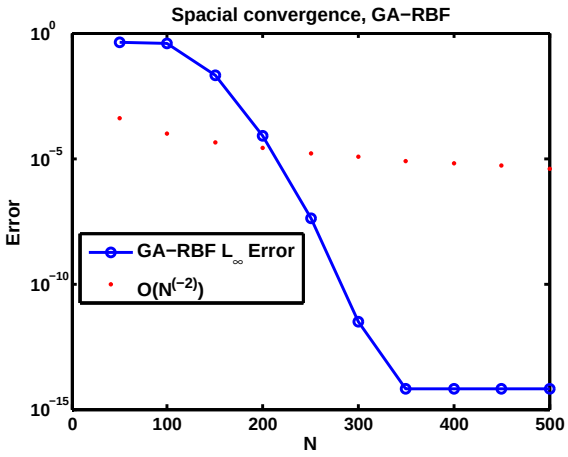
Code Listing 6.6.1

Main RBF Pseudospectral file (BBM)

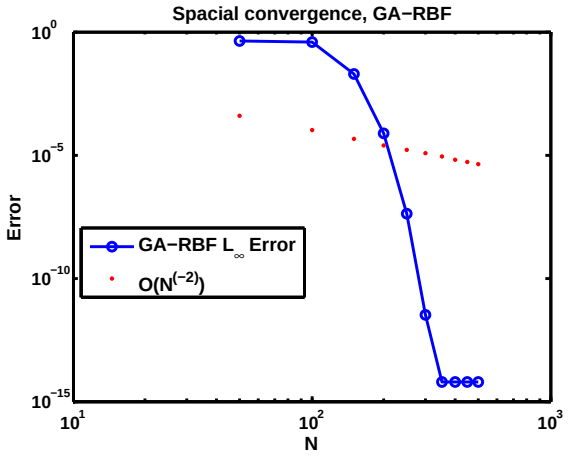
```
% Approximates the solution to the BBM equation which is given by
% u_t + u_x + uu_x - u_{xxt} = 0.  The spatial domain is [-100,100] with no
% flux boundary conditions.  The approximation is done by a Gaussian radial
% basis function spectral method with shape parameter 0.8.  ODE113 is used.
% A uniform grid is used for the RBF centers.
clear all;  close all;  clc;

n = 300;  L1  = 100;
dt = 1e-1;  Tfin = 10;  nmax = floor(Tfin/dt);
tspan = 0 : dt : Tfin;  x = linspace ( -L1, L1, n )';
shape = 0.8;  %Shape parameter
cx = (shape)*ones(n,1);  [Ax,D1x,D2x] = deal(zeros(n));
for j = 1 : n
    [Ax(:,j),D1x(:,j),D2x(:,j)] = gau(x,x(j),cx(j));
end
D1x = D1x / ( Ax );  D2x = D2x / ( Ax );
%Zero flux
D1x(1,:) = zeros(size(D1x(1,:)));  D1x(end,:) = zeros(size(D1x(1,:)));
D2x(1,:) = zeros(size(D1x(1,:)));  D2x(end,:) = zeros(size(D1x(1,:)));

c_s = (5/3)*(1/10) + 1;    %Profile speed
U = @(x,t) 3*(c_s-1)*(sech( 0.5*sqrt((c_s-1)/c_s)*(x - c_s*t) )).^2;

RHS_u = @(t,u) - 0.5*D1x*(u.^2) - D1x*u; %BBM right hand side

I = eye(n);
LL = I - D2x;  %Mass matrix (I-D2x)*U =-D1x*U -0.5*D1x*(U^2)

init = U(x,0.0);  %Initial conditon
options = odeset('RelTol',2.3e-14,'AbsTol',1e-16,'Mass',LL);
[t,w] = ode113(@(t,u) RHS_u(t,u),tspan,init,options);
W1 = w(end,:);  W1 = W1';

Err = norm(W1 - U(x,Tfin),inf)
```



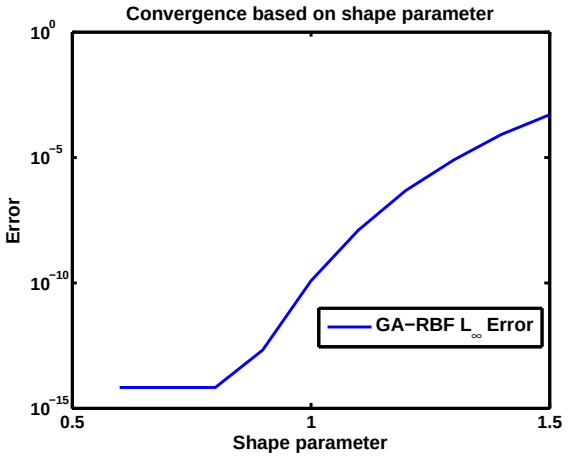
(a) Dominant errors ($N = 350$)



(b) Semi–log ($N = 350$)



(c) Log–log ($N = 350$)



(d) Effect of shape parameter ($N = 350$)

Figure 6.13: Spatial convergence for GA–RBF spectral method (single soliton initial condition). Notice in figure 6.13a the right boundary is providing dominant errors.

6.7 Radial basis function pseudospectral method for a Boussinesq system

Bona, Chen, and Saut examine varations of the classical Boussinesq system and related generalizations in [8] and [9]. These systems are well known to model long waves of small amplitude, which have a variety of applications. The family of Boussinesq systems we will consider have the form

$$\eta_t + u_x + (u\eta)_x - au_{xxx} - b\eta_{xxt} = 0 \tag{6.9}$$

$$u_t + \eta_x + uu_x - c\eta_{xxx} - du_{xxt} = 0, \tag{6.10}$$

where the constants a, b, c, and d are real constants that are related by the parameters μ, λ, and $0 \leq \theta \leq 1$:

$$a = 0.5(\theta^2 - 1/3)\lambda, \qquad b = 0.5(\theta^2 - 1/3)(1 - \lambda), \tag{6.11a}$$

$$c = 0.5(1 - \theta^2)\mu, \qquad d = 0.5(1 - \theta^2)(1 - \mu). \tag{6.11b}$$

The parameters μ and λ are arbitrary. Exact soliton solutions to the family of systems in equations (6.9) and (6.10) are derived in [15]. For $a = 0$ a solution to the system in equations (6.9) and (6.10) is given by

$$\eta(x,t) \equiv -1 \tag{6.12}$$

$$u(x,t) = (1 - d\rho)C_s + 3dC_s\text{sech}^2[0.5\sqrt{\rho}(x - x_0 - C_s t)], \tag{6.13}$$

where C_s (profile speed), x_0 (initial profile position), and $\rho \geq 0$ are arbitrary constants.

	One Soliton ($N = 550$)
System parameters $a = b = c$	0
System parameters d	1/3
ρ	0.5
Speed C_s	1.7
Initial position x_0	0
Final time (T)	1
Spatial domain length	$[-40, 80]$
Radial basis function	Multiquadratic (MQ)
RBF shape parameter	0.4
Associated figure	6.15

Table 6.4: Test problem for the MQ–RBF spectral method (Boussinesq system).

The system in equations (6.12) and (6.13) can be written as a semidiscrete system

$$[\mathbf{I} - b\mathbf{D_{xx}}](\vec{\eta})_t = -\mathbf{D_x}\vec{u} - \mathbf{D_x}(\vec{u}. * \vec{\eta}) + a\mathbf{D_{xxx}}\vec{u} \tag{6.14}$$

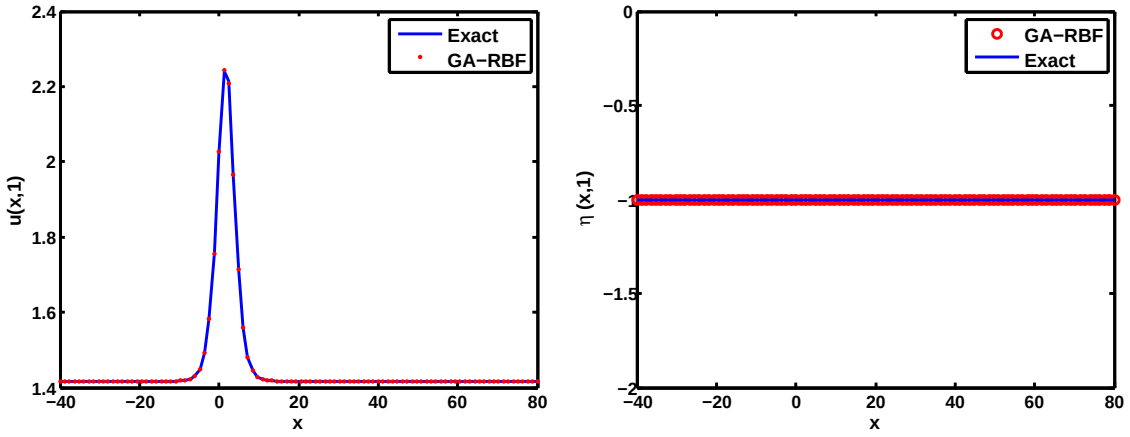
$$[\mathbf{I} - d\mathbf{D_{xx}}](\vec{u})_t = -\mathbf{D_x}\vec{\eta} - \vec{u}. * (\mathbf{D_x}\vec{u}) + c\mathbf{D_{xxx}}\vec{\eta}, \tag{6.15}$$

where the $.*$ operation is performed elementwise. The vectors $\vec{u}$, and $\vec{\eta}$ are defined as

$$\vec{u} = \begin{bmatrix} u(x_1,t) \\ u(x_2,t) \\ \vdots \\ u(x_N,t) \end{bmatrix}, \quad \vec{\eta} = \begin{bmatrix} \eta(x_1,t) \\ \eta(x_2,t) \\ \vdots \\ \eta(x_N,t) \end{bmatrix}.$$

Using one of MATLAB's ODE routines we can advance the coupled system (6.14) and (6.15) in time. The mass matrix option will be used, since a linear system is to be solved at each time step. Code Listing 6.7.1 has an example implementation. Figure 6.15 has a plot with an analysis of the test case described in table 6.4. The errors are only examined for $u(x,t)$. For the system parameters chosen the function η is constant, and the MQ–RBF approximation calculates η to machine precision.

In figure 6.15a, one can see that the dominant errors occur at the boundaries. The rate of convergence is slower, but still better than 8th order convergence. Figures 6.15b and 6.15c display the convergence rates as the number of RBF centers is increased. A plot of the approximation and exact solution are given in figure 6.14. Notice that $u(x,1)$ is shifted up from the origin (see equation (6.13)).



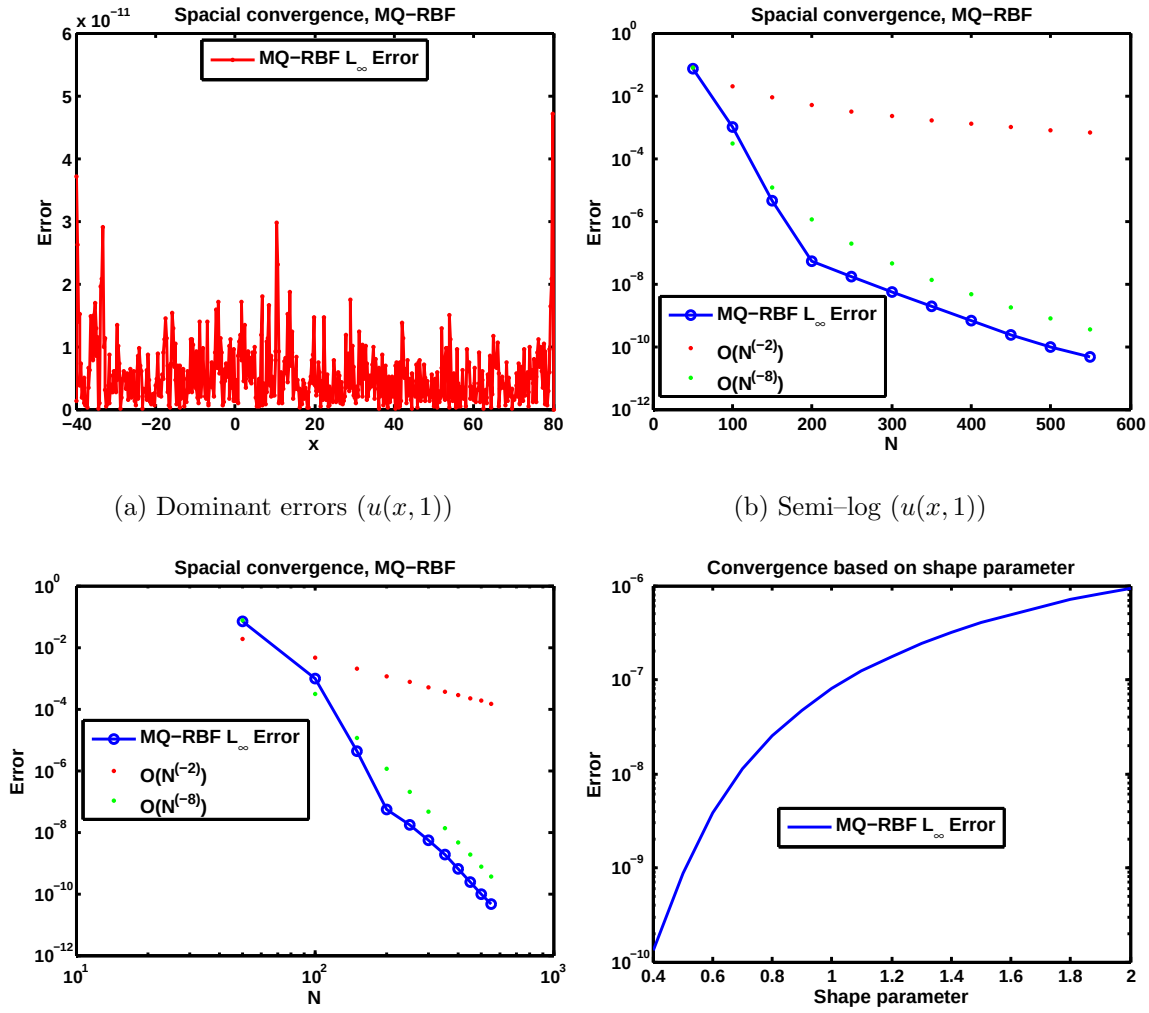
(a) $u(x,1)$ (b) $\eta(x,1)$

Figure 6.14: Plots of exact solution and MQ–RBF approximation.

Code Listing 6.7.1

Main RBF Pseudospectral file

```
% Approximates the solution to a Boussinesq system given by
% n_t + u_x + (un)_x + a*u_{xxx} - b*n_{xxt} = 0         (1)
% u_t + n_x + (uu_x) + c*n_{xxx} - d*u_{xxt} = 0,        (2)
% where n = n(x,t), u = u(x,t), a = b = c = 0, and d = 1/3.
% See Chen 1997 for information about soliton solutions to this system.
%The spatial domain is [-40,80] with zero flux boundary conditions.  The
% approximation is done by a multiquadratic radial basis function spectral
% method with shape parameter 0.4.  Matlab's ODE113 is used to evolve the
% equations in time.  A uniform grid is used for the RBF centers.
clear all;  close all;  clc;
N = 350;  dt = 1e-1;  Tfin = 1;  tspan = 0 : dt : Tfin;
L = 80;   d = 1/3;  C_s = (21/3)*(1/10) + 1; rho = 0.5;  x0 = 0;

u  = @(x,t) (1-d*rho)*C_s+3*d*C_s*rho*sech(0.5*sqrt(rho)*(x+x0-C_s*t)).^2;
eta= @(x,t) -1*ones(size(x));

x = linspace(-L/2,L,N)';  Nx = length(x);
cx = (0.4)*ones(Nx,1);    [Ax,D1x,D2x] = deal(zeros(Nx));
for j=1:Nx
    [Ax(:,j),D1x(:,j),D2x(:,j)] = mq(x,x(j),cx(j));
end

%Set up interpolation and differentiation matrices
D1x = D1x / (Ax);  D2x = D2x / (Ax);
D1x(1,:) = zeros(size(D1x(1,:)));  D1x(end,:) = zeros(size(D1x(1,:)));
D2x(1,:) = zeros(size(D1x(1,:)));  D2x(end,:) = zeros(size(D1x(1,:)));

I = eye(N);  ETA = eta(x,0);  U   = u(x,0);  init = [ETA; U];

%Right hand side function for ODE solver
RHS_u = @(t,q) [-D1x*q(N+1:end)-D1x*(q(N+1:end).*q(1:N));...
    -D1x*q(1:N) - 0.5*D1x*(q(N+1:end).^2)];

L2 = I - d*D2x;  L2 = blkdiag(I,L2);%need a mass matrix for u_{xxt}
options = odeset('RelTol',2.3e-14,'AbsTol',1e-16,'Mass',L2);
[t,w] = ode113(@(t,q) RHS_u(t,q),tspan,init,options);
W = w(end,:)';  U = W(N+1:end);

Err = norm(U - u(x,Tfin),inf)
figure(1),  plot(x, abs(U-u(x,Tfin) ),'r.-')
```



(a) Dominant errors ($u(x,1)$)

(b) Semi–log ($u(x,1)$)

(c) Log–log ($u(x,1)$)

(d) Effect of shape parameter ($u(x,1)$)

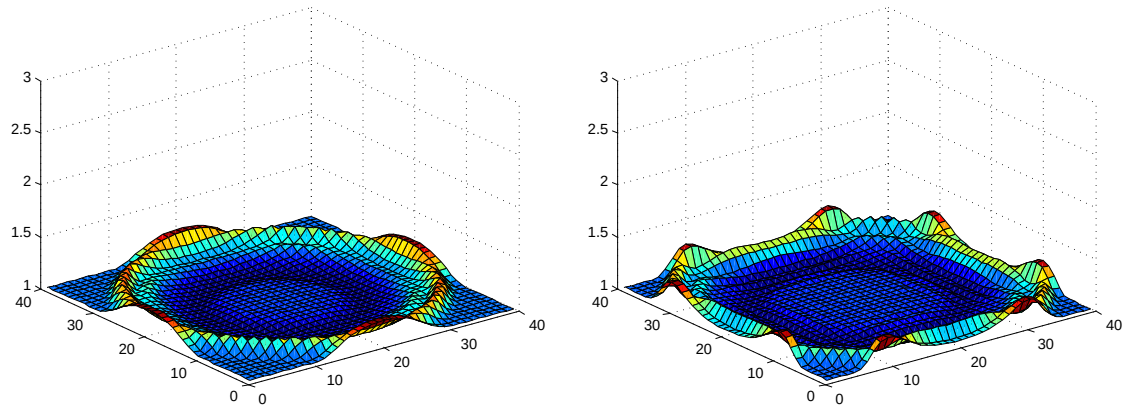
Figure 6.15: Spatial convergence for MQ–RBF spectral method (single soliton initial condition). Notice in figure 6.15a the boundaries are providing dominant errors. The errors are comparing the function $u(x,t)$ ($\eta(x,t) \equiv 1$, and approximating a constant function in this case is trivial).

6.8 Radial basis function pseudospectral method for 1D/2D shallow water equations

Code Listings 6.8.1 and 6.8.2 have implementations of a RBF pseudospectral method for the 1D and 2D SW equations. Notice that it is much easier to implement different boundary conditions than the Fourier spectral approach given in chapter 5. One can replace the implicit explicit manual time stepping (described in section 5.2) with one of MATLAB's ODE routines. See sections 6.7 and 6.9 for examples of applying RBF spectral methods to PDE systems. The derivatives are calculated a bit differently for systems with two space dimensions . Directly below is some pseudo code related to this calculation:

```
I = eye(n);
D1x = D1x / (Ax);           %Compute 1D derivative on [x_a,x_b]
D1y = D1y / (Ay);           %Compute 1D derivative on [y_a,y_b]
Lx = sparse(kron(I,D1x));  %Compute partial derivative in x direction
Ly = sparse(kron(D1x,I));  %Compute partial derivative in y direction
```

The matrices $\mathbf{D_x}$ and $\mathbf{D_y}$ are technically one–dimensional, and are calculated via equation 6.5. Using the Kronecker tensor product we can construct two–dimensional RBF discretized differentiation operators. More information on this particular technique can be found in [82] and [27].



(a) Reflecting boundaries

(b) Reflecting boundaries

Figure 6.16: RBF spectral method for 2D SW equations (same water droplet in a bathtub initial condition used in figure 5.7).

Code Listing 6.8.1

Main RBF Pseudospectral file

```
% Approximates a solution to the 1D SW equations
clear all;  close all;  clc;
n = 2^(8);  dt = 1e-2;  Tfin = 15;  nmax = floor(Tfin/dt);  g = 9.80;
L = 200;  x = ( linspace ( 0, L, n ) )';

shape = 5;  %shape parameter
cx = (shape)*ones(n,1);
[Ax,D1x,D2x,D3x] = deal(zeros(n));
for j = 1 : n
    [Ax(:,j),D1x(:,j),D2x(:,j),D3x(:,j)] = mq(x,x(j),cx(j));
end
D1x = D1x / (Ax);
D2x = D2x / (Ax);

%Initial conditions
h =  2.0 + sin ( (2*pi/L) * x );  v = zeros(size(h));
u = v;  uh = u.*h;
%Right hand side functions h_t = ..., (uh)_t = ...
RHS_h  = @(h,uh) -D1x*(uh);
RHS_uh = @(h,uh) -D1x*( (uh).^2 ./ h + 0.5*g*h.^2 );

for jj = 1 : nmax
    k1 = dt*RHS_h(h       ,uh);  %     RK4 step for h
    k2 = dt*RHS_h(h + k1/2,uh);
    k3 = dt*RHS_h(h + k2/2,uh);
    k4 = dt*RHS_h(h + k3  ,uh);
    h  = h + (1/6)*( k1 + 2*(k2 + k3) + k4 );

    k1 = dt*RHS_uh(h,uh);        %     RK4 step for uh
    k2 = dt*RHS_uh(h,uh + k1/2);
    k3 = dt*RHS_uh(h,uh + k2/2);
    k4 = dt*RHS_uh(h,uh + k3);
    uh  = uh + (1/6)*( k1 + 2*(k2 + k3) + k4 );
    h(1) = h(end-1);  h(end) = h(2);   %Periodic boundary conditions
    uh(1) = uh(end-1);  uh(end) = uh(2);
    H(:,jj) = h;
end
[X,T] = meshgrid(x,linspace(0,Tfin,size(H,2)));
figure(1),  surf(X,T,H'),  shading interp
```

Code Listing 6.8.2

Main RBF Pseudospectral file

```
% A 2D SWE simulation.  Water droplet in a rectangular paralelipiped.
clear all;  close all;  clc;
n = 80;  dt = 1e-2;  Tfin = 45;  nmax = floor(Tfin/dt);  g = 9.8;
L1 = 100;  x  = ( linspace ( -L1, L1, n ) )';  y = x;
[XX,YY] = meshgrid(x,y);  xx = XX(:); yy = YY(:);

shape = 0.2393;  %shape parameter
cx = shape*ones(n,1);
%Set up RBF matrices
[Ax,D1x] = deal(zeros(n));
for j = 1 : n
    [Ax(:,j),D1x(:,j)] = mq(x,x(j),cx(j));
end
D1x = D1x / (Ax);  I = eye(n);
D1y = D1x;  %Compute 1D derivatives
Lx = sparse(kron(I,D1x));  Ly = sparse(kron(D1x,I));  %Compute partial derivatives
%Initial conditions
h = 1.0 + exp( -(xx.^2+yy.^2)/(2*L1) );  H = zeros(n, n);  U = H;  V = H;
H = H + reshape(h,n , n);  vh = zeros(size(h));  uh = vh;
grid = surf(H);  axis([0 n 0 n 1 3]);  hold all;  %plot simulation

RHS_h  = @(h,uh,vh) -Lx*(uh) - Ly*(vh);
RHS_uh = @(h,uh,vh) -Lx*( (uh).^2 ./ h + 0.5*g*h.^2 ) - Ly*((uh./h).*vh);
RHS_vh = @(h,uh,vh) -Ly*( (vh).^2 ./ h + 0.5*g*h.^2 ) - Lx*((uh./h).*vh);

while ( 1 == 1 )  %for jj = 1 : nmax
    set(grid ,'zdata', H);  drawnow                %plot simulation
    h  = h  + (dt)* RHS_h(h,uh,vh);
    uh = uh + (dt)*RHS_uh(h,uh,vh);
    vh = vh + (dt)*RHS_vh(h,uh,vh);

    u = uh ./ h;  U(:,:) = reshape(u,n, n); %Reflecting boundary conditions
    U(:,1) = 0.0;  U(:,end) = 0.0;
    U(1,:) = -U(2,:);  U(end,:) = -U(end-1,:);
    uh = reshape(U,n^2, 1) .* h;

    v = vh ./ h;  V(:,:) = reshape(v,n, n); %Reflecting boundary conditions
    V(1,:) = 0.0;  V(end,:) = 0.0;
        V(:,1) = -V(:,2);  V(:,end) = -V(:,end-1);
    vh = reshape(V,n^2, 1) .* h;

    H(:,:) = reshape(h,n, n);                  %Consistency conditions
    H(:,1) = H(:,2);  H(:,n) = H(:,n-1);       %(see Cleve Moler reference)
    H(:,n) = H(:,n-1);
    H(1,:) = H(2,:);  H(n,:) = H(n-1,:);
end
```

6.9 Radial basis function pseudospectral method for the fully nonlinear 1D Serre–Green Nagdhi equations

The fully non–linear 1D Serre–Green Nagdhi (SGN) equations are given by the system of PDEs

$$h_t + (uh)_x = 0 \tag{6.16}$$

$$u_t + (0.5u^2 + gh)_x = \beta h^{-1}[h^3(u_{xt} + uu_{xx} - u_x^2)]_x, \tag{6.17}$$

where g is the acceleration due to gravity, and $\beta = 1/3$. For spectral methods it is easier to work with the following equivalent system (see [25])

$$\eta_t + [u(d+\eta)]_x = 0 \tag{6.18}$$

$$q_t + [qu - 0.5u^2 + g\eta - 0.5(d+\eta)^2 u_x^2]_x = 0, \tag{6.19}$$

$$q - u + \beta(d+\eta)^2 u_{xx} + (d+\eta)\eta_x u_x = 0, \tag{6.20}$$

where $\eta = \eta(x,t)$ is the free surface elevation ($h(x,t) = d + \eta(x,t)$, we will assume d is constant) and $q = q(x,t)$ is a conserved quantity of the form $q(x,t) = uh - \beta[h^3 u_x]_x$. Equations (6.18) and (6.19) have time dependent derivatives, however, equation (6.20) has no time dependent derivatives. The SGN equations admit exact solitary wave solutions:

$$\eta(x,t) = a \cdot \operatorname{sech}(0.5\kappa(x - ct))^2, \tag{6.21}$$

$$u(x,t) = \frac{c\eta}{d+\eta}, \tag{6.22}$$

where $c = \sqrt{g(d+a)}$ is the wave speed, a is the wave amplitude, and $(\kappa d)^2 = a/(\beta(d+a))$.

6.9.1 Discretization

We take a radial basis function approach. To begin the collocation, partition the spatial domain (an interval) as $x_1, x_2, \ldots, x_N$, and suppose that N centers $y_1, y_2, \ldots y_N$ have been selected. Then the RBF interpolation and differentiation matrices need to be constructed. The RBF interpolation matrix $\mathbf{A}$ has entries

$$\mathbf{A}_{ij} = \phi(\|x_i - y_j\|_2).$$

The first and second RBF order evaluation matrices are given by

$$\mathbf{D_1}_{ij} = \frac{\partial}{\partial x_i}\phi(\|x_i - y_j\|_2), \quad \mathbf{D_2}_{ij} = \frac{\partial^2}{\partial x_i^2}\phi(\|x_i - y_j\|_2),$$

for $i, j = 1, \ldots, N$. Then the first and second RBF differentiation matrices denoted by $\mathbf{D_x}$ and $\mathbf{D_{xx}}$, respectively, are defined as $\mathbf{D_x} = \mathbf{D_1}\mathbf{A}^{-1}$ and $\mathbf{D_{xx}} = \mathbf{D_2}\mathbf{A}^{-1}$.
Let the variables $\vec{x}$, $\vec{\eta}$, $\vec{q}$ and , $\vec{u}$ be given by

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad \vec{\eta} = \begin{bmatrix} \eta(x_1,t) \\ \eta(x_2,t) \\ \vdots \\ \eta(x_N,t) \end{bmatrix}, \quad \vec{q} = \begin{bmatrix} q(x_1,t) \\ q(x_2,t) \\ \vdots \\ q(x_N,t) \end{bmatrix}, \quad \text{and} \quad \vec{u} = \begin{bmatrix} u(x_1,t) \\ u(x_2,t) \\ \vdots \\ u(x_N,t) \end{bmatrix}.$$

In addition, let $\vec{d} \in \mathbb{R}^N$ be a vector with d in each component. Given $\vec{p} \in \mathbb{R}^N$, let the function diag : $\mathbb{R}^N \to \mathbb{R}^{N\times N}$ be defined as

$$(\text{diag}\{\vec{p}\})_{ij} = \begin{cases} \vec{p}_i, \text{ if } i = j \\ 0, \text{ otherwise,} \end{cases}$$

for $i, j = 1, 2, \ldots, N$. The equations (6.18), (6.19) , and (6.20) can be expressed as a semidiscrete system

$$\vec{\eta}_t + [\mathbf{D_x}(\vec{d} + \vec{\eta})] = \vec{0} \tag{6.23}$$

$$\vec{q}_t + \mathbf{D_x}[\vec{q}. * \vec{u} - 0.5(\vec{u})^2 + g\vec{\eta} - 0.5\text{diag}\{(\vec{d} + \vec{\eta})^2\}(\mathbf{D_x}\vec{u})^2] = \vec{0}, \tag{6.24}$$

$$\vec{q} - \vec{u} + \text{diag}\{\beta(\vec{d} + \vec{\eta})^2\}(\mathbf{D_{xx}}\vec{u}) + \text{diag}\{(\vec{d} + \vec{\eta}). * (\mathbf{D_x}\vec{\eta})\}(\mathbf{D_x}\vec{u}) = \vec{0}. \tag{6.25}$$

In equation (6.25), the .$*$ operator is element–wise multiplication. Also, in equations (6.23), (6.24), and (6.25) the square operator on vectors is element–wise. The method of lines can be employed from here to fully discretize the 1D Serre-Green Nagdhi equations. Equations (6.23) and (6.24) are of evolution type, and equation (6.25) can be treated like an elliptic PDE in the variable $\vec{u}$. To initialize η and u, equations (6.21) and (6.22) are used. The differentiation matrix $\mathbf{D_x}$ is used to initialize $q(x, t) = uh - \beta[h^3 u_x]_x$. Sample code using MATLAB's `ode113` (variable order Adams–Bashforth–Moulton PECE solver) can be found in Code Listing 6.9.1 and 6.9.2.

Algorithm (RBF spectral method) for 1D SGN

- Step 1: Select a RBF, RBF centers, collocation points, and a time step.
- Step 2: Construct the required RBF interpolation and differentation matrices outside of the main time stepping loop.
- Step 3: Use an ODE solver to advance the coupled evolution equations (6.23) and (6.24) to time level t_{n+1}.
- Step 4: Solve the linear system in equation (6.25) at the time level t_n. This updates the variable u to time level t_{n+1}.
- Step 5: Repeat steps 3 and 4 until final time is reached.

6.9.2 Test Cases for the 1D Serre–Green Nagdhi equations

For the fully non–linear 1D Serre–Green Nagdhi equations we examine three test cases. Two come from Bonneton et al. [10], and the other from Dutykh et al. [25].

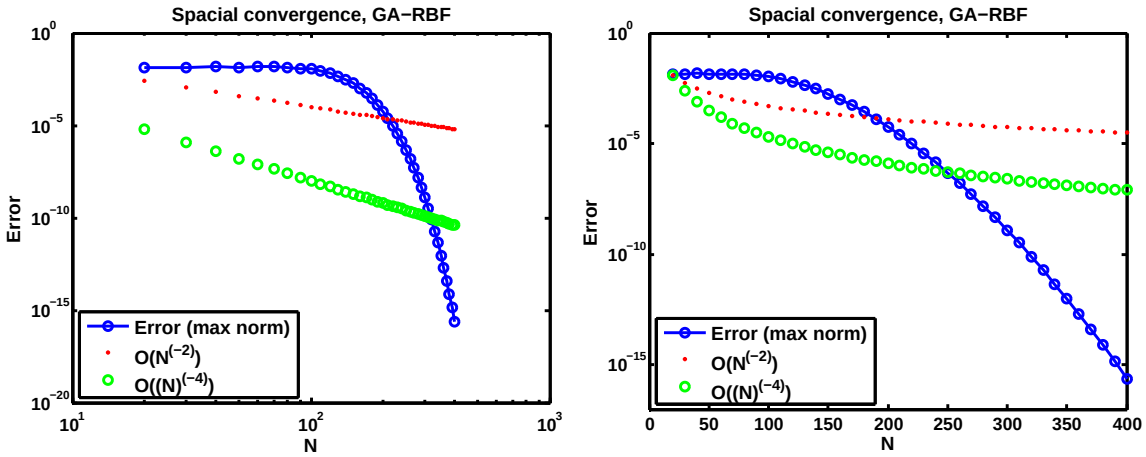
	One Soliton	One Soliton	One Soliton
Amplitude (a)	0.1	0.025	0.05
Depth above bottom (d)	0.5	0.5	1
Speed (c)	2.4343	2.2771	1.0247
Gravity (g)	9.8765	9.8765	1
Final time (T)	3	3	2
Spatial domain length	$[-30, 30]$	$[-50, 50]$	$[-100, 100]$
Radial basis function	Gaussian (GA)	Gaussian (GA)	Gaussian (GA)
RBF shape parameter	2	2	1
Reference	Bonneton et al. [10]	Bonneton et al. [10]	Dutykh et al. [25]
N	400	350	400
Associated figure	6.19a and 6.19b	6.19c and 6.19d	6.17

Table 6.5: Test problems for the 1D SGN equations.

The errors measured below are for the function $\eta(x, t)$ (free surface elevation).

Kim in [58] and Bonneton et al. in [10] have studies of finite volume operator splitting methods applied to the 1D SGN equations. For the test cases examined in both [58] and [10], a second order convergence rate is observed with a Strang splitting. Figure 6.19 confirms a near–spectral convergence rate for the same test cases described in [10] (the spatial domains in table 6.5 are slightly larger; also, since the solution decays rapidly for large x, zero flux boundary conditions are imposed). The values of N are much more moderate in the GA–RBF method. For instance, the largest Δx used in [10] is $\Delta x = 1/256$. For a domain of $[-15, 15]$, this corresponds to 7681 evenly spaced grid points. The RBF spectral method uses dense, highly nonnormal, ill conditioned matrices; so a grid spacing of that magnitude is not tractable. In figure 6.19 one can see that the relative error is near machine precision for $N = 500$, which corresponds to a $\Delta x = 0.0601$ on a domain of length 30.

As far as the author is aware, these results for the RBF pseudospectral method applied to the 1D SGN equations are new. In [25] a Fourier pseudospectral method is applied to the 1D SGN equations.



(a) Log–log ($\eta(x,2)$)

(b) Semi–log ($\eta(x,2)$)

Figure 6.17: Spatial convergence for Gaussian–RBF spectral method applied to the Dutykh et al. test cases [25]. Error is in the infinity norm, $\|\text{approx} - \text{exact}\|_\infty$.

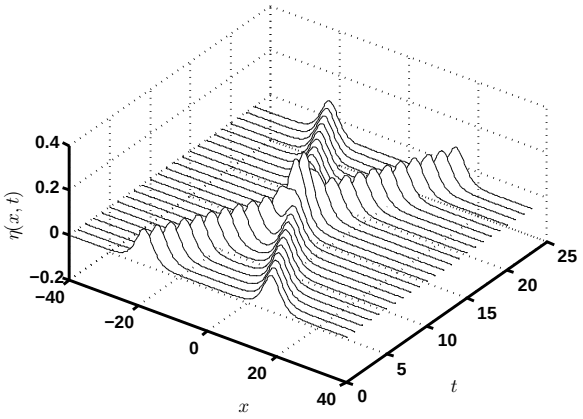
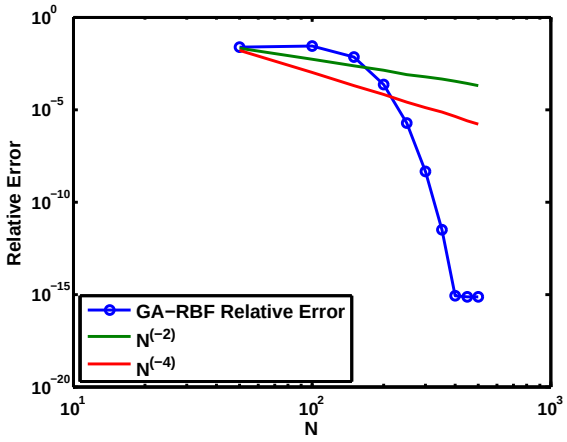
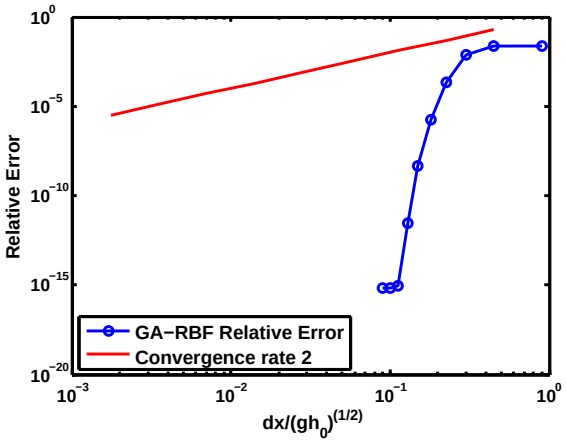


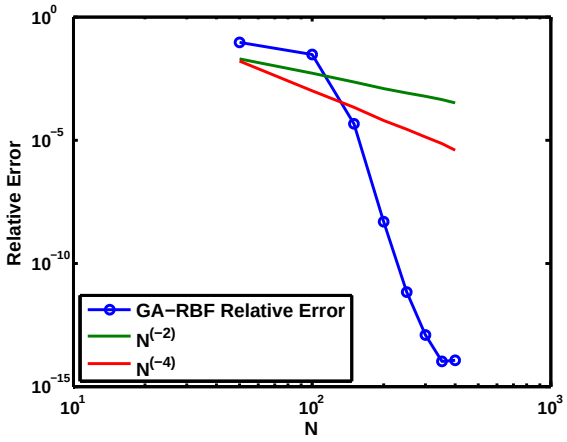
Figure 6.18: Gaussian–RBF spectral method simulation of a head on collision. Relevant parameters: $\epsilon = 2$, $N = 300$, $L = 30$, $T = 36$, $a = 0.15$, and $x_0 = \pm 20$. This test case can also be found in [25].



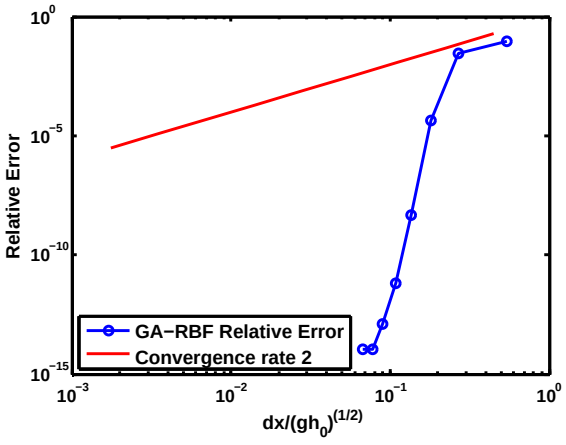
(a) $h_0 = a = 0.025$, $(\eta(x, 3))$



(b) $h_0 = a = 0.025$, $(\eta(x, 3))$



(c) $h_0 = a = 0.1$, $(\eta(x, 3))$



(d) $h_0 = a = 0.1$

Figure 6.19: Spatial convergence for Gaussian–RBF spectral method applied to the Bonneton et al. test cases [10]. Error is relative, $\|\text{approx} - \text{exact}\|_\infty / \|\text{exact}\|_\infty$.

Code Listing 6.9.1

Main RBF Pseudospectral file

```
% Approximates a solution to the nonlinear 1D Serre-Green Nagdhi system
% given by the equations
% n_t + ((d+n)u)_x                          = 0,        (1)
% q_t + (qu -0.5u^2+gn-0.5(d+n)^2u_x^2)_x = 0,        (2)
% q - u + (1/3)(d+n)^2u_xx + (d+n)n_xu_x  = 0.        (3)
% The spatial domain is [-100,100] with zero flux boundary conditions.  The
% approximation is done by a Gaussian radial basis function pseudospectral
% method with shape parameter 1.  Needs gau.m and RHS.m.

clear all;  close all;  clc;
dt = 1e-1;  Tfin = 3;
tspan = 0:dt:Tfin;

n = 400;
L  = 50;

x = ( linspace ( -L, L, n ) )';
Nx = length(x);
cx = (2)*ones(Nx,1);  %shape parameters

[Ax,D1x,D2x] = deal(zeros(Nx));
for j = 1 : Nx
    [Ax(:,j),D1x(:,j),D2x(:,j)] = gau(x,x(j),cx(j));
end
D1x = D1x / ( Ax );  D2x = D2x / ( Ax );
D1x(1,:) = zeros(size(D1x(1,:)));  D1x(end,:) = zeros(size(D1x(1,:)));
D2x(1,:) = zeros(size(D1x(1,:)));  D2x(end,:) = zeros(size(D1x(1,:)));

% Various physical paramters
d = 0.5;
g = (1/(0.45*sqrt(d)))^2;
a = 0.025;  %0.025,0.1
BETA = 1.0 / 3.0;
c = sqrt(g*(d+a));
kappa = sqrt(3*a)/(d*sqrt(a+d));

% Various physical paramters
eta  = @(x,t) a * sech( 0.5*kappa*(x - c * t)).^2;
u    = @(x,t) c*eta(x,t) ./ (d + eta(x,t));

%Initial conditions
eta0  = eta(x,0.0);
```

```
u_x   = (D1x*(u(x,0)));
u_xx  = (D2x*(u(x,0)));
eta_x = D1x*(eta0);
q0    = u(x,0) - (d+eta0).*((BETA)*(d+eta0).*u_xx + eta_x.*u_x);
U = u(x,0);  Q = q0;  ETA = eta0;
init = [ETA; Q];

%for higher accuracy set RelTol to 2.3e-14
options = odeset('RelTol',2.3e-11,'AbsTol',eps);
[t,w] = ode113(@(t,q) RHS(t,q,0,D1x,D2x,g,d,BETA), tspan,init,options);
W  = w(end,:);  W1 = W(1:n)';

error3 = norm( W1 - eta(x,Tfin) , inf)
error4 = norm( W1 - eta(x,Tfin) , inf)/norm( eta(x,Tfin) , inf)
```

Code Listing 6.9.2

RBF Pseudospectral function file

```
% This function file provides the right hand side from the system of ODEs
% created by the method of lines.
% INPUT VARIABLES:
% **t       , time
% **q       , input from previous time step
% **options , options for Matlab's ode ode routines
% **D1x,D2x , RBF first and second order differentiation matrices
% **g,d,BETA, various physical constants
function f = RHS(t,q,options,D1x,D2x,g,d,BETA)
n  = length(D1x);  I = eye(n);
ETA  = q(1:n);  ETA_x = D1x*ETA;
Q = q(n+1 : 2*n);

%Set up and approximate u via an elliptic equation at time level t_n
L   = ( BETA*diag((d+ETA).^2)*D2x + diag((d+ETA).*ETA_x)*D1x - I );
U   = L \ (-Q);

%Set up right hand side for the coupled equations (1) and (2)
rhs1 = -D1x*((d+ETA) .*U );
rhs2 = -D1x*( Q.*U - 0.5*(U).^2 + g*ETA - 0.5*(d+ETA).^2.*((D1x*U).^2) );
f = [rhs1;rhs2];
```

Chapter 7

CONCLUSIONS AND FUTURE DIRECTIONS

A brief conclusion and possible areas of further study are presented in this chapter.

7.1 Conclusions

Hyperbolic problems pose many challenges for any numerical method for PDEs. In the case of hyperbolic PDEs that model small–amplitude long waves, spectral methods can provide useful approximations - especially when the PDEs admit smooth solitary solutions. In this situation, as we have demonstrated, spectral methods can act as high accuracy reference solutions. When analytic solutions are not known, as in the small dispersion limit for the KdV and CH equations, spectral methods can act as low investment approximation. They are much easier to implement and analyze than the current state of the art asymptotic and perturbation theories.

Fourier spectral methods are classic numerical techniques and are well established for scalar PDEs. The main issues that they suffer from in this context are aliasing (Gibbs phenomena) and periodic boundary conditions. For scalar PDEs that admit smooth solutions that decay rapidly for large x, both of these issues can be remedied. There are workarounds for non–periodic boundary conditions based on even/odd periodic function extension, however they are a little cumbersome to implement. The most common ways to handle aliasing are: to increase the number of grid points and reduce the time step, Orszag's 2/3–rule, and spectral viscosity.

For nonsmooth data, the standard Fourier spectral method breaks down. This can be seen in our study of the CH equation. The evolution of peakons is not captured accurately by standard Fourier spectral methods. This is a harsh limitation of the Fourier spectral method, because for nonsmooth data the convergence rate is much lower than first or second order.

Fourier spectral methods can be applied to systems of PDEs (see chapter 5), however, the usefulness of this is debatable. For optimal results smooth data is necessary, and nonlinear hyperbolic systems are well known to generate nonsmooth data. The restriction to periodic boundary conditions can also be awkward, and result in a numerical method that is not robust.

There are many different types of spectral methods, simply change the basis and you can arrive at different spectral methods (for example Chebyshev, Legendre, etc.). Radial basis functions were originally applied to scattered node approximation problems in high dimensions. More recently they have been applied in a wide range of areas, including PDEs. RBF

spectral methods provide an interesting alternative to the classical spectral and pseudospectral methods. This is because of their generality, RBFs have been shown to generalize all of the legacy pseudospectral methods. In addition they provide a wide range of flexibility. For instance, there is an assortment of basis functions to choose from. So you are not confined to a particular basis (as in the Fourier, Chebyshev, Legendre, etc. spectral methods). RBFs are independent of coordinate system, dimension as well as domain geometry. RBF centers are freely customizable, as are RBF shape parameters. The RBF shape parameters determine how flat the basis functions are, and they are modifiable at each RBF center.

In chapter 6 RBF spectral methods are showcased for PDEs that model shallow water/small–amplitude long wave phenomena. When smooth data is used, spectral accuracy is observed in the spatial dimensions. So, in that sense, the RBF spectral methods perform just as well as the Fourier based methods. Since RBF spectral methods use differentiation matrices, the can also handle more general boundary conditions.

RBF spectral methods suffer from aliasing, and in general they also do not handle nonsmooth data well. To deal with aliasing, spectral or hyper viscosity can be utilized. Global RBF spectral methods work with dense, highly nonnormal, ill conditioned matrices. This hinders the scalability of global RBF spectral methods. Local RBF differentation applied to RBF spectral methods (also known as radial basis function finite differences) address this particular issue. RBF–FD introduce interpolation and differentiation matrices that have controllable sparsity, and are much more suitable for scaling to larger problems. Fourier and RBF spectral methods both work best with smooth data.

7.2 Future Directions

There are a number of possible improvements for spectral methods in the case of PDEs that model shallow water/small–amplitude long wave phenomena. It might be interesting to investigate a piecewise approximation for the CH equation (with peakon data propagating). Since the peakon has a finite but discontinuous derivative, it might be possible to do a spectral approximation on either side of the discontinuity. This piecewise approximation can then be extend to the small dispersion limit CH equation (with peakon data propagating) and could act as a reference solution.

Another area that might be of interest is to examine the usefulness of applying spectral methods to various generalizations of the scalar equations studied in this thesis. For instance the Kadomtsev-Petviashvili equation, is thought of as a $2+1$ dimensional analog to the KdV equation. There are many Boussinesq approximations that are deemed important to the study of shallow water waves and solitary waves. Perhaps the most famous of these approximations is the "Boussinesq equation" (sometimes called the *good* or *bad* Boussinesq equation) [37].

Of course other hyperbolic systems of PDEs can be examined. Since RBFs can scale to high dimensions and are meshfree regardless of dimension or irregular geometry; RBF based approximations may provide interesting alternatives to the more popular finite difference,

finite volume, and finite element (including DG) methods.

RBF spectral methods applied to hyperbolic PDEs are relatively new and are currently being researched. There are many avenues to explore:

- How to select a RBF? (Infinitely smooth? Piecewise smooth? Compact support? Discontinuous? etc.)
- How to efficiently solve the RBF interpolating system (see equation 6.2)?
- How to select RBF centers? (Adaptive? Variable? etc.)
- How to select shape parameters? (Adaptive? Variable? etc.)
- How to incorporate adaptive mesh refinement?
- How to further improve RBF–FDs?
- How to guarantee eigenvalue stability in MOL? (Any techniques other than hyperviscosity?)
- How to scale to large problems? (Parallel computing? See [7] and [94])
- How to apply RBF collocation to hyperbolic systems? (There has been some success for the SW equations on the sphere [30])

BIBLIOGRAPHY

[1] A. Gelb and E. Tadmor. Enhanced spectral viscosity approximations for conservation laws. *Applied Numerical Mathematics 33*, pages 3–21, 2000.

[2] S. Abenda, T. Grava, and C. Klein. Numerical Solution of the Small Dispersion Limit of the Camassa-Holm and Whitham Equations and Multiscale Expansions. *SIAM J. Appl. Math.*, 70:2797–2821, 2010.

[3] K. Anastasiou and C. T. Chan. Solution of the 2D shallow water equations using the finite volume method on unstructured triangular meshes. *International Journal for Numerical Methods in Fluids*, 24(11):1225–1245, 1997.

[4] H. Ashi. Numerical methods for stiff systems. *PhD thesis, University of Nottingham.*, 2008.

[5] T. B. Benjamin, J. L. Bona, and J. J. Mahony. Model equations for long waves in nonlinear dispersive systems. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 272(1220):pp. 47–78, 1972.

[6] H. Berland, B. Skaflestad, and W. M. Wright. EXPINT—A MATLAB Package for Exponential Integrators. *ACM Trans. Math. Softw.*, 33(1), 2007.

[7] E. F. Bollig, N. Flyer, and G. Erlebacher. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *Journal of Computational Physics*, 231(21):7133–7151, 2012.

[8] J. L. Bona, M. Chen, and J.-C. Saut. Boussinesq Equations and Other Systems for Small-Amplitude Long Waves in Nonlinear Dispersive Media. I: Derivation and Linear Theory. *Journal of Nonlinear Science*, 12(4):283–318, 2002.

[9] J. L. Bona, M. Chen, and J.-C. Saut. Boussinesq equations and other systems for small-amplitude long waves in nonlinear dispersive media: II. The nonlinear theory. *Nonlinearity*, 17(3):925, 2004.

[10] P. Bonneton, F. Chazel, D. Lannes, F. Marche, and M. Tissier. A splitting approach for the fully nonlinear and weakly dispersive Green-Naghdi model. *Journal of Computational Physics*, 230:1479–1498, February 2011.

[11] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. 2000.

[12] R. Bronson. *Schaum's Outline of Modern Introductory Differential Equations: With Laplace Transforms, Numerical Methods, Matrix Methods [and] Eigenvalue Problems.* Schaum's outline series. McGraw-Hill, 1973.

[13] R. Camassa and D. D. Holm. An integrable shallow water equation with peaked solitons. *Phys. Rev. Lett.*, 71:1661–1664, Sep 1993.

[14] J. Certaine. The solution of ordinary differential equations with large time constants. *Mathematical methods for digital computers. Wiley*, pages 128–132, 1963.

[15] M. Chen. Exact solutions of various boussinesq systems. *Applied Mathematics Letters*, 11(5):45 – 49, 1998.

[16] M. E. Chenoweth. A local radial basis function method for the numerical solution of partial differential equations. *Master's thesis, Marshall University.*, 2012.

[17] İ. Dağ and Y. Dereli. Numerical solutions of KdV equation using radial basis functions. *Applied Mathematical Modelling*, 32(4):535–546, 2008.

[18] M. L. Dahlby. Geometric integration of nonlinear wave equations. *Master's thesis, Norwegian University of Science and Technology.*, 2007.

[19] C. Dawson and J. Proft. Discontinuous and coupled continuous/discontinuous galerkin methods for the shallow water equations. *Computer Methods in Applied Mechanics and Engineering*, 191(41):4721–4746, 2002.

[20] E. M. de Jager. *On the origin of the Kortewegde Vries equation.* arXiv:math/0602661v1 [math.HO]. 2006.

[21] M. Dehghan and A. Shokri. A numerical method for KdV equation using collocation and radial basis functions. *Nonlinear Dynamics*, 50(1-2):111–120, 2007.

[22] T. A. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3):413–422, 2002.

[23] T. A. Driscoll and A. R. H. Heryudono. Adaptive residual subsampling methods for radial basis function interpolation and collocation problems. *Computers & Mathematics with Applications*, 53(6):927–939, 2007.

[24] D. R. Durran. *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics.* J.E.Marsden and others. Springer, 1999.

[25] D. Dutykh, D. Clamond, P. Milewski, and D. Mitsotakis. Finite volume and pseudo-spectral schemes for the fully nonlinear 1D Serre equations. *ArXiv e-prints*, April 2011.

[26] D. Dutykh, Th. Katsaounis, and D. Mitsotakis. Finite volume methods for unidirectional dispersive wave models. *International Journal for Numerical Methods in Fluids*, 71(6):717–736, 2010.

[27] G. E. Fasshauer. *Meshfree Approximations Methods with MATLAB.* Interdisciplinary Mathematical Sciences. World Scientific, 2007.

[28] G. E. Fasshauer and J. G. Zhang. On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms*, 45(1-4):345–368, 2007.

[29] N. Flyer and B. Fornberg. Radial basis functions: Developments and applications to planetary scale flows.

[30] N. Flyer, E. Lehto, S. Blaise, G. B. Wright, and A. St-Cyr. A guide to RBF-generated finite differences for nonlinear transport: Shallow water simulations on a sphere. *Journal of Computational Physics*, 231(11):4078 – 4095, 2012.

[31] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1998.

[32] B. Fornberg and N. Flyer. Accuracy of radial basis function interpolation and derivative approximations on 1-D infinite grids. *Advances in Computational Mathematics*, 23(1-2):5–20, 2005.

[33] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2):869–892, 2011.

[34] B. Fornberg and C. Piret. On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. *Journal of Computational Physics*, 227(5):2758 – 2780, 2008.

[35] B. Fornberg, G. Wright, and E. Larsson. Some observations regarding interpolants in the limit of flat radial basis functions. *Computers & mathematics with applications*, 47(1):37–55, 2004.

[36] R. Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, DTIC Document, 1979.

[37] J. de Frutos, T. Ortega, and J. M. Sanz-Serna. Pseudospectral Method for the "Good" Boussinesq Equation. *Mathematics of Computation*, 57(195):pp. 109–122, 1991.

[38] T. W. Gamelin. *Complex Analysis.* Undergraduate Texts in Mathematics. U.S. Government Printing Office, 2001.

[39] T. Grava and C. Klein. Numerical solution of the small dispersion limit of Korteweg-de Vries and Witham equations. *Comm. Pure Appl. Math*, pages 1623–1664.

[40] T. Grava and C. Klein. Numerical study of a multiscale expansion of the Korteweg-de Vries equation. *Proc. Royal. Soc.*, 464:733–755, 2008.

[41] T. Grava and C. Klein. Numerical study of a multiscale expansion of the Korteweg-de Vries equation and Painleve-II equation. *Proc. R. Soc. A*, (464):733–757, 2008.

[42] X. Raynaud H. Kalisch. Convergence of a spectral projection of the Camassa-Holm equation. *Numerical Methods for Partial Differential Equations*, 22(5):1197–1215, 2006.

[43] R. Haberman. *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*. Pearson Prentice Hall, Pearson Education, Incorporated, 2004.

[44] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, 1971.

[45] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill Series in Computer Science. WCB/McGraw-Hill, 1997.

[46] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007.

[47] N. J. Higham. *Accuracy and Stability of Numerical Algorithms: Second Edition*. Society for Industrial and Applied Mathematics, 2002.

[48] H. Holden and X. Raynaud. A convergent numerical scheme for the Camassa-Holm equation based on multipeakons. *Discrete Contin. Dyn. Syst.*, 14:505–523, 2006.

[49] D. D. Holm, J. E. Marsden, and T. S. Ratiu. Euler-Poincaré models of ideal fluids with nonlinear dispersion. 1998.

[50] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[51] K. B. Howell. *Principles of Fourier Analysis*. Studies in Advanced Mathematics. Taylor & Francis, 2001.

[52] M. Yousuff Hussaini and Thomas A. Zang. Spectral Methods in Fluid Dynamics. *Institute for Computer Applications in Science and Engineering*, 1986.

[53] L. W. Johnson and R. D. Riess. *Numerical Analysis*. Addison-Wesley world student series. Addison-Wesley, 1977.

[54] H. Kalisch. Instability of solitary waves for a nonlinearly dispersive equation. *Discrete Contin. Dyn. Syst.*, 10:709–717, 2004.

[55] H. Kalisch and J. Lenells. Numerical study of traveling-wave solutions for the Camassa-Holm equation. *Chaos, Solitons and Fractals*, 25:287–298, 2005.

[56] E. J. Kansa. Multiquadricsa scattered data approximation scheme with applications to computational fluid-dynamicsi surface approximations and partial derivative estimates. *Computers & Mathematics with applications*, 19(8):127–145, 1990.

[57] E.J. Kansa and R. E. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Computers & Mathematics with Applications*, 24(12):99–120, 1992.

[58] J. Kim. Finite volume methods for Tsunamis genereated by submarine landslides. *PhD thesis, University of Washington.*, 2014.

[59] A.-K. Kassam. L. N. Trefethen. Fourth-Order Time-Stepping for Stiff PDEs. *SIAM Journal of Scientific Computing*, 26(4):1214–1233, 2002.

[60] R. J. LeVeque. On the interaction of nearly equal solitons in the KdV equation. *SIAM J. Appl. Math.*, 47(2):254–262, 1987.

[61] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[62] R. J. LeVeque, D. L. George, and M. J. Berger. Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, 20:211–289, 5 2011.

[63] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.

[64] R. D. Nair. Diffusion experiments with a global discontinuous Galerkin shallow-water model. *Monthly Weather Review 137.10*, 2009.

[65] C. D. Levermore P. D. Lax. The small dispersion limit of the Kortewegde Vries equation. I, II, III. *Pure Appl. Math.*, 36(3):253290, 1983.

[66] D. A. Pope. An exponential method of numerical integration of ordinary differential equations. *Communications of the ACM*, 6(8):491–493, 1963.

[67] X. Raynaud. On A Shallow Water Wave Equation. *PhD thesis, Norwegian University of Science and Technology.*, 2006.

[68] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2-3):193–210, 1999.

[69] P. C. Mathews. S. M. Cox. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–430455, 2002.

[70] S. A. Sarra. Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations. 2009.

[71] S. A. Sarra and D. Sturgill. A random variable shape parameter strategy for radial basis function approximation methods. *Engineering Analysis with Boundary Elements*, 33:1239–1245, 2009.

[72] W. E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Academic Press, 1991.

[73] A. Senthilkuma. BBM equation with non-constant coefficients. *Turk. J. Math.*, 37:652–664, 2013.

[74] T. Grava T. Claeys. The KdV hierarchy: universality and a Painleve transcendent.

[75] T. Grava T. Claeys. Universality of the break-up profile for the KdV equation in the small dispersion limit using the Riemann-Hilbert approach. *Comm. Math. Phys.*, 286(3):979–1009, 2009.

[76] T. Grava T. Claeys. Painleve II Asymptotics near the Leading Edge of the Oscillatory Zone for the Korteweg de Vries Equation in the Small-Dispersion Limit. *Comm. Pure and Appl. Math.*, 63:203– 232, 2010.

[77] T. Grava T. Claeys. Solitonic asymptotics for the Korteweg-de Vries equation in the small dispersion limit. *SIAM J. Math. Analysis*, 42:2132–2154, 2010.

[78] L. N. Trefethen. T. Schmelzer. Evaluating matrix functions for exponential integrators via Caratheodory-Frejer approximation and contour integrals. *Report Numer 06/20, Numerical Analysis Group, Oxford University Computing Laboratory, Oxford, UK*, 2006.

[79] E. Tadmor. Convergence of Spectral Methods for Nonlinear Conservation Laws. *SIAM J. Numer. Anal.*, 26:30–44, 1989.

[80] E. Tadmor. Super Viscosity and Spectral Approximations of Nonlinear Conservation Laws. *Numerical Methods for Fluid Dynamics IV, M. J. Baines and K. W. Morton, eds,*, pages 69–82, 1993.

[81] Manuscript Submitted To, Nate Bottman, and Bernard Deconinck. pp. xxx kdv cnoidal waves are spectrally stable.

[82] L. N. Trefethen. *Spectral Methods in MATLAB*. Software, Environments, and Tools. Society for Industrial and Applied Mathematics, 2000.

[83] L. N. Trefethen. *Approximation Theory and Approximation Practice*. Applied mathematics. Society for Industrial and Applied Mathematics, 2013.

[84] L. N. Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[85] L. N. Trefethen and J. A. C. Weideman. THE EXPONENTIALLY CONVERGENT TRAPEZOIDAL RULE. 2013.

[86] T. Trogdon, S. Olver, and B. Deconinck. Numerical inverse scattering for the Kortewegde Vries and modified Kortewegde Vries equations. *Physica D: Nonlinear Phenomena*, 241(11):1003 – 1025, 2012.

[87] M. Uddin, S. Haq, and M. Ishaq. RBF-Pseudospectral Method for the Numerical Solution of Good Boussinesq Equation. *Applied Mathematical Sciences*, 6(49):2403–2410, 2012.

[88] S. ul Islam, S. Haq, and A. Ali. A meshfree method for the numerical solution of the RLW equation.

[89] S. Venakides. The Zero Dispersion Limit of the Korteweg-de Vries Equation With Periodic Initial Data. *Transactions of the American Mathematical Society*, 301(1):pp. 189–226, 1987.

[90] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. NATO Asi Series. Series C, Mathematical and Physical Science. Springer, 1994.

[91] G. B. Whitham. Linear and nonlinear waves. *John Wiley, New York*, 1974.

[92] G. Wright and B. Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. pages 1391–1395, 2006.

[93] Y. Xu and C.-W. Shu. A local discontinuous Galerkin method for the Camassa-Holm equation. *SIAM Jornal on Numerical Analysis*, 46:1998–2021, 2008.

[94] Rio Yokota, LA Barba, and Matthew G Knepley. Petrbfa parallel¡ i¿ o¡/i¿(¡ i¿ n¡/i¿) algorithm for radial basis function interpolation with gaussians. *Computer Methods in Applied Mechanics and Engineering*, 199(25):1793–1804, 2010.

Appendix A

WHERE TO FIND THE FILES

Please see https://github.com/msfabien/ for the sample codes used in this thesis. For correspondence (bugs, improvements, comments, etc.) please email `fabien@rice.edu` or `msfabien@uw.edu`

Appendix B

AUXILIARY FILES

Here are some of the auxiliary files used in this thesis. Credit goes to Alfa R.H. Heryudono[1] for these code samples.

[1]Department of Mathematics at the University of Massachusetts Dartmouth

Code Listing B.0.1

Main RBF Pseudospectral file

```
% This function file provides the 1-D multiquadric radial basis function
% and its subsequent derivatives.  Derivatives higher than four will have
% to be included manually.
% INPUT VARIABLES:
% **x       , computational grid                     | vector of size (Nx1)
% **xc      , radial basis function centers          | vector of size (Mx1)
% **c       , radial basis function shape parameters| vector of size (Mx1)
% OUTPUT VARIABLES:
% **phi     , MQ-RBF interpolation matrix            | matrix of size (NxM)
% **phi1    , MQ-RBF first order evaluation matrix   | matrix of size (NxM)
% **phi2    , MQ-RBF second order evaluation matrix  | matrix of size (NxM)
% **phi#    , MQ-RBF # order evaluation matrix       | matrix of size (NxM)
function [phi,phi1,phi2,phi3,phi4] = mq(x,xc,c)

f = @(r,c) sqrt((c*r).^2 + 1);

r = x - xc;
phi = f(r,c); % phi(||x-xc||_2)

if nargout > 1
% 1-st derivative
 phi1 = (c^2)*r./phi;
    if nargout > 2
    % 2-nd derivative
    phi2 = (c^2)./(phi.^3);
        if nargout > 3
        % 3-rd derivative
        phi3 = -3*(c^4)*r./(phi.^5);
            if nargout > 4
            % 4-th derivative
            phi4 = 12*(c^4)*((c*r).^2-0.25)./(phi.^7);
            end
        end
    end
end
```

Code Listing B.0.2

Main RBF Pseudospectral file

```
% This function file provides the 1-D guassian radial basis function and
% its subsequent derivatives.  Derivatives higher than four
% will have to be included manually.
% INPUT VARIABLES:
% **x       , computational grid                    | vector of size (Nx1)
% **xc      , radial basis function centers         | vector of size (Mx1)
% **c       , radial basis function shape parameters| vector of size (Mx1)
% OUTPUT VARIABLES:
% **phi     , GA-RBF interpolation matrix           | matrix of size (NxM)
% **phi1    , GA-RBF first order evaluation matrix  | matrix of size (NxM)
% **phi2    , GA-RBF second order evaluation matrix | matrix of size (NxM)
% **phi#    , GA-RBF # order evaluation matrix      | matrix of size (NxM)
function [phi,phi1,phi2,phi3,phi4] = gau(x,xc,c)

f = @(r,c) exp(-(c*r).^2);

r = x - xc;
phi = f(r,c);  % phi(||x-xc||_2)
if nargout > 1
% 1-st derivative
        phi1 = -2*r*c^2.*exp(-(c*r).^2);
    if nargout > 2
    % 2-nd derivative
        phi2 = 2*c^2*exp(-c^2*r.^2).*(2*c^2*r.^2 - 1);
        if nargout > 3
        % 3-rd derivative
            phi3 = -4*c^4*r.*exp(-c^2*r.^2).*(2*c^2*r.^2 - 3);
            if nargout > 4
            % 4-th derivative
                phi4 = 4*c^4*exp(-c^2*r.^2).*(4*c^4*r.^4 - 12*c^2*r.^2 + 3);
            end
        end
    end
end
```

Code Listing B.0.3

Main RBF Pseudospectral file

```
% This function file provides the 1-D inverse multiquadric radial basis
% function and its subsequent derivatives.  Derivatives higher than four
% will have to be included manually.
% INPUT VARIABLES:
% **x       , computational grid                    | vector of size (Nx1)
% **xc      , radial basis function centers         | vector of size (Mx1)
% **c       , radial basis function shape parameters| vector of size (Mx1)
% OUTPUT VARIABLES:
% **phi     , IMQ-RBF interpolation matrix           | matrix of size (NxM)
% **phi1    , IMQ-RBF first order evaluation matrix  | matrix of size (NxM)
% **phi2    , IMQ-RBF second order evaluation matrix | matrix of size (NxM)
% **phi#    , IMQ-RBF # order evaluation matrix      | matrix of size (NxM)
function [phi,phi1,phi2,phi3,phi4] = imq(x,xc,c)

f = @(r,c) 1./((c*r).^2 + 1);

r = x - xc;
phi = f(r,c);

if nargout > 1
% 1-st derivative
phi1 = -2*r*c^2./((c*r).^2 + 1).^2;
    if nargout > 2
    % 2-nd derivative
        phi2 = (6*c^4*r.^2 - 2*c^2)./(c^2*r.^2 + 1).^3;
        if nargout > 3
        % 3-rd derivative
        phi3 = -(24*c^4*r.*(c^2*r.^2 - 1))./(c^2*r.^2 + 1).^4;
            if nargout > 4
            % 4-th derivative
            phi4 = 24*(5*c^8*r.^4 - 10*c^6*r.^2 + c^4)./(c^2*r.^2 + 1).^5;
            end
        end
    end
end
```

VITA

Maurice S. Fabien obtained his Bachelor's degree in mathematics from the University of Washington. He then obtained the degree of Master of Science from the Department of Applied Mathematics from The University of Washington in 2014. After this, he pursued a PhD from the department of Computational and Applied Mathematics at Rice University.