Air Force Institute of Technology
AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-23-2018

Radial Basis Function Generated Finite Differences for the Nonlinear Schrodinger Equation

Justin Ng

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Mathematics Commons, and the Special Functions Commons

Recommended Citation

Ng, Justin, "Radial Basis Function Generated Finite Differences for the Nonlinear Schrodinger Equation" (2018). *Theses and Dissertations*. 1740.
https://scholar.afit.edu/etd/1740

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact AFIT.ENWL.Repository@us.af.mil.

Radial Basis Function Generated Finite Differences for the Nonlinear Schrodinger Equation

THESIS

Justin Ng, 2d Lt, USAF

AFIT-ENC-MS-18-M-004

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENC-MS-18-M-004

RADIAL BASIS FUNCTION GENERATED FINITE DIFFERENCES FOR THE NONLINEAR SCHRODINGER EQUATION

THESIS

Presented to the Faculty
Department of Mathematics & Statistics
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Applied Mathematics

Justin Ng, B.S.
2d Lt, USAF

March 2018

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENC-MS-18-M-004

RADIAL BASIS FUNCTION GENERATED FINITE DIFFERENCES FOR THE NONLINEAR SCHRODINGER EQUATION

THESIS

Justin Ng, B.S.
2d Lt, USAF

Committee Membership:

Maj Jonah A. Reeger, Ph.D. (Chair)

Benjamin F. Akers, Ph.D. (Member)

William P. Baker, Ph.D. (Member)

AFIT-ENC-MS-18-M-004

Abstract

Solutions to the one-dimensional and two-dimensional nonlinear Schrodinger (NLS) equation are obtained numerically using methods based on radial basis functions (RBFs). Periodic boundary conditions are enforced with a non-periodic initial condition over varying domain sizes. The spatial structure of the solutions is represented using RBFs while several explicit and implicit iterative methods for solving ordinary differential equations (ODEs) are used in temporal discretization for the approximate solutions to the NLS equation. Splitting schemes, integration factors and hyperviscosity are used to stabilize the time-stepping schemes and are compared with one another in terms of computational efficiency and accuracy. This thesis shows that RBFs can be used to numerically solve the NLS with reasonable accuracy. Integration factors and splitting methods yield improvements in stability at the cost of computation time; both methods produce solutions of similar accuracy while splitting methods are slightly less expensive to implement than integration factors (computation times were of the same order of magnitude). The use of hyperviscosity can lead to an improvement in stability but can also lead to increased errors if the relevant parameters are not chosen carefully.

AFIT-ENC-MS-18-M-004

I would like to dedicate this work to one of my college roommates (name omitted due to Air Force policy). This thesis would not be complete if he went unmentioned. We went through the math program at the Academy together. It's only appropriate that we finish our graduate education in math together before our careers separate us.

Acknowledgements

I would like to express my sincere gratitude to my advisor *Maj. Jonah A. Reeger, Ph.D* for his patience, motivation, enthusiasm and immense knowledge. Writing this thesis would not have been possible without his continuous mentorship and guidance.

Justin Ng

Contents

Page

Abstract iv

Acknowledgements vi

List of Figures viii

List of Tables xvi

I. Introduction 1

- 1.1 Motivation 1
- 1.2 Radial Basis Functions 2
  - A Brief History 2
  - Interpolation 3
  - Augmentation of Polynomial Terms 6
  - Linear Operator Approximation 8
  - Motivating the Use of RBF-Generated Finite Differences 10
- 1.3 Theory of Ordinary Differential Equations 11
  - Runge-Kutta Methods 12
  - Stability and Stiffness 13
  - Handling Instability 15
- 1.4 Previous Work 17
- 1.5 Organization of Thesis 18

II. The One-Dimensional Case 19

- 2.1 Methodology 19
- 2.2 Hyperviscosity 21
- 2.3 Integrating Factors 25
- 2.4 Splitting Methods 29
- 2.5 Comparison of Methods and Observations 37

III. The Two-Dimensional Case 42

- 3.1 Methodology 42
- 3.2 Results 47

IV. Conclusion 55

- 4.1 Concluding Remarks 55
- 4.2 Future Considerations 55

Bibliography 57

List of Figures

Figure Page

1 Effects of a bandwidth reduction algorithm on the sparsity pattern of the RBF-FD Matrix using a 1000-node stencil and a k-d tree algorithm to find the nearest 25 neighbors. The reverse Cuthill-Mckee algorithm was used to permute the sparse RBF-FD matrix into band matrix form with smaller bandwidth 11

2 Stability regions of the Runge-Kutta methods of orders 2 and 4 in the complex plane. The region in red illustrates the values of λk for which RK4 is unstable; RK4 is stable for all values of λk within the green and cyan region. RK2 is only stable for λk contained within the cyan region. 14

3 The region of absolute stability for AB1-AM1. Values of λk for which the method is stable are contained within the stability domain in green. 15

4 Stability region of BDF2, an implicit method. BDF2 is stable for all values of λk outside the region colored in red. 16

5 (TOP) Solutions with and without hyperviscosity and the (BOTTOM) associated spectra of D are compared with one another to demonstrate the effects of hyperviscosity ($\beta = 4, \gamma = 1$) on the spectrum of a differential operator. (BOTTOM-RIGHT) Eigenvalues with positive real part are shifted over to the left half-plane to ensure stability. 200 nodes, 20 nearest neighbors and a polynomial order of 5 were used to compute the RBF-FD matrix over the domain $x \in [0, 8\pi]$ while RK4 with a step size of $5 \cdot 10^{-4}$ was used to propagate solutions in time 24

6 Log10 absolute error plot demonstrating the effects of hyperviscosity ($\beta = 2, \gamma = 1$) on numerical approximation errors; adding hyperviscosity increases numerical errors. 180 nodes, 12 nearest neighbors and a polynomial order of 9 were used to compute the RBF-FD matrix over the domain $x \in [0, 8\pi]$ while RK4 with a step size of $5 \cdot 10^{-3}$ was used to propagate solutions in time. Errors were plotted at $t = 17.1405$ 25

Figure Page

7 Comparison between solutions obtained using the integrating factor method with and without hyperviscosity ($\beta = 2, \gamma = 0.00075$). 200 nodes, 20 nearest neighbors and a polynomial order of 5 were used to compute the RBF-FD matrix over $x \in [0, 8\pi]$ in conjunction with RK4 with a step size of $5 \cdot 10^{-2}$ 27

8 Log10 convergence plot comparing numerical solutions at $t = 1$ with and without Integrating Factors using a first order ODE solver (AB1-AM1 Predictor Corrector Method) for 1D NLS. 140 nodes, 10 nearest neighbors and a polynomial order of 7 were used to compute the RBF-FD matrix over $x \in [0, 30]$ and hyperviscosity was computed with $\beta = 3$ and $\gamma = 0.001$ 28

9 Log10 convergence plot comparing numerical solutions at $t = 1$ with and without Integrating Factors using a fourth order ODE solver for 1D NLS. Accuracy of solutions without integrating factors is limited by the spatial discretization, i.e. the RBF-FD matrix. 600 nodes, 20 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $x \in [0, 50]$ and hyperviscosity was computed with $\beta = 3$ and $\gamma = 0.0001$ 30

10 1^{st} order splitting convergence plot comparing solutions with and without hyperviscosity ($\beta = 2, \gamma = 0.0001$) at $t = 1$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part 32

11 The maximum of $u(x, t)$ over x plotted versus t. Solutions were produced using a 1^{st} order splitting scheme with $k = 5 \cdot 10^{-3}$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part 33

Figure Page

12 The maximum of the absolute error over x plotted versus t. Solutions were produced using a 1^{st} order splitting scheme with $k = 5 \cdot 10^{-3}$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part 34

13 Strang splitting convergence plot at $t = 1$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 5$ and $\gamma = 0.1$. RK2 was used to solve the nonlinear part of the PDE and BDF-2 was used to solve the linear part 35

14 The maximum of $u(x,t)$ over x plotted versus t. Solutions were produced using Strang splitting, a 2^{nd} order splitting scheme, with $k = 0.1257$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. RK2 was used to solve the nonlinear part of the PDE and BDF2 was used to solve the linear part 36

15 $\log_{10}$ of maximum of the absolute error over x plotted versus t. Solutions were produced using Strang splitting, a 2^{nd} order splitting scheme, with $k = 0.1257$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. RK2 was used to solve the nonlinear part of the PDE and BDF2 was used to solve the linear part 37

Figure Page

16 Log10 convergence plots are shown comparing numerical solutions of the one-dimensional NLS equation at $t = 0.1$ obtained using several methods. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. (TOP) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. Hyperviscosity was computed with $\beta = 2$ and $\gamma = 0.03$. (BOTTOM) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. Hyperviscosity was computed with $\beta = 2$ and $\gamma = 0.0001$. 39

17 Timing results for several methods excluding hyperviscosity; the preprocessing costs of computing the RBF-FD matrix and its eigendecomposition were not accounted for. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. Each method has rate $O\left(k^{-1}\right)$ for computation cost. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. 40

Figure Page

18 Computation time versus the L_∞ error for several methods—hyperviscosity was not used. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. 41

19 Modulus of the initial condition with $T_x = 60$ and $T_y = 6$ 42

20 The vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are illustrated in the $v - w$ plane. They are the rotated variants of $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. 45

21 The vectors $\hat{\mathbf{p}}_1$ and $\hat{\mathbf{p}}_2$ are illustrated in the $v - w$ plane. They point to the two points p_1 and p_2, respectively, on a cross section of the torus centered at the origin in the v-w plane. 45

22 Nearest neighbors (circled in red) of a node (circled in green) on a torus with $R = 1$ and $r = 0.5$ using the custom angular distance metric defined by (26). Nearest neighbors from the torus are properly mapped back onto the two-dimensional grid 46

23 Nearest neighbors (circled in red) of a node (circled in green) on a torus with $R = 1$ and $r = 0.5$ using the default Euclidean distance. Since Euclidean distance is not preserved when points are conformally mapped from a two-dimensional grid onto a torus, nearest neighbors are incorrectly computed 46

Figure Page

24 Log10 convergence plots are shown comparing numerical solutions of the two-dimensional NLS equation at $t = 0.1$ obtained using several methods. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. (TOP) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. Hyperviscosity was computed with $\beta = 4$ and $\gamma = 0.01$. (BOTTOM) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. Hyperviscosity was computed with $\beta = 5$ and $\gamma = 0.1$. 49

25 Comparison between numerical solutions at $t = 0.1$ with and without Integrating Factors using a fourth order ODE solver (RK4) for 2D NLS. 1000 nodes were used (100 in x and 10 in y) with 72 nearest neighbors, polynomial order 7, $T_x = 50$ and $T_y = 5$. Hyperviscosity was computed with $\gamma = 0.05$ and $\beta = 3$ 50

26 Error between D applied to the initial condition and $\Delta u(x, y, 0)$; 1000 nodes were used (100 in x and 10 in y) with 72 nearest neighbors, polynomial order 7, $T_x = 50$ and $T_y = 5$. The accuracy of solutions using higher order ODE solvers is limited by the spatial discretization, i.e. the RBF-FD matrix D 51

27 (TOP) Error between D applied to the initial condition and $\Delta u(x, y, 0)$; 36000 nodes were used (600 in x and 60 in y) with 132 nearest neighbors, polynomial order 10, $T_x = 60$ and $T_y = 6$. (BOTTOM) Comparison between numerical solutions at $t = 0.1$ with Integrating Factors using RK4 for 2D NLS; accuracy is still limited by the spatial discretization despite the fact that 36000 nodes were used. Hyperviscosity was computed using $\beta = 4$ and $\gamma = 2.5$ 52

28 Timing results for several methods excluding hyperviscosity; the preprocessing costs of computing the RBF-FD matrix and its eigendecomposition were not accounted for. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. Each method has rate $O\left(k^{-1}\right)$ for computation cost. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. 53

Figure Page

29 Computation time versus the L_∞ error for several methods—hyperviscosity was not used. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. Solutions to the two-dimensional variant of the NLS equation were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. 54

List of Tables

Table Page

1 Examples of Radial Basis Functions $\phi(r)$ 5

RADIAL BASIS FUNCTION GENERATED FINITE DIFFERENCES FOR THE NONLINEAR SCHRODINGER EQUATION

I. Introduction

The work presented in this thesis documents the use of radial basis functions (RBFs) in solving partial differential equations (PDEs). In particular, RBF based numerical methods for solving the nonlinear Schrodinger equation (NLS) are considered. This chapter provides the relevant literature and background information required to carry out this research; in no way is this literature review intended to be exhaustive.

1.1 Motivation

The NLS equation arises in various physical contexts. It appears in the fields of nonlinear waves, electromagnetics and plasma physics where it represents a wave packet equation [1]. The equation of interest is of the form

$$i\frac{\partial u}{\partial t} = \triangle u + \kappa |u|^2 u, \quad i^2 = -1, \tag{1}$$

where u is a complex valued function and κ is a real-valued scalar. The application of primary concern is the propagation of a laser through a medium. Equation (1) can be used to model the propagation of a high energy laser (HEL) beam in a medium whose index of refraction depends on the wave amplitude in the form of the paraxial equation [1, 2, 3]. The paraxial equation is essentially (1), but with different scaling and with $|u|^2$ replaced by a different potential [3]. In the context of HEL, equation

(1) models the electromagnetic field, $u(\mathbf{x},t)$, with $t \in \mathbb{R}$ being the direction of the beam propagation and $\mathbf{x} \in \mathbb{R}^d$ the transverse direction [1]. The dimension, d, is taken to be 1 or 2 in this work and the Laplacian is only in the transverse directions.

Although analytical consideration of various aspects of the NLS equation has historically been of great interest (see, e.g. [2]), this will not be the primary focus of this thesis. Let $\mathbf{x} = [x_1 \; x_2 \; \ldots \; x_d]^T$. We will only be interested in periodic numerical solutions defined by the periodic boundary condition

$$u(x_1 + T_1, x_2, \ldots, x_d, t) = u(x_1, x_2 + T_2, \ldots, x_d, t) = \ldots = u(\mathbf{x}, t), \tag{2}$$

where T_i is the period of the i^{th} spatial coordinate, $i = 1, 2, \ldots, d$.

This thesis will produce numerical solutions by discretizing the spatial domain of equation (1) by means of RBF-generated finite differences and then using standard numerical techniques for solving ordinary differential equations (ODEs) to evolve the solution in t. Ultimately, the results of this thesis will be used to inform a laser modeling effort that couples the NLS equation to the Navier-Stokes equations, so a novel integrating factors method and split-step methods for solving equation (1), as well as numerical instabilities will be of great concern to improve computational efficiency. The remainder of this chapter will introduce RBFs and the motivation behind using them for solving equation (1).

1.2 Radial Basis Functions

A Brief History.

RBFs have been used to solve PDEs with high accuracy for $d = 1, 2, 3$ [4]. These methods can be easily extended to higher dimensions, which, in some cases, make them superior to traditional numerical methodologies for solving PDEs that arise in the physical sciences [4]. The RBF-based approach to solving PDEs originated from Hardy's work in 1971 when he proposed using RBFs for multi-dimensional interpolation with scattered data [5]. Numerical methods such as finite-difference and finite-element methods (FEM) rely on data points that are connected together by a topological map called a mesh. In such a mesh, the connection between each data point in the simulation domain is used to define differential operators to solve PDEs. RBFs do not require any information on the relationship between nodes to construct mathematical operators. Instead, they rely on the interaction of each node with all of the data points in the domain to establish a system of algebraic equations for the whole domain of the problem without the need for a predefined mesh, making them ideal for scattered nodes. The utility that RBFs provided in interpolating scattered data naturally led to meshfree methods for solving PDEs. In 1990, Kansa showed that RBFs were capable of providing accurate derivative approximations given known function values at scattered data locations, which opened up the door for using RBFs for approximating solutions to PDEs over meshless grids given their outstanding geometric flexibility and potential for spectral accuracy [6].

Interpolation.

In many areas of applied mathematics, it is necessary to determine an unknown function given only discrete samples of the function or equations governing the behavior of the unknown function (e.g. algebraic equations, ODEs or PDEs). To determine the unknown function, a set of basis functions (e.g. polynomials, trigonometric functions or Bessel functions) whose span forms a subspace of the set of continuous functions is chosen. The unknown function is then approximated as a linear combination of the basis functions, with the weights of the linear combination chosen to satisfy the governing equations, or so that the approximation matches the discrete samples exactly (i.e. interpolation).

Given a set of basis functions $\{F_j(\mathbf{x})\}_{j=1}^N$, the linear combination

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j F_j(\mathbf{x}) = c_1 F_1(\mathbf{x}) + \cdots + c_N F_N(\mathbf{x}), \tag{3}$$

is an interpolant of $f(\mathbf{x})$ if the unknown coefficients $\{c_j\}_{j=1}^N$ are chosen so that

$$s(\mathbf{x}_j) = f(\mathbf{x}_j) \tag{4}$$

at a set of N points $\{\mathbf{x}_j\}_{j=1}^N$. In general, equations (4) can be written as N linear equations for the N unknown weights. That is, constraining the interpolant to the data $f(\mathbf{x}_j)$ at location $\mathbf{x}_j$ for $j = 1, \ldots, N$, we obtain the following linear system

$$\begin{bmatrix} F_1(\mathbf{x}_1) & F_2(\mathbf{x}_1) & \cdots & F_N(\mathbf{x}_1) \\ F_1(\mathbf{x}_2) & F_2(\mathbf{x}_2) & \cdots & F_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ F_1(\mathbf{x}_N) & F_2(\mathbf{x}_N) & \cdots & F_N(\mathbf{x}_N) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix}.$$

In multidimensional interpolation, there are some considerations: is the data scattered or located on a grid? Can the basis set actually approximate the unknown continuous function?

A natural choice for the basis in one-dimension is polynomials. Because of this, when approximating a multivariate function (e.g. surfaces), one may consider the use of multivariate polynomials. However, interpolating data with location-independent basis functions, such as multivariate polynomials, is an ill-posed problem [7]. For multi-dimensional data sets, two nodes can be moved continuously along a closed path so that they end up interchanged. As a result, two rows are interchanged in the matrix of the linear system that arises from (3), i.e. the determinant of the coefficient matrix changed sign. Thus, by continuity, the determinant is zero at some point along the path. This is true for all closed paths connecting the two points that do not contain any of the nodes other than the two being interchanged, implying that singular systems can arise from a plethora of different node configurations. RBFs do not suffer from this issue since the definition of the basis function depends explicitly on the data [4]. RBFs can also be used on mesh-free data sets where data nodes may be scattered; they provide total geometric flexibility [4].

Table 1 lists several examples of RBFs. The non-negative parameter ϵ affects the shape of infinitely smooth RBFs and m is a natural number. The argument r of the RBFs is a radial argument that measures the distance from a given point, often called a center, under a given norm on $\mathbb{R}^d$. The most common choice is $r_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_j\|_2$ (with $\mathbf{x}_j$ being the center). When considering interpolation, the points $\mathbf{x}_j$ are the interpolation points. Hence, the interpolant becomes

Table 1. Examples of Radial Basis Functions $\phi(r)$

Piecewise Smooth RBFs	
Monomial (MN)	r^{2m+1}
Thin Plate Spline (TPS)	$r^{2m} \ln r$
Infinitely Smooth RBFs	
Gaussian (GA)	$e^{-(\epsilon r)^2}$
Multiquadric (MQ)	$\sqrt{1+(\epsilon r)^2}$
Inverse Multiquadric (IMQ)	$1/\sqrt{1+(\epsilon r)^2}$
Hyperbolic Secant (HS)	$\operatorname{sech}(\epsilon r)$

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|). \tag{5}$$

The weights c_j are determined by the N interpolation conditions

$$f(\mathbf{x}_i) = s(\mathbf{x}_i) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x}_i - \mathbf{x}_j\|), \quad i = 1, \dots, N,$$

which give rise to the following linear system

$$A\mathbf{c} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} = \mathbf{f}. \tag{6}$$

Typically, A is a symmetric $N \times N$ matrix and $\mathbf{c}$ can be found using row-reduction algorithms.

Augmentation of Polynomial Terms.

The coefficient vector $\mathbf{c}$ is guaranteed to exist and be unique if A is invertible. In the case of GA, MQ, IMQ and HS RBFs where $\epsilon > 0$, the matrix is non-singular [4]. However, interpolation using MN and TPS RBFs can often lead to a singular matrix. To remedy this, the basis set is augmented by polynomial terms and some extra constraints are introduced to guarantee a positive definite interpolation matrix, which is a much stronger condition than non-singularity [4]. Even when A is guaranteed to be non-singular, it is often still useful to modify (5) to, for instance, include polynomial terms to reduce Runge phenomenon-type boundary oscillations [8], for instance. In this case, (5) becomes

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{M} c_k^p \pi_k(\mathbf{x}) \tag{7}$$

with the constraints

$$\sum_{j=1}^{N} c_j \pi_k(\mathbf{x}_j) = 0, \quad \text{for} \quad k = 1, \ldots, M, \tag{8}$$

where M is the total number of additional polynomial terms and depends on the maximum order of the multivariate polynomials and $\{\pi_k\}_{k=1}^{M}$ is the set of polynomials in $\mathbb{R}^d$ up to degree m. The constraint (8) forces the weight vector to be orthogonal to the polynomial space, which guarantees non-singularity [4]. The inclusion of polynomials up to linear terms for $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{x} = [x\ y]^T$, yields the interpolant

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + c_1^p + c_2^p x + c_3^p y,$$

with the following constraints:

$$\sum_{j=1}^{N} c_j = 0, \quad \sum_{j=1}^{N} c_j x_j = 0, \quad \sum_{j=1}^{N} c_j y_j = 0.$$

In this two-dimensional example, $M = (m+1)(m+2)/2$ where m is the highest degree of the augmented bivariate polynomials. In general, (7) and (8) produces the linear system

$$\hat{A}\hat{\mathbf{c}} = \begin{bmatrix} A & P \\ P^T & 0_{M\times M} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{c}^p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0_{M\times 1} \end{bmatrix} = \hat{\mathbf{f}}, \tag{9}$$

where

$$P = \begin{bmatrix} \pi_1(\mathbf{x}_1) & \pi_2(\mathbf{x}_1) & \pi_3(\mathbf{x}_1) & \cdots & \pi_M(\mathbf{x}_1) \\ \pi_1(\mathbf{x}_2) & \pi_2(\mathbf{x}_2) & \pi_3(\mathbf{x}_2) & \cdots & \pi_M(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \pi_1(\mathbf{x}_N) & \pi_2(\mathbf{x}_N) & \pi_3(\mathbf{x}_N) & \cdots & \pi_M(\mathbf{x}_N) \end{bmatrix}, \quad \mathbf{c}^p = \begin{bmatrix} c_1^p \\ c_2^p \\ \vdots \\ c_M^p \end{bmatrix}.$$

The coefficients c_k^p can be interpreted as Lagrange multipliers that constrain the RBF coefficients to the space $P^T\mathbf{c} = \mathbf{0}$ so that (9) is guaranteed to have a unique solution [9]—this idea will be expanded upon in the following subsection. Including low-order polynomials, even just up to linear terms, can significantly improve the accuracy of the RBF interpolant at domain boundaries [4, 9]. Additionally, they can also improve the accuracy of derivative approximations [4].

Linear Operator Approximation.

Function approximation and interpolation is, again, useful when a set of equations governing an unknown function is given. These equations often involve linear operators whose actions on the basis set is taken to approximate the action on the unknown function. Consider the linear operator L and let

$$\mathbf{\Phi}^{(i)} = \left[L\phi(\|\mathbf{x}-\mathbf{x}_1\|)\big|_{\mathbf{x}=\mathbf{x}_i} \;\dots\; L\phi(\|\mathbf{x}-\mathbf{x}_N\|)\big|_{\mathbf{x}=\mathbf{x}_i} \;\dots\; L\pi_1(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i} \;\dots\; L\pi_M(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i} \right]^T$$

for each $i = 1, \dots, N$. Assuming that $\hat{A}$ is invertible and given that it is symmetric, applying L to (7) at location $\mathbf{x}_i$ gives

$$\begin{aligned} Lf(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i} \approx Ls(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i} &= \left[\sum_{j=1}^{N} c_j L\phi(\|\mathbf{x}-\mathbf{x}_j\|) + \sum_{k=1}^{M} c_k^p L\pi_k(\mathbf{x}) \right] \Bigg|_{\mathbf{x}=\mathbf{x}_i} \\ &= \hat{\mathbf{c}}^T \mathbf{\Phi}^{(i)} \\ &= \left(\hat{A}^{-1} \hat{\mathbf{f}} \right)^T \mathbf{\Phi}^{(i)} \\ &= \hat{\mathbf{f}}^T (\hat{A}^T)^{-1} \mathbf{\Phi}^{(i)} \\ &= \hat{\mathbf{f}}^T \hat{A}^{-1} \mathbf{\Phi}^{(i)} \\ &= \hat{\mathbf{f}}^T \mathbf{w}^{(i)}, \end{aligned} \tag{10}$$

where

$$\mathbf{w}^{(i)} = [w_1 \;\dots\; w_N \; w_{N+1} \;\dots w_{N+M}]^T$$

and

$$\hat{A}\mathbf{w}^{(i)} = \mathbf{\Phi}^{(i)}. \tag{11}$$

Hence, as long as $\hat{A}$ is invertible, computing the approximation (10) at $\mathbf{x} = \mathbf{x}_i$ reduces to solving the linear system (11). This means that an approximation to $Lf(\mathbf{x})$ can be found without computing the interpolation coefficients. In the case of a local operation like a derivative, the weights w_j, $j = 1, \ldots, N$, may be used to approximate the linear operator L at location $\mathbf{x}_i$ or $Ls(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}$. Notice that the $N+1, N+2, \ldots, N+M$ entries of $\hat{\mathbf{f}}$ are zero so $Lf(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i} = \hat{\mathbf{f}}^T\mathbf{w}^{(i)} = \sum_{j=1}^{N} f(x_j)w_j^{(i)}$ and we find that only the weights $w_1, \ldots, w_N$ should be used; $w_{N+1}, \ldots, w_{N+M}$ are "dummy" entries [4]. See Section 5.1.4 of [4] for a discussion on why only the weights $w_1, \ldots, w_N$ are used. All in all, approximating L at each node requires solving (11) for each $i = 1, \ldots, N$ to obtain $L \approx [\mathbf{w}^{(1)}\ \mathbf{w}^{(2)}\ \cdots\ \mathbf{w}^{(N)}]^T$.

In [9], it is noted that the weight vector $\mathbf{w}^{(i)}$ can be viewed as the solution to the constrained linear least-squares problem

$$\min_{\mathbf{w}^{(i)}} \left\| \frac{1}{2}\left(\mathbf{w}^{(i)}\right)^T \hat{A}\mathbf{w}^{(i)} - \mathbf{w}^{(i)}\mathbf{\Phi}^{(i)} \right\| \quad \text{subject to} \quad P^T\mathbf{w}^{(i)} = \mathbf{p}^{(i)}, \tag{12}$$

where $\mathbf{p}^{(i)} = \left[L\pi_1(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i}\ L\pi_2(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i}\ \ldots L\pi_M(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_i}\right]^T$. The constraint here can be thought of as enforcing exactness for polynomials up to degree m of the approximate operator. Treating the polynomial coefficients (i.e. $\mathbf{c}^p$) as Lagrange multipliers, (12) can be solved by finding the minimum with respect to $\mathbf{w}^{(i)}$ and $\mathbf{c}^p$ of the Lagrangian

$$\mathcal{L}(\mathbf{w}^{(i)}, \mathbf{c}^p) = \frac{1}{2}\left(\mathbf{w}^{(i)}\right)^T \hat{A}\mathbf{w}^{(i)} - \mathbf{w}^{(i)}\mathbf{\Phi}^{(i)} + \left(\mathbf{c}^p\right)^T\left(P^T\mathbf{w}^{(i)} - \mathbf{p}^{(i)}\right). \tag{13}$$

The Lagrangian (13) is convex and has a unique minimum [9], so it follows that $\nabla_{\mathbf{w}^{(i)},\mathbf{c}^p}\mathcal{L}(\mathbf{w}^{(i)}, \mathbf{c}^p) = 0$ is equivalent to the system (11).

Motivating the Use of RBF-Generated Finite Differences.

Since calculating the weights for each location costs $O((N+M)^3)$ operations using traditional row-reduction algorithms, approximating L globally costs $O(N(N+M)^3)$ operations. This cost can be reduced if the LU decomposition of $\hat{A}$ is obtained prior to computing the weights. In this case, the preprocessing cost of decomposing $\hat{A}$ into LU is $O(N+M)^3$. Computing the weights for each $\mathbf{x}_i$ reduces to solving $L\mathbf{y} = \mathbf{\Phi}^{(i)}$ for $\mathbf{y}$ by forward substitution, which costs $O(N+M)^2$ operations, and then solving $U\mathbf{w}^{(i)} = \mathbf{y}$ for $\mathbf{w}^{(i)}$ by back substitution, which costs another $O(N+M)^2$ operations. Hence, computing the weights for each node only costs $O((N+M)^2)$ operations. Approximating L globally reduces to $O(N(N+M)^2)$ operations with a preprocessing cost of $O(N+M)^3$.

Computational costs can be reduced by means of RBF-generated finite differences (RBF-FD). RBF-FD generates approximate linear operators in a similar manner to finite differences, that is, by considering piecewise instead of global interpolation. Given a point $\mathbf{x}_i$, L is instead approximated for each node locally by considering an interpolant of $f(\mathbf{x})$ over the set containing $\mathbf{x}_i$ and its $n-1$ nearest neighbors. Conceptually, with N nodes across the entire domain, separate domains surrounding each node composed of its $n-1$ nearest neighbors are overlapped. For practical purposes, it is assumed that $n \ll N$. In general, a k-d tree algorithm (*knnsearch* in MATLAB's statistical toolbox) is used to find the $n-1$ nearest neighbors of a node when calculating the weights for L. Solving (11) for each $\mathbf{x}_i$ in this case produces one row of the approximate operator, so there are N linear systems of size $(n+M) \times (n+M)$ to solve. Thus, the total cost of approximating L using this local method is $O(N(n+M)^3)$, which is a significant improvement over $O(N(N+M)^2)$ when N is large and n is reasonably chosen. In general, for a fixed level of accuracy, a global approximation will typically require fewer nodes than an RBF-FD approximation, but approximations at each node will require more computation time [4]. It is worth mentioning that the differences between an RBF-FD approach and a global RBF approach for approximating derivatives is comparable to those between finite differences and Fourier-based pseudospectral approximations [4].

The resulting RBF-FD matrix that approximates L is quite sparse since each row contains only n nonzero entries. Further improvements in computational efficiency can be achieved by applying bandwidth reduction algorithms (*symrcm* or *symamd* in MATLAB) [4]. The result of the reverse Cuthill-Mckee algorithm (*symrcm* in MATLAB) applied to an RBF-FD matrix is shown in figure 1. This algorithm permutes sparse matrices so that nonzero elements are closer to the diagonal.

1.3 Theory of Ordinary Differential Equations

For a time-dependent PDE such as equation (1), once the spatial domain is properly discretized, solutions may be obtained by treating the semi-discrete PDE as an ODE. Only first-order ODEs will be discussed in this thesis. First-order ODEs of the

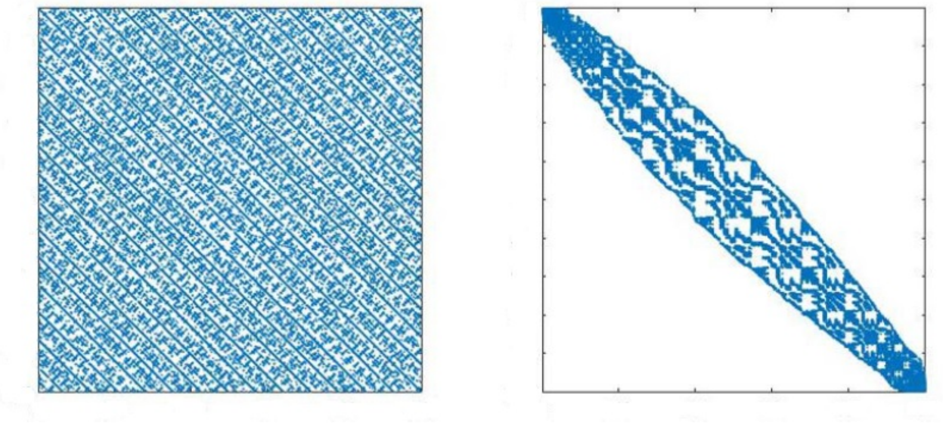


Figure 1. Effects of a bandwidth reduction algorithm on the sparsity pattern of the RBF-FD Matrix using a 1000-node stencil and a k-d tree algorithm to find the nearest 25 neighbors. The reverse Cuthill-Mckee algorithm was used to permute the sparse RBF-FD matrix into band matrix form with smaller bandwidth

form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = g(t, u),$$

may be solved with numerical schemes if the initial condition $(t_0, u(t_0))$ is explicitly given. Numerical schemes or ODE solvers give approximate solutions at distinct values of t. The numerical solution at a mesh point, t_j, is denoted $\tilde{u}_j$ and is different from the exact solution $u(t_j)$.

Runge-Kutta Methods.

Runge-Kutta methods are a family of single-step ODE solvers that have the form

$$\tilde{u}_{j+1} = \tilde{u}_j + k\xi,$$

where k is the temporal step size or the distance between two general mesh points, i.e. $k = t_{j+1} - t_j$, and

$$\xi = a_1 h_1 + a_2 h_2 + \cdots + a_p h_p$$

where h_i are slope estimates, a_i are the associated weights of the slope estimates and p is the number of estimates used [10]. The most widely used Runge-Kutta method, also known as fourth-order Runge-Kutta or simply RK4, uses $p = 4$ with

$$\begin{aligned} h_1 &= g(t_n, \tilde{u}_j), \\ h_2 &= g\left(t_j + \frac{k}{2}, \tilde{u}_j + \frac{k}{2} h_1\right), \\ h_3 &= g\left(t_j + \frac{k}{2}, \tilde{u}_j + \frac{k}{2} h_2\right), \\ h_4 &= g\left(t_j + k, \tilde{u}_j + k h_3\right), \\ \xi &= \frac{1}{6}\left(h_1 + 2h_2 + 2h_3 + h_4\right). \end{aligned}$$

In general, a Runge-Kutta method becomes increasingly accurate as more slope estimates are used, and larger step sizes can be used to maintain a given accuracy. However, the number of function evaluations also increases with p. Also, the numerical scheme is not guaranteed to behave in a controlled fashion. Hence, the behavior of the numerical solution for a fixed value $t > 0$ as $k \to 0$ must be considered.

Stability and Stiffness.

When solving an ODE (e.g. a semi-discrete initial boundary value problem), it is useful to know if a numerical scheme exhibits stability. A method is defined to be stable if slight changes in the initial data produces slight changes in the numerical solution [10]. Further, a numerical method is said to be absolutely stable if the slight perturbations in the numerical solution are bounded and shrink over time [10]. Thus, a numerical method that is absolutely stable is extremely desirable.

Suppose that a given numerical method, with step-size k, is used to solve the linear ODE

$$\frac{dy}{dt} = \lambda y, \quad t \geq 0, \quad y(0) = y_0,$$

where λ may be complex. Allowing λ to be complex comes from the fact that in practice, we are usually solving systems of ODEs. The region of absolutely stability of the numerical method is the region in the complex plane, D, such that if $\lambda k \in D$, then $y(t) \to 0$ as $t \to \infty$ for all initial values y_0. Figure 2 illustrates the stability regions for the Runge-Kutta methods of orders 2 and 4. These two explicit methods will be used throughout this work to solve the NLS equation.

When considering dispersive equations like the NLS equation, it is important to discern how far along the imaginary axis the region of absolute stability extends to encompass the purely imaginary eigenvalues of the dispersive operator [11]. The step size k is chosen so that the spectrum of a differential operator is scaled within

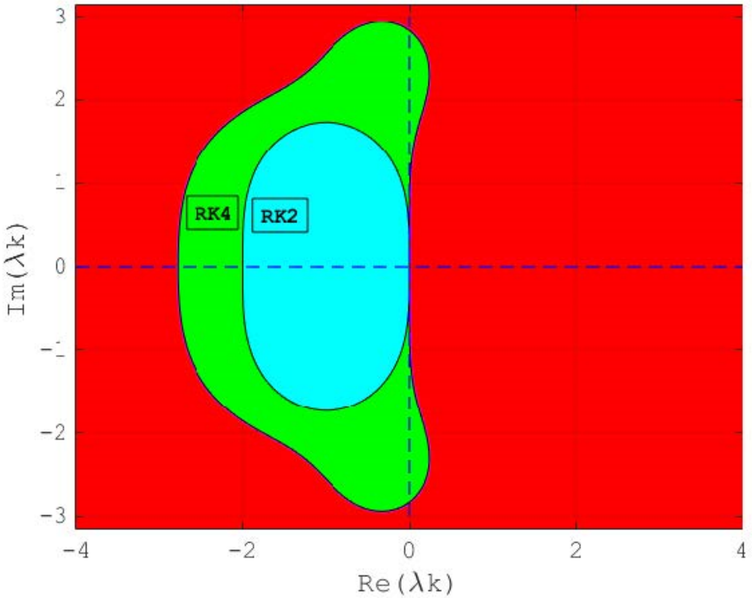


Figure 2. Stability regions of the Runge-Kutta methods of orders 2 and 4 in the complex plane. The region in red illustrates the values of λk for which RK4 is unstable; RK4 is stable for all values of λk within the green and cyan region. RK2 is only stable for λk contained within the cyan region.

the stability domain. Ideally, an ODE solver's stability region would encompass the entire left-half plane, i.e. $\text{Re}(\lambda) \leq 0$ so that numerical solutions do not grow in time. In this case, k can be chosen based on accuracy considerations, rather than on stability considerations. In [11], it was shown that the region of absolute stability of the Adams-Bashforth (AB) and Adams-Moulton (AM) predictor-corrector method of order 1 extends along the imaginary axis, making it useful for approximating solutions to first-order differential equations containing dispersive terms—this first order multistep method (AB1-AM1) will be used throughout this work. The stability region for AB1-AM1 is shown in figure 3 and is depicted in green. The stability domain in green contains a portion of the imaginary axis, making it viable for solving the NLS equation.

The stability domain of a numerical scheme ultimately sheds light on how large of a step size one can take and also gives insight on how to modify a problem to manipulate its eigenvalues appropriately. This can lead to improvements in computational

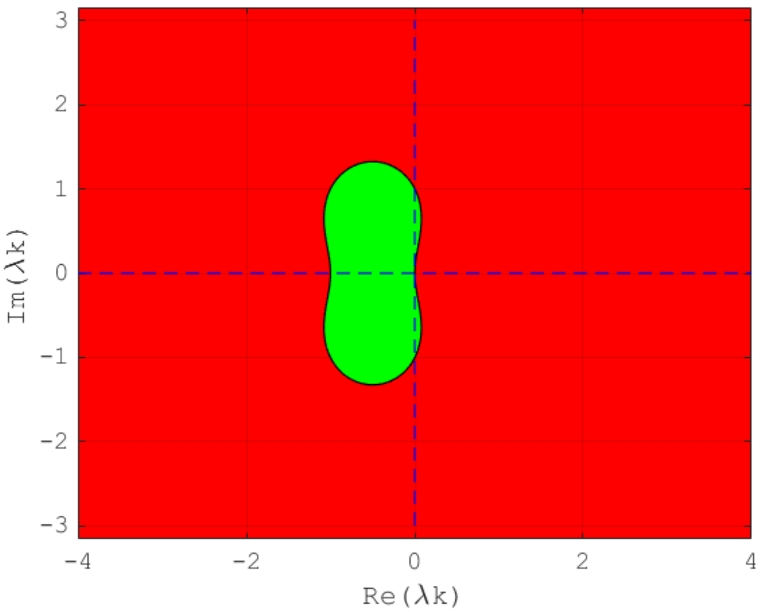


Figure 3. The region of absolute stability for AB1-AM1. Values of λk for which the method is stable are contained within the stability domain in green.

efficiency. Hence, the region of absolute stability of a numerical method for solving ODEs is one of the most important factors determining its performance.

Further complicating the selection of ODE solvers is the notion of stiffness. A stiff differential equation can be defined as a differential equation for which certain numerical methods for solving the equation are unstable, that is, the problem has poorly scattered eigenvalues, i.e. an unfeasible spectrum [10]. In this case, the step size can be taken to be extremely small, but this is not always practical or ideal. Equation (1) is stiff, which we will see in the next section. For this reason, stability requirements, rather than those of accuracy, will constrain our time steps.

Handling Instability.

It is important to know that the stability regions of implicit numerical schemes used for solving ODEs typically encompass a larger portion of the left-half plane compared to explicit schemes. Hence, implicit methods have better stability properties than explicit methods. An implicit method that will be used throughout this work is the backward differentiation formula (BDF) of order 2, or BDF2. The stability region for BDF2 is shown in figure 4 and is depicted in green. The stability region of BDF2 encompasses the entire left-half plane, making it suitable for producing numerical solutions to stiff equations. Moreover, the method will not exhibit instabilities that are often seen when explicit methods are used with larger step sizes.

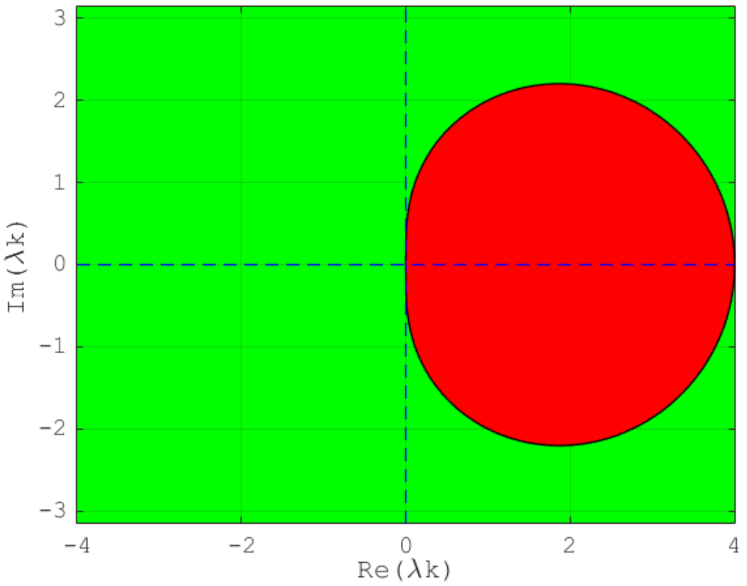


Figure 4. Stability region of BDF2, an implicit method. BDF2 is stable for all values of λk outside the region colored in red.

Although implicit methods allow for larger time steps and better stability, they can potentially be more computationally demanding since they often generate root finding problems or require other sophisticated algorithms [10]. On the other hand, explicit methods, albeit less stable, produce numerical solutions in terms of known quantities at previous time steps which makes them less computationally expensive. Operator splitting, or the fractional step method, is a popular method used to deal with ODEs that are stiff by combining the stability properties of implicit methods with the computational simplicity of explicit methods. In splitting, an ODE is treated as an equation with a stiff linear term coupled with a nonlinear term. Conceptually, instead of employing a single explicit or implicit method to solve an ODE, an implicit method is applied to the stiff term and an explicit method is applied to the non-stiff term. A popular splitting method, known as Strang splitting, gives approximations that are of second-order accuracy [12]. More attention will be given to splitting methods in chapter II.

Another technique used to stabilize time-stepping schemes involves adding or subtracting a constant multiple of a high-order Laplacian to the right-hand side of a semi-discrete PDE, also known as a hyperviscosity filter [4]. For purely dispersive problems, this approach controls stiffness or the scatter of eigenvalues by shifting the eigenvalues of the differential operator from the right-half plane to the left-half plane while keeping them reasonably close to the imaginary axis [13, 14]. Hyperviscosity will be used to handle the stiffness of the differential operator associated with equation (1) in chapters II and III.

1.4 Previous Work

Previous treatments of the NLS include Fourier collocation methods [15, 16] and finite-difference methods [15, 17]. The split-step method was used to discretize the time variable for the numerical solution of the NLS in [15]. In [18], a Fourier collocation method was used to discretize the spatial domain of the two-dimensional NLS equation and integrating factors were used for time integration to reduce computational costs without a significant reduction in accuracy. The NLS equation was solved numerically using radial basis functions in [19] to compare the errors associated with different RBFs; given an optimal shape parameter, it was found that GA RBFs yielded smaller errors than MQ and IMQ RBFs. In [20], a variant of the NLS equation was solved using three Fourier-based methods to compare the errors associated with each method. Duo and Zhang showed that the split-step Fourier method was more accurate for studying the long-term behaviors of solutions when compared to the Crank-Nicolson Fourier method and the relaxation Fourier method and has a lower computational cost than the two other methods [20].

1.5 Organization of Thesis

The goal of this thesis is to investigate computationally efficient uses of RBFs for solving (1). Chapter II presents an analysis of numerical solutions to the one-dimensional variant of (1) produced by RBF-FD methods in terms of accuracy and stability. Both explicit and implicit schemes are considered to propagate solutions in time. Integrating factors and hyperviscosity are also considered to stabilize such time-stepping schemes. Solutions to the two-dimensional variant of (1) are then analyzed in chapter III, which will be produced using methods similar to those introduced in chapter II.

II. The One-Dimensional Case

In this section, the accuracy and stability of several methods used to numerically solve the one-dimensional variant of (1) where $\mathbf{x} \in \Omega \subset \mathbb{R}^1$ are analyzed. Thus, (1) reduces to

$$\frac{\partial}{\partial t}u(x,t) = -i\left(\frac{\partial^2}{\partial x^2}u(x,t) + \kappa|u(x,t)|^2 u(x,t)\right), \tag{14}$$

where u depends only on x and t, and the periodic boundary condition defined by (2) is enforced. The use of integrating factors and hyperviscosity to remedy the stiffness of the equation is also discussed.

2.1 Methodology

As mentioned in chapter I, differential operators can be approximated with RBFs either locally or globally and both cases will require N nodes across the full domain. We define a stencil to be the geometric arrangement of points that is used to compute a numerical approximation at a particular node, e.g. the collection of the node and its $n-1$ nearest neighbors. Global approximation requires that the total number of weights calculated within the global stencil of N nodes be equal to N. This requires solving (6), a problem with a large, full matrix, to calculate all the weights for the operator. Local approximation of the operator using RBF-FD becomes much more attractive when the the number of nodes n in each local stencil (i.e. the set containing a node and its $n-1$ nearest neighbors) is much less than N. For this reason, only RBF-FD will be used to solve (1) by approximating the term containing the Laplacian operator—we will call this finite-dimensional operator D from here on out. With $\kappa = 2$, this approach transforms (14) into the following system of ODEs:

$$\frac{\partial}{\partial t}\mathbf{u}(t) = -i\left(D\mathbf{u}(t) + 2\begin{bmatrix} |u(\mathbf{x}_1,t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2,t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N,t)|^2 \end{bmatrix}\mathbf{u}(t)\right), \qquad (15)$$

where

$$\mathbf{u}(t) = \begin{bmatrix} u(\mathbf{x}_1,t) \\ u(\mathbf{x}_2,t) \\ \vdots \\ u(\mathbf{x}_N,t) \end{bmatrix} \qquad (16)$$

given the N nodes $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \Omega$ used to construct D.

The RBF-FD matrix D is found by first discretizing the domain Ω into N nodes. Starting with the first node $\mathbf{x}_1$, a local stencil Ω_1 is created containing $\mathbf{x}_1$ and its $n-1$ nearest neighbors. The operator L in equation (10) is replaced with the Laplacian operator Δ (in this chapter, $\Delta = \frac{\partial^2}{\partial x^2}$) and the system (11) is then solved for each element within Ω_1 to produce the weights for an approximation of $\Delta u(\mathbf{x},t)$ over Ω_1. The weights used for the approximation of $\Delta u(\mathbf{x},t)\big|_{\mathbf{x}=\mathbf{x}_1}$ within the local stencil are stored in the first row of D; the weights must be stored so that the column indices of the weights within D agree with the row indices of $u(\mathbf{x},t)$ evaluated at $\mathbf{x}_1$ and its $n-1$ nearest neighbors in the vector (16). For instance, if $\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \mathbf{x}_{k_3}, \ldots, \mathbf{x}_{k_{(n-1)}}$ are the nearest neighbors of $\mathbf{x}_k$, then row k of D has zeros except for entries $k_1, k_2, \ldots, k_n$, where $w_1, w_2, \ldots, w_n$ are placed instead. This process is repeated until weights are computed and stored within D for every node in Ω.

With the initial condition $u(x,0) = \operatorname{sech}(x - \frac{T}{2})\exp\left(i(x - \frac{T}{2})/2\right)$, (14) has the solitary wave solution [19, 21]

$$u(x,t) = \operatorname{sech}\left(\left(x - \frac{T}{2}\right) + t\right) \mathrm{e}^{i\left(\frac{1}{2}\left(x-\frac{T}{2}\right)-\frac{3}{4}t\right)}. \tag{17}$$

Numerical solutions with and without hyperviscosity using $T_1 = 8\pi$ will be compared to (17).

All work in this section will be carried out using GA RBFs. The choice of the shape parameter ϵ is highly dependent on N and n [4]. In order to account for this, ϵ is chosen such that condition number of the matrix in (11), $\hat{A}$, is kept below 10^8 as N increases. This is achieved by choosing a small initial value for ϵ and increasing ϵ until the condition number is within the tolerance of 10^8.

2.2 Hyperviscosity

Since (1) is a dispersion-type PDE, we should expect a purely imaginary spectrum [4]. It would be ideal to step the equation in time with a classical numerical method for ODEs whose stability region extends far along the imaginary axis. However, using RBF-FD-based spatial discretization causes the eigenvalues of D to scatter into the right half of the complex plane, producing solutions that grow unbounded with time even if extremely small step sizes are used [13]. The scattering of these eigenvalues gets worse as N increases [13]. As N increases, the eigenvalues for which $\mathrm{Re}(\lambda) > 0$ causes exponential growth in the solution. This issue is compounded when the eigenvalues are not sufficiently close to the imaginary axis [4]. If D has eigenvalues that are not sufficiently close to the imaginary axis and/or have large positive real parts, a hyperviscosity filter can be applied to minimize the exponential growth of solutions.

A method that is becoming increasingly common to stabilize RBF-FD for dispersive PDEs adds or subtracts a constant multiple of a high-order Laplacian to the right hand side of (1) [4, 13]. Thus, the stabilized variant of (1) is

$$\frac{\partial u}{\partial t} = -i\left(\Delta u + 2|u|^2 u\right) + (-1)^{\beta+1}\gamma N^{-\beta}\Delta^{\beta} u = -i\left(\Delta u + 2|u|^2 u\right) + Hu,$$

where $\beta \in \mathbb{N}\backslash\{1\}$ is the hyperviscosity order and γ is a scaling parameter [13]. Since the stencil used to approximate the hyperviscosity term is the same stencil used to produce D, there is little additional cost per time step [14]. With β and γ chosen appropriately, the inclusion of hyperviscosity leaves the physically relevant eigenvalues of D largely intact but shifts the ones on the right half of the complex plane to the opposite half [13]. Although polynomial terms are used to compute D, they will not be used to compute H in this thesis.

It is worth mentioning that there is a strategy for choosing an appropriate value for γ. If γ is too large, there is a risk of adding too much hyperviscosity which results in eigenvalues that end up far out in the left half-plane, which potentially decreases the accuracy of the numerical time-stepping scheme and forces k to be small [14]. If the effects of the hyperviscosity filter are too minimal, i.e. γ is too small, there is the possibility that eigenvalues will still exist on the right half-plane. It was found experimentally in [13, 22] that γ ranging from $O(1)$ to $O(10^{-2})$ is reasonable enough to stabilize PDEs with dispersive operators.

There is also a strategy for choosing an appropriate value for β. The purpose of hyperviscosity is to suppress highly oscillatory modes and physically irrelevant modes [22]. As the stencil size increases, the accuracy of the RBF-FD method is expected to increase. Hence, as n or N increases, more physical modes of the problem are being represented with greater accuracy. As the hyperviscosity order β increases, more physical modes are preserved [4], so the order of the hyperviscosity filter should increase as the stencil size gets larger (see [14] for an illustration); β should typically be in the range of 2 to 10, but could be higher [13].

The process of computing the hyperviscosity operator can be simplified if GA or MN RBFs are used. Regular Laguerre polynomials, $p_\beta(r)$, can be used to fill in the right hand side vector of (11) using the relation [4]

$$\Delta^\beta \phi(r) = \epsilon^{2\beta} p_\beta(r)\phi(r)$$

if GA RBFs ($\phi(r) = \mathrm{e}^{-(\epsilon r)^2}$) are used to compute the hyperviscosity filter in a two-dimensional spatial domain (for d dimensions, generalized Laguerre polynomials are used instead [4]). This approach to computing powers of the Laplacian does not require computing derivatives or the use of the binomial theorem. This makes GA RBFs an attractive choice since they can be used to compute hyperviscosity of any order with relative ease. For MN RBFs (i.e. $\phi(r) = r^m$ with m odd), the relation [4]

$$\Delta\phi(r) = m(m + d - 2)r^{m-2} \tag{18}$$

can be used to compute a hyperviscosity operator of order $\beta = 1$ if $m \geq 3$ in dimension d; equation (18) can be applied repeatedly to obtain higher orders of Δ.

To demonstrate how hyperviscosity can be used to stabilize certain numerical schemes, (15) was solved numerically using an explicit ODE solver. The one-dimensional Laplacian, $\frac{\partial^2}{\partial x^2}$, and the hyperviscosity filter was approximated by means of RBF-FD with $N = 200, n = 20$ and the maximum polynomial order set to 5. Once the spatial domain was discretized with equally spaced nodes, the fourth order Runge-Kutta (RK4) method was used to solve the equation in time with a time step of $k = 5 \cdot 10^{-4}$. Figure 5 summarizes the results. In figure 5, the eigenvalues of D with positive real part seen in the bottom-left subplot are contributing to the exponential growth of the solution observed in the top-left subplot. In the bottom-right subplot, where a hyperviscosity filter is used, eigenvalues with positive real part are shifted to the left-half plane which stabilizes the time-stepping scheme as shown by the top-right subplot.

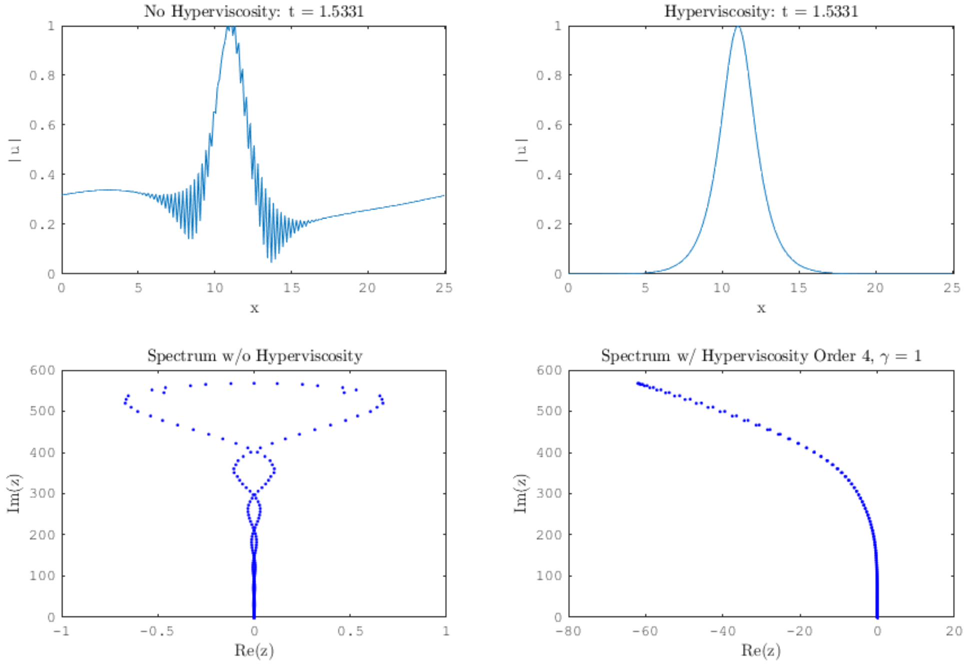
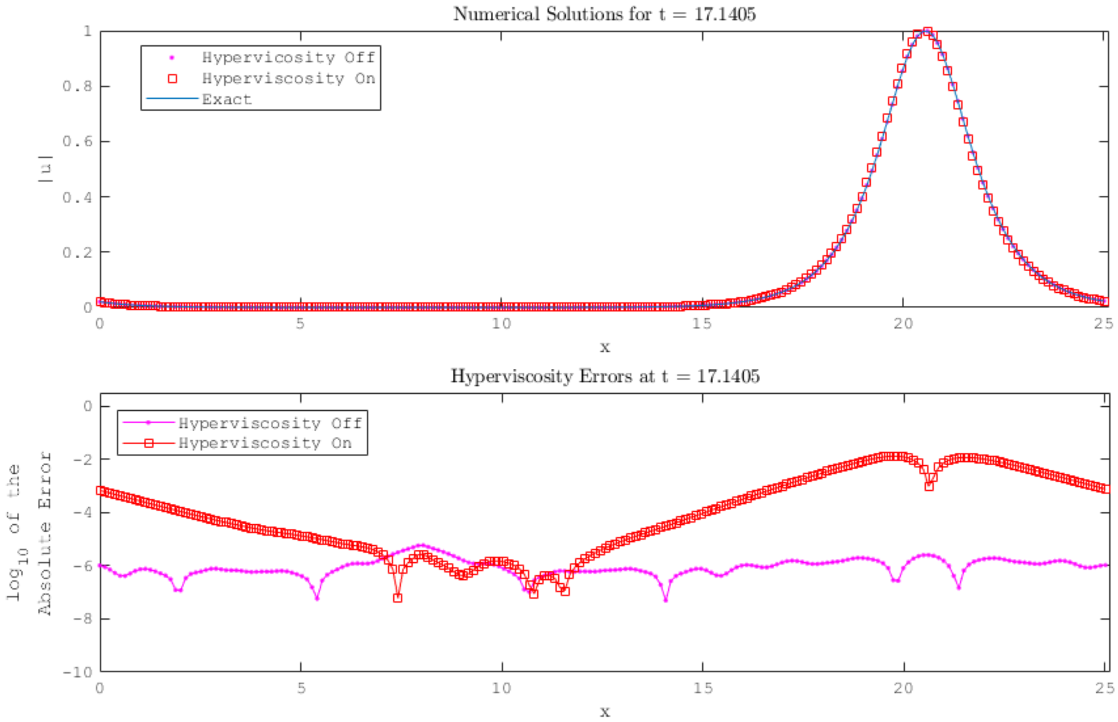


Figure 5. (TOP) Solutions with and without hyperviscosity and the (BOTTOM) associated spectra of D are compared with one another to demonstrate the effects of hyperviscosity ($\beta = 4, \gamma = 1$) on the spectrum of a differential operator. (BOTTOM-RIGHT) Eigenvalues with positive real part are shifted over to the left half-plane to ensure stability. 200 nodes, 20 nearest neighbors and a polynomial order of 5 were used to compute the RBF-FD matrix over the domain $x \in [0, 8\pi]$ while RK4 with a step size of $5 \cdot 10^{-4}$ was used to propagate solutions in time

Adding hyperviscosity comes at a slight cost. Numerical errors, especially near the peak of the propagating wave, increase with the application of the hyperviscosity filter. Figure 6 illustrates this drawback. Errors can be controlled by manipulating the order of the filter, β, and the scaling constant, γ. Achieving an optimal spectrum and minimizing errors is ultimately a balancing act. Further research in this area (errors associated with hyperviscosity) could be conducted and a parameter study could be performed to find optimal values, but this will not be an objective of this thesis.


Numerical Solutions for t = 17.1405

Hyperviscosity Errors at t = 17.1405

Figure 6. Log10 absolute error plot demonstrating the effects of hyperviscosity ($\beta = 2, \gamma = 1$) on numerical approximation errors; adding hyperviscosity increases numerical errors. 180 nodes, 12 nearest neighbors and a polynomial order of 9 were used to compute the RBF-FD matrix over the domain $x \in [0, 8\pi]$ while RK4 with a step size of $5 \cdot 10^{-3}$ was used to propagate solutions in time. Errors were plotted at $t = 17.1405$

2.3 Integrating Factors

Much of the instability that arises from numerically solving (1) can be attributed to the linear term, $-i\Delta u$, or the eigenvalues of D. An integrating factor can be introduced to transform (14) into an equation with a more favorable spectrum.

Since D is diagonalizable, it can be factored as

$$D = V\Lambda V^{-1} \tag{19}$$

where V is a square $N \times N$ matrix whose j^{th} column is the eigenvector $\mathbf{v}_j$ of D and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e. $\Lambda_{jj} = \lambda_j$. Substituting (19) into (15) gives

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t) &= -i\left(V\Lambda V^{-1}\mathbf{u}(t) + 2\begin{bmatrix} |u(\mathbf{x}_1,t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2,t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N,t)|^2 \end{bmatrix}\mathbf{u}(t)\right), \\
\frac{\mathrm{d}}{\mathrm{d}t}V^{-1}\mathbf{u}(t) + i\Lambda V^{-1}\mathbf{u}(t) &= -2iV^{-1}\begin{bmatrix} |u(\mathbf{x}_1,t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2,t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N,t)|^2 \end{bmatrix}\mathbf{u}(t).
\end{aligned} \tag{20}$$

Multiplying equation (20) through by the integrating factor $\mathrm{e}^{i\Lambda t}$ (a matrix exponential) yields

$$\begin{aligned}
\mathrm{e}^{i\Lambda t}\frac{\mathrm{d}}{\mathrm{d}t}V^{-1}\mathbf{u}(t) + i\mathrm{e}^{i\Lambda t}\Lambda V^{-1}\mathbf{u}(t) &= -2i\mathrm{e}^{i\Lambda t}V^{-1}\begin{bmatrix} |u(\mathbf{x}_1,t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2,t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N,t)|^2 \end{bmatrix}\mathbf{u}(t), \\
\frac{\mathrm{d}}{\mathrm{d}t}\left[\mathrm{e}^{i\Lambda t}V^{-1}\mathbf{u}(t)\right] &= -2i\mathrm{e}^{i\Lambda t}V^{-1}\begin{bmatrix} |u(\mathbf{x}_1,t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2,t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N,t)|^2 \end{bmatrix}\mathbf{u}(t).
\end{aligned} \tag{21}$$

Letting $W = V\mathrm{e}^{-i\Lambda t}$ and $\mathbf{C}(t) = \mathrm{e}^{i\Lambda t}V^{-1}\mathbf{u}(t)$ and substituting into (21) produces

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{C}(t) = -2iW^{-1}B(t)W\mathbf{C}(t), \tag{22}$$

where $B(t)$ is the diagonal matrix with $B(t)_{kk} = \left|\sum_{j=1}^{N} V_{kj}\mathrm{e}^{-i\Lambda_{jj}t}C(\mathbf{x}_j,t)\right|^2$ and $C(\mathbf{x}_j,t) = \mathrm{e}^{i\Lambda t}V^{-1}u(\mathbf{x}_j,t)$. This gives the new initial condition $\mathbf{C}(0) = V^{-1}\mathbf{u}(0)$. Thus, (22) is solved for $\mathbf{C}(t)$ from which $\mathbf{u}(t)$ can be obtained using the relation $\mathbf{u}(t) = W\mathbf{C}(t)$.

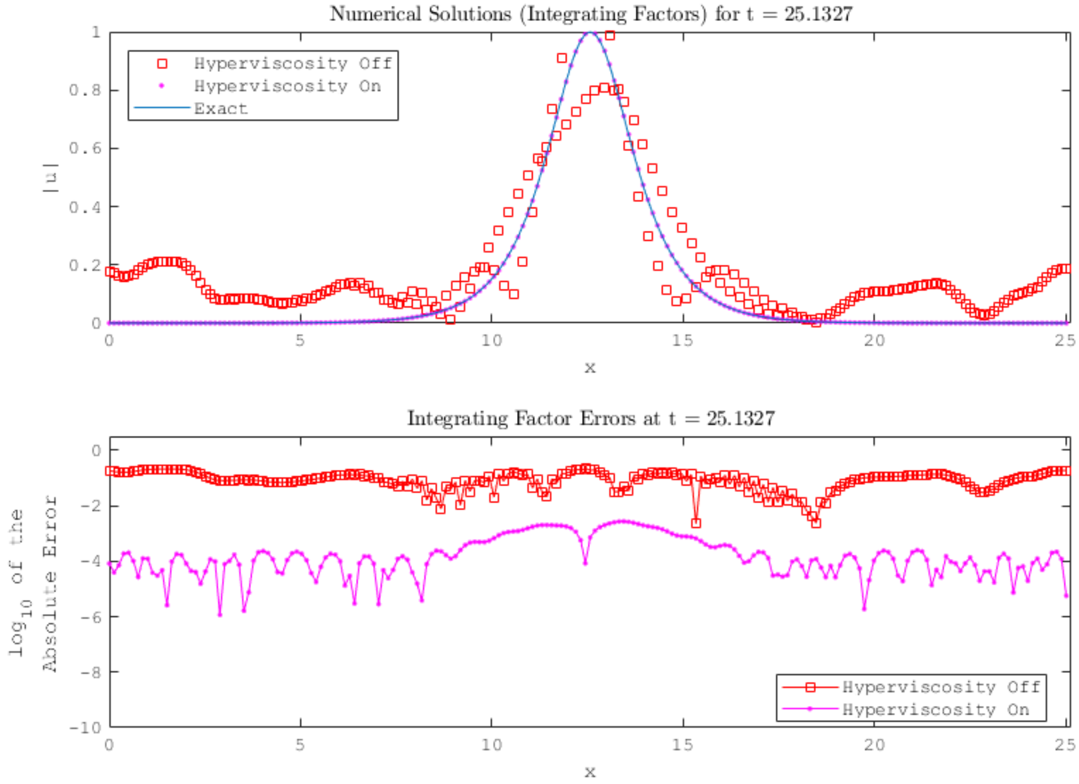


Figure 7. Comparison between solutions obtained using the integrating factor method with and without hyperviscosity ($\beta = 2, \gamma = 0.00075$). 200 nodes, 20 nearest neighbors and a polynomial order of 5 were used to compute the RBF-FD matrix over $x \in [0, 8\pi]$ in conjunction with RK4 with a step size of $5 \cdot 10^{-2}$

We note that (22) no longer contains a linear term; much of the instability inherent in (14) was due to the linear term containing the Laplacian operator. Hyperviscosity can be used in conjunction with an integrating factor to achieve even greater improvements in stability by simply replacing D in (15) with $D + iH$ where H is the hyperviscosity filter. Thus, an improvement in numerical stability should be expected when solving (22) with larger step sizes.

In figure 7, (22) was solved with RK4 using a step size two orders of magnitude lower than that used to solve (14). Moreover, (22) was solved for nearly the entire length of a period ($T_1 = 8\pi$) before major instabilities were observed in the case where hyperviscosity was not applied in the top subplot of figure 7.

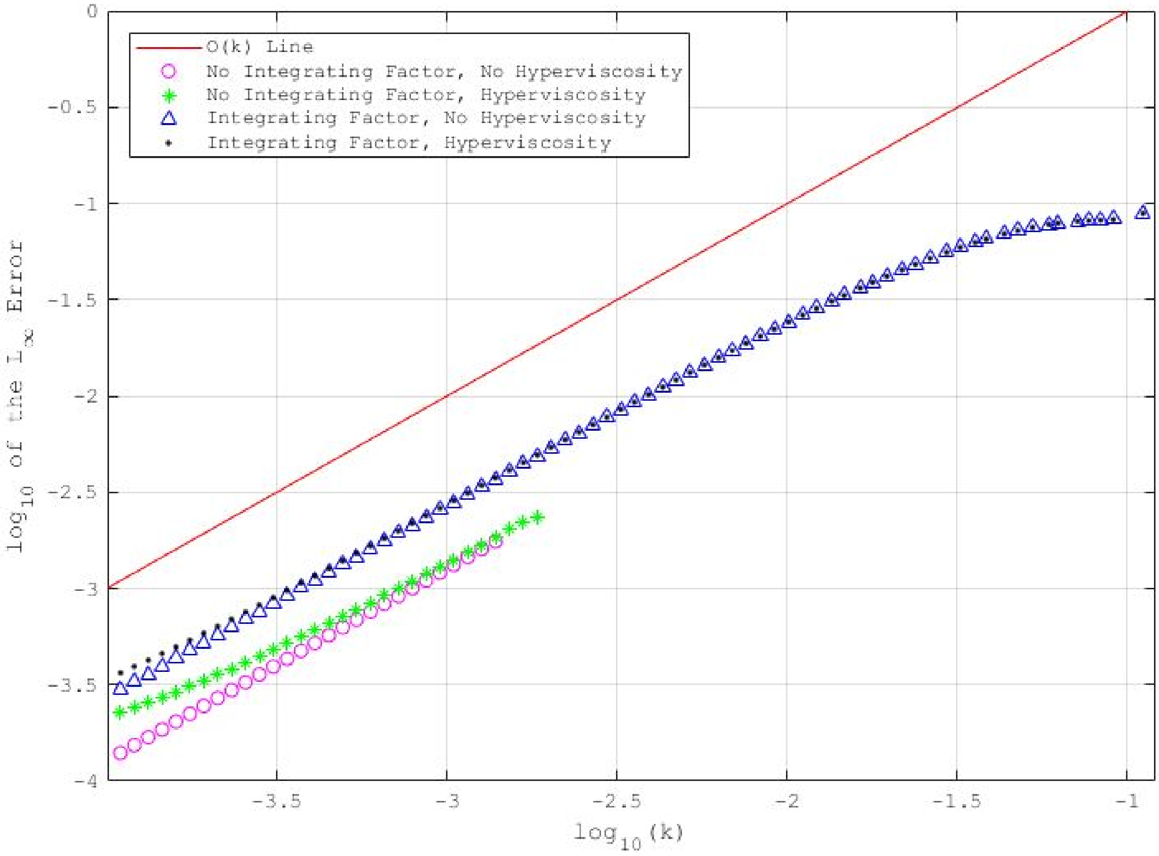


Figure 8. Log10 convergence plot comparing numerical solutions at $t = 1$ with and without Integrating Factors using a first order ODE solver (AB1-AM1 Predictor Corrector Method) for 1D NLS. 140 nodes, 10 nearest neighbors and a polynomial order of 7 were used to compute the RBF-FD matrix over $x \in [0, 30]$ and hyperviscosity was computed with $\beta = 3$ and $\gamma = 0.001$

Figure 8 provides a summary of the comparisons between solutions produced with and without integrating factors. Larger step sizes can be used when integrating factors are used to solve (22). As expected, errors are larger when hyperviscosity is applied.

Additionally, errors are also slightly higher when integrating factors are used. These results are not just limited to first order ODE solvers. Figure 9 confirms the fact that RK4 achieves fourth order convergence when integrating factors are applied. However, we begin to see that the accuracy of the numerical solver used is limited by the spatial discretization because numerical errors reach a minimum and no longer decrease for $\log_{10}(k) < -3.2$. It is important to also note that periodic boundary conditions are enforced on an initial condition that is not periodic, which affects the accuracy of the solutions at the boundary. This can be mitigated by increasing the domain size since $u(x, 0) \to 0$ rapidly as $|x| \to \infty$. Hence, utilizing higher order methods requires us to increase the number of nodes and/or number of nearest neighbors used to produce D and/or increase the domain size, which can be computationally expensive. Likewise, as the size of the matrix D grows, so do its eigenvalues, requiring smaller step sizes for stability. This computational cost compounds when we consider the fact that the use of integrating factors requires computing the eigendecomposition of D, which is a major drawback when N is large. To reconcile stability with computational costs, we resort to splitting methods for ODEs.

2.4 Splitting Methods

As mentioned previously, much of the instability that arises in numerically solving the NLS can be attributed to the fact that D has a wide range of eigenvalues. Since at least one eigenvalue of D has relatively large negative real part, we require k to be small when using an explicit solver. Therefore, it is natural to use implicit methods, especially A-stable methods such as the backward Euler and trapezoidal methods for stiff problems. In splitting, the stiff linear term is often best treated using an implicit ODE solver, while the nonlinear term is best treated explicitly. Conveniently, we can treat the NLS as a PDE with a stiff linear term coupled with a nonlinear term. We

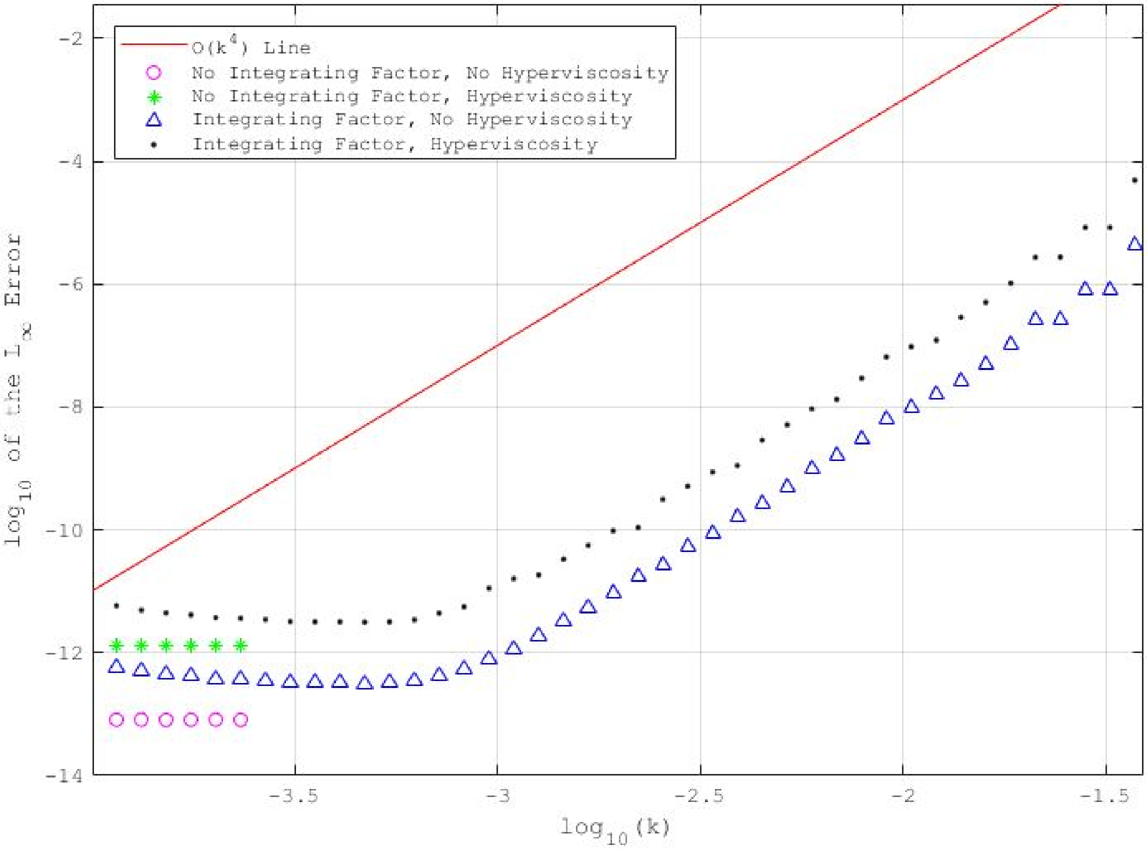


Figure 9. Log10 convergence plot comparing numerical solutions at $t = 1$ with and without Integrating Factors using a fourth order ODE solver for 1D NLS. Accuracy of solutions without integrating factors is limited by the spatial discretization, i.e. the RBF-FD matrix. 600 nodes, 20 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $x \in [0, 50]$ and hyperviscosity was computed with $\beta = 3$ and $\gamma = 0.0001$

can write (15) as

$$\frac{\partial}{\partial t}\mathbf{u}(t) = \mathcal{L}\mathbf{u}(t) + \mathcal{N}\mathbf{u}(t), \tag{23}$$

with

$$\mathcal{L}\mathbf{u}(t) = -iD\mathbf{u}(t) \quad \text{and} \quad \mathcal{N}\mathbf{u}(t) = -2i \begin{bmatrix} |u(\mathbf{x}_1, t)|^2 & 0 & \cdots & 0 \\ 0 & |u(\mathbf{x}_2, t)|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |u(\mathbf{x}_N, t)|^2 \end{bmatrix} \mathbf{u}(t).$$

In the method of splitting, we can approximate the solution $\mathbf{u}(t)$ by considering $\mathcal{L}\mathbf{u}(t)$ and $\mathcal{N}\mathbf{u}(t)$ as acting successively. We do this by successively solving the equations

$$\frac{\partial}{\partial t}\mathbf{u}(t) = \mathcal{N}\mathbf{u}(t), \quad \frac{\partial}{\partial t}\mathbf{u}(t) = \mathcal{L}\mathbf{u}(t) \tag{24}$$

by using the solution of the former equation as the initial condition for the latter equation; the order in which we solve the equations does not matter. There is no splitting error if $\mathcal{N}\mathcal{L} = \mathcal{L}\mathcal{N}$. But in general, $\mathcal{N}$ and $\mathcal{L}$ do not commute. In practice, the exact operators $\mathcal{N}$ and $\mathcal{L}$ are replaced by their numerical approximations. The splitting error is $O(k^2)$ for each time step, and has first order global accuracy [15]—a convergence plot (see Figure 10) was generated to confirm this. For simple splitting, figure 10 demonstrates that hyperviscosity has relatively little impact on the convergence of the scheme if the hyperviscosity order is appropriately chosen ($\beta = 2$ was used in figure 10).

We immediately see the appeal of splitting: one may sequentially use separate methods to evolve each physical term, using solution methods that may be highly tuned to that particular form of operator. To demonstrate this, first order splitting was used to solve (23). The forward Euler method, an explicit method, was used to solve the nonlinear equation in (24). The backward Euler method, an implicit method, was used in conjunction with a QR decomposition to solve the linear equation in (24).

Although first order splitting leads to an improvement in stability, the method is not necessarily viable for solving NLS. In figure 11, numerical dissipation can be observed as the amplitude of the wave being propagated is being dampened as solutions are evolved in time. This dampening effect can be observed regardless of whether or not hyperviscosity is applied to the differential operator. In figure 12, errors become quite large as t increases. The growth in maximum absolute error appears to slow down over time as the maximum absolute error approaches 1. Since solutions

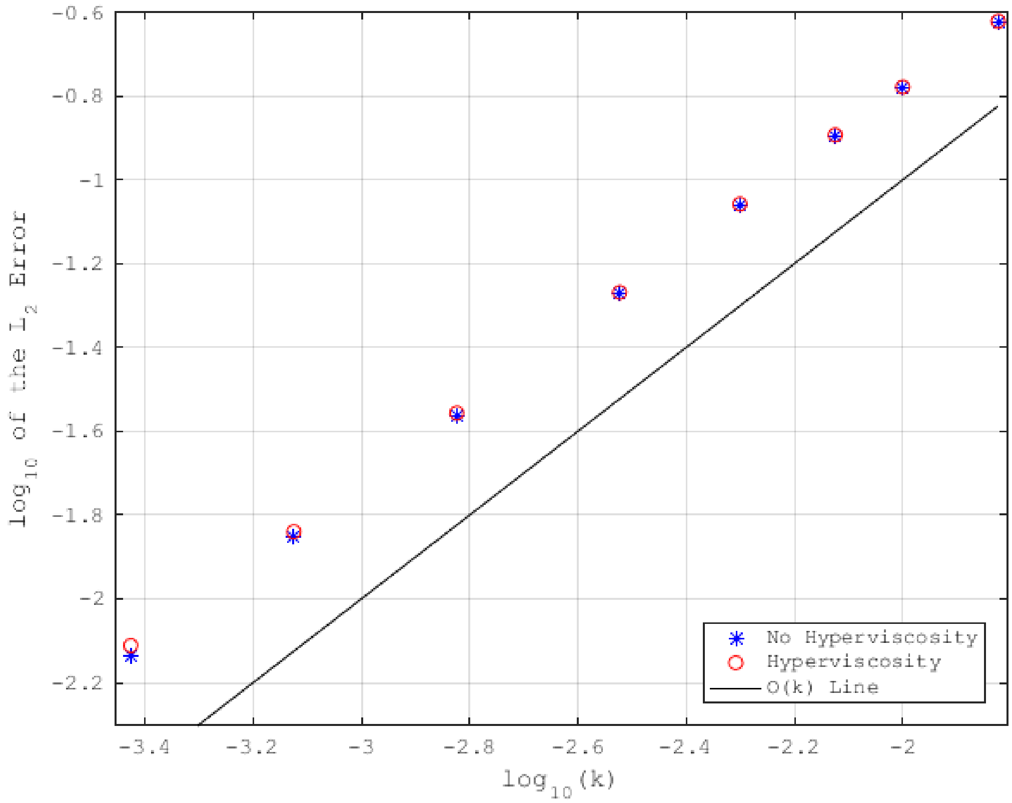


Figure 10. 1^{st} order splitting convergence plot comparing solutions with and without hyperviscosity ($\beta = 2, \gamma = 0.0001$) at $t = 1$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part

are not necessarily growing exponentially, figure 12 shows that numerical dispersion is present in addition to dissipation since there is a mismatch in phase between the numerical solution and the analytical solution. Hence, it is reasonable to conclude that first order splitting is not ideal for solving NLS and we must seek out a higher order splitting scheme.

Further improvements in the splitting error can be made by considering different splitting schemes. We can achieve second order accuracy by employing Strang splitting [12]. Figure 13 shows that Strang splitting is indeed a second order scheme before errors are limited by the spatial discretization. Figure 13 also shows that hy-

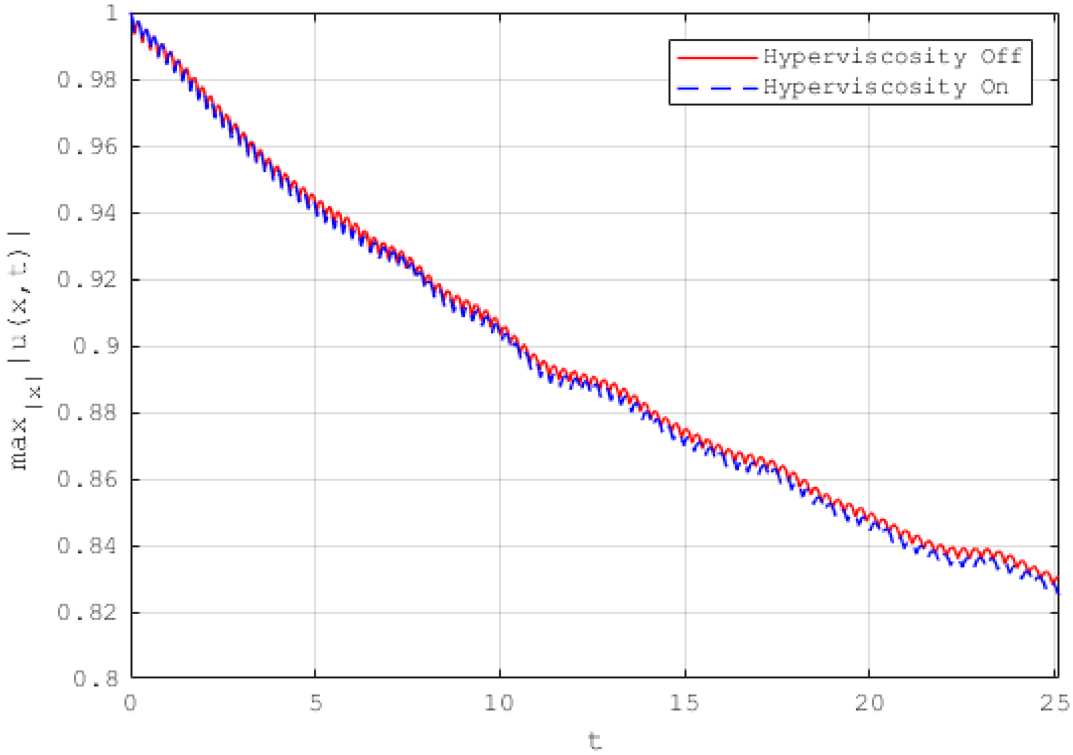


Figure 11. The maximum of $u(x,t)$ over x plotted versus t. Solutions were produced using a 1^{st} order splitting scheme with $k = 5 \cdot 10^{-3}$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part

perviscosity can affect the accuracy of numerical approximations if the parameters β and/or γ are not chosen properly. The errors produced with hyperviscosity are higher than those produced without hyperviscosity. Additionally, errors decrease at a lower rate when hyperviscosity is applied when reasonably optimal parameters are not chosen.

It is worth mentioning that Strang splitting does not require much more computation than first order splitting. Consider the differential equation of the form

$$\frac{\mathrm{d}y}{\mathrm{d}t} = (L_1 + L_2)y,$$

with initial condition $y(0) = y_0$ and L_1 and L_2 are arbitrary operators. For each time

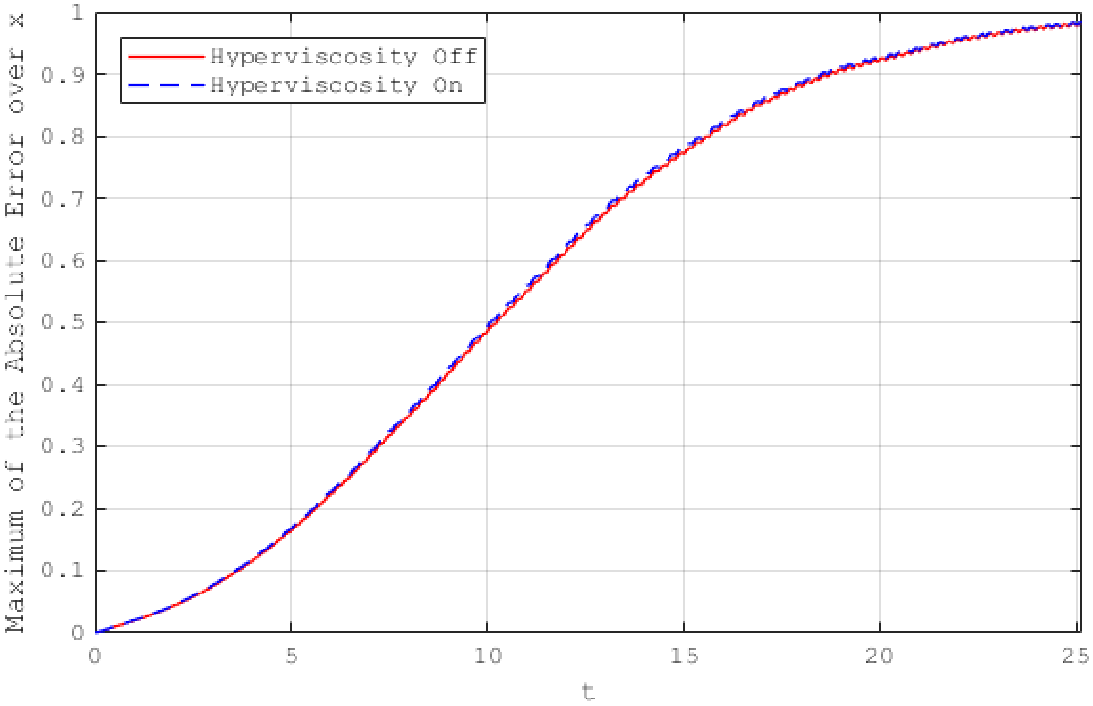


Figure 12. The maximum of the absolute error over x plotted versus t. Solutions were produced using a 1^{st} order splitting scheme with $k = 5\cdot 10^{-3}$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. Forward Euler was used to solve the nonlinear part of the PDE and backward Euler was used to solve the linear part

step, we solve $y' = L_1 y$ for a half time step; then solve $y' = L_2 y$ for a full time step; and finally, solve $y' = L_1 y$ for a half time step. Thus, the Strang splitting scheme is implemented as

$$
\begin{aligned}
\tilde{y}_1 &= \mathrm{e}^{L_1 k/2} y_0, \quad & \bar{y}_1 &= \mathrm{e}^{L_2 k} \tilde{y}_1, \quad & y_1 &= \mathrm{e}^{L_1 k/2} \bar{y}_1, \\
\tilde{y}_2 &= \mathrm{e}^{L_1 k/2} y_1, \quad & \bar{y}_2 &= \mathrm{e}^{L_2 k} \tilde{y}_2, \quad & y_2 &= \mathrm{e}^{L_1 k/2} \bar{y}_2, \\
& & &\vdots & & \\
\tilde{y}_F &= \mathrm{e}^{L_1 k/2} y_{F-1}, \quad & \bar{y}_F &= \mathrm{e}^{L_2 k} \tilde{y}_F, \quad & y_F &= \mathrm{e}^{L_1 k/2} \bar{y}_F,
\end{aligned}
$$

for F time steps. We note that y_j and $\tilde{y}_{j+1}$ for $j = 1, \ldots, F-1$ can be combined into

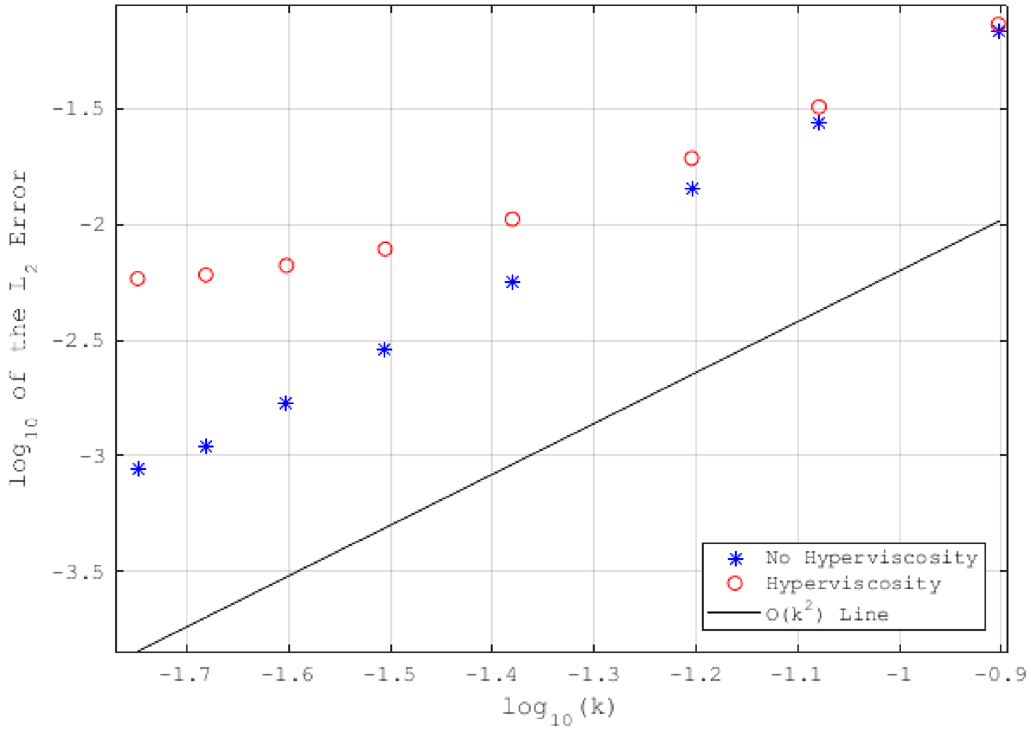


Figure 13. Strang splitting convergence plot at $t = 1$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 5$ and $\gamma = 0.1$. RK2 was used to solve the nonlinear part of the PDE and BDF-2 was used to solve the linear part

a single step so that the scheme becomes

$$\begin{aligned}
\tilde{y}_1 &= \mathrm{e}^{L_1 k/2} y_0, \quad & \bar{y}_1 &= \mathrm{e}^{L_2 k} \tilde{y}_1, \\
\tilde{y}_2 &= \mathrm{e}^{L_1 k} \bar{y}_1, \quad & \bar{y}_2 &= \mathrm{e}^{L_2 k} \tilde{y}_2, \\
& & \vdots & \\
\tilde{y}_F &= \mathrm{e}^{L_1 k} \bar{y}_{F-1}, \quad & \bar{y}_F &= \mathrm{e}^{L_2 k} \tilde{y}_F, \\
& & y_F &= \mathrm{e}^{L_1 k/2} \bar{y}_F.
\end{aligned}$$

Hence, to implement Strang splitting, we only need to solve $y' = L_1 y$ for a half time step on the initial and final time steps. In between, we use the simple splitting scheme presented earlier. In this manner, we obtain second order accuracy without much more computation. In the context of the NLS equation, the linear dispersive operator of the equation $\mathcal{L}$ is treated as L_1 while the nonlinear operator $\mathcal{N}$ is treated as L_2. We proceed by using the Strang splitting scheme to evolve (23) in time by solving the linear part with BDF2, a multi-step implicit method, using RK2 to start the scheme. RK2 was used to solve the nonlinear part.

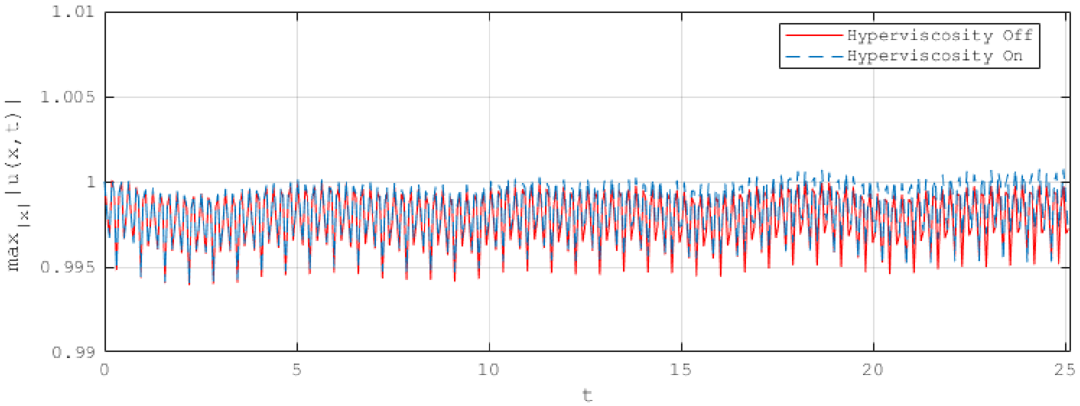


Figure 14. The maximum of $u(x,t)$ over x plotted versus t. Solutions were produced using Strang splitting, a 2^{nd} order splitting scheme, with $k = 0.1257$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. RK2 was used to solve the nonlinear part of the PDE and BDF2 was used to solve the linear part

All in all, the Strang splitting scheme produces significantly better results than the simple splitting scheme. In figure 14, numerical dissipation is not as drastic as that in figure 11. In figure 14, we observe that the peak of the wave being propagated does tend to oscillate around 1, but the amplitudes of the oscillations are reasonably low. Moreover, hyperviscosity does not appear to have an influence on numerical dissipation since results produced using hyperviscosity are similar to those produced without hyperviscosity. In figure 15, we can observe error propagation and the effect hyperviscosity has on the absolute error. For $t > 4$, the maximum absolute errors over x produced using hyperviscosity are greater than those produced without hyperviscosity—this difference grows as t increases, but not to the point where Strang

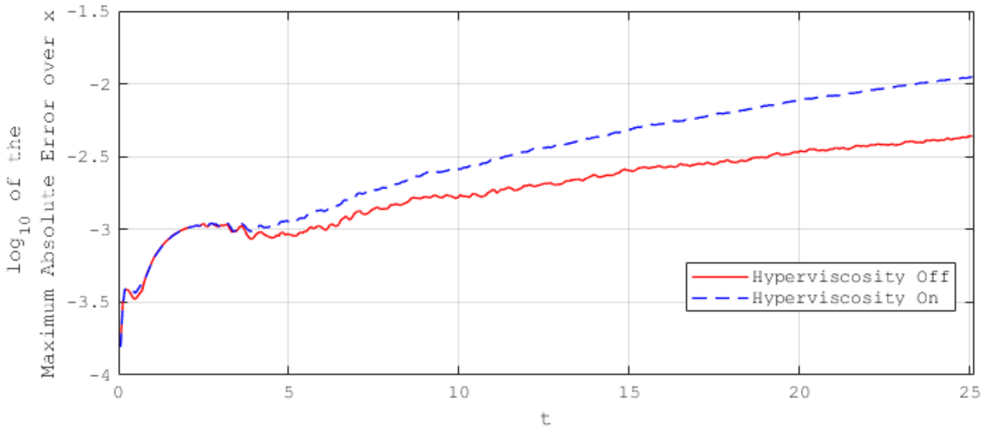


Figure 15. $\log_{10}$ of maximum of the absolute error over x plotted versus t. Solutions were produced using Strang splitting, a 2^{nd} order splitting scheme, with $k = 0.1257$. 120 nodes, 12 nearest neighbors and a polynomial order of 10 was used to discretize the spatial domain while hyperviscosity was computed using $\beta = 3$ and $\gamma = 0.01$. RK2 was used to solve the nonlinear part of the PDE and BDF2 was used to solve the linear part

splitting becomes inviable. It is also worth mentioning that we were able to produce superior results with a significantly larger step size—$k = 0.1257$ for Strang splitting in figures 14 and 15 versus $k = 5 \cdot 10^{-3}$ for simple splitting in figures 11 and 12.

2.5 Comparison of Methods and Observations

The use of splitting schemes, integrating factors and hyperviscosity allows for larger step sizes. In figure 16, for $\log_{10}(k) > -2.5$, the scheme used without integration factors operator splitting, or hyperviscosity was unstable when first order methods were used. However, when hyperviscosity was included, there was an improvement in stability since larger step sizes were able to be taken. This came at the cost of increased errors for $\log_{10}(k) > -2.8$. As expected, larger step sizes can be taken when using integration factors or operator splitting. When hyperviscosity used in conjunction with integration factors or operator splitting, errors increase and are no longer consistent with the order of method beyond a certain step size despite providing a marginal improvement in stability. These observations agree favorably with the the results in figure 16 obtained using second order methods; the Strang splitting scheme without hyperviscosity actually attains an order of accuracy greater than 2 for $\log_{10}(k) < -3.1$.

For fixed $N = 500$, $n = 24$ and $m = 11$, the costs of evolving solutions in time with several first and second order schemes for $x \in [0, 50]$ and $t \in [0, 0.1]$ are illustrated in figure 17. Computing costs are relatively high when integrating factors are used because a larger number of matrix multiplications must be performed to solve the ODE (22) and obtain $\mathbf{u}(t)$ from the relation $\mathbf{u}(t) = W\mathbf{C}(t) = V\mathrm{e}^{-i\Lambda t}\mathbf{C}(t)$. The majority of the computing costs associated with the first order splitting scheme comes from the fact that a row-reduction algorithm was used to implement the implicit forward Euler method used to solve the linear portion of the NLS equation. Hence, improved stability or the ability to take larger step sizes comes at the cost of more operations. Figure 17 does not account for any preprocessing costs. It is important to note that the implicit Euler method used for operator splitting required a QR decomposition of the RBF-FD matrix while the integration factors required its eigendecomposition, so the total computing costs associated with these stabilization methods are much higher than what is illustrated in figure 17. The computation times associated with first order methods in figure 17 show a similar trend to those associated with second order methods. When hyperviscosity is not used, splitting schemes appear to have an advantage over integration factors in terms of accuracy. Splitting schemes also have slightly lower computing costs than integration factors, however, this difference is negligible since the computing costs are of the same order of magnitude.

Figures 16 and 17 are summarized by figure 18. Errors produced by each method decrease appropriately as computation time increases. For first order methods, the errors decrease linearly according to computation time. The errors associated with the second order methods behave as expected as well.

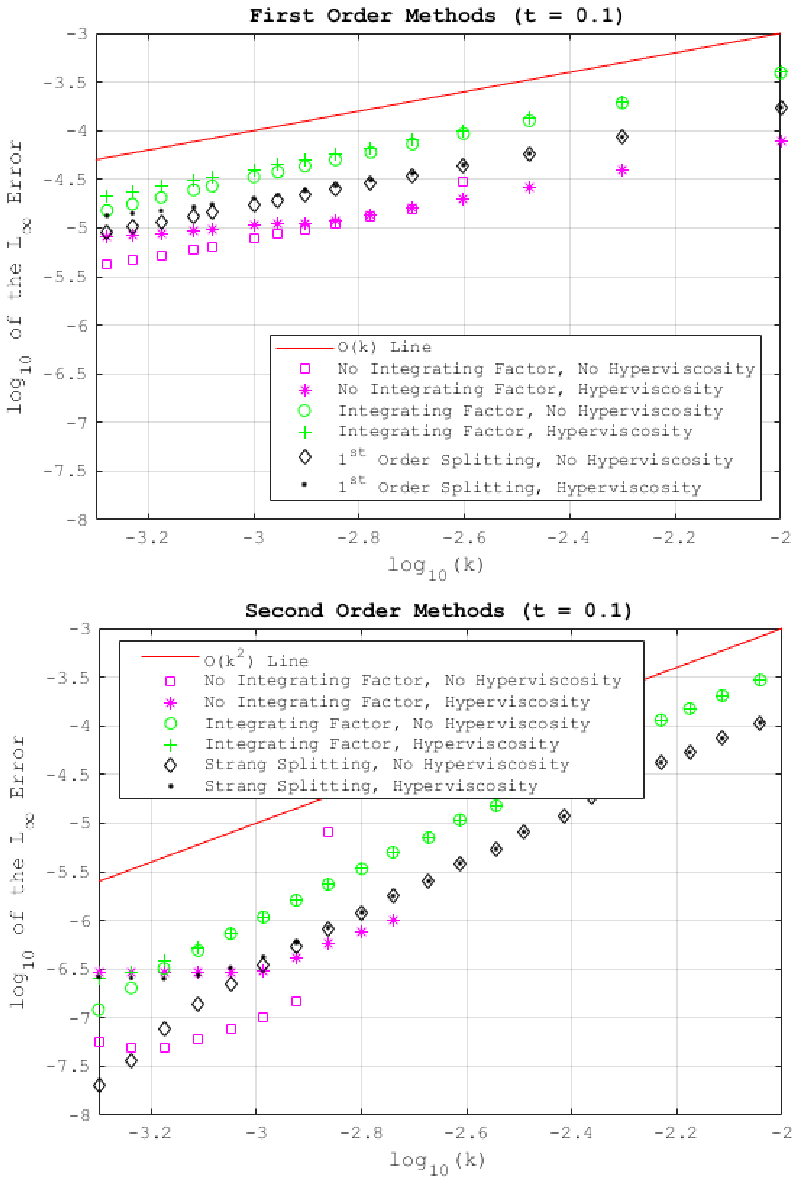


Figure 16. Log10 convergence plots are shown comparing numerical solutions of the one-dimensional NLS equation at $t = 0.1$ obtained using several methods. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. (TOP) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. Hyperviscosity was computed with $\beta = 2$ and $\gamma = 0.03$. (BOTTOM) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. Hyperviscosity was computed with $\beta = 2$ and $\gamma = 0.0001$.

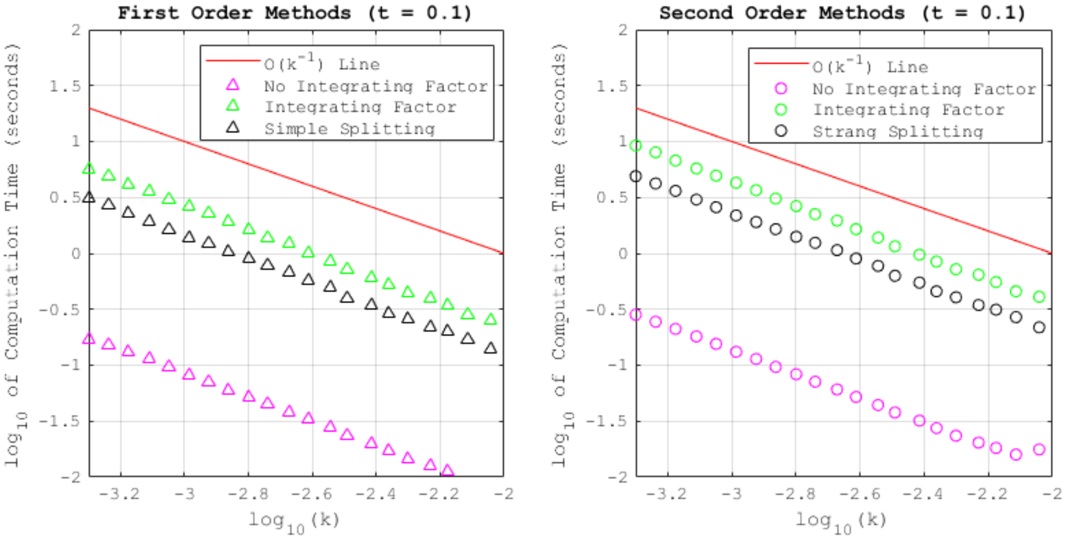


Figure 17. Timing results for several methods excluding hyperviscosity; the preprocessing costs of computing the RBF-FD matrix and its eigendecomposition were not accounted for. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. Each method has rate $O\left(k^{-1}\right)$ for computation cost. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2.

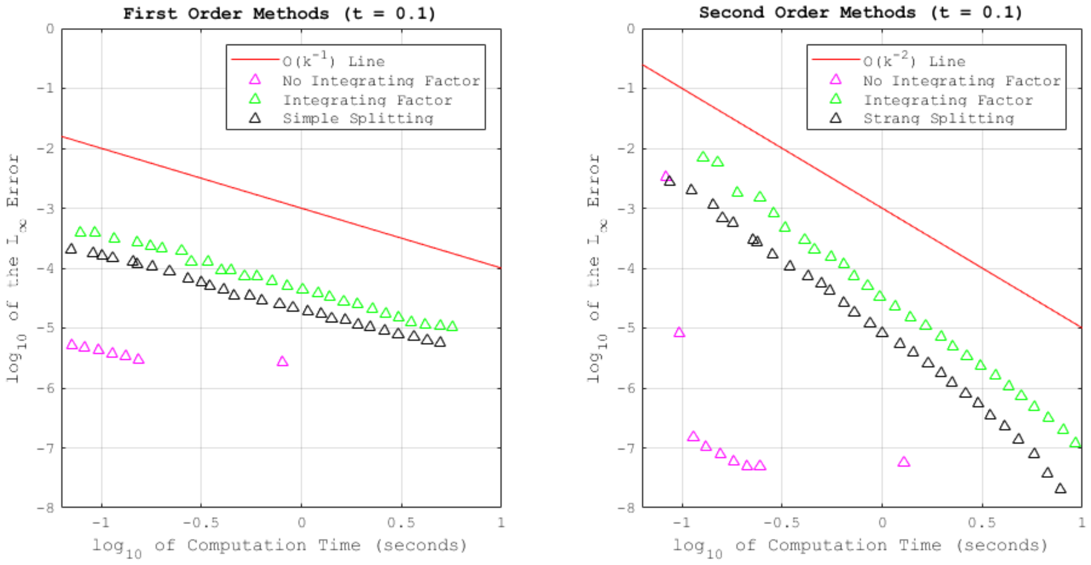


Figure 18. Computation time versus the L_∞ error for several methods—hyperviscosity was not used. D was computed using 500 nodes, 24 nearest neighbors and a polynomial order of 11 were used to compute the RBF-FD matrix over $x \in [0, 50]$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2.

III. The Two-Dimensional Case

In this section, we produce and analyze numerical solutions, with and without hyperviscosity, of the two-dimensional variant of (1) where $\mathbf{x} = [x\ y]^T \in \mathbb{R}^2$ and $\kappa = 2$,

$$\frac{\partial}{\partial t}u(x,y,t) = -i\left(\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u(x,y,t)+2|u(x,y,t)|^2u(x,y,t)\right), \tag{25}$$

where periodicity in x and y will be enforced. More specifically, u will be T_x-periodic in x and T_y-periodic in y. As shown in figure 19, the initial condition that will be used is

$$u(x,y,0) = \operatorname{sech}\left(x-\frac{T_x}{2}\right)\exp\left(i\frac{\left(x-\frac{T_x}{2}\right)}{2}\right).$$

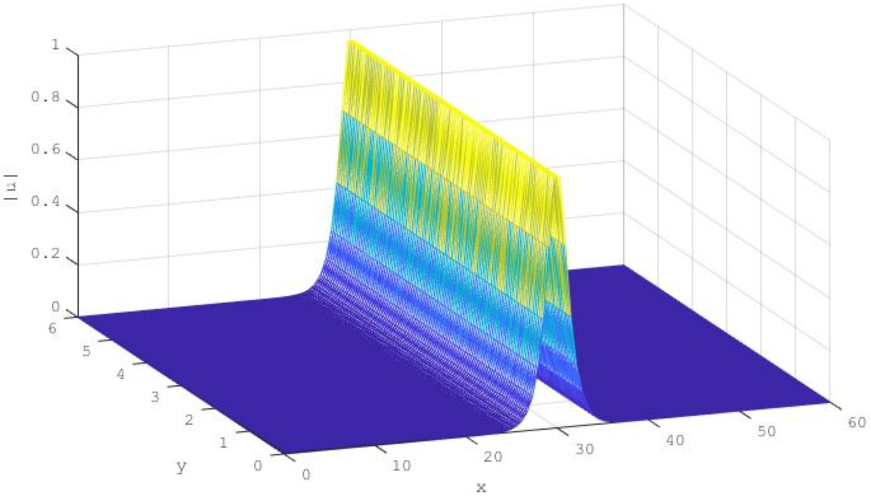


Figure 19. Modulus of the initial condition with $T_x = 60$ and $T_y = 6$

3.1 Methodology

Solutions will be produced using the MN RBF r^7 while powers of the Laplacian for the hyperviscosity term will be computed using GA RBFs. As in chapter II, ϵ will be chosen such that condition number of the matrix in (11), $\hat{A}$, is kept below 10^8 and polynomial terms will not be used to compute hyperviscosity. It is not neccessary to use the same type of RBF for the spatial discretization and hyperviscosity term, as the role of hyperviscosity is simply to shift eigenvalues to the left half of the complex plane [4], as seen in figure 5.

Periodic boundary conditions can be enforced on (25) through the conformal mapping of nodes from a two-dimensional grid onto the surface of a torus as shown in figures 22 and 23. A specific node (circled in green) and its nearest neighbors (circled in red) are shown on a two-dimensional grid in the left subplot and are shown again in the right subplot after they are mapped onto a torus. The torus used to compute the nearest neighbors for each node is defined parametrically as

$$
\begin{aligned}
u(x,y) &= \left(R + r\cos\left(2\pi\frac{x}{T_x}\right)\right)\cos\left(2\pi\frac{y}{T_y}\right),\\
v(x,y) &= \left(R + r\cos\left(2\pi\frac{x}{T_x}\right)\right)\sin\left(2\pi\frac{y}{T_y}\right),\\
w(x,y) &= r\sin\left(2\pi\frac{x}{T_x}\right),
\end{aligned}
$$

where R is the distance from the center of the torus to the center of the circle that is rotated to generate the torus and r is the radius of the circle.

Let p_1 and p_2 be two points on the torus and let $\mathbf{b}_1$ and $\mathbf{b}_2$ be vectors from the origin to the points p_1 and p_2, respectively, and let $\mathbf{b}_1^{(p)}$ and $\mathbf{b}_2^{(p)}$ be the orthogonal projections of $\mathbf{b}_1$ and $\mathbf{b}_2$ onto the $u-v$ plane. The metric used to compute the nearest neighbors for each node on the torus is

$$d(p_1,p_2) = \sqrt{\theta_1^2 + \theta_2^2}, \tag{26}$$

where θ_1 is the angle between $\mathbf{b}_1^{(p)}$ and $\mathbf{b}_2^{(p)}$ and θ_2 is the angular distance between p_1

and p_2 projected onto a circle of radius r centered at the origin on the $v - w$ plane. θ_1 can be found by taking the inner product between vectors $\mathbf{b}_1^{(p)}$ and $\mathbf{b}_2^{(p)}$ by the relation

$$\theta_1 = \frac{T_y}{2\pi} \cos^{-1} \left(\frac{\left\langle \mathbf{b}_1^{(p)}, \mathbf{b}_2^{(p)} \right\rangle}{\left\| \mathbf{b}_1^{(p)} \right\| \left\| \mathbf{b}_1^{(p)} \right\|} \right).$$

To find θ_2, the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ must first be rotated onto the $v - w$ plane. This is accomplished by using the transformation

$$\begin{bmatrix} \cos \eta_i & -\sin \eta_i & 0 \\ \sin \eta_i & \cos \eta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

where η_i is the angle between $\mathbf{b}_i^{(p)}$ and the $v-$axis for $i = 1, 2$. Using the appropriate trigonometric identities, the matrix (27) can be expressed as

$$\mathcal{R}_i = \frac{1}{\sqrt{u_{\mathbf{b}_i}^2 + v_{\mathbf{b}_i}^2}} \begin{bmatrix} v_{\mathbf{b}_i} & -u_{\mathbf{b}_i} & 0 \\ u_{\mathbf{b}_i} & v_{\mathbf{b}_i} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where $u_{\mathbf{b}_i}$ is the $u-$component of $\mathbf{b}_i$ and $v_{\mathbf{b}_i}$ is the $v-$component of $\mathbf{b}_i$ for $i = 1, 2$. Let

$$\mathbf{p}_1 = \mathcal{R}_1 \mathbf{b}_1 \quad \text{and} \quad \mathbf{p}_2 = \mathcal{R}_2 \mathbf{b}_2$$

to obtain

$$\hat{\mathbf{p}}_1 = \mathbf{p}_1 - R \frac{\mathbf{b}_1^{(p)}}{\left\| \mathbf{b}_1^{(p)} \right\|} \quad \text{and} \quad \hat{\mathbf{p}}_2 = \mathbf{p}_2 - R \frac{\mathbf{b}_2^{(p)}}{\left\| \mathbf{b}_2^{(p)} \right\|}.$$

Figure 20 provides an illustration of the two rotated vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ in the $v - w$ plane while figure 21 provides an illustration of the vectors $\hat{\mathbf{p}_1}$ and $\hat{\mathbf{p}_2}$.

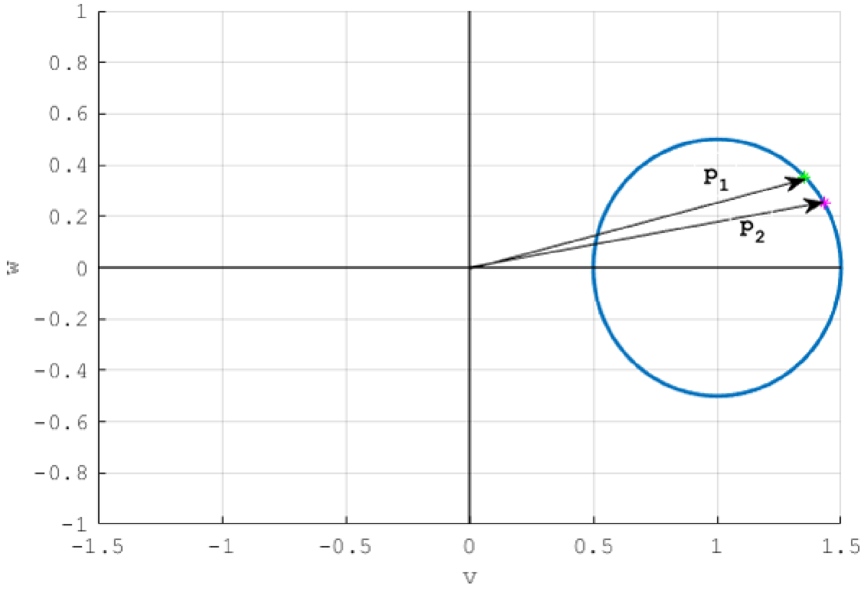


Figure 20. The vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are illustrated in the $v - w$ plane. They are the rotated variants of $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively.

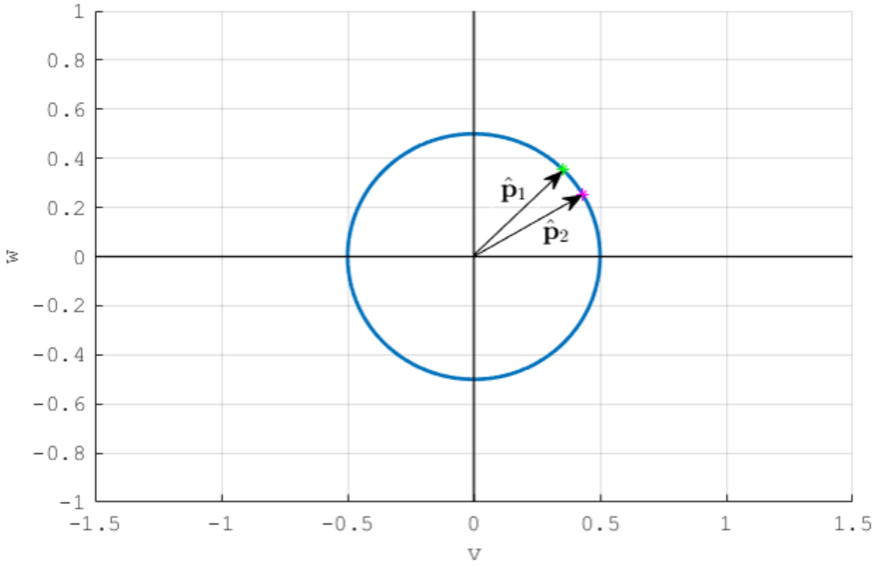


Figure 21. The vectors $\hat{\mathbf{p}_1}$ and $\hat{\mathbf{p}_2}$ are illustrated in the $v - w$ plane. They point to the two points p_1 and p_2, respectively, on a cross section of the torus centered at the origin in the v-w plane.

The angle θ_2 can then be computed as

$$\theta_2 = \frac{T_x}{2\pi} \cos^{-1}\left(\frac{\langle \hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2 \rangle}{\|\hat{\mathbf{p}}_1\| \, \|\hat{\mathbf{p}}_2\|}\right).$$

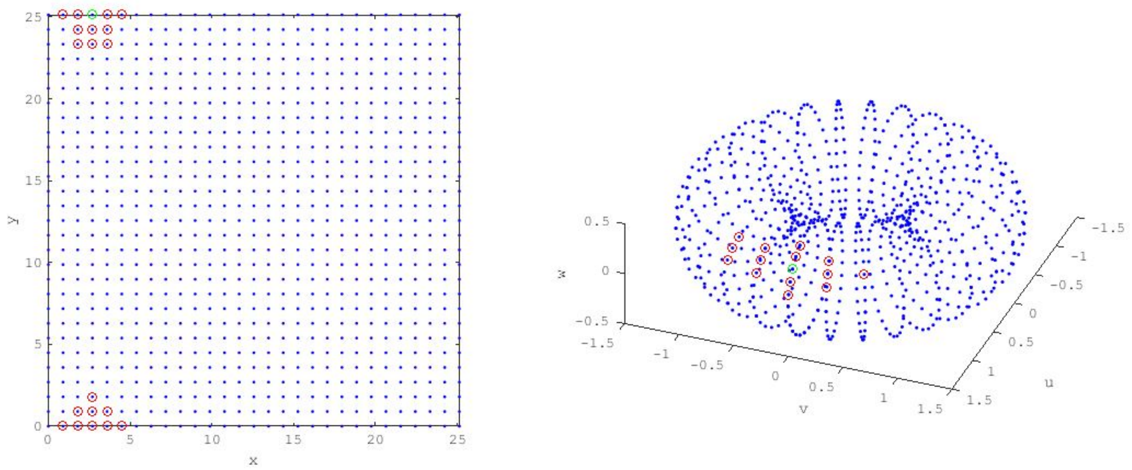


Figure 22. Nearest neighbors (circled in red) of a node (circled in green) on a torus with $R = 1$ and $r = 0.5$ using the custom angular distance metric defined by (26). Nearest neighbors from the torus are properly mapped back onto the two-dimensional grid

The metric (26) was used to compute nearest neighbors on the torus instead of the default Euclidean distance because the Euclidean distances between points on a two-dimensional grid are not preserved when they are mapped onto a torus; figures 22 and 23 illustrate why using the default Euclidean distance to compute a node's nearest neighbors is not recommended.

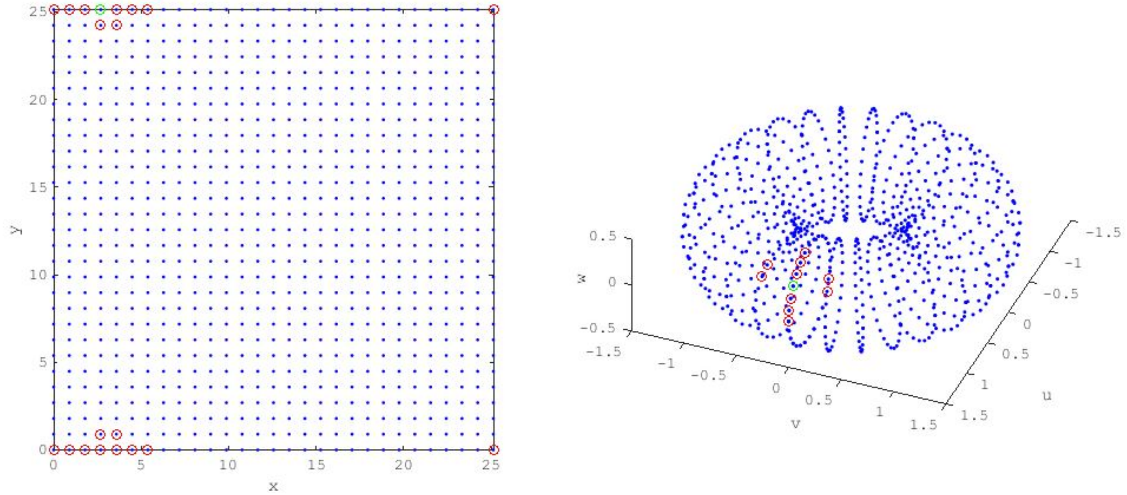


Figure 23. Nearest neighbors (circled in red) of a node (circled in green) on a torus with $R = 1$ and $r = 0.5$ using the default Euclidean distance. Since Euclidean distance is not preserved when points are conformally mapped from a two-dimensional grid onto a torus, nearest neighbors are incorrectly computed

3.2 Results

Solutions of varying accuracies are given in figures 24 and 25. As we saw in chapter II, larger step sizes can be taken when integrating factors and splitting schemes are employed to solve (14). Figures 24 and 25 show this to be true as well when solving equation (25). Figure 24 compares favorably with figure 16. Additionally, as we saw in figure 9, the accuracy of our solutions are limited by the spatial discretization as seen in figure 25 since errors neither decrease or increase and stay at around $10^{-4.8}$ regardless of what k is. Limited accuracy can also be attributed to the fact that periodic boundary conditions are enforced on an initial condition that is not periodic, as mentioned in chapter II. Figure 26 gives us some insight into how accurate we should expect our solutions to be. In figure 26, the absolute error attains a maximum around 10^{-4} at roughly $x = 25$. Since the error between D applied to the initial condition and $\Delta u(x, y, 0)$ is around 10^{-4}, we expect the accuracy of our solutions to be around the same order. This is accurately reflected in figure 25 where solutions attain errors around $10^{-4.8}$.

Since the accuracy of our solutions is limited by the spatial discretization of (25), we can simply increase N and n at the cost of computation time and memory. Additionally, as mentioned in chapter II, increasing the domain size can also lead to improvements in accuracy since $u(\mathbf{x}, 0) \to 0$ rapidly as $|\mathbf{x}| \to \infty$. Figure 27 compares solutions found using integrating factors with a large number of nodes ($N = 36000$) and a large domain size ($T_x = 60, T_y = 6$); the accuracy of our solutions is still limited by the error between D applied to the initial condition and $\Delta u(x, y, 0)$, but larger step sizes can be used.

With 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 used to compute D over $\mathbf{x} \in [0, T_x] \times [0, T_y]$, where $T_x = 50$ and $T_y = 5$, the costs of evolving solutions in time with several first and second order schemes for $t \in [0, 0.1]$

are illustrated in figure 28. The computing costs in figure 28 is comparable to the computing costs illustrated in figure 17. First and second order splitting outperforms integration factors in terms of computing costs—this advantage is somewhat negligible given that their computing costs are still of the same order. We note that splitting methods also outperform integration factors in terms of accuracy based off of figure 24.

Figures 24 and 28 are summarized by figure 29. Errors produced by each first order method decrease appropriately as computation time increases. For second order methods, when integrating factors or splitting are not used, errors do not follow a familiar trend due to stability restrictions. However, errors produced with integrating factors and splitting behave as expected. Figure 29 suggests that splitting is superior to integration factors.

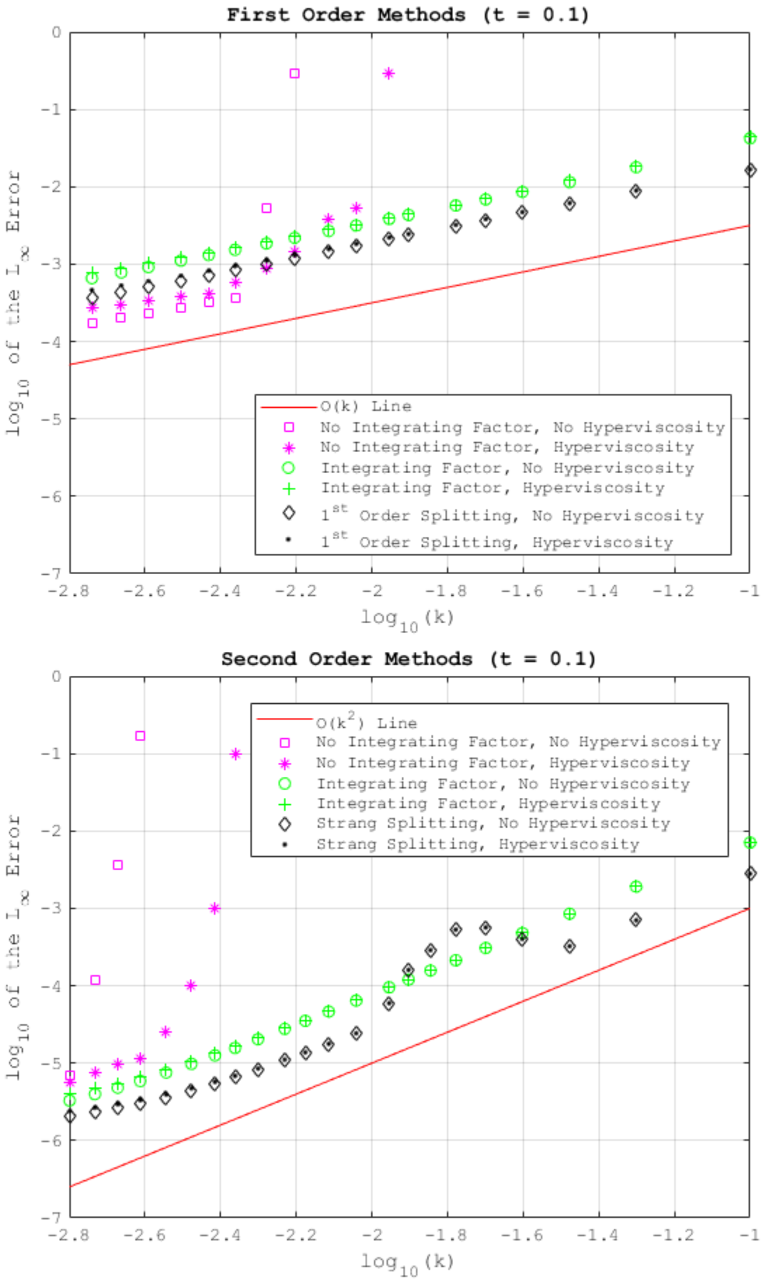


Figure 24. Log10 convergence plots are shown comparing numerical solutions of the two-dimensional NLS equation at $t = 0.1$ obtained using several methods. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. (TOP) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. Hyperviscosity was computed with $\beta = 4$ and $\gamma = 0.01$. (BOTTOM) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2. Hyperviscosity was computed with $\beta = 5$ and $\gamma = 0.1$.

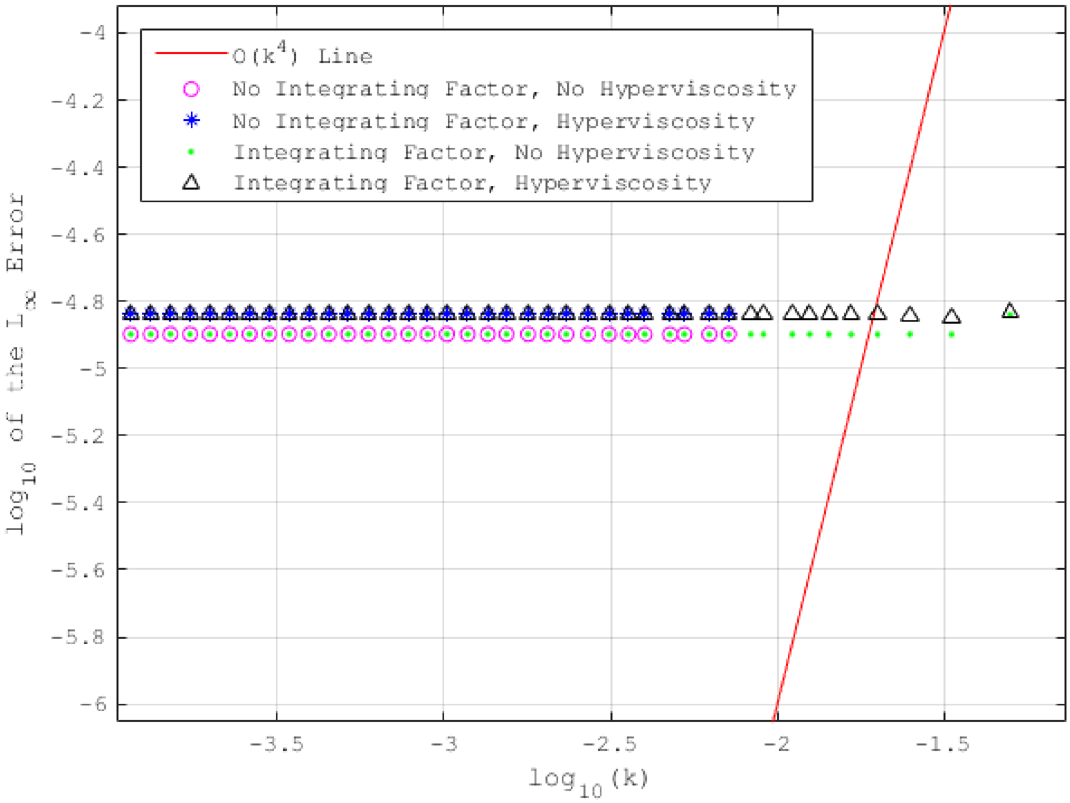


Figure 25. Comparison between numerical solutions at $t = 0.1$ with and without Integrating Factors using a fourth order ODE solver (RK4) for 2D NLS. 1000 nodes were used (100 in x and 10 in y) with 72 nearest neighbors, polynomial order 7, $T_x = 50$ and $T_y = 5$. Hyperviscosity was computed with $\gamma = 0.05$ and $\beta = 3$

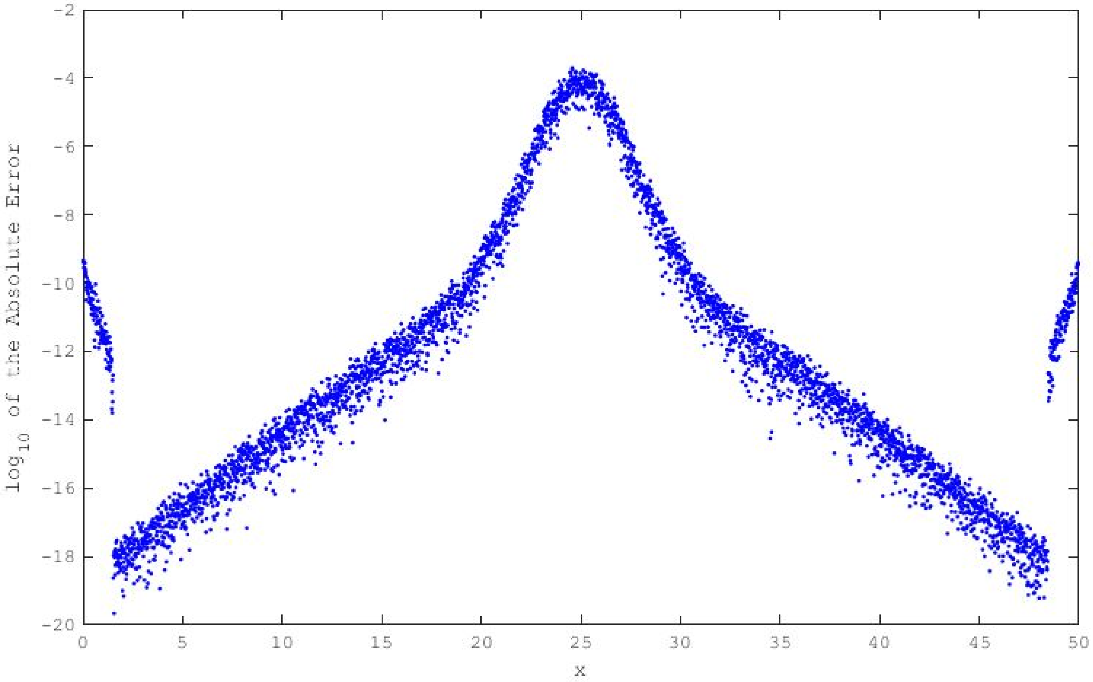


Figure 26. Error between D applied to the initial condition and $\Delta u(x, y, 0)$; 1000 nodes were used (100 in x and 10 in y) with 72 nearest neighbors, polynomial order 7, $T_x = 50$ and $T_y = 5$. The accuracy of solutions using higher order ODE solvers is limited by the spatial discretization, i.e. the RBF-FD matrix D

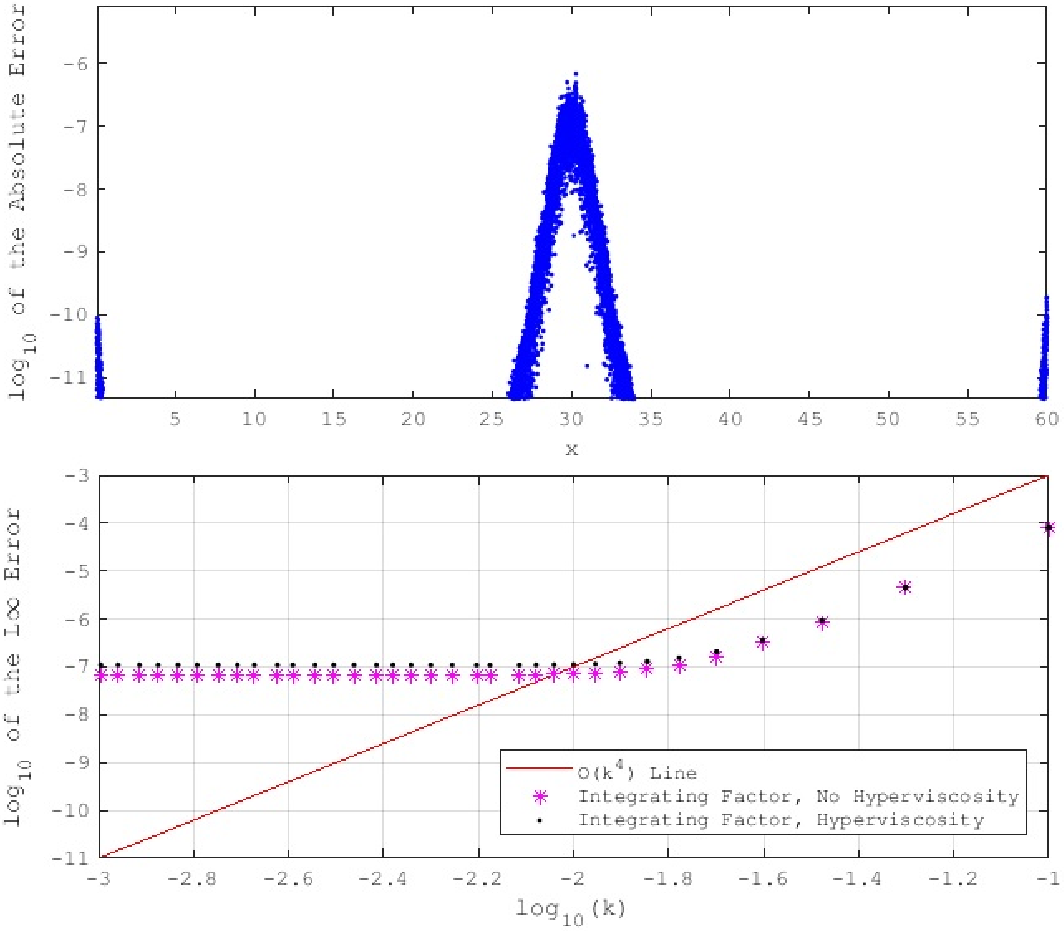


Figure 27. (TOP) Error between D applied to the initial condition and $\Delta u(x, y, 0)$; 36000 nodes were used (600 in x and 60 in y) with 132 nearest neighbors, polynomial order 10, $T_x = 60$ and $T_y = 6$. (BOTTOM) Comparison between numerical solutions at $t = 0.1$ with Integrating Factors using RK4 for 2D NLS; accuracy is still limited by the spatial discretization despite the fact that 36000 nodes were used. Hyperviscosity was computed using $\beta = 4$ and $\gamma = 2.5$

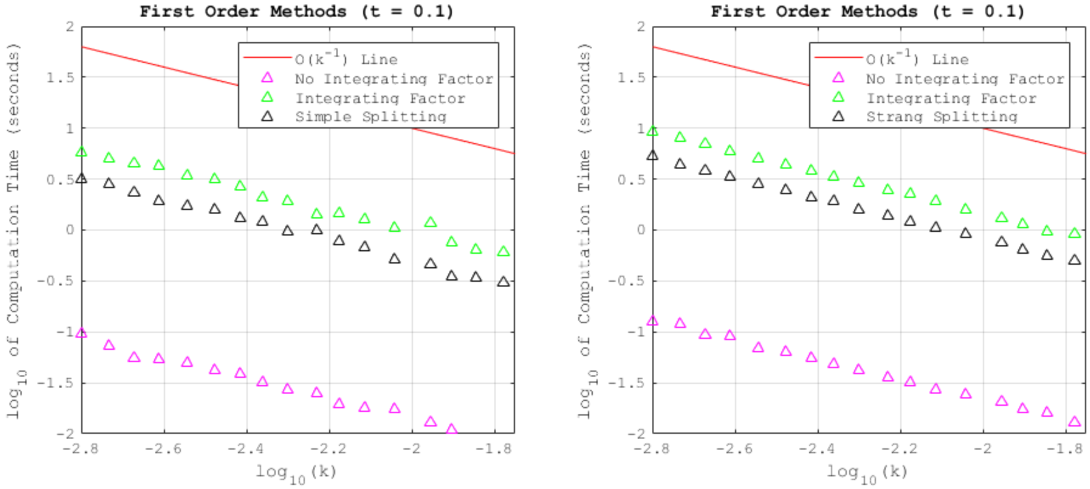


Figure 28. Timing results for several methods excluding hyperviscosity; the preprocessing costs of computing the RBF-FD matrix and its eigendecomposition were not accounted for. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. Solutions were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. Each method has rate $O\left(k^{-1}\right)$ for computation cost. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2.

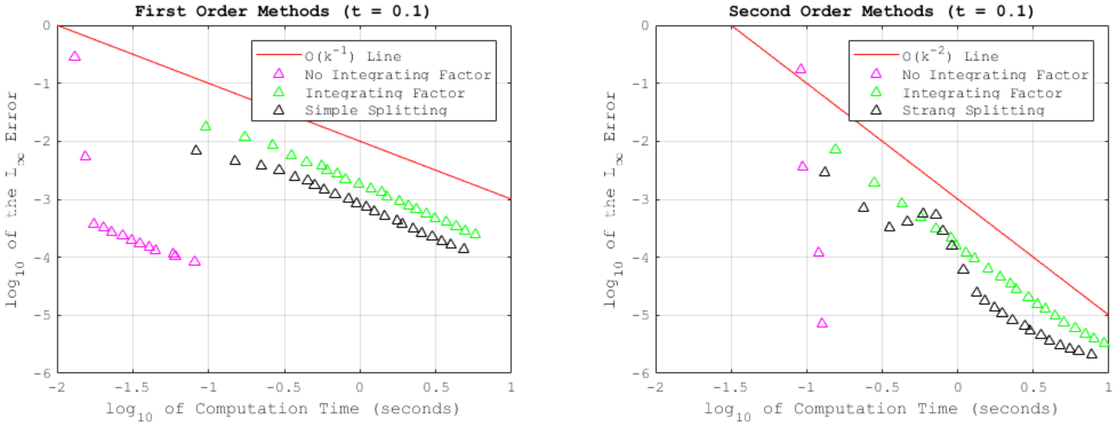


Figure 29. Computation time versus the L_∞ error for several methods—hyperviscosity was not used. 4000 nodes (200 in x, 20 in y), 130 nearest neighbors and a polynomial order of 10 were used to compute the RBF-FD matrix over $\mathbf{x} \in [0, T_x] \times [0, T_y]$ where $T_x = 50$ and $T_y = 5$. Solutions to the two-dimensional variant of the NLS equation were evolved in time from $t = 0$ to $t = 0.1$ for each method using different step sizes on a machine with an Intel Xeon E5-2687W v3 3.1 GHz 10-core (20 logical cores) processor. (LEFT) AB1-AM1 predictor-corrector method was used to propagate solutions in time for non-splitting methods. For the first order splitting scheme, an explicit Euler method was used to solve the nonlinear portion while an implicit Euler scheme was used to solve the linear portion. (RIGHT) RK2 was used to propagate solutions in time for non-splitting methods. For the Strang splitting scheme, RK2 was used to solve the nonlinear portion while BDF2 was used to solve the linear portion. Given that BDF2 is an implicit multi-step method, unknown starting values were determined using RK2.

IV. Conclusion

4.1 Concluding Remarks

RBFs can be used to numerically solve the NLS with reasonable accuracy because errors behave as expected based on their orders of accuracy when produced with various ODE solvers. Hyperviscosity, operator splitting and integration factors can be used to stabilize time-stepping schemes so that larger step sizes can be taken when using higher order methods. Results in chapter II are in close agreement with those in chapter III. Integrating factors and splitting schemes yield improvements in stability at the cost of computation time. Splitting schemes have lower computing costs and attain lower errors when compared to integration factors, making them superior in terms of computing costs and accuracy. Larger step sizes can be taken when hyperviscosity is employed at the expense of higher errors unless optimal values for β and γ are used.

4.2 Future Considerations

The use of hyperviscosity as a time stabilization method allows for smaller step sizes at the cost of increased errors. Deliberating the source of these errors is an important avenue for future research. Hyperviscosity was not computed using any polynomial terms in chapters II and III and the inclusion of polynomial terms can potentially have an effect on the behavior of the operator. Moreover, the work here only considered the use of GA RBFs to construct the hyperviscosity filter. Exploring the use of other RBFs to compute hyperviscosity should also be considered to determine the errors of hyperviscosity computed using different RBFs. Lastly, a parameter study should be conducted to determine optimal values for β and γ based off of the size and/or spectral gap of the RBF-FD matrix. Much of the work on hyperviscosity found in the literature simply gave ranges for optimal values based off of heuristics.

The work presented only made use of first and second order splitting schemes. Higher order operator splitting methods combined with RBF-FD should be considered in future work using the composition techniques discussed in [23] for potential improvements in accuracy.

Results produced in chapters II and III using higher order methods, especially fourth order methods, were found to be limited by the spatial discretization of the differential operator. Blindly increasing the size of the domain, the polynomial order, the number of RBF centers and nearest neighbors by means of a heuristic approach can yield a small improvement in the accuracy of solutions at the added expense of computation time. A parameter study should be conducted to determine an optimal combination of the polynomial order, number of RBF centers and number of nearest neighbors that can be used to construct the RBF-FD matrix in order to obtain a desired degree of accuracy. Periodic initial conditions should also be considered. Solving the NLS equation subject to periodic boundary conditions using a periodic initial condition may lead to an improvement in accuracy at the boundary of the domain, thus eliminating the need to increase the domain size.

Bibliography

1. G. Whitham, *Linear and Nonlinear Waves.* New York: John Wiley, 1974.
2. M. J. Ablowitz, *Nonlinear Dispersive Waves: Asymptotic Analysis and Solutions.* Cambridge: Cambridge University Press, 2011.
3. C. Sulem and P. Sulem, *The Nonlinear Schrodinger Equation: Self-Focusing and Wave Collapse.* New York: Springer Science and Business Media, 1999.
4. B. Fornberg and N. Flyer, *A Primer on Radial Basis Functions with Applications to the Geosciences.* Philadelphia: SIAM, 2015.
5. R. L. Hardy, “Multiquadric equations of topography and other irregular surfaces,” *J. Geophys. Res.*, vol. 76, pp. 1905–1915, 1971.
6. E. J. Kansa, “Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics - i: Surface approximations and partial derivative estimates,” *Comput. Math. Appl.*, vol. 19, pp. 127–145, 1990.
7. P. C. Jr., “n-parameter families and best approximation,” *Pacific Journal of Mathematics*, vol. 93, pp. 1013–1027, 1959.
8. B. Fornberg and J. Zuev, “The runge phenomenon and spatially variabled shape parameters in rbf interpolation,” *Comput. Math. Appl.*, vol. 54, pp. 379–398, 2007.
9. N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett, “On the role of polynomials in rbf-fd approximations: Interpolation and accuracy,” *Journal of Computational Physics*, vol. 321, pp. 21–38, 2016.
10. R. L. Burden and J. D. Faires, *Numerical Analysis.* New York: Brooks/Cole Publishing Company, sixth ed., 1997.

11. M. L. Ghrist, B. Fornberg, and J. A. Reeger, "Stability ordinates of adams predictor-corrector methods," *BIT Numerical Mathematics*, vol. 55, pp. 733–750, 2015.

12. G. Strang, "On the construction and comparison of different splitting schemes," *SIAM J. Numer. Anal.*, vol. 5, pp. 506–517, 1968.

13. N. Flyer, B. Fornberg, and G. B. Wright, "Radial basis function-generated finite differences: A mesh-free method for computational geosciences," *Handbook of Geomathematics*, pp. 1–30, 2013.

14. B. Fornberg and E. Lehto, "Stabilization of rbf-generated finite difference methods for convective pdes," *Journal of Computational Physics*, vol. 93, pp. 2270–2285, 2011.

15. J. A. C. Weideman and B. M. Herbst, "Split-step methods for the solution of the nonlinear schrodinger equation," *SIAM J. Numer. Anal.*, vol. 23, pp. 485–507, 1986.

16. H. Berland, B. Owren, and B. Skaflestad, "Solving the nonlinear schrodinger equation using exponential integrators," *Journal of Modeling, Identification and Control*, vol. 27, pp. 201–218, 2006.

17. T. I. Lakoba, "Instability of the finite-difference split-step method on the background of a soliton of the nonlinear schrodinger equation," *Numerical Methods for Partial Differential Equations*, vol. 32, pp. 1002–1023, 2016.

18. R. Zhang, J. Zhu, X. Yu, M. Li, and A. F. D. Loula, "A conservative spectral collocation method for the nonlinear schrodinger equation in two dimensions," *Applied Mathematics and Computation*, vol. 310, pp. 194–203, 2017.

19. Y. Dereli, D. Irk, and I. Dag, “Soliton solutions for nls equation using radial basis functions,” *Chaos, Solitons and Fractals*, vol. 42, pp. 1227–1233, 2009.

20. S. Duo and Y. Zhang, “Mass-conservative fourier spectral methods for solving the fractional nonlinear schrodinger equation,” *Computers and Mathematics with Applications*, vol. 71, pp. 2257–2271, 2016.

21. L. R. T. Gardner, G. A. Gardner, S. I. Zaki, and Z. E. Sahrawi, “B-spline finite element studies of the nonlinear schrodinger equation,” *Computer Methods in Applied Mechanics and Engineering*, vol. 108, pp. 303–318, 1993.

22. N. Flyer, E. Lehto, S. Blaise, G. 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*, vol. 231, pp. 4078–4095, 2012.

23. H. Yoshida, “Construction of higher order symplectic integrators,” *Physics Letters A*, vol. 150, 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*
22–03–2018

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From — To)*
Sept 2016 — Mar 2018

4. TITLE AND SUBTITLE
Radial Basis Function Generated Finite Differences
for the Nonlinear Schrodinger Equation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Ng, Justin, Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENC-MS-18-M-004

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
OFFICE OF NAVAL RESEARCH
875 N. Randolph St.
Arlington VA 22203-1995

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A.
Approved for Public Release; distribution unlimited.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Solutions to the one-dimensional and two-dimensional nonlinear Schrodinger (NLS) equation are obtained numerically using methods based on radial basis functions (RBFs). Periodic boundary conditions are enforced with a non-periodic initial condition over varying domain sizes. The spatial structure of the solutions is represented using RBFs while several explicit and implicit iterative methods for solving ordinary differential equations (ODEs) are used in temporal discretization for the approximate solutions to the NLS equation. Splitting schemes, integration factors and hyperviscosity are used to stabilize the time-stepping schemes and are compared with one another in terms of computational efficiency and accuracy. This thesis shows that RBFs can be used to numerically solve the NLS with reasonable accuracy. Integration factors and splitting methods yield improvements in stability at the cost of computation time; both methods produce solutions of similar accuracy while splitting methods are slightly less expensive to implement than integration factors (computation times were of the same order of magnitude). The use of hyperviscosity can lead to an improvement in stability but can also lead to increased errors if the relevant parameters are not chosen carefully.

15. SUBJECT TERMS
Radial Basis Functions, RBF, Nonlinear Schrodinger Equation, NLS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj. Jonah. A. Reeger, AFIT/ENC
U	U	U	UU	77	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, x3320; jonah.reeger@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18