Numerical and Analytical Exploration of the Semi-Discrete Massive Thirring Model: Thesis Presented in Fulfillment of the Bachelor of Science Thesis Project

Leeor Greenblat

Supervisor: Dr. Pelinovsky

April 10, 2019

Abstract

The massive Thirring system is a completely integrable system of nonlinear partial differential equations in (1+1) dimensions. Computational approaches to understanding the solutions of this system require that the spatial variable be discretized. In a recent paper, Joshi and Pelinovsky (2019) obtain an integrable semi-discretization of the massive Thirring system in laboratory coordinates. In this thesis, we use von Neumann stability analysis to demonstrate that while explicit methods of numerically solving this system are unconditionally unstable, there are alternative, neutrally stable, semi-implicit schemes that are available. Furthermore, we generate a numerical scheme that begins with initial data and inverts two finite difference equations before using a semi-implicit solver to find the solution at the next timestep. Despite the unconditional stability of this implicit solver, the error between the numerical and exact solution grows with each iteration, preventing the acquisition of stable numerical solutions.

Contents

1 Introduction to the Massive Thirring Model

The massive Thirring model (MTM) was released in 1958 by Walter Thirring after many years of stagnation in relativistic field theory [10]. This model can be understood as a nonlinear adaptation of the Dirac equation [2]. In (1 + 1) dimensions and in laboratory coordinates, the MTM consists of two nonlinear partial differential equations

$$i\left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right) + v = |v|^2 u \tag{1}$$

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

where $u(x,t), v(x,t) : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$. Prior to the release of the MTM, quantum field models such as the Lee model, static neutral scalar theory, and the static isotropic spin independent pair theory only pertained to systems in the non-relativistic limit [10]. The MTM differs in this regard as it is in fact a relativistic field theory.

The MTM is a completely integrable system. This was shown in the work of Kuznetsov and Mikhailov in [6] in which the Hamiltonian of the system was expressed purely with canonical momenta. This implies several properties, including the existence of exact N-soliton solutions, an infinite number of conserved quantities, and the existence of Lax operators [1]. Furthermore, it also ensures that a Backlund transformation exists between solutions [3]. Given these interesting properties, it is desirable to be able to study the solutions of the MTM numerically. To do this, an integrable semi-discretization of the MTM is required. While discretizations of the MTM exist in characteristic coordinates [7, 11], no integrable discretization of the MTM in laboratory coordinates has been available until Joshi and Pelinovsky obtained one in [5]. This semi-discretization, in which time remains continuous and the spatial variable is discretized, is given by

$$\begin{cases} 4i\frac{dU_n}{dt} + Q_{n+1} + Q_n + \frac{2i}{h}(R_{n+1} - R_n) + U_n^2(\bar{R}_n + \bar{R}_{n+1}) \\ \quad - U_n(|Q_{n+1}|^2 + |Q_n|^2 + |R_{n+1}|^2 + |R_n|^2) - \frac{ih}{2}U_n^2(\bar{Q}_{n+1} - \bar{Q}_n) = 0 \\ -\frac{2i}{h}(Q_{n+1} - Q_n) + 2U_n - |U_n|^2(Q_{n+1} + Q_n) = 0 \\ R_{n+1} + R_n - 2U_n + \frac{ih}{2}|U_n|^2(R_{n+1} - R_n) = 0. \end{cases} \tag{3}$$

In this semi-discretization, the spatial coordinate is evaluated on a lattice of grid points given by $x_h = hn$ where h denotes the spacing between lattice sites and n is an integer. One natural and necessary condition that must be satisfied by (3) is that as h approaches 0, (1) - (2) are recovered. As h approaches 0, the second equation in (3) is

$$-2i\frac{\partial Q}{\partial x} + 2U - 2|U|^2 Q = O(h).$$

In the limit as h approaches 0 we obtain

$$-i\frac{\partial Q}{\partial x} + U - |U|^2 Q = 0. \tag{4}$$

Likewise, for the third equation of (3), we get the following in the continuum limit:

$$2R - 2U = O(h)$$

so that

$$R = U.$$

Finally, for the first equation of (3), we get the following in the continuum limit:

$$2i\frac{\partial U}{\partial t} + Q + i\frac{\partial R}{\partial x} + U^2\bar{R} - U|Q|^2 - U|R|^2 = O(h).$$

Replacing R with U in the above equation, we obtain, in the limit as h approaches 0,

$$2i\frac{\partial U}{\partial t} + Q + i\frac{\partial U}{\partial x} - U|Q|^2 = 0. \tag{5}$$

Let us now set $U(x,t) = u(x,t-x)$ and $Q(x,t) = v(x,t-x)$. Then, the partial derivatives of U and Q are obtained by using the chain rule:

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{\partial u}{\partial t} \\
\frac{\partial U}{\partial x} &= \frac{\partial u}{\partial x} - \frac{\partial u}{\partial t} \\
\frac{\partial Q}{\partial t} &= \frac{\partial v}{\partial t} \\
\frac{\partial Q}{\partial x} &= \frac{\partial v}{\partial x} - \frac{\partial v}{\partial t}.
\end{aligned}$$

Making these replacements into (4) yields

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

which is the same as (2). Likewise, making the same replacements in (5) yields

$$i\left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right) + v = |v|^2 u$$

which is the same as (1). Thus we can conclude that the semi-discretization (3) returns the MTM (1) - (2) in the continuum limit.

2 Stability of the Time Evolution near the Zero Equilibrium

In this section, we discuss the stability of the solutions, U_n, Q_n, and R_n, near the zero equilibrium. Under this condition, (3) can be linearized as follows

$$\begin{cases}
4i\frac{dU_n}{dt} + Q_{n+1} + Q_n + \frac{2i}{h}(R_{n+1} - R_n) = 0 \\
-\frac{2i}{h}(Q_{n+1} - Q_n) + 2U_n = 0 \\
R_{n+1} + R_n - 2U_n = 0.
\end{cases} \tag{6}$$

Recall that the given a system of linear partial differential equations, the general solution can be constructed using a linear combination of separable solutions [8]. To assess the time evolution of the solutions of the semi-discrete linear system, we consider a single mode in a possible linear combination. If an arbitrary mode grows in time, we can conclude that the solution is not stable. These modes are of the form

$$\begin{aligned} U_n &= \hat{U}e^{in\theta}e^{-i\omega t} \\ R_n &= \hat{R}e^{in\theta}e^{-i\omega t}, \\ Q_n &= \hat{Q}e^{in\theta}e^{-i\omega t} \end{aligned}$$

where θ ranges from 0 to π. In substituting these solutions into the linearized system, we obtain a dispersion relation, $\omega(\theta)$, which encodes information about the behavior of a given mode. There are three possibilities: Im(ω) > 0, Im(ω) $= 0$, Im(ω) < 0.

If Im(ω) > 0 for a real value of θ, the mode grows exponentially in time and the solution is unstable. Conversely, if Im(ω) < 0 the mode decays exponentially in time, and the solution is stable. Finally, when Im(ω) = 0, the solution does not decay nor grow, but generally propagates as a linear wave [1]. Substituting these solutions into (6), we generate the following system of equations.

$$\begin{bmatrix} 4\omega & e^{i\theta}+1 & \frac{2i}{h}(e^{i\theta}-1) \\ 2 & \frac{-2i}{h}(e^{i\theta}-1) & 0 \\ -2 & 0 & e^{i\theta}+1 \end{bmatrix} \begin{bmatrix} \hat{U} \\ \hat{Q} \\ \hat{R} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

If we are to avoid the trivial solution, $(\hat{U}, \hat{Q}, \hat{R}) \neq (0,0,0)$, we must ensure the determinant of the matrix is 0. Thus, (7) is true if and only if det(A) = 0. That is if

$$\frac{8}{h^2}(e^{i\theta}-1)^2 + (e^{i\theta}+1)\Big[-2(e^{i\theta}+1) - \frac{8i\omega}{h}(e^{i\theta}-1)\Big] = 0.$$

Using the identities

$$\sin(\theta) = \frac{e^{i\theta} - e^{-i\theta}}{2i} \qquad \cos(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

we obtain the following equation

$$-\frac{32}{h^2}\sin^2\left(\frac{\theta}{2}\right)e^{i\theta} - 8e^{i\theta}\cos^2\left(\frac{\theta}{2}\right) - \frac{8i\omega}{h}2ie^{i\theta}\sin(\theta) = 0$$

which can be simplified to

$$\omega\sin\theta = \frac{h}{16}\left[\frac{32}{h^2}\sin^2\left(\frac{\theta}{2}\right) + 8\cos^2\left(\frac{\theta}{2}\right)\right]$$

$$\omega = \frac{h}{2\sin\theta}\left[\cos^2\left(\frac{\theta}{2}\right) + \frac{4}{h^2}\sin^2\left(\frac{\theta}{2}\right)\right] \in \mathbb{R}$$

ω is a real number, and thus Im(ω) = 0. From this, we can conclude that near the zero equilibrium, the Fourier mode will not grow or decay, but propagate as a linear wave.

3 Numerical Investigations of the Semi-Discrete MTM

The goal of the numerical simulation was to obtain stable numerical solutions $U(x = hn, t), Q(x = hn, t)$, and $R(x = hn, t)$ to (3) for time $t > 0$ beginning with initial data $U(x = hn, 0)$ at $t = 0$. The initial data $U(x = hn, 0)$ was obtained by evaluating the exact solution of the the semi-discrete MTM at $t = 0$. This solution, given by (8), is a soliton and was obtained by Pelinovsky and Xu using the Backlund-Darboux transformation in [9].

$$U = \frac{2i\bar{\lambda}\bar{\gamma}(\lambda^2 - \bar{\lambda}^2)e^{\bar{\eta}(t)+\xi(t)}}{\lambda^2(h\bar{\lambda}^2 - 2i)e^{\bar{\xi}(t)+\xi(t)} + |\gamma|^2|\lambda|^2(h\lambda^2 - 2i)e^{\bar{\eta}(t)+\eta(t)}} \tag{8}$$

where $\xi_n(t)$ and $\eta_n(t)$ are given by

$$\xi_n(t) = n\log\left(\lambda + \frac{2i}{h\lambda}\right) + \frac{1}{2}i\lambda t^2 \qquad \eta_n(t) = n\log\left(-\lambda + \frac{2i}{h\lambda}\right) - \frac{it}{2\lambda^2}$$

and λ affects the characteristic width of the soliton. The first part of the numerical procedure was to obtain Q and R at $t = 0$. This was achieved by inverting the second and third equations of (3) under Dirichlet boundary conditions. The function used for this inversion can be found in Appendix C. With $U, Q,$ and R known at $t = 0$, one can proceed to find U at the next timestep using the first equation of (3) . At this point, the newly computed U is used to obtain R and Q. This process repeats itself until the procedure has completed the specified number of iterations. As is seen in the next section, explicit calculation U at the next timestep using Euler's method or Heun's method is a numerically unstable procedure. That is, any numerical error present in the initial conditions accumulates with each timestep. This is in contrast to a stable or neutrally stable procedure, in which errors decay or remain constant respectively with each iteration [4].

3.1 Stability of the Explicit Scheme

The stability of the evolutionary step is analyzed using von Neumman stability analysis [4]. This method of stability analysis works by expressing the numerical solution of (6) as a Fourier series

$$U_n = \sum_{i=1}^{M} \hat{U}(t)e^{in\theta}, \quad R_n = \sum_{i=1}^{M} \hat{R}(t)e^{in\theta}, \quad Q_n = \sum_{i=1}^{M} \hat{Q}(t)e^{in\theta}.$$

If the coefficient, $\hat{U}(t)$ grows in time, we can conclude that the numerical solutions and corresponding error experience growth as well. This indicates an unstable numerical scheme. We define the amplification factor G as the ratio of amplitude of a mode at the next timestep to the amplitude at the current timestep. That is

$$G = \frac{\hat{U}^{k+1}}{\hat{U}^{k}}.$$

Inserting a single mode of U and R into the third equation of (6) , we see that $\hat{R}$ is related to $\hat{U}$ by

$$\hat{R} = \frac{2\hat{U}}{e^{i\theta} + 1}.$$

Likewise, inserting a single mode of $\hat{Q}$ and $\hat{U}$ into the second equation of (6), we see that $\hat{U}$ and $\hat{Q}$ are related by

$$\hat{Q} = \frac{-ih\hat{U}}{e^{i\theta} - 1}.$$

These relations are then inserted into the first equation of (6), yielding

$$\frac{d\hat{U}}{dt} = \left[\frac{h}{4}\frac{e^{i\theta}+1}{e^{i\theta}-1} - \frac{1}{h}\frac{e^{i\theta}-1}{e^{i\theta}+1}\right]\hat{U} = -i\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]\hat{U}.$$

Replacing the derivative with a finite difference yields

$$\frac{\hat{U}^{k+1} - \hat{U}^{k}}{\tau} = -i\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]\hat{U}^{k}.$$

Under Euler's method, we see that $\hat{U}^{k+1}$ at the next timestep is

$$\hat{U}^{k+1} = \hat{U}^{k} - i\tau\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]\hat{U}^{k} \tag{9}$$

Under this scheme, the modulus of G is

$$|G| = \sqrt{1 + \tau^2\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]^2}$$

which exceeds 1 for all τ. Thus, we can conclude that the Euler explicit method is unconditionally unstable. Under Heun's method, the explicit scheme is

$$\hat{U}^{k+1} = \hat{U}^{k} - \frac{i\tau}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]\hat{U} - \frac{i\tau}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]\hat{U}^{k+1}.$$

Replacing $\hat{U}^{k+1}$ using (9), we see that G is given by

$$G = 1 - i\tau\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right] - \frac{\tau^2}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]^2$$

and likewise that $|G|$ is given by

$$|G| = \sqrt{1 + \frac{\tau^4}{4}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]^4}.$$

Again $|G| > 1$, thus the explicit Heun's method is unconditionally unstable. Therefore, an implicit scheme is required for the evolution step.

3.2 Stability of the Semi-Implicit Scheme

In the semi-implicit scheme, we start with

$$\frac{\hat{U}^{k+1} - \hat{U}^k}{\tau} = -\frac{i}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right](\hat{U}^{k+1} + \hat{U}^k).$$

The corresponding amplification factor is

$$G = \frac{1 - \frac{i\tau}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]}{1 + \frac{i\tau}{2}\left[\frac{h}{4}\cot\left(\frac{\theta}{2}\right) + \frac{1}{h}\tan\left(\frac{\theta}{2}\right)\right]}$$

so that $|G| = 1$. From this, we can conclude that an implicit scheme is neutrally stable. Since the implicit method is stable, we devised a semi-implicit numerical scheme which is given by

$$4i\frac{U^{k+1} - U^k}{\tau} + \frac{1}{2}(Q_{n+1}^{k+1} + Q_n^{k+1}) + \frac{1}{2}(Q_{n+1}^k + Q_n^k) + \frac{i}{h}(R_{n+1}^{k+1} - R_n^{k+1}) + \frac{i}{h}(R_{n+1}^k - R_n^k) = F_n^k \tag{10}$$

where F_n^k is given by

$$F_n^k = U_n(|Q_{n+1}|^2 + |Q_n|^2 + |R_{n+1}|^2 + |R_n|^2) + \frac{ih}{2}U_n^2(\bar{Q}_{n+1} - \bar{Q}_n) - U_n^2(\bar{R}_n + \bar{R}_{n+1}).$$

In practice, however, the future values of U are placed on the left hand side, and the current values of U are placed on the right hand side. This yields

$$U_n^{k+1} + \frac{\tau}{8i}(Q_{n+1}^{k+1} + Q_n^{k+1}) + \frac{\tau}{4h}(R_{n+1}^{k+1} - R_n^{k+1}) = U_n^k - \frac{\tau}{8i}(Q_{n+1}^k + Q_n^k) - \frac{\tau}{4h}(R_{n+1}^k - R_n^k) + \frac{\tau}{4i}F_n^k.$$

The implementation of the scheme can be found in Appendix A.

3.3 Growth of the Error in Numerical Solutions

Despite the fact that $|G| = 1$ for all modes, the error between the numerical and exact solutions increased with each iteration. The error was computed as the Euclidean norm of the quantity $U - U_{exact}$ where U is the numerical solution and U_{exact} is the solution solving (3) given by (8).

To see the growth in error, consider the following the example. In this example, we define the computational domain to extend from -15 to 15. The number of lattice points in the computational domain was selected to be 81, giving a lattice spacing of 0.375. The initial data was obtained by evaluating (8) at $t = 0$ with the parameters $\lambda = \frac{1+i}{\sqrt{2}}$ and $\gamma = 1$. The plot of this graph can be found in Figure 1.

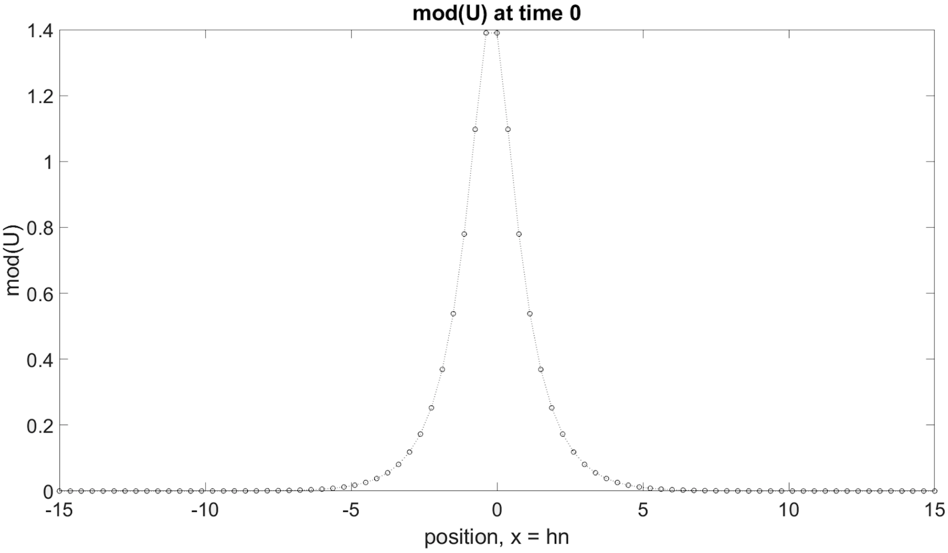


Figure 1: Plot of the initial data at time 0.

In this example, we set the timestep τ to be 0.0002 and allowed the procedure to run for 1000 iterations. The error between the exact solution and the numerical solution was recorded after each iteration. As can be seen in Figure 2, the error grew with each iteration.

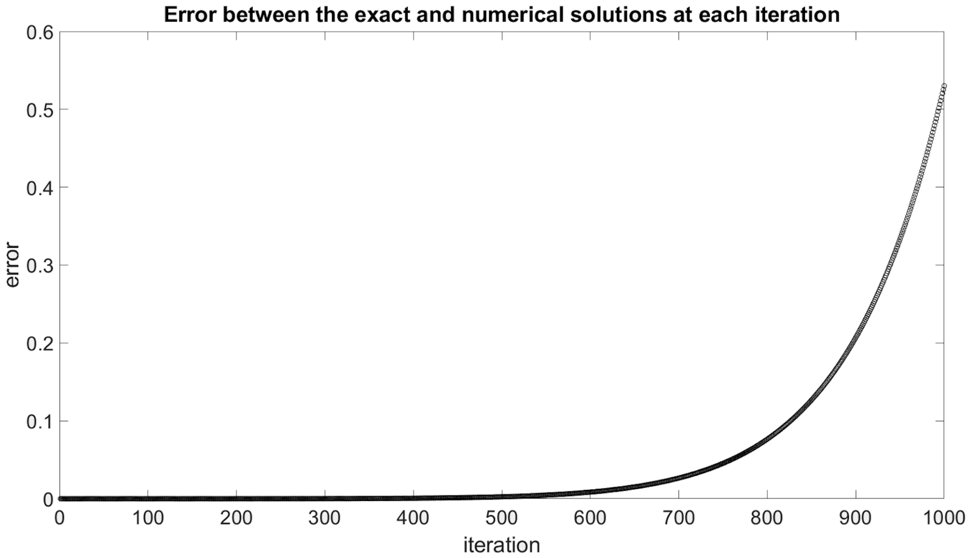


Figure 2: The Euclidean norm of the difference between the exact solution and the numerical solution. There is a drastic increase in the error.

The source of the error, as can be seen in Figure 3, is due to the growth in the solution at the left end of the computational domain.

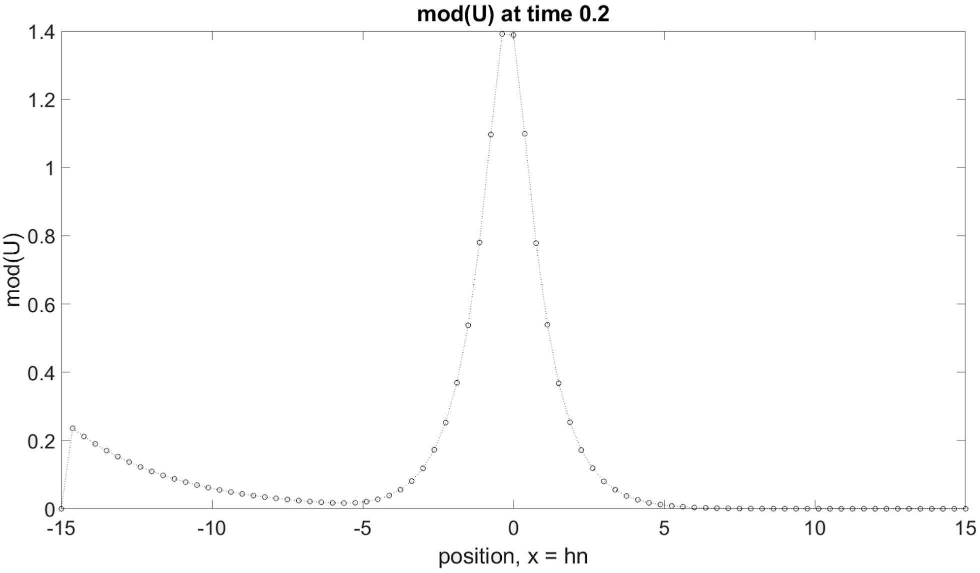


Figure 3: The plot of the modulus of the numerical solution at time 0.2, after 1000 iterations.

Ideally, the solution at the left side of the domain would look similar to the solution at the right end of the domain. In Figure 4, we see both the numerical and exact solutions for U at time 0.2.

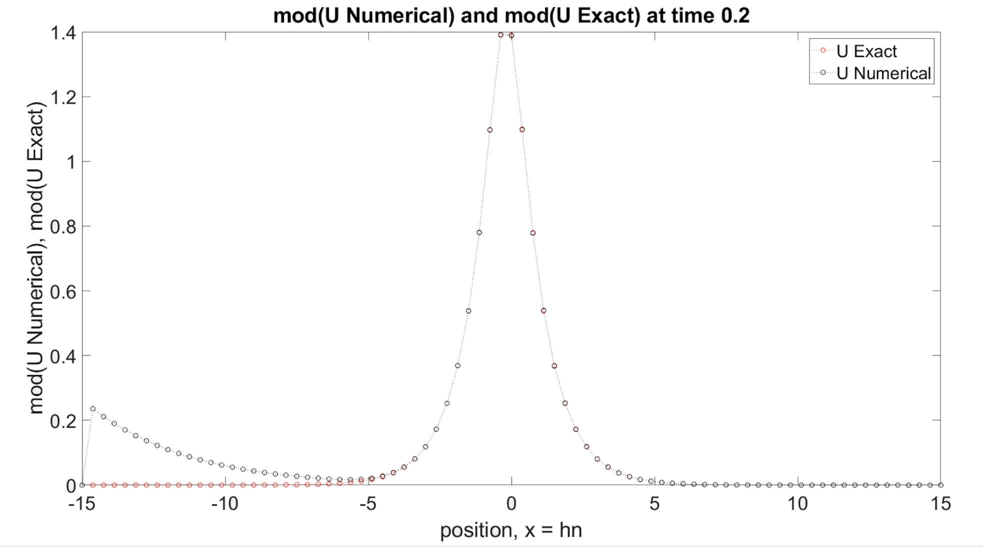


Figure 4: The modulus of the exact solution for U in red vs the modulus of numerical solution for U in black. By time 0.2, a noticeable difference has accumulated between the two solutions. The total error at this point was 0.5301.

Given the stability of the scheme established in section 3.2, other sources of error were considered to explain why a dissonance between the numerical and exact solutions was observed. It is believed that the source of the error is from truncation error arising as a result of operating upon a finite computational domain.

3.4 Absorbing Layers

To address this possibility, absorbing layers were incorporated into the numerical scheme with the intention of damping the growth of the solution at the edge of the domain. This involved selecting a damping parameter, δ, as well as a depth, r, over which to apply the absorbing layers. However, as the below data indicate, this was futile in suppressing the growth of the error.

Error between the exact and numerically computed solution ($|U - U_{exact}|^2$)

Strength (δ) / Depth (r)	0.01	0.1	1	10
1 grid point	4.8215×10^{-4}	4.78×10^{-4}	4.3787×10^{-4}	1.4397×10^{-4}
2 grid points	4.8281×10^{-4}	4.8457×10^{-4}	5.0152×10^{-4}	6.2645×10^{-4}
3 grid points	4.8159×10^{-4}	4.7244×10^{-4}	3.8493×10^{-4}	2.1015×10^{-4}
4 grid points	4.8353×10^{-4}	4.9173×10^{-4}	5.6991×10^{-4}	1.1000×10^{-3}
5 grid points	4.8018×10^{-4}	4.5839×10^{-4}	2.5177×10^{-4}	1.1000×10^{-3}

Each cell corresponds to the error between the numerical solution and the exact solution for a given (δ, r). The error between the numerical solution and the exact solution in the absence of any absorbing layers was 4.8261×10^{-4}. The data in this table corresponds to a timestep of 0.0002 with 1000 iterations. The domain extended from -15 to 15 and the number of lattice sites used was 51, corresponding to a lattice spacing, h, of 0.6. The exact solution has $\lambda = \frac{1+i}{\sqrt{2}}$ and $\gamma = 1$.

As is seen in the above table, moving horizontally across a given row does not result in a significant reduction in the error. This implies that increasing the damping factor does not conceal the truncation error. Likewise, moving down a given column does not result in a significant change in the error. This shows that increasing the depth over which the absorbing layers are applied has no significant effect on error growth. Due to the lack of success using absorbing layers, a second, slightly improved numerical scheme was devised in which the terms $U_n^2(\bar{R}_n + \bar{R}_{n+1})$ and $-\frac{ih}{2}U_n^2(\bar{Q}_{n+1} - \bar{Q}_n)$ in the first equation of (3) are calculated in the implicit phase instead of being calculated along with the terms in F_n^k. The corresponding implicit scheme is given by

$$\begin{aligned} U_n^{k+1} &+ \frac{\tau}{8i}(Q_{n+1}^{k+1} + Q_n^{k+1}) + \frac{\tau}{4h}(R_{n+1}^{k+1} - R_n^{k+1}) + \frac{\tau}{8i}U_n^2(\bar{R}_{n+1} + \bar{R}_n) - \frac{\tau h}{16}U_n^2(\bar{Q}_{n+1} - \bar{Q}_n) \\ &= U_n^k - \frac{\tau}{8i}(Q_{n+1}^k + Q_n^k) - \frac{\tau}{4h}(R_{n+1}^k - R_n^k) - \frac{\tau}{8i}U_n^2(\bar{R}_{n+1} + \bar{R}_n) + \frac{\tau h}{16}U_n^2(\bar{Q}_{n+1} - \bar{Q}_n) + \frac{\tau}{4i}F_n^k \end{aligned}$$

where F_n^k is given by

$$F_n^k = U_n(|Q_{n+1}|^2 + |Q_n|^2 + |R_{n+1}|^2 + |R_n|^2).$$

The implementation of this scheme can be found in Appendix B. Under this new scheme, very similar results were obtained. Beginning with the same initial data depicted in Figure 1, with a computational domain extending form -15 to 15 containing 81 grid points (h = 0.375), we see that after 1000 iterations with timestep 0.0002, the emergence of the same phenomenon at the left hand side of the domain as can be seen in Figure 5.

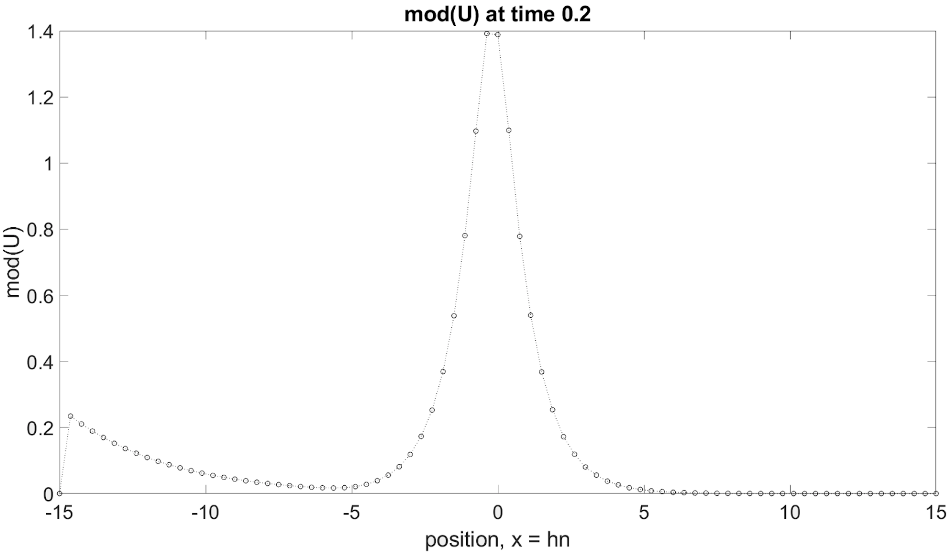


Figure 5: The plot of the modulus of the numerical solution at time 0.2, after 1000 iteration. This result was obtained using the improved numerical scheme.

As was observed in the numerical scheme given by (10), a similar profile for error growth was observed. The error at each iteration can be seen in Figure 6.

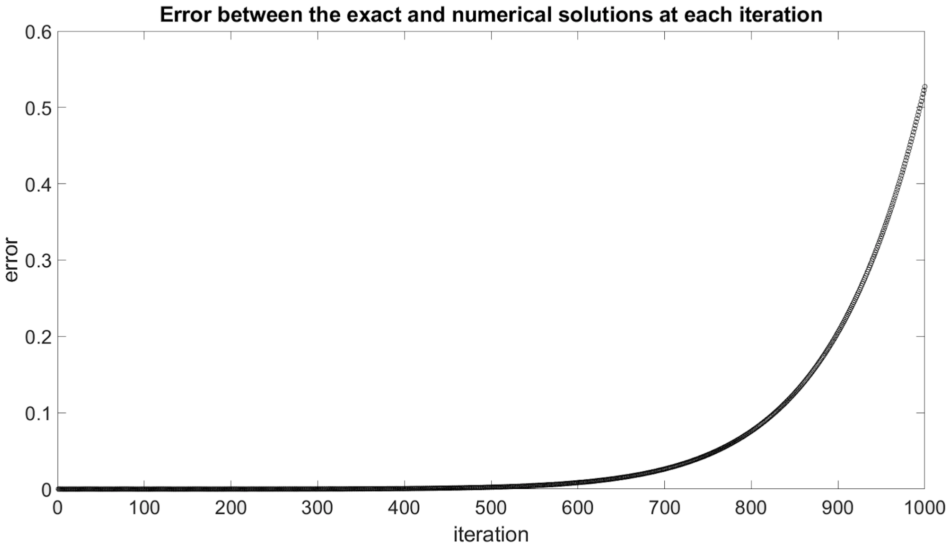


Figure 6: The Euclidean norm of the difference between the exact solution and the numerical solution. This result was obtained using the improved numerical scheme.

As before, the source of the error comes the disparity between the exact solution and the numerical solution at the left side of the domain, as can be seen in Figure 7.

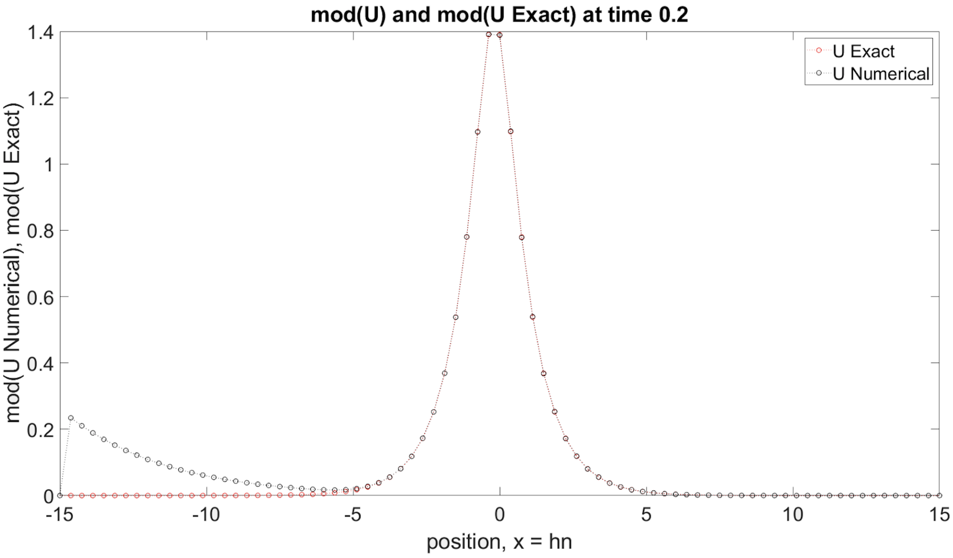


Figure 7: A comparison of the numerical solution of U in black and the exact solution for U in red obtained using the improved numerical scheme. The error between the two solutions after 1000 iterations was 0.5272.

This modification did not result in a reduction in error, even with the use of absorbing layers. In the below table, we show the error for a given strength, δ, and depth, r, as was done for the original semi-implicit scheme (10).

Error between the exact and numerically computed solution ($|U - U_{exact}|^2$)

Depth (r) \ Strength (δ)	0.01	0.1	1	10
1 grid point	5.9451 $\times 10^{-4}$	5.9041 $\times 10^{-4}$	5.5089 $\times 10^{-4}$	2.7348 $\times 10^{-4}$
2 grid points	5.9514 $\times 10^{-4}$	5.9664 $\times 10^{-4}$	6.112 $\times 10^{-4}$	7.2156 $\times 10^{-4}$
3 grid points	5.9395$\times 10^{-4}$	5.8487 $\times 10^{-4}$	4.9848 $\times 10^{-4}$	1.8778 $\times 10^{-4}$
4 grid points	5.9581 $\times 10^{-4}$	6.0337 $\times 10^{-4}$	6.7587 $\times 10^{-4}$	1.2000 $\times 10^{-3}$
5 grid points	5.9258$\times 10^{-4}$	5.7121 $\times 10^{-4}$	3.7274 $\times 10^{-4}$	9.8779 $\times 10^{-4}$

Each cell corresponds to the error between the numerical solution and the exact solution for a given (δ, r). The error between the numerical solution and the exact solution in the absence of any absorbing layers was 5.9497×10^{-4}. The data in this table corresponds to a timestep of 0.0002 with 1000 iterations. The domain extended from -15 to 15 and the number of lattice sites used was 51, corresponding to a lattice spacing, h, of 0.6. The exact solution has $\lambda = \frac{1+i}{\sqrt{2}}$ and $\gamma = 1$.

As was the case for the original semi-implicit scheme, moving across a given row or down a given column does not result in a significant reduction in error.

4 Conclusion

In this thesis, we have shown that near the zero-equilibrium, the solutions of (3) propagate as a linear wave. We have further shown that explicit methods of obtaining numerical solutions to the semi-discrete MTM are unstable, but that a neutrally stable, semi-implicit scheme exists. Despite the stability of the scheme, the error between the numerical and exact solutions could not be contained. Applying absorbing layers to the edges of the computational domain did not improve the results. Further researchers may wish to determine the precise source of the error being observed. A deeper analysis of the exact solutions of (3) may provide insight regarding the growth in the error of the numerical solution. Finally, different numerical schemes may be employed in obtaining solutions in addition to the ones tried in this investigation.

5 Acknowledgements

Many thanks go to Dr. Pelinovsky for providing guidance and technical expertise throughout the project.

References

[1] M. Ablowitz, and H. Segur, *Solitons and the inverse scattering transform*, Society for Industrial and Applied Mathematics, Philadelphia, 1981.

[2] N. Boussad, and A. Comech, *On spectral stability of the nonlinear Dirac equation*, Journal of Functional Analysis. **271** (2016) 1462-1524.

[3] A. Contreras, D.E Pelinovsky, and Y. Shimabukuro, *L^2 orbital stability of Dirac solitons in the massive Thirring model*, Communications in Partial Differential Equations. **41** (2016), 227-225.

[4] E. Isaacson and H.B. Keller, *Analysis of numerical methods*, Dover Publications, New York, 1994.

[5] N. Joshi, and D.E Pelinovsky, *Integrable semi-discretization of the massive Thirring system in laboratory coordinates*, J. Phys. A: Math. Theor. **52** (2019) 03LT01 .

[6] E.A. Kuznetzov and A.V. Mikhailov, *On the complete integrability of the two-dimensional classical Thirring model*, Theor. Math. Phys. **30** (1977), 193200.

[7] F.W. Nijhoff, H.W. Capel, and G.R.W. Quispel, *Integrable lattice version of the massive Thirring model and its linearization*, Phys. Lett. A **98** (1983), 8386.

[8] J. Oliver and P. Rosenau, *The construction of special solutions to partial differential equations.* Physics Letters. **114A** (1986) no. 3, 107-112.

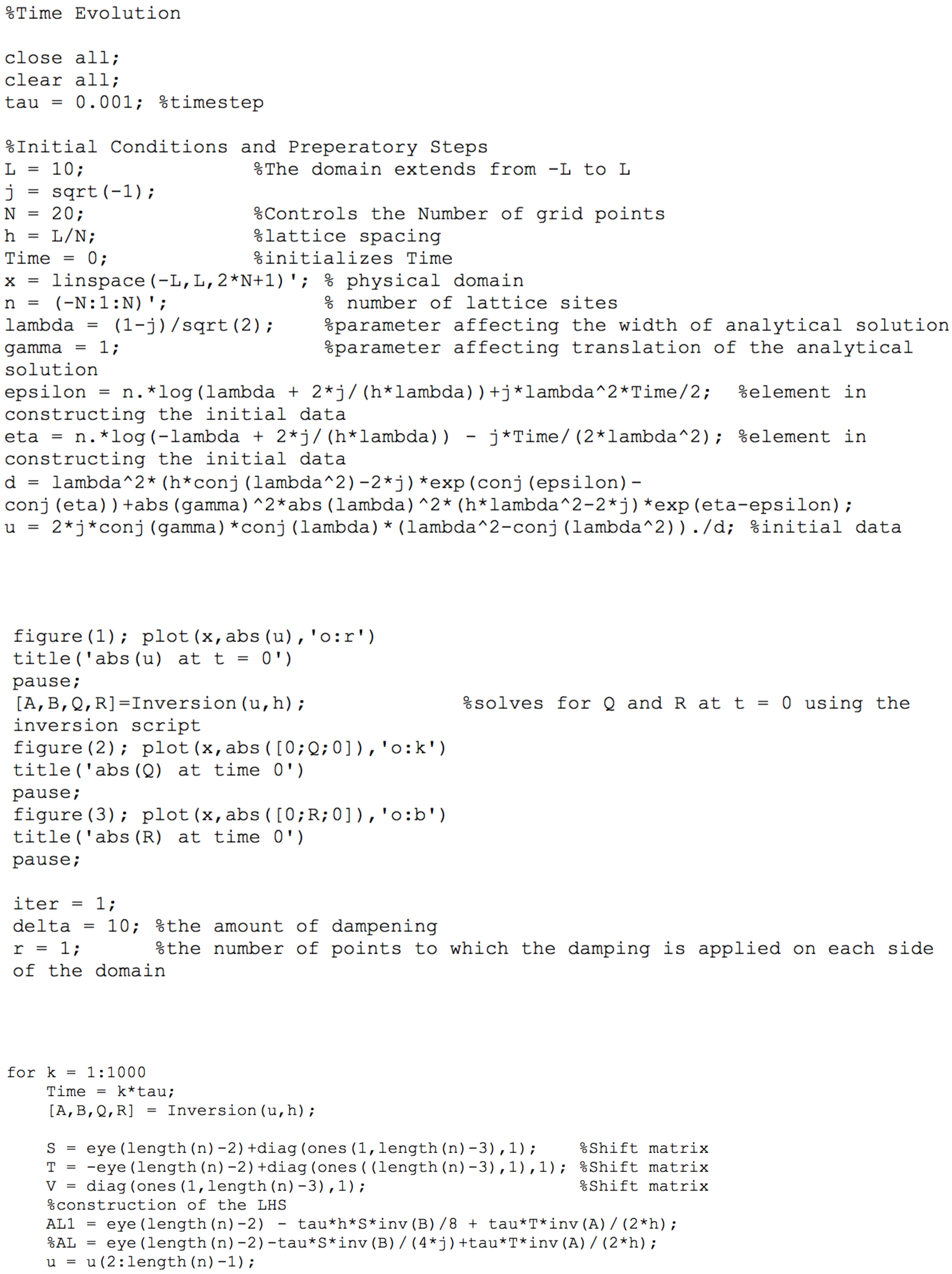
[9] D.E. Pelinovsky and T. Xu, *Darboux transformation and analytical solutions of semi-discretized massive Thirring model*, (2019).

[10] W.E. Thirring, *A soluble relativistic field theory*,Annals of Physics. **3** (1958) 91-112.

[11] T. Tsuchida, *Integrable discretizations of derivative nonlinear Schrdinger equations*, J. Phys. A: Math. Gen. **35** (2002) 7827.

A Time Evolution Script (Original)

Below is an image of the original semi-implicit scheme used in finding U at the next timestep.



```
%Time Evolution

close all;
clear all;
tau = 0.001; %timestep

%Initial Conditions and Preperatory Steps
L = 10;                 %The domain extends from -L to L
j = sqrt(-1);
N = 20;                 %Controls the Number of grid points
h = L/N;                %lattice spacing
Time = 0;               %initializes Time
x = linspace(-L,L,2*N+1)'; % physical domain
n = (-N:1:N)';             % number of lattice sites
lambda = (1-j)/sqrt(2);    %parameter affecting the width of analytical solution
gamma = 1;                 %parameter affecting translation of the analytical
solution
epsilon = n.*log(lambda + 2*j/(h*lambda))+j*lambda^2*Time/2;  %element in
constructing the initial data
eta = n.*log(-lambda + 2*j/(h*lambda)) - j*Time/(2*lambda^2); %element in
constructing the initial data
d = lambda^2*(h*conj(lambda^2)-2*j)*exp(conj(epsilon)-
conj(eta))+abs(gamma)^2*abs(lambda)^2*(h*lambda^2-2*j)*exp(eta-epsilon);
u = 2*j*conj(gamma)*conj(lambda)*(lambda^2-conj(lambda^2))./d; %initial data



figure(1); plot(x,abs(u),'o:r')
title('abs(u) at t = 0')
pause;
[A,B,Q,R]=Inversion(u,h);                %solves for Q and R at t = 0 using the
inversion script
figure(2); plot(x,abs([0;Q;0]),'o:k')
title('abs(Q) at time 0')
pause;
figure(3); plot(x,abs([0;R;0]),'o:b')
title('abs(R) at time 0')
pause;

iter = 1;
delta = 10; %the amount of dampening
r = 1;      %the number of points to which the damping is applied on each side
of the domain



for k = 1:1000
    Time = k*tau;
    [A,B,Q,R] = Inversion(u,h);

    S = eye(length(n)-2)+diag(ones(1,length(n)-3),1);     %Shift matrix
    T = -eye(length(n)-2)+diag(ones((length(n)-3),1),1); %Shift matrix
    V = diag(ones(1,length(n)-3),1);                      %Shift matrix
    %construction of the LHS
    AL1 = eye(length(n)-2) - tau*h*S*inv(B)/8 + tau*T*inv(A)/(2*h);
    %AL = eye(length(n)-2)-tau*S*inv(B)/(4*j)+tau*T*inv(A)/(2*h);
    u = u(2:length(n)-1);
    %construction of the RHS
    F = -
u.^2.*(conj(S*R))+j*h*u.^2.*(conj(T*Q))/2+u.*(abs(V*Q).^2+abs(Q).^2+abs(V*R).^2+
abs(R).^2);
    %absorbing layer
    if r > 0
        F = F-4*j*delta*[u(1:r);zeros(length(n)-2-2*r,1);u(length(n)-1-
r:length(n)-2)];
    end

    AR1 = eye(length(n)-2) + tau*h*S*inv(B)/8 - tau*T*inv(A)/(2*h);
%construction of the right hand side

    u = AL1\(AR1*u+tau*F/(4*j)); %Solves for u at the next time step

    epsilon = n.*log(lambda + 2*j/(h*lambda))+j*lambda^2*Time/2;
    eta = n.*log(-lambda + 2*j/(h*lambda)) - j*Time/(2*lambda^2);
    d = lambda^2*(h*conj(lambda^2)-2*j)*exp(conj(epsilon)-
conj(eta))+abs(gamma)^2*abs(lambda)^2*(h*lambda^2-2*j)*exp(eta-epsilon);
    uExact = 2*j*conj(gamma)*conj(lambda)*(lambda^2-conj(lambda^2))./d;

    Error(k) = norm([0;u;0]-uExact);
```

Figure 8: The semi-implicit scheme used to generate U at the next timestep. This script calls the function Inversion, which is displayed in Appendix C.

B Time Evolution Script (Improved)

```
%Time Evolution

close all;
clear all;
tau = 0.0002; %timestep

%Initial Conditions and Preperatory Steps
L = 15;                 %The domain extends from -L to L
j = sqrt(-1);
N = 40;                 %Controls number of grid points
h = L/N;                %lattice spacing
Time = 0;               %initializes Time
x = linspace(-L,L,2*N+1)'; % physical domain
n = (-N:1:N)';              % number of lattice sites
lambda = (1+j)/sqrt(2);     %parameter affecting the width of the analytical
solution
gamma = 1;                  %parameter affecting translation of the analytical
solution
epsilon = n.*log(lambda + 2*j/(h*lambda))+j*lambda^2*Time/2;  %element in
constructing the initial data
eta = n.*log(-lambda + 2*j/(h*lambda)) - j*Time/(2*lambda^2); %element in
constructing the initial data
d = lambda^2*(h*conj(lambda^2)-2*j)*exp(conj(epsilon)-
conj(eta))+abs(gamma)^2*abs(lambda)^2*(h*lambda^2-2*j)*exp(eta-epsilon);
u = 2*j*conj(gamma)*conj(lambda)*(lambda^2-conj(lambda^2))./d; %initial data
```

```
iter = 1;   %used later for graphics output
delta = 0;  %the amount of dampening
r = 0;      %the number of points to which the damping is applied on each side
of the domain

for k = 1:1000
    Time = k*tau;                %redefine time so it increases with the
iteration
    [A,B,Q,R] = Inversion(u,h);  %Creates Q and R
    u = u(2:length(u)-1);        %truncates the first and last elements

    S = eye(length(n)-2)+diag(ones(1,length(n)-3),1);      %Shift matrix
    T = -eye(length(n)-2)+diag(ones((length(n)-3),1),1);  %Shift matrix
    V = diag(ones(1,length(n)-3),1);                        %Shift matrix
    V1 = diag(ones(1,2*length(n)-5),1);
    S1 = [S,zeros(length(n)-2)];
```

```
iter = 1;   %used later for graphics output
delta = 0;  %the amount of dampening
r = 0;      %the number of points to which the damping is applied on each side
of the domain

for k = 1:1000
    Time = k*tau;                %redefine time so it increases with the
iteration
    [A,B,Q,R] = Inversion(u,h);  %Creates Q and R
    u = u(2:length(u)-1);        %truncates the first and last elements

    S = eye(length(n)-2)+diag(ones(1,length(n)-3),1);      %Shift matrix
    T = -eye(length(n)-2)+diag(ones((length(n)-3),1),1);  %Shift matrix
    V = diag(ones(1,length(n)-3),1);                        %Shift matrix
    V1 = diag(ones(1,2*length(n)-5),1);
    S1 = [S,zeros(length(n)-2)];
```

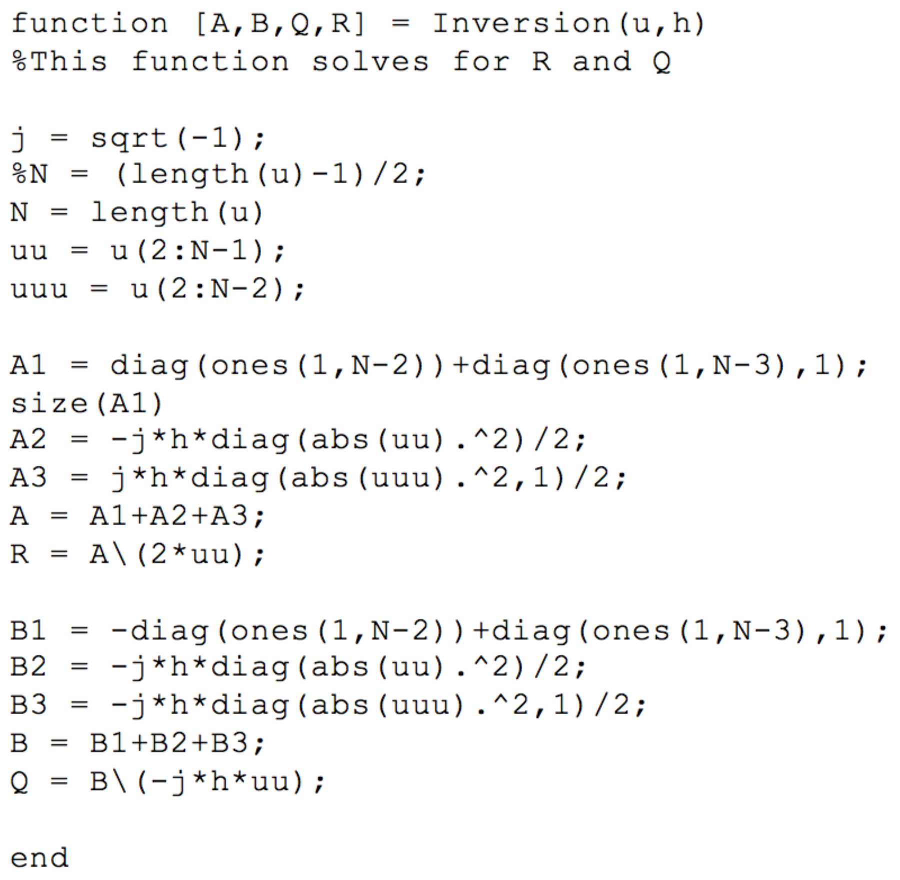
```
      S2 = [S1;[zeros(length(n)-2),S]];                      %larger version of S

      T1 = [T,zeros(length(n)-2)];
      T2 = [T1;[zeros(length(n)-2),T]];                      %larger version of T

      B1 = [inv(B),zeros(length(n)-2)];
      B2 = [B1;[zeros(length(n)-2),-conj(inv(B))]];          % Larger version of
  inv(B)

      A1 = [inv(A),zeros(length(n)-2)];
      A2 = [A1;[zeros(length(n)-2),conj(inv(A))]];
      A3 = [zeros(length(n)-2),conj(inv(A))];
      A4 = [A3;[inv(A),zeros(length(n)-2)]];                 %Creates a larger
  version of inv(A)

      B3 = [zeros(length(n)-2),conj(inv(B))];
      B4 = [B3;[-inv(B),zeros(length(n)-2)]];                %creates a larger
  version of inv(B)
      UUbar = [u;conj(u)];                                   %concatenates U with
  its conjugate
      QQbar = [Q;conj(Q)];                                   %concatenates Q with
  its conjugate
      RRbar = [R;conj(R)];                                   %concatenates R with
  its conjugate
```

```
    %construction of the LHS
    AL2 = eye(2*length(n)-4) - tau*h*S2*B2/8 + tau*T2*A2/(2*h)+
tau*diag((UUbar).^2)*S2*A4/(4*j)-tau*j*(h^2)*diag((UUbar).^2)*T2*B4/16;
    %u = u(2:length(n)-1);

    %construction of the RHS
    F1 = UUbar.*(abs(V1*QQbar).^2+abs(QQbar).^2+abs(V1*RRbar).^2+abs(RRbar).^2);
    %absorbing layer
    if r > 0
        F1 = F1-4*j*delta*[[u(1:r);zeros(length(n)-2-2*r,1);u(length(n)-1-
r:length(n)-2)];conj([u(1:r);zeros(length(n)-2-2*r,1);u(length(n)-1-r:length(n)-
2)])];
    end


    AR2 = eye(2*length(n)-4) + tau*h*S2*B2/8 - tau*T2*A2/(2*h) -
tau*diag((UUbar).^2)*S2*A4/(4*j) + (h^2)*j*tau*diag((UUbar).^2)*T2*B4/16;
    UUbar = AL2\(AR2*UUbar+tau*F1/(4*j)); %Solves for u at the next time step

    u = UUbar(1:length(UUbar)/2);         %extracts U from the concatenated
vector




    epsilon = n.*log(lambda + 2*j/(h*lambda))+j*lambda^2*Time/2;
    eta = n.*log(-lambda + 2*j/(h*lambda)) - j*Time/(2*lambda^2);
    d = lambda^2*(h*conj(lambda^2)-2*j)*exp(conj(epsilon)-
conj(eta))+abs(gamma)^2*abs(lambda)^2*(h*lambda^2-2*j)*exp(eta-epsilon);
    uExact = 2*j*conj(gamma)*conj(lambda)*(lambda^2-conj(lambda^2))./d;
%determines U at the next time




     Error(k) = norm([0;u;0]-uExact);  % calculated error


      % graphical display
     if k == 5000*iter
         figure(1); plot(x,abs([0;u;0]),'o:r')
         title(['mod(U) at time ' num2str(Time)])
         pause;
         figure(4); plot(linspace(1,k,k),Error,'o:r');
         pause;
         RG = [0; R; 0];
         QG = [0;Q;0];
         figure(2); plot(x,abs(QG),'o:b')
         title(['abs(Q) at time ' num2str(Time)])
         pause;
         figure(3); plot(x,abs(RG),'o:g');
         title(['abs(R) at time ' num2str(Time)])
         pause;
         iter = iter + 1;
     end
     u = [0;u;0];
 end
```

```
%additional graphics
plot(x,abs(u),'o:k')
title(['mod(U) at time 0.2'])
xlabel('position, x = hn')
ylabel('mod(U)')
ax = gca;
ax.FontSize = 25;
pause;
plot(linspace(1,k,k),Error,'o:k')
title(['Error between the exact and numerical solutions at each iteration'])
xlabel('iteration')
ylabel('error')
ax = gca;
ax.FontSize = 25;
ax = gca;
ax.FontSize = 25;
pause;
plot(x,abs(uExact),'o:r')
hold on
plot(x,abs(u),'o:k')
title(['mod(U) and mod(U Exact) at time ' num2str(Time)])
xlabel('position, x = hn')
ylabel('mod(U Numerical), mod(U Exact)')
ax = gca;
ax.FontSize = 25
legend('U Exact','U Numerical')
hold off
pause;
norm(uExact-u) %final error after last iteration
```

Figure 9: The improved semi-implicit scheme.

C Inversion Function



```
function [A,B,Q,R] = Inversion(u,h)
%This function solves for R and Q

j = sqrt(-1);
%N = (length(u)-1)/2;
N = length(u)
uu = u(2:N-1);
uuu = u(2:N-2);

A1 = diag(ones(1,N-2))+diag(ones(1,N-3),1);
size(A1)
A2 = -j*h*diag(abs(uu).^2)/2;
A3 = j*h*diag(abs(uuu).^2,1)/2;
A = A1+A2+A3;
R = A\(2*uu);

B1 = -diag(ones(1,N-2))+diag(ones(1,N-3),1);
B2 = -j*h*diag(abs(uu).^2)/2;
B3 = -j*h*diag(abs(uuu).^2,1)/2;
B = B1+B2+B3;
Q = B\(-j*h*uu);

end
```

Figure 10: The function ”Inversion”. This function solves for R and U using the last two equations of (3).