Int. J. Open Problems Compt. Math., Vol. 2, No. 1, March 2009

Numerical Computation of a Nonlocal Double Obstacle Problem

Samir K. Bhowmik[1,2]
1. KdV Institute for Mathematics, University of Amsterdam, Amsterdam, Netherlands , S.K.Bhowmik@uva.nl

and

2. Department of Mathematics, University of Dhaka, Dhaka 1000, Bangladesh, bhowmiksk@gmail.com.

Communicated by Feras M. Al Faqih

Abstract

We consider a nonlocal double obstacle problem. This type of problems comes in various biological and physical situations, e.g., in phase transition models. We focus on numerical approximations and fast computation of such a model. We start with considering piece-wise basis functions for spatial approximation followed by implicit Euler's method for time integration and then Newton's method for solving resulting non linear system. Then we apply various linear system solver to get a time efficient technique to solve the model. We also attempt Fourier transform in space as well.

Keywords: *partial integro-differential equation, phase transition models, piecewise constant functions,* MATLAB*, Fourier transforms, Newton's method, Jacobian, iterative methods.*

1 Introduction

Many problems from phase transitions and various biological processes have been modeled by reaction-diffusion equations, and the diffusion terms usually

involve the Laplacian differential operator are well known [18]. An alternative model based on a convolution integral operator [16, 1, 4, 9, 14] in place of the diffusion term arises in some cases, phase transition models [16, 1, 4, 9], dynamics of neurons in the brain model [14], population dynamics models [15]. Some problems contain both local as well as nonlocal operators [6]. Study in convolution model of phase transitions (initial value problem) is of ongoing interest and a lot about numerical analysis is yet to be done.

Many models of phase transitions have been constructed using thermodynamics concepts. Materials with fine mixture of phases are a familiar and well studied phenomenon, especially in the metallurgy field. For an example the separation of a binary alloy into two phases is a technologically important phenomenon. Such mixtures raise a lot of questions. But the integro-differential models focuses on the process is that of phase transition. A phase separation occurs leading to a fine scaled mixture of phases.

In this article we study Numerical approximation and fast computation of the integro-differential equation representing model of phase transitions

$$u_t(x,t) = \varepsilon \mathbb{L}u(x,t) + f(u(x,t)) \tag{1}$$

where

$$\mathbb{L}u(x,t) = \int_\Omega J(x-y)u(y,t)dy - u(x,t)\int_\Omega J(x-y)dy$$

with initial condition $u(x,0) = u_0(x)$, $x \in \Omega$ where $\Omega \subseteq \mathbb{R}$, $f(u)$ is a bistable nonlinearity for the associated ordinary differential equation

$$u_t = f(u) \tag{2}$$

and $J(x-y)$ is a kernel that measures interaction between particles at position x and at position y within the block or between the blocks with $A1) J(.) \geq 0$, $A2) J(x)$ is symmetric. Here it is assumed that the effect of close neighbours x and y is greater than that from more distant ones; the spatial variation is incorporated in $J(x-y)$. u represents the density(concentration) at point x in Ω of a binary material and $\varepsilon \geq 0$ is interaction length. This type of model based on a convolution integral operator arises in firing rate models in neuronal networks [14] also. We can define $u(x,t)$ in the following way.

Let us consider a binary alloy consisting many atoms of species X and Y and it is specially divided into different sites. It is assumed that each site

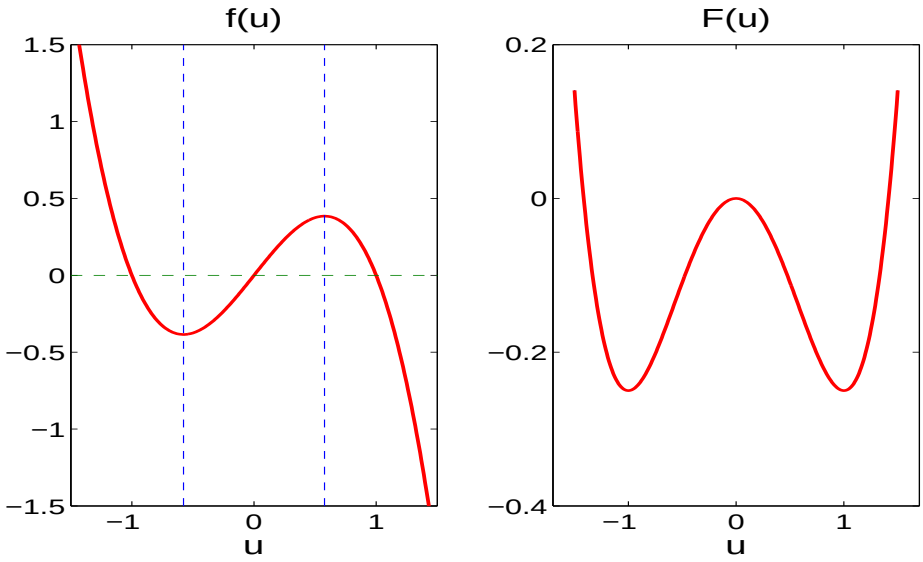

Figure 1: The figure shows $f(u)$ and $F(u)$.

contains atoms from X and Y. The dynamics of u is as follows. If $u(x,t)$ is continuous at (x,t), then its rate of change with time at that point is given by (2). We will assume unless stated otherwise that $f(u)$ is the derivative of a double-well potential $F(u)$ with two local minima at $u = \pm 1$, not necessarily of equal depth , representing the bulk energy density of a state with u constant shown in the Figure1 with $f(u) = u - u^3$. The ODE $u_t = -u(u^2 - 1)$ has two attractors $u = \pm 1$ and $u = 0$ is an unstable equilibrium. So, the free energy is reduced locally by continuous change in u. Let us consider for some $-1 < u_1 < 0 < u_2 < 1$

$$f'(u) < 0, \quad \forall \quad (-1, u_1) \cup (u_2, 1) \qquad \text{and} \qquad f'(u) > 0 \quad \forall \quad (u_1, u_2).$$

Then $(-1, u_1)$ and $(u_2, 1)$ are called metastable intervals and (u_1, u_2) is called the spinodal interval. In Figure 1 the region between the vertical dashed lines is called the spinodal region and outside the region is called the metastable region.

Note that the convolution equation (1) is the L^2-gradient flow of the free energy functional

$$E(u) = \varepsilon \int_\Omega \int_\Omega J(x-y)\,(u(y) - u(x))^2\, dxdy + \int_\Omega F(u)dx \tag{3}$$

where u, and J are defined above and $F'(u) = -f(u)$. In (3) the first integral penalizes inhomogeneous materials and the second integral penalizes states with all u for which $u \notin \{-1, 1\}$. If we consider a binary alloy, for $u \approx \pm 1$ both of the materials of the alloy is in either of the two pure phases, values in between indicate a corresponding mixture of them. It is to be noted that the minima of F have equal depth if $\int_{-1}^{1} f du = 0$. It is conceivable that a stable state u exists taking values near -1 on an interval $(-\infty, a]$ and values 1 on an interval of the form $[b, \infty)$ with a simple transition occurring on $[a, b]$.

In [1] authors studied traveling wave solutions and the stationary solutions of discrete version of (1). They performed existence, uniqueness and stability of solutions of (1). In [5] author described solutions of (1) with their existence, smoothness and stability. Results on stability and convergence of solutions of the time dependent IDE (1) can also be found in [12]. Hutson and Grinfeld [12] show that for small values of ε every solutions converges in L^1 to an equilibrium. Discussion about coarsening of solutions, stability and numerical approximation of (1) can be found in detail in [9, 7]. In these articles authors present stability and sample numerical of solutions using piecewise constant basis functions.

Here we present some numerical computational techniques that speeds up computation. In Section 2 we present approximation piecewise constant basis functions with Galerkin approach in space and show numerical results in Section 3. Then we motivate ourselves to present some linear system solvers that can speed up the computation followed by sample results in Section 4 and Section 5 respectively. We finish with Fourier spectral method for the problem in Section 6.

2 Numerical approximation of the problem in space

Here we concentrate on numerical approximation of the problem (1) in space considering periodic domain $\Omega = [0, 1]$. Redefining the infinite domain problem to a 1-periodic is well presented in [2, 8]. We start with approximating the problem with piecewise constant basis function with Galerkin procedure. We describe a Galerkin method with exact and approximate integrals. We divide

the interval Ω into N equal subintervals (Ω_j) with space mesh h and we define $x_j = jh$, and $x_{j-\frac{1}{2}} = \frac{x_{j-1}+x_j}{2}$. The piecewise constant approximation u_h of u has value $\hat{u}_j = u_h(x_{j+\frac{1}{2}}, t)$ on the space interval $\Omega_j = [x_j, x_{j+1}]$.

Here we discuss the Galerkin approximation of (1) with piecewise constant polynomials in space for the special one dimensional case. We define the basis function $\phi_{j+\frac{1}{2}}(x)$ as

$$\phi_{j+\frac{1}{2}}(x) = \begin{cases} 1 & \text{when } x \in (x_j, x_{j+1}), \\ 0 & \text{otherwise.} \end{cases}$$

We write the approximate solution u_h as

$$u_h(x,t) = \sum_{j=0}^{N-1} \hat{u}_{j+\frac{1}{2}}(t)\phi_{j+\frac{1}{2}}(x) \tag{4}$$

where $\hat{u}_{j+\frac{1}{2}}(t) \in \mathbb{R}$, for all $j = 0, 1, 2, \cdots, N-1$. Substituting (4) for u in (1), multiplying by $\phi_{k+\frac{1}{2}}$ for each $k = 0, 1, 2, \cdots, N-1$ and integrating over Ω, we get the standard Galerkin approximation

$$\left(\phi_{k+\frac{1}{2}}, \frac{du_h}{dt}\right) = (\phi_{k+\frac{1}{2}}, \varepsilon \mathbb{L} u_h) + (\phi_{k+\frac{1}{2}}, f(u_h)). \tag{5}$$

Here $(\cdot,\cdot)$ is the usual L_2 inner product and $f(u_h) = \sum_{j=0}^{N-1} f(\hat{u}_{j+\frac{1}{2}})\phi_{j+\frac{1}{2}}(x)$. From (5) we have

$$\begin{aligned}
\sum_{j=0}^{N-1} \left(\phi_{k+\frac{1}{2}}, \phi_{j+\frac{1}{2}}\right) \frac{d\hat{u}_{j+\frac{1}{2}}}{dt} &= \sum_{j=0}^{N-1} (\phi_{k+\frac{1}{2}}, \varepsilon \mathbb{L}\phi_{j+\frac{1}{2}})\hat{u}_{j+\frac{1}{2}} + \sum_{j=0}^{N-1} (\phi_{k+\frac{1}{2}}, \phi_{j+\frac{1}{2}}) f(\hat{u}_{j+\frac{1}{2}}) \\
\iff \left(\phi_{k+\frac{1}{2}}, \phi_{k+\frac{1}{2}}\right) \frac{d\hat{u}_{j+\frac{1}{2}}}{dt} &= \sum_{j=0}^{N-1} (\phi_{k+\frac{1}{2}}, \varepsilon \mathbb{L}\phi_{j+\frac{1}{2}})\hat{u}_{j+\frac{1}{2}} + (\phi_{k+\frac{1}{2}}, \phi_{k+\frac{1}{2}}) f(\hat{u}_{k+\frac{1}{2}}) \\
\iff h\frac{d\hat{u}_{j+\frac{1}{2}}}{dt} &= hf(\hat{u}_{k+\frac{1}{2}}) + \sum_{j=0}^{N-1} (\phi_{k+\frac{1}{2}}, \varepsilon \mathbb{L}\phi_{j+\frac{1}{2}})\hat{u}_{j+\frac{1}{2}} \\
\Rightarrow \frac{d\underline{u}_h}{dt} &= \varepsilon A \underline{u}_h + f(\underline{u}_h)
\end{aligned} \tag{6}$$

where, for $j \neq k$, the elements of A are

$$\begin{aligned}
a_{j,k} &= \frac{1}{h}(\phi_{k+\frac{1}{2}}, \mathbb{L}\phi_{j+\frac{1}{2}}) = \frac{1}{h}\int_{x_k}^{x_{k+1}} \left(\int_0^1 J(x-y)\left(\phi_{j+\frac{1}{2}}(y) - \phi_{j+\frac{1}{2}}(x)\right) dy\right) dx \\
&= \frac{1}{h}\int_{x_k}^{x_{k+1}} \int_{x_j}^{x_{j+1}} J(x-y) dy dx \quad \text{if } j \neq k
\end{aligned}$$

and when $j = k$

$$\begin{aligned} a_{k,k} &= \frac{1}{h}\int_{x_k}^{x_{k+1}} \left(\int_0^1 J(x-y)\left(\phi_{k+\frac{1}{2}}(y) - \phi_{k+\frac{1}{2}}(x)\right) dy \right) dx \\ &= \frac{1}{h}\int_{x_k}^{x_{k+1}} \int_{x_k}^{x_{k+1}} J(x-y)dydx - \frac{1}{h}\int_{x_k}^{x_{k+1}} \int_0^1 J(x-y)dydx. \end{aligned}$$

Using the midpoint approximation for the integrals we get

$$a_{j,k} = hJ\left(x_{k+\frac{1}{2}} - x_{j+\frac{1}{2}}\right), \quad j \neq k, \text{ and}$$

$$a_{k,k} \approx \frac{h^2}{h} J(x_{k+\frac{1}{2}} - x_{k+\frac{1}{2}}) - \frac{h^2}{h}\sum_{k=0}^{N-1} J(x_{k+\frac{1}{2}} - x_{j+\frac{1}{2}}) = -h \sum_{k=0, j\neq k}^{N-1} J(x_{k+\frac{1}{2}} - x_{j+\frac{1}{2}}).$$

2.1 Numerical results

Here we present some experimental results obtained from the approximation (6) using MATLAB built in function **ode15s**. There is a parameter $\varepsilon \geq 0$ multiplied with the coefficient matrix A in (6). That can result a stiff differential equation for larger values of ε. It is well known that explicit solvers converge slowly for the problems with stiffness [11, 13]. **ode15s** is designed to solve a system of differential equations using implicit solvers and it converges faster than explicit solvers for problems with stiffness, for more details please see [2, 11, 19]. We set $RelTol = 10^{-12}$, $AbsTol = 10^{-15}$ to solve the system of differential equations inside the MATLAB **ode15s** solver. It is to mention that we set the tolerance inside the framework of MATLAB so that we get accurate solution in time with machine precision that one can see from exact solution. This choice is made to experiment the accuracy of spatial approximation (6).

We approximate the solution with kernel function $J(x) = \sqrt{\frac{100}{\pi}}e^{-100x^2}$ and $f(u) = u - u^3$. The initial condition is $u(x, 0) = \sin(8\pi x^2)$. In Figure 2 we plot $u(x, t)$ at various choice of t with parameter $\varepsilon = 0.48$, and $N = 128$ space elements. Then we plot the equilibrium solutions of (6) for several values of ε in Figure 3 again using $N = 128$ space elements. Now it is almost obvious to ask how accurate such a piecewise constant approximation is. To investigate that numerically, we begin by estimating the approximation error in solutions of (6). Here we show the computational error and computational time taken for solving (6) using the MATLAB built in function **ode15s** with $RelTol = 10^{-12}$,

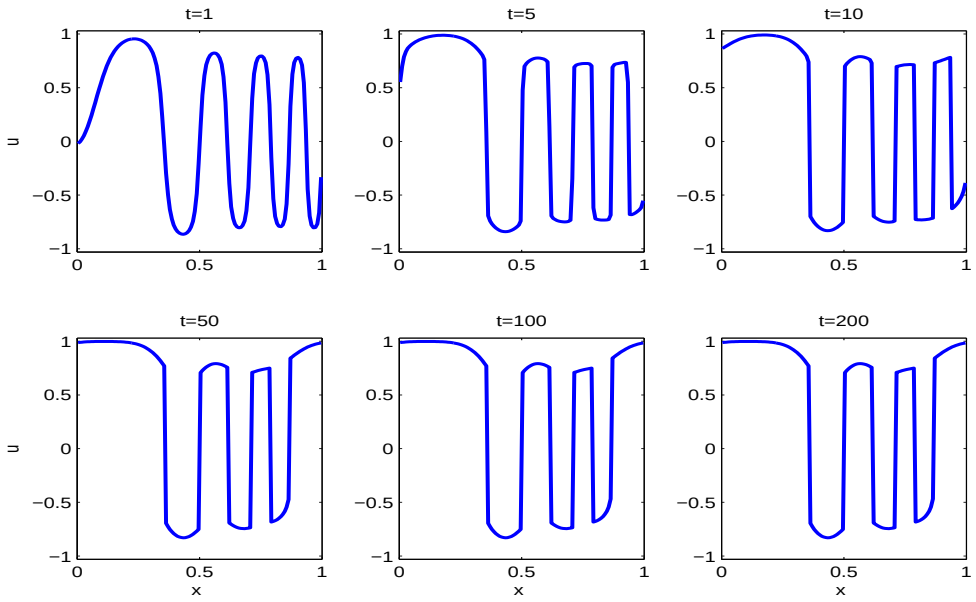

Figure 2: We plot the approximate solution at different times for fixed $\varepsilon = 0.48$, $u_0 = \sin(8\pi x^2)$ and 128 space elements. Here we use the MATLAB built in function **ode15s** with $RelTol = 10^{-12}$, $AbsTol = 10^{-15}$.

$AbsTol = 10^{-15}$ to solve the system of differential equations. Here we compute the error of the approximate solutions. The solutions approximated using N space elements are defined by $u_N(\cdot, t)$. In Figure 4 we plot the discrete L_2 norm of the approximation error $\|u_N(\cdot, t) - u_{\frac{N}{2}}(\cdot, t)\|$ against N for each $N = 2^i$ where $i = 1, 2, ..., 9$. We record the CPU time for each computation. From the right subplot of Figure 4 we observe that the rate of convergence of piecewise constant approximation in space appears to be $\mathcal{O}(h)$. From the left subplot of Figure 4 we observe that computational cost is high with the set up we used to solve the problem. We will look at ways to speed up the calculation in Section 3.

3 The full discrete problem and computational issues

We show in the results of Section 2.1 that calculations for this problem can be expensive. As we choose MATLAB implicit solver ode15s in earlier section, here

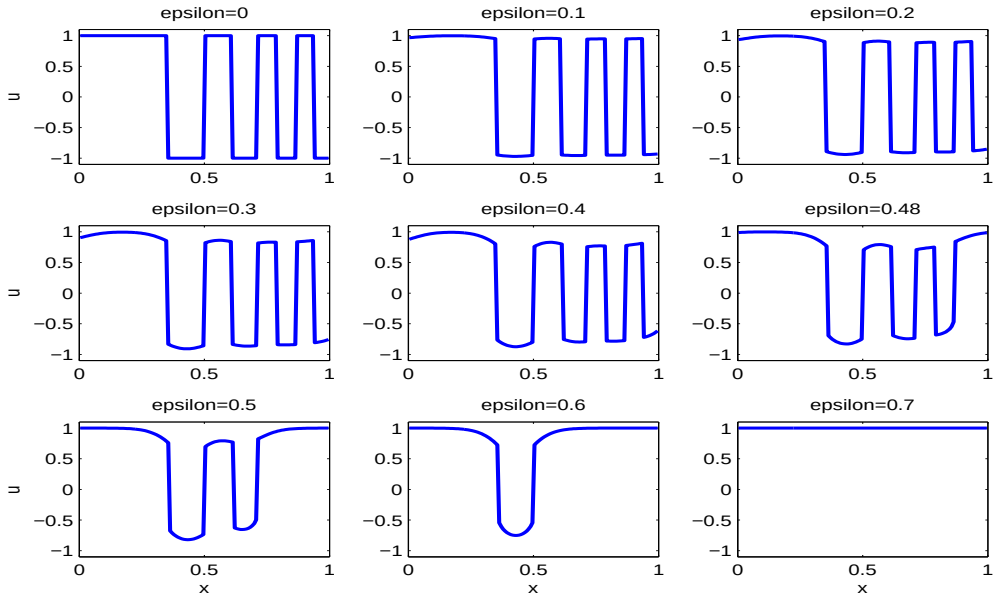

Figure 3: Here we show the effect of ε on the approximate solution at $t = 100$ with 128 space elements, and $u_0 = \sin(8\pi x^2)$. Theses are close to equilibrium solutions. Here we use the MATLAB built in function **ode15s** with $RelTol = 10^{-12}$, $AbsTol = 10^{-15}$.

we start with implicit Euler method to solve (6) in time followed by Newton's method for the resulting nonlinear system of equations. We try and examine couple of techniques to solve the system of linear system equations that arises in each Newton iteration with the aim of speed the computation up.

Applying the implicit Euler approximation to the ODEs (6) gives the approximation

$$\underline{u}_h^{n+1} - \varepsilon \Delta t A \underline{u}_h^{n+1} - \Delta t f(\underline{u}_h^{n+1}) - \underline{u}_h^n = 0 \tag{7}$$

which is a nonlinear system of equations for $\underline{u}_h^{N+1}$ with given $\underline{u}_h^N$, constants Δt, ε and matrix A. To solve (7) for $\underline{u}_h^{N+1}$ let us consider

$$F(\underline{u}_h^{n+1}) \equiv \underline{u}_h^{n+1} - \varepsilon \Delta t A \underline{u}_h^{n+1} - \Delta t f(\underline{u}_h^{n+1}) - \underline{u}_h^n = 0. \tag{8}$$

It is very important to solve (8) efficiently to cut the computational cost. One efficient way to solve the problem is to use Newton's method. We examine various direct, iterative and approximate techniques to solve the system of linear equations that arise in each Newton's iteration step.

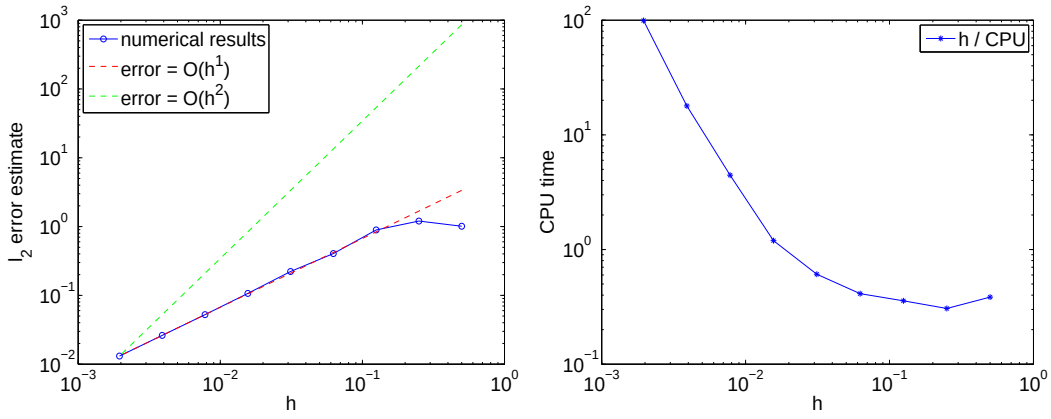

Figure 4: In these subplots we show estimated error, and computational time for the space discretisation used to approximate the problem. Here we use the MATLAB built in function **ode15s** with $RelTol = 10^{-12}$, $AbsTol = 10^{-15}$.

Newton's Method for Nonlinear Systems of Equations

Newton's method is one of the most common and popular methods to solve a system of nonlinear equations. Here we consider

$$F(\underline{v}) = 0. \tag{9}$$

We use the Jacobian matrix $\mathcal{J}(\underline{v}) = \frac{\partial F(\underline{v})}{\partial \underline{v}}$. To avoid the computation of the inverse of the Jacobian matrix, the method can be divided into the steps in the following algorithm.

Algorithm 1. *Newton's method to solve nonlinear system of equations*
To solve the system of nonlinear equations (9) approximately with tolerance TOL:

Step 1*: start with an initial data* $\underline{v}_0$*, set* $\underline{dv} = 0$*, number of iterations* N*.*

Step 2*: Calculate* $\underline{dv}$ *such that* $\mathcal{J}(\underline{v}^k)\underline{dv} = F(\underline{v}^k)$ *and then update* $\underline{v}^{k+1} = \underline{v}^k - \underline{dv}$*, for* $k \leq N$ *and check the convergence of solutions.*

The main problem in the implementation of Newton's method discussed above for our problem is to solve the resulting system of linear equations at each step. To avoid using the full Jacobian matrix, we investigate several

approximate techniques to solve the system with desired accuracy and observe the efficiency of the techniques.

For large system of equations, most work is done solving the linear system of equations defined in **Step 2** of the Newton's Algorithm 1. An efficient way to solve the system can speed up computation. We examine several linear system solving techniques to get the best approach to solve the system of linear equation arises in each Newton iteration. This is usually done by evaluating and LU factorizing the Jacobian matrix $\mathcal{J}(v^k)$ and performing back substitution for $\underline{dv}$. However, when $\mathcal{J}(v^k)$ is very large it is natural to consider alternatives such as iterative methods to speed up the computation.

We have done some experiments to solve the linear system in the **Step** 2 of the Newton iteration. We also approximate the Jacobian matrix $\mathcal{J}(v^k)$ using continuous Fourier transform and $(2, 2)$ pade approximation to get a computation time efficient method. Then with that approximate Jacobian we solve the linear system using LU factorization. We compared this result with various other solvers like direct method, LU factorization, several iterative techniques which are discussed, in detail, in [3, 17] and references there in. We start with approximating the Jacobian of (9) and then compare results with various other linear solvers.

3.1 Jacobian approximation

Here we use Fourier transforms and pade approximation to find a approximate Jacobian. Here we consider the infinite domain problem to perform Fourier transform to get a time dependent differential equation. Then we use Pade approximation [17] followed by the implicit Euler method in time to get an approximate Jacobian. Before the main discussion let us introduce definition of Fourier transform and its inverse. If $u \in L_2(\mathbb{R})$, then the Continuous Fourier Transform (CFT) of $u(x)$ in space can be defined as

$$\hat{u}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u(x) e^{-ix\xi} dx \equiv (Fu)(\xi). \tag{10}$$

If $u, \hat{u} \in L_2(\mathbb{R})$, then the inverse Fourier transform is defined as

$$u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{u}(\xi) e^{ix\xi} d\xi \equiv \left(F^{-1}\hat{u}\right)(x).$$

Considering $\Omega = \mathbb{R}$ and applying Fourier transform on (1) we have

$$\hat{u}_t = \varepsilon q(\xi)\hat{u}(\xi,t) + \widehat{f(u)} \tag{11}$$

where $q(\xi) = \sqrt{2\pi}\left(\hat{J}(\xi) - \hat{J}(0)\right)$. Now if we consider $J(x) = \sqrt{\frac{\nu}{\pi}}\exp(-\nu x^2)$, then $\hat{J}(\xi) - \hat{J}(0) = \exp(-\frac{\xi^2}{4\nu}) - \hat{J}(0) \approx \frac{-2\xi^2}{8\nu+\xi^2}$ using $(2,2)$ pade approximation [17] where ξ is small. Now $-\xi^2 \equiv CFT\left(\frac{\partial^2}{\partial x^2}\right)$. Now using the above approximation to $q(\xi)$ we can approximate (11) as

$$\begin{aligned}
\hat{u}_t &= \frac{-2\varepsilon\xi^2}{8\nu+\xi^2}\hat{u}(\xi,t) + \widehat{f(u)} \\
\Leftrightarrow (8\nu+\xi^2)\hat{u}_t &= -2\varepsilon\xi^2\hat{u}(\xi,t) + (8\nu+\xi^2)\widehat{f(u)} \\
\Leftrightarrow (8\nu - \frac{\partial^2}{\partial x^2})u_t &= 2\varepsilon\frac{\partial^2}{\partial x^2}u(x,t) + (8\nu - \frac{\partial^2}{\partial x^2})f(u)
\end{aligned} \tag{12}$$

We can approximation (12) using second order central difference in space and implicit Euler's in time as

$$(8\nu I - T)(u^{n+1} - u^n) = 2\varepsilon\Delta t T u^{n+1} + \Delta t(8\nu - T)f(u^{n+1}) \tag{13}$$

where I in an identity matrix and T is a triangular matrix as $T = \frac{1}{h^2}(1,-2,1)$ with some boundary conditions imposed on it. We compare (7) and (13) to approximate

$$A \approx 2(8\nu I - T)^{-1}T \Rightarrow Au \approx 2(8\nu I - T)^{-1}(Tu).$$

Also we have Jacobian of (7) as

$$\mathcal{J} = I - \varepsilon\Delta t A - \Delta t f_u(u) \approx (8\nu I - T)^{-1}\left[(8\nu I - T) - \varepsilon\Delta t 2T - \Delta t(8\nu I - T)f_u(u)\right]$$

considering $A \approx 2(8\nu I - T)^{-1}T$. Using the approximate Jacobian we replace $\mathcal{J}dy = -F(u)$ by

$$\left[(8\nu I - T) - \varepsilon\Delta t 2T - \Delta t(8\nu I - T)f_u(u)\right]dy = (8\nu I - T)F(u)$$

where $F(u)$ is defined in (8). We consider the following restriction:

- $$\dot{u}|_{x=0} = \int_0^1 J(0-y)(u(y,t) - u(0,t))dy + f(u)|_{x=0} \quad \text{and}$$
$$\dot{u}|_{x=1} = \int_0^1 J(1-y)(u(y,t) - u(1,t))dy + f(u)|_{x=1}.$$

3.2 Choice of Iterative methods to solve linear system of equations

Here for simplicity we consider the linear system in **Step 2** of the Algorithm 1 of the form

$$\mathcal{J}\underline{dy} = b \tag{14}$$

where $\mathcal{J} = T_1 + D$, T_1 is a toeplitz matrix, D is a diagonal matrix with $D = \frac{d}{du}f(u)$, $b = -F(\underline{v})$. To solve the system of linear equations (14) let us consider the following two cases with different choices of T_1 and D

- if $\rho(T_1^{-1}D) < 1$, we can approximate the solution of the above linear system $\mathcal{J}\underline{dy} = (T_1 + D)\underline{dy} = b$, where T_1 and D defined above and u, b are column vectors, as

$$T_1\underline{dy}^{k+1} = b - D\underline{dy}^k \quad \text{with } k = 0, 1, 2, 3, \cdots .$$

- if $\rho(D^{-1}T_1) < 1$, we can approximate the solution of the above linear system $\mathcal{J}u = (T_1 + D)u = b$, where T_1 and D defined above and $\underline{dy}$, b are column vectors, as

$$D\underline{dy}^{k+1} = b - T_1\underline{dy}^k \quad \text{with } k = 0, 1, 2, 3, \cdots .$$

Now here is some choices of D and T_1. Let us consider the initial choice as $T_1 = T_{1_0}$, with zeros on the diagonal and a diagonal matrix $D = D_0$, then let us consider $D = D_0 + aI$, $T_1 = T_{1_0} - aI$ for any $a \in \mathbb{R}$.

3.2.1 Motivation for the choice of the Iterations

To solve the initial value problem (IVP) (6) with initial condition $u(t_0) = u_0$ using backward Euler's method, in each step we get a system of nonlinear equations. To solve the nonlinear system we use the Newton's method. In each step of the Newton's method we need to solve a system of linear equation of the form $\mathcal{J}u = b$ where

$$\mathcal{J} = I - \Delta t\varepsilon A - \Delta t\frac{d}{du}f(u)$$

Now we have $\frac{d}{du}f(u) = 1 - 3u^2$. On the steady state $u \to \pm 1$. So we have $\frac{d}{du}f(u) \approx -2$. Then also we have

$$\varepsilon A = \varepsilon(T_{1_0} - D_0) = \varepsilon(T_{1_0} - I)$$

as for the periodic domain $D_0 = qI$ and $q \to 1$ on the equilibrium position. So we have

$$\mathcal{J} \;=\; I - \Delta t\varepsilon(T_{1_0} - I) + 2\Delta t I = (I + 2\Delta t + \Delta t\varepsilon)I - \Delta t\varepsilon T_{1_0} = D + T_1,$$

where $D = (I + 2\Delta t + \Delta t\varepsilon)I$ and $T_1 = -\Delta t\varepsilon T_{1_0}$. Thus the spectral radius of

$$\rho(D^{-1}T_1) = \frac{\Delta t\varepsilon}{1 + 2\Delta t + \Delta t\varepsilon}\rho(T_{1_0}) \Rightarrow \rho(D^{-1}T_1) < \frac{\Delta t\varepsilon}{1 + 2\Delta t + \Delta t\varepsilon} < 1.$$

And hence the successive approximations $D\underline{dy}^{s+1} = b - T_1\underline{dy}^s$ converges to the solution of $\mathcal{J}\underline{dy} = b$, which is actually the Jacobi iteration of $(D - T)\underline{dy} = b$.

To solve the linear system (14) with several iterative techniques the matrices

1. $T_J = D^{-1}(L + U)$ is used for Jacobi iteration with $x^s = T_J x^{s-1} + C$; where $C = D^{-1}b$.

2. $T_{GS} = (D-L)^{-1}U$ is used for Gauss-Seidel iteration with $x^s = T_{GS}x^{s-1} + C$; where $C = (D - L)^{-1}b$.

3. $T_{SOR} = (D - aL)^{-1}[(1 - a)D + aU]$ is used for SOR iteration with any real a, where $x^k = T_{SOR}.x^{k-1} + C$; where $C = a(D - aL)^{-1}b$. It is to be noted that with $0 < a < 1$ this technique is called under-relaxation methods whereas with $1 < a$, it is called over-relaxation methods.

3.3 Linear algebra implementation

Here we discuss the results obtained from various linear system solvers introduced in Section 3. In most cases, the system size for the calculation is $N = 2^8$, and tolerance as tol= 10^{-12} and $u_0 = \sin(10\pi x^2)$. Figure 5 show total cpu time and total number of Newton iteration taken to reach steady states of the ODE (6) for all the techniques discussed above. It is a clear evidence of dominance of the approximate Jacobian technique discussed in Section 3.1 compared to others used here to solve (6). Here we also observe that Jacobi iteration technique also work well and it works faster for a very big time stepping($\Delta t = 1$) as well. We also notice that though fewer Newton iterations are taken by a direct and LU solver, the CPU time is less for the iterative techniques as well as approximate Jacobian version of computation, and hence they give the faster

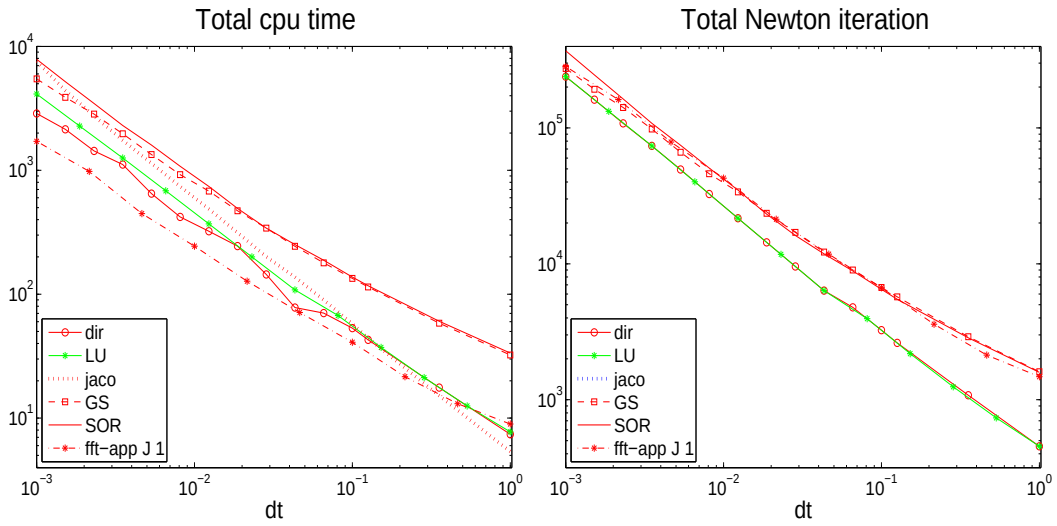

Figure 5: These figures compare CPU time and total number of Newton iteration taken by each method with different choice of Δt shown in the x-axis. Here in all the cases $u_0 = \sin(10\pi x^2)$, $\varepsilon = 0.48$ and $Tol = 10^{-10}$. Here in the Legend dir stands for direct solver, LU for LU factorization, jaco for Jacobian iteration, GS for Gauss saidel, SOR for relaxation with $a = 1.012$ in (we tried $a = .98$ that gives same speed up) and fft-app J1 stands for approximate Jacobian version of computation.

computation. Here we consider the time interval $[0, 100]$ in all computation as from Figures 2 - 3 we observe that when h is sufficiently small $u(x, 100)$ reaches the steady state.

4 Fourier Spectral Approximation:

Here we solve the time dependent problem (1) by Fourier spectral method on $\Omega = [-\pi, \pi]$. This study is motivated from [10, page 131] and [19, page 110]. Specially we follow the steps taken by Trefethen [19, page 110] where he used a Fourier spectral method for the KdV equation

$$u_t + uu_x + u_{xxx} = 0$$

on $[-\pi, \pi]$. He used the fourth order Runge-Kutta method for time discretisation. We use and modify his code to fit in our problem. Applying the

continuous Fourier transformation discussed in the Section 3.1 to (1) where $\Omega = \mathbb{R}$ we get

$$\hat{u}_t(\xi) = \varepsilon\sqrt{2\pi}\left(\hat{J}(\xi) - \hat{J}(0)\right)\hat{u}(\xi) + \widehat{f(u)} \tag{15}$$

where the term $\sqrt{2\pi}\left(\hat{J}(\xi) - \hat{J}(0)\right)$ is defined in (11). Let us consider $\hat{U}(\xi, t) = \exp(-\varepsilon\mu t)\hat{u}(\xi, t)$ where $\mu = \sqrt{2\pi}\left(\hat{J}(\xi) - \hat{J}(0)\right)$. Then

$$\hat{U}_t = \exp(-\varepsilon\mu t)\left(\hat{u}_t - \varepsilon\mu\hat{u}\right). \tag{16}$$

So using (16) in (15) we get

$$\begin{aligned} & \hat{U}_t = (\exp(-\varepsilon\mu t)\widehat{f(u)} = (\exp(-\varepsilon\mu t)\mathbf{F}\left(f\left(\mathbf{F}^{-1}\hat{u}\right)\right) \\ \Rightarrow \quad & \hat{U}_t - (\exp(-\varepsilon\mu t)\mathbf{F}\left(f\left(\mathbf{F}^{-1}\left(\exp(-\varepsilon\mu t)\hat{U}\right)\right)\right) = 0 \end{aligned} \tag{17}$$

where $\mathbf{F}$ denote the Fourier transform operator defined by (10). In Fourier space we can discretize the problem (17) in time and we use the fourth order **Runge-Kutta** method for that purpose. The resulting solution can be transferred to the real domain using inverse Fourier transform. For computation we use MATLAB function **FFT** and **IFFT** to compute Fourier transform and inverse Fourier transform of functions in $[-\pi, \pi]$ which is appropriate since we are not interested in the effect of boundary conditions here. The computational costs and computational complexity of FFT and IFFT are well presented in [17, 19]. The following Figure 6 shows the result with $u(x, 0) = \frac{1}{2}\exp(-|x|)$ and $\varepsilon = 0.48$.

5 Conclusions and restrictions

Several numerical approximation schemes to solve the nonlocal model of phase transitions has been addressed. This study was to perform numerical approximations and implemen- tations to get a computationally efficient technique to solve (1). We considered piecewise constant basis functions for spatial approximation followed by implicit time solver. Then we experimented various linear algebra tools to get a time efficient technique. We also use Fourier spectral method in space followed by fourth order Runge-Kutta method in time. In our study we notice that experimented spatial approximation is of $\mathcal{O}(h)$ accurate computationally, but use of a general ordinary differential equation solver is

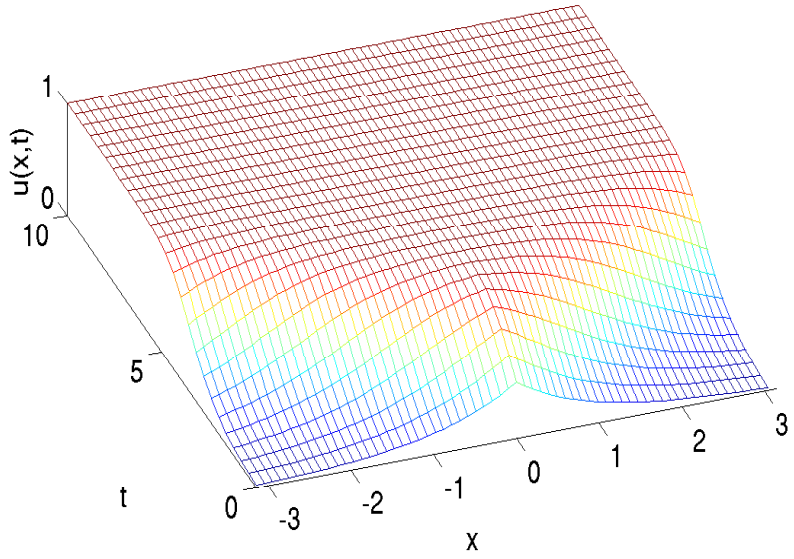

Figure 6: This figure shows solution using Fourier spectral approximation in space and RK4 in time with $u(x,0) = \frac{1}{2}\exp(-|x|)$ and $\varepsilon = 0.48$.

costly in terms of computer time. Performing several linear algebra tools we notice that an implicit solver coupled with Jacobian approximation for Newton's method as well as Jacobian iteration for linear system inside each Newton's iteration outperformed all other solvers presented in this article. There was a problem of getting appropriate initial function for the approximate Jacobian computation. We noticed that computational time depends the choice of initial function used for the Jacobian approximation. In this study we noticed that Fourier spectral method can be proposed but not very efficient due to the nonlinear part. Also there is a restriction on kernel function and initial function for the Fourier spectral method, needed to be Fourier integrable.

Acknowledgment: The author would like to thank Professor Dugald B Duncan, Mathematics Department, Heriot-Watt University, UK for his kind and cordial help and advise while the work was being done. This research was supported by MACS studentship, Heriot-Watt University, Edinburgh, UK and travel and study leave grant from the University of Dhaka, Dhaka, Bangladesh.

References

[1] Peter W Bates and Adam Chmaj. A discrete convolution model for phase transitions. *Archive for Rational Mechanics and Analysis*, 150(4):281–305, 1999.

[2] Samir K. Bhowmik. *Numerical approximation of a nonlinear partial integro-differential equation.* PhD thesis, PhD Dissertation, Heriot-Watt University, Edinburgh, UK, April, 2008.

[3] Richard L. Burden and J. Douglas Faires. *Numerical Analysis.* Thomson, seventh edition, 2004.

[4] Fengxin Chen. Uniform stability of multidimensional travelling waves for the nonlocal allen-chan equation. *Fifth Mississippi State Conferrence on Differential Equations and Computational Simulations, Electronic Journal of Differential Equations*, Conference 10:109–113, 2003.

[5] Adam Chmaj and Xiaofeng Ren. Homoclinic solutions of an integral equation: Existence and stability. *Journal of Differential Equations*, 155:17–43, 1999.

[6] Jerome Coville and Louis Dupaigne. propagation speed of travelling fronts in non local reaction-diffusion equations. *ELSEVIER, Nonlinear Analysis*, 60:797–819, 2005.

[7] *Applied Linear Algebra, Course notes for MSc Module in the Department of Mathematics, Heriot Watt University, 2006.*

[8] Pascal Delaunay. *The Numerical Approximation of IDE's Arising in Phase Transitions.* PhD thesis, Second year placement report, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK., 2003.

[9] Dugald B. Duncan, M Grinfeld, and I Stoleriu. Coarsening in an integro-differential model of phase transitions. *Euro. Journal of Applied Mathematics*, 11:511–523, 2000.

[10] Bengt Fornberg. *A practical guide to pseudospectral methods.* Cambridge monographs on applied and computational mathematics, 1996.

[11] Desmond J. Higham and Nicholas J. Higham. MATLAB *Guide*. SIAM, 2000.

[12] V. Hutson and M. Grinfeld. Non-local dispersal and bistability. *Euro. Journal of Applied Mathematics, Cambridge university press*, 17:211–232, Feb 2006.

[13] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge texts in applied Mathematics, Cambridge University press, 1996.

[14] Carlo R. Laing and William C. Troy. Pde methods for nonlocal models. *SIAM J. Applied dynamical systems*, 2(3):487–516, 2003.

[15] Jan Medlock and Mark Kot. Spreading disease: integro-differential equations old and new. *Mathematical Biosciences*, 184:201–222, 2003.

[16] Xiaofeng Ren Peter W. Bates, Paul C. Fife and Xuefeng Wang. Travelling waves in a convolution model for phase transitions. *Archive for Rational Mechanics and Analysis*, 138(2):105–136, July 1997.

[17] William H. Press, Saul A Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C*. Cambridge university press, 2002.

[18] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth and Brooks, Cole Advanced Books and Software, Pacific Grove, California,, 1989.

[19] Lloyd N. Trefethen. *Spectral Methods in Matlab*. SIAM, 2000.