Solving the Cahn-Hilliard Equation using finite differences and spectral methods

Author	**NetID**
Cristian Lacey	clacey
Amlan Sinha	amlans
Sijie Tong	sijiet

APC 523: Numerical Algorithms for Scientific Computing

Department of Mechanical and Aerospace Engineering
Princeton University, Princeton, NJ, USA

May 14, 2019

Contents

List of Figures

List of Tables

1 Background of the Cahn-Hilliard Equation

The Cahn-Hilliard equation was proposed to study the spontaneous phase separation of a mixture of liquids or solids into two different phases. This process is called spinodal decomposition. Typically, the homogeneous mixture is stable at a high temperature. When suddenly cooled down below a critical temperature, two thermodynamically stable phases with different concentrations form. The original mixture will separate into these two phases and the domains of these phases will grow and coarsen with time.

In the Cahn-Hilliard theory, a binary mixture is described by a order parameter $c(\vec{x})$, which can be the concentration in the mixture. $c = \pm 1$ indicates two different domains respectively. The time evolution of the order parameter $c(\vec{x})$ is described by a diffusion-like equation

$$\frac{\partial c}{\partial t} = D\nabla^2 \frac{\delta f}{\delta c} \tag{1}$$

where f is the free energy density and D is the diffusion constant. The free energy density usually assumes a Ginzburg-Landau energy functional

$$f = \frac{1}{2}\gamma |\nabla c|^2 + V(c) \tag{2}$$

where potential $V(c)$ describes the energy in the bulk of the phases. For the spinodal decomposition of a binary mixture,

$$V(c) = \frac{1}{4}(c^2 - 1)^2 \tag{3}$$

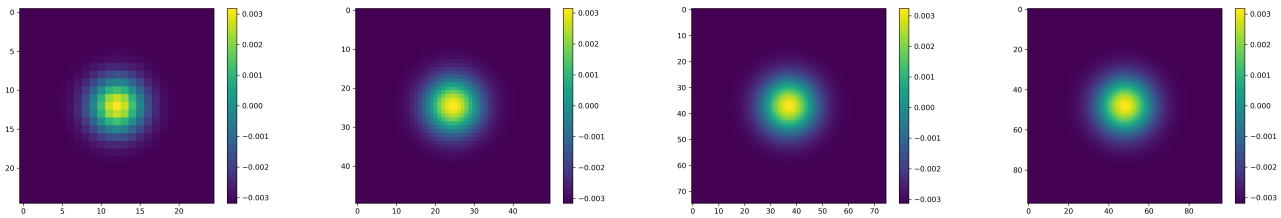
This potential $V(c)$ is a double-well potential which has two minima at $c = \pm 1$. These two minima correspond to two phases. The first term in the free energy density Eq. (2) describes the interfacial energy between two phases. It adds more energetic penalty if the gradient ∇c is large, so the system will tend not to have sharp interface. This term also accounts for the coarsening process in the spinodal decomposition since the energy function favors reduction of the interface. After carrying out the variational calculation in Eq. (1), we arrive at the Cahn-Hilliard equation,

$$\frac{\partial c}{\partial t} = D\nabla^2 (c^3 - c - \gamma \nabla^2 c) \tag{4}$$

2 Project Goals

The main objective is to solve the Cahn-Hilliard Equation using two different spatial methods: finite differences method and the spectral methods. We then compare their performance in terms of runtime and accuracy. In order to simplify the problem, the constants D and γ have been set to one. We discretize the spatial domain and impose periodic boundary conditions. Since we intend to solve the partial differential equation with different grid sizes, we need to find a way to impose the initial condition such that it is independent of the grid size chosen. Use of a bivariate normal distribution function over the whole spatial

domain guarantees this grid independence. Because the function and spatial domain remain consistent throughout the problem, the initial condition remains independent of the grid size. Our spatial domain is square, with x and y both in the interval [-25,25] and time in [0,60]. We refer to the side length as L = 50, and the global simulation time as T = 60. These domains are large enough to observe a trend in the solutions without requiring an excessive total runtime. For each of the two methods, we implement multiple spatial discretization schemes and integration schemes. We compare a second order central difference scheme with a fourth order central difference scheme for spatial discretization in the finite differences method. We also compare implicit integration schemes (First Order Backward Euler, Second Order Crank-Nicolson) with explicit integration schemes (First Order Forward Euler, Fourth Order Runge-Kutta) with the finite differences method. We expect the implicit methods to be numerically stable, allowing us to take larger time steps and reduce runtime. We implement a First Order Explicit, a Fourth Order Runge-Kutta and a First Order Semi-Implicit temporal scheme for the spectral method. The different methods were bundled in a class structure. By doing so, we are able to easily switch between different spatial and temporal schemes for different methods.



(a) 25×25 grid (b) 50×50 grid (c) 75×75 grid (d) 100×100 grid

Figure 1: Initial condition with a bivariate normal distribution over the spatial domain $L = [-25, 25]$ with four different grid sizes.

3 Implementation of Spatial and Temporal Schemes

Since we are solving the Cahn-Hilliard equation in 2D, the phase concentration, c, is initially defined as an $N \times N$ matrix. This matrix is then 'unraveled' as a $N^2 \times 1$ vector using column-major form. Operators in physical and Fourier space are then defined as $N^2 \times N^2$ matrices that operate on the vector c through matrix multiplication. The forms of the operators depend on the spatial scheme, as laid out in the following sections. In addition, depending on the temporal scheme, the Cahn-Hilliard equation is discretized differently in time. The implicit schemes require iterative methods of solution.

3.1 Finite Differences

In two dimensions, the Laplacian operator is defined in terms of Kronecker products of the identity matrix, I, and the second derivative operator in matrix form, D^2.

$$\nabla^2_{N^2 \times N^2} = I_{N \times N} \otimes D^2_{N \times N} + D^2_{N \times N} \otimes I_{N \times N} \tag{5}$$

Depending on the spatial scheme, the form of the D^2 matrix changes, altering the Laplacian. However, the form of the Laplacian is constant from time-step to time-step for a given run, allowing it to be pre-computed once during the initialization of a CahnHilliard() object and then passed as an argument to the time-stepping method.

3.1.1 Second Order Central Differences

The Second Order Central Difference formula (with spatial error of $\mathcal{O}(dx^2)$) was used to build one form of the Laplacian.

$$\frac{\partial^2 c[i][j]}{\partial x^2} \approx \frac{c[i][j+1] - 2c[i][j] + c[i][j-1]}{dx^2} \tag{6}$$

Since the boundary conditions are periodic, the same coefficients just 'wrap around' on the first and last rows of D^2, introducing ones in the lower left and upper right corners.

$$D^2 = \frac{1}{dx^2} \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 & -2 & 1 \\ 1 & 0 & \dots & 0 & 0 & 1 & -2 \end{bmatrix} \tag{7}$$

3.1.2 Fourth Order Central Differences

The Fourth Order Central Difference formula (with spatial error of $\mathcal{O}(dx^4)$) was used to build an alternate form of the Laplacian.

$$\frac{\partial^2 c[i][j]}{\partial x^2} \approx \frac{-c[i][j+2] + 16c[i][j+1] - 30c[i][j] + 16c[i][j-1] - c[i][j-2]}{12dx^2} \tag{8}$$

Again, since the boundary conditions are periodic, the coefficients 'wrap around' on the first and last couple rows of D^2, introducing additional non-zero elements near the corners.

$$D^2 = \frac{1}{12dx^2}\begin{bmatrix} -30 & 16 & -1 & 0 & \dots & -1 & 16 \\ 16 & -30 & 16 & -1 & \dots & 0 & -1 \\ -1 & 16 & -30 & 16 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 16 & -30 & 16 & -1 \\ -1 & 0 & \dots & -1 & 16 & -30 & 16 \\ 16 & -1 & \dots & 0 & -1 & 16 & -30 \end{bmatrix} \tag{9}$$

3.2 Temporal Schemes

Two explicit and two implicit time-stepping methods were implemented. The explicit methods involve only spatial derivatives of c at the current time-step, while the implicit methods involve those at the next time-step. The implicit methods therefore require a solution to an equation of the form $Ac^{k+1} = b(c^k, c^{k+1})$, where k corresponds to the vector c at the current time-step, and $k+1$ at the next time-step. The forms of A and b change depending on the specific scheme. Since b turns out to be a function of c^{k+1}, an initial guess for c^{k+1} is made using an initial First Order Forward Euler step, and b is evaluated using this value. The equation $Ac^{k+1} = b$ is then solved repeatedly using the method of conjugate gradient, each time re-evaluating b with the new c^{k+1} until convergence is attained within a specified tolerance.

3.2.1 First Order Forward Euler

For first order forward Euler, spatial derivatives are taken at the current time-step.

$$\frac{\partial c}{\partial t} = \nabla^2(c^3 - c - \nabla^2 c) \tag{10}$$

$$\frac{c^{k+1} - c^k}{dt} = \nabla^2((c^k)^3 - (c^k) - \nabla^2(c^k)) \tag{11}$$

Isolating c^{k+1}, an explicit equation can be formed, with c^k, dt, and ∇^2 all known.

$$c^{k+1} = c^k + dt\nabla^2((c^k)^3 - (c^k) - \nabla^2(c^k)) \tag{12}$$

This scheme is $\mathcal{O}(dt)$ accurate in time.

3.2.2 Explicit Fourth Order Runge-Kutta

Defining $f(c^k)$ as $\nabla^2((c^k)^3 - (c^k) - \nabla^2(c^k))$, then an estimate for c^{k+1} can be constructed from evaluations of f.

$$\begin{aligned} k_1 &= dtf(c^k) \\ k_2 &= dtf(c^k + \frac{k_1}{2}) \\ k_3 &= dtf(c^k + \frac{k_2}{2}) \\ k_4 &= dtf(c^k + k_3) \\ c^{k+1} &= c^k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{13}$$

This scheme is $\mathcal{O}(dt^4)$ accurate in time.

3.2.3 First Order Backward Euler

For First Order Backward Euler, spatial derivatives are taken at the next time-step.

$$\frac{c^{k+1} - c^k}{dt} = \nabla^2((c^{k+1})^3 - (c^{k+1}) - \nabla^2(c^{k+1})) \tag{14}$$

Collecting c^{k+1} terms on the left-hand-side (except for the element-wise cubed term):

$$\underbrace{[I + dt\nabla^2 + dt\nabla^2\nabla^2]}_{A} c^{k+1} = \underbrace{c^k + dt\nabla^2(c^{k+1})^3}_{b} \tag{15}$$

The system $Ac^{k+1} = b$ is then solved iteratively. This scheme is $\mathcal{O}(dt)$ accurate in time.

3.2.4 Second Order Crank-Nicolson

For Second Order Crank-Nicolson, spatial derivatives are taken as the average of those at the current and next time-step.

$$\frac{c^{k+1} - c^k}{dt} = \frac{1}{2}\nabla^2((c^k)^3 - (c^k) - \nabla^2(c^k)) + \frac{1}{2}\nabla^2((c^{k+1})^3 - (c^{k+1}) - \nabla^2(c^{k+1})) \tag{16}$$

Collecting c^{k+1} terms on the left-hand-side (except for the element-wise cubed term):

$$\underbrace{[I + \frac{dt}{2}\nabla^2 + \frac{dt}{2}\nabla^2\nabla^2]}_{A} c^{k+1} = \underbrace{c^k + \frac{dt}{2}\nabla^2(c^{k+1})^3 + \frac{dt}{2}\nabla^2((c^k)^3 - (c^k) - \nabla^2(c^k))}_{b} \tag{17}$$

The system $Ac^{k+1} = b$ is then solved iteratively. This scheme is $\mathcal{O}(dt^2)$ accurate in time.

3.3 Spectral Methods

We define $g = c^3 - c$ which includes the non-linearity in the equation. Then the Cahn-Hilliard equation Eq. (4) can be written down in Fourier space as follows

$$\frac{\partial \tilde{c}}{\partial t} = -k^2(\tilde{g} + k^2\tilde{c}) \tag{18}$$

where $\tilde{c}, \tilde{g}$ are the Fourier components of c and g respectively, $k^2 = k_x^2 + k_y^2$ in 2D, k_x and k_y are the wavenumbers in x and y directions respectively. We assume the 2D physical domain is of size $L \times L$ and take $N \times N$ sampling points, so that the spacing in Fourier space is $2\pi/L$. We use the FFT routine in Numpy to carry out the discrete Fourier transform. To update in time, we implement Forward Euler and Fourth Order Runge-Kutta, which are explicit. We also implement a first order semi-implicit method which is more stable in time than explicit methods.

3.3.1 First Order Forward Euler

Spatial terms in Fourier space are taken at the current time step and the time derivative is discretized as follows

$$\frac{\tilde{c}^{j+1} - \tilde{c}^j}{dt} = -k^2(\tilde{g}^j + k^2\tilde{c}^j) \tag{19}$$

Then $\tilde{c}$ at next time step is updated using

$$\tilde{c}^{j+1} = \tilde{c}^j - dtk^2(\tilde{g}^j + k^2\tilde{c}^j) \tag{20}$$

To calculate $\tilde{g}$ at the next time step, we first carry out the discrete Fourier transform of $\tilde{c}^{j+1}$. Then we calculate g in real space since this calculation will only involve products. Finally, we perform a discrete Fourier transform on g. If we kept all the calculations in Fourier space, then we would have to do the convolution in Fourier space for the nonlinear terms. The calculation for convolution would cost more time.

3.3.2 Fourth Order Runge-Kutta

We define $f(\tilde{g}^j, \tilde{c}^j) = -k^2(\tilde{g}^j + k^2\tilde{c}^j)$. Then the estimate for $\tilde{c}^{j+1}$ is as follows

$$\begin{aligned}
d_1 &= dtf(\tilde{g}^j, \tilde{c}^j) \\
d_2 &= dtf(\tilde{g}_1^{j+1/2}, \tilde{c}^j + d_1/2) \\
d_3 &= dtf(\tilde{g}_2^{j+1/2}, \tilde{c}^j + d_2/2) \\
d_4 &= dtf(\tilde{g}_1^{j+1}, \tilde{c}^j + d_3) \\
\tilde{c}^{j+1} &= \tilde{c}^j + \frac{1}{6}(d_1 + 2d_2 + 2d_3 + d_4)
\end{aligned} \tag{21}$$

To calculate $\tilde{g}_1^{j+1/2}$, we first perform an inverse discrete Fourier transform on $\tilde{c}_1^{j+1/2} = \tilde{c}^j + d_1/2$ to obtain $c_1^{j+1/2}$ in real space, then calculate the product to obtain $g_1^{j+1/2}$ in real space, on which we perform one last discrete Fourier transform. A similar procedure applies to the calculation of $\tilde{g}_2^{j+1/2}$ and $\tilde{g}_1^{j+1}$.

3.3.3 First Order Semi-Implicit method

Instead of taking all the spatial terms at the current time step like in the explicit methods, we take the $k^2\tilde{c}$ at the next time step and the nonlinear term, $\tilde{g}$, at the current time step. Then the time derivative is discretized as follows

$$\frac{\tilde{c}^{j+1} - \tilde{c}^j}{dt} = -k^2(\tilde{g}^j + k^2\tilde{c}^{j+1}) \tag{22}$$

Then $\tilde{c}$ at next time step is updated using

$$\tilde{c}^{j+1} = \frac{\tilde{c}^j - dtk^2\tilde{g}^j}{1 + k^4 dt} \tag{23}$$

4 Comparison of Second and Fourth Order Central Differences

In order to decide which of the two spatial schemes to use for the finite difference method, we decided to compare the runtimes of the Second Order Central Difference and the Fourth Order Central Difference schemes. We ran the code with a 100×100 spatial grid and a $0.001s$ time step with both a First Order Forward Euler and a (explicit) Fourth Order Runge Kutta integration scheme. The time step was chosen small enough to ensure numerical stability, an issue we will address in the next section. With the First Order Euler, the total runtime for the Second Order Central Difference was $80.863s$ while the total runtime for the Fourth Order Central Difference was $99.052s$. On the other hand, with the (explicit) Fourth Order Runge Kutta, the total runtime for the Second Order Central Difference was $208.446s$ while the total runtime for the Fourth Order Central Difference was $246.966s$. As expected, the runtime complexity for the two spatial methods are not drastically different because they both involve similar matrix operations, only the Laplacian defined using a Fourth Order Central Difference is marginally less sparse. Recall that the error with a Fourth Order Central Difference scheme is $\mathcal{O}(dx^4)$ while the error with a Second Order Central Difference is $\mathcal{O}(dx^2)$. Since the runtimes are almost identical, we decided to use the Fourth Order Central Difference scheme for the finite difference method due to its higher accuracy.

5 Explicit and Implicit Time Methods with Finite Differences

Next, we decided to compare the runtimes of the different integration schemes. We ran the code with a 100×100 spatial grid and the Fourth Order Central Difference spatial scheme. For each integration scheme, we found the largest time step we could use while ensuring a numerically stable solution. We adjusted the

number of time steps accordingly such that the total time remained the same for the different integration schemes, and measured the total runtime.

	Integration Scheme	dt_{max} **(s)**	**Runtime (s)**
Explicit Methods	First Order Forward Euler	0.001	89.136
	Fourth Order Runge Kutta	0.001	232.442
Implicit Methods	First Order Backward Euler	0.1	794.635
	Second Order Crank Nicolson	0.5	179.085

Table 1: Comparing the total runtimes for explicit and implicit methods

In order to ensure numerical stability with the explicit methods, we needed to use very small time steps. On the other hand, as expected, the implicit methods allowed us to use comparatively larger time steps. However, even with the largest time step we could use, the total runtime for the First Order Backward Euler was much bigger compared to the explicit methods. However, the Second Order Crank Nicolson allowed us to take a appreciably large time step, yielding a runtime which was bigger than the runtime of the First Order Forward Euler but much smaller than that of the Fourth Order Runge Kutta Method. It is important to note that the error on the Second Order Crank Nicolson was $\mathcal{O}(dt^2)$ while the error on the Fourth Order Runge Kutta was $\mathcal{O}(dt^4)$. Since the runtimes for these two methods were not drastically different, we decided to choose the integration scheme which yielded a lower error. As we will discuss in further detail in the next two sections, the poor performance of the implicit methods may be due to the non-linearity in the governing equations.

6 Investigation of "Linearized" Cahn-Hilliard Equation

We investigated if the poor performance of the implicit methods is truly due to the non-linearity of the equation by carrying out numerical calculations of a "Linearized" Cahn-Hilliard equation. We take out the nonlinear term c^3 resulting in the following linear equation.

$$\frac{\partial c}{\partial t} = \nabla^2(-c - \nabla^2 c) \tag{24}$$

For the two implicit methods we investigated, we use time steps much bigger than the largest allowed time steps (listed in section 5) for nonlinear Cahn-Hilliard equation and measure the runtime.

	Integration Scheme	dt **(s)**	**Runtime (s)**
Implicit Methods	First Order Backward Euler	1	10.01
	Second Order Crank Nicolson	2	7.29

Now the two implicit methods can run much faster. This is because we can choose a much larger time

step and the equation is now linear. The equation doesn't have to be solved iteratively in the implicit methods. The results in this section confirm our explanation for the poor performance of the implicit methods. In the following sections for the comparison of different schemes to solve Cahn-Hilliard equation, we will therefore employ the explicit methods in time for the finite difference method.

7 Comparison of Explicit Methods with Finite Differences

We now compare the two explicit methods with each other, and attempt to derive a bound on the error for the best out of the two explicit methods. First, we compare the First Order Forward Euler with the Fourth Order Runge Kutta method while using a Fourth Order Central Difference scheme in space. In order to compare the results from the two integration schemes, we run the code with a 100×100 grid with the same time step and compute the relative error between the two solutions at ten time slices as follows.

$$Relative\ Error, \epsilon_k = \left| \frac{\Phi_{k,Euler} - \Phi_{k,Runge-Kutta}}{\Phi_{k,Runge-Kutta}} \right| \ \forall\ k \in \{1, 10\} \tag{25}$$

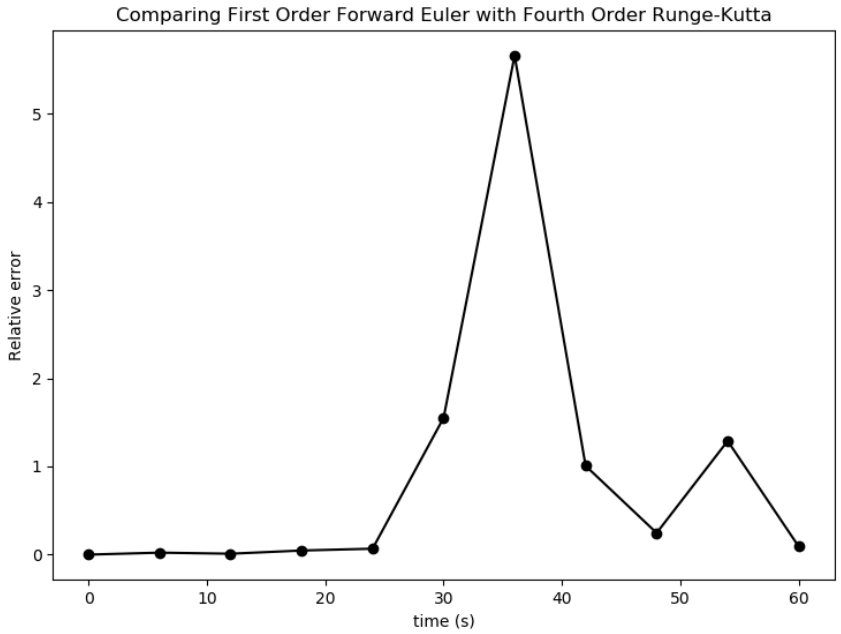


Figure 2: Comparing the two explicit methods at different time slices shows that the First Order Euler performs significantly worse than the Fourth Order Runge Kutta method. Because we know that the Fourth Order Runge Kutta method yields a lower temporal discretization error, it is natural to trust the solution from the Runge-Kutta method more than the First Order Euler method.

Because the solution obtained with the First Order Euler Method is significantly different from the one obtained with the Fourth Order Runge Kutta method, and because we trust the higher order Runge Kutta scheme to produce a lower discretization error, we decided to use the Fourth Order Runge Kutta method as our main integration scheme. However, it would be helpful to be able to somehow quantify the error with the Fourth Order Runge Kutta method. In order to estimate the error, a Fifth Order Runge Kutta method was implemented. The relative error between the fifth order solution and the fourth order solution gave an upper bound on the error on the fifth order solution. The error bound provided a numerical threshold below which we could not trust our solutions. This gives us sufficient justification to use the Fourth Order Runge Kutta scheme with a Fourth Order Central Difference spatial formula as the true solution.

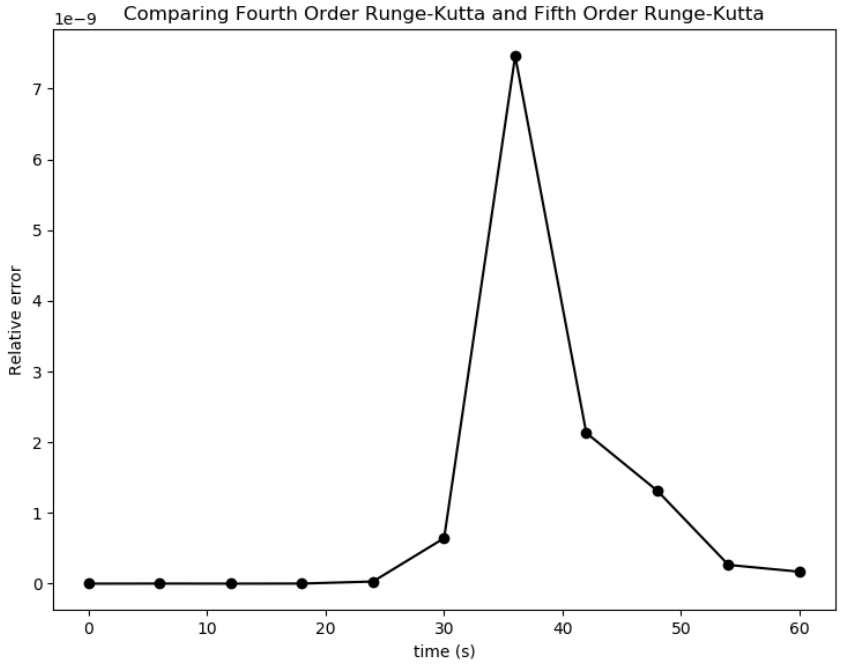


Figure 3: As seen in the plot above, the difference between the Fourth Order Runge-Kutta Method and the Fifth Order Runge-Kutta Method is insignificant.

8 Comparison of Finite Difference Methods and Spectral Methods

We use the Fourth Order Runge-Kutta scheme with a Fourth Order Central Difference to solve for the solution of the Cahn-Hilliard equation and use that solution as the true solution. In order to investigate the performance of spectral methods, we use the fourth order Runge Kutta scheme to update the Fourier spectrum in time. We use the same time step to run the numerical calculations of both methods with different spatial resolutions. This time step is chosen such that numerical stability is ensured for all the spatial resolutions we use. Specifically, we keep the system size $L = 50$ always the same. We choose the number of sampling points in one direction $N = 25, 40, 50, 100$ and we use the time step $dt = 0.0004s$ which is stable for the finest grid in the calculation for comparison. Here the largest time step we can use is constrained by the spectral method with Fourth Order Runge-Kutta scheme. We also need a metric to measure the error. We use Fourth Order Central Difference with fourth order Runge-Kutta scheme and use an even finer grid ($N = 200$) with a smaller time step $dt = 0.00008s$ to solve the Cahn-Hilliard equation. This solution is then used to measure the error of the solutions generated with a coarser grid and larger

time step. To calculate the error, we choose a few time slices and then calculate the average error over these time slices.

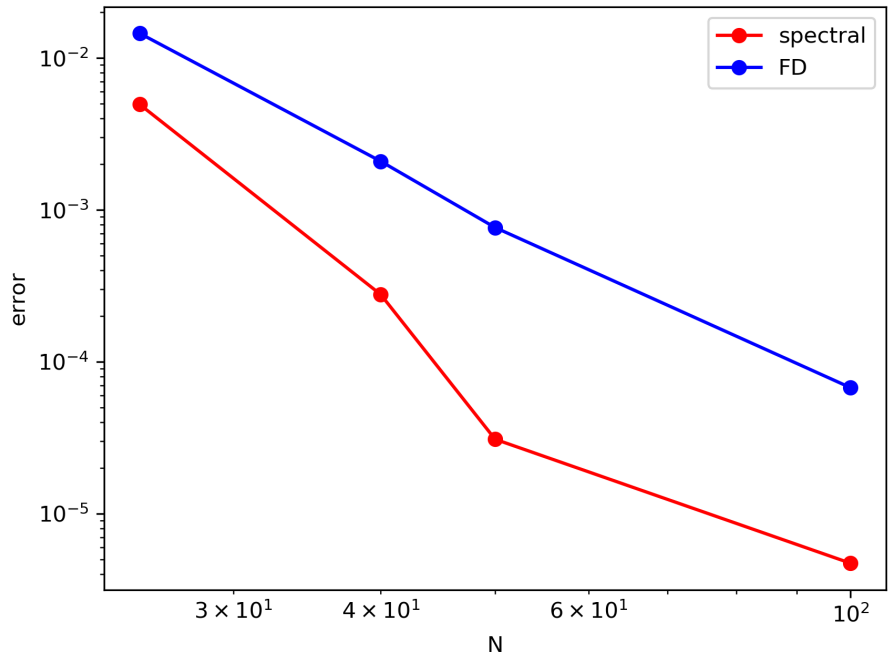


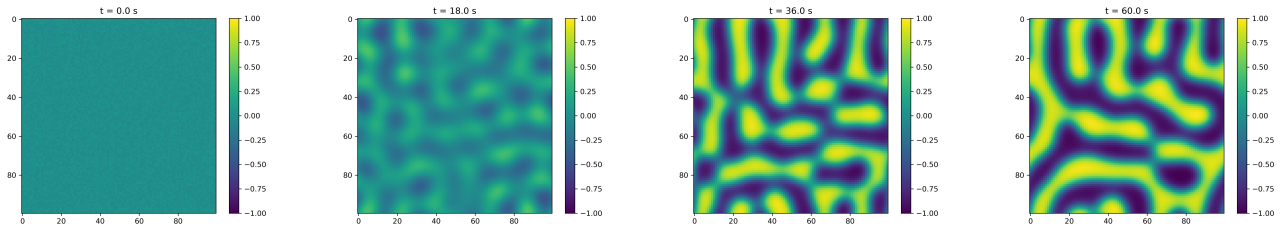
Figure 4: The error of the spectral method and fourth order finite difference method as a function of the number of grid points N

In Fig.4, we present the error for two methods as a function of the number of grid points in a log-log plot. The error for the finite difference method almost goes linearly with the number of grid points. The exponent obeys the order of accuracy required by the Fourth Order Central Difference scheme. The error for spectral method has a jump in the middle. This might indicate that choosing the number of grid points around that jump can capture some major Fourier modes of this problem. It can be seen from the comparison of the error for two methods that Fourier spectral methods need fewer sampling points than the Fourth Order Central Difference method in order to reach the same order of accuracy. For example, if we want to reach the accuracy of Fourth Order Central Difference with $N = 100$, we only need $N \approx 45 - 50$ for spectral method. We can see from the following table of runtime that using spectral method can reach that accuracy with less runtime. Although if we want to reach the accuracy of fourth order central difference with a coarser grid, like $N = 40$, we need $N \approx 25$ for spectral method. For this accuracy, it takes more time for spectral method. So we believe spectral method is better to reach higher accuracy.

N	Runtime of 4^{th} order central difference (s)	Runtime of spectral method (s)
25	32.55	87.16
40	54.72	119.91
50	73.50	146.53
100	233.98	381.69
200 (**metric**)	4856.79 (dt=0.00008s)	

9 Simulation for Optimal Method and Random Initial Conditions

Selecting the spectral spatial method with fourth order Runge-Kutta time integration as the best balance between computational cost and accuracy, we now change our initial conditions. Previously, we were initializing the concentration with a function so that the initial conditions would be comparable when changing the grid resolution. A more physically meaningful initial condition for the problem of phase separation, however, is a series of random values close to zero, bounded by the absolute value of some specified number. As time evolves, one should be able to observe regions of different phases represented by concentrations of -1 and 1.



(a) t = 0 seconds (b) t = 18 seconds (c) t = 36 seconds (d) t = 60 seconds

Figure 5: Simulated phase separation using random initial fluctuations bounded by $|\delta| \leq 0.05$, spectral in space, RK4 in time, on a 100×100 grid for $L = 50$, $T = 60$. Color represents phase concentration. x and y values correspond to jth and ith entries in c[i][j] matrix, respectively.

10 Summary of Results

We initially set out to investigate the relative applicability of multiple spatial and temporal schemes to solving the Cahn-Hilliard equation. Beginning with finite differences, we found that using a Fourth Order Central Difference was only marginally more computationally expensive versus the less-accurate Second Order Central Difference, deciding that the added accuracy outweighed the corresponding increase in cost. Continuing our investigation of finite differences, we compared four different temporal schemes (two explicit, two implicit). We expected to find that the added stability provided by implicit methods would allow us to

increase our time-step enough to offset the additional computational cost per time-step through a reduction in the number of steps. We instead discovered that the implicit methods were not sufficiently stable to provide a significant advantage over the explicit methods. According to linear theory, the implicit methods should be stable. To prove that the non-linearity of the Cahn-Hilliard equation is indeed responsible for the observed limitation in the implicit time-steps, we removed the non-linear term. Again, comparing multiple time methods for this linear version of the Cahn-Hilliard equation, we observed what we initially expected to - the implicit time steps could be chosen to be sufficiently large so as to make implicit methods more efficient than explicit ones. Despite the slight efficiency advantage of Crank-Nicolson, we decided that the added time-accuracy of an explicit fourth order Runge-Kutta method overshadowed the minor increase in runtime. We then used an embedded Runge-Kutta scheme and constructed an estimate for a 'true' solution with a conservative bound on the error. Fourth Order Central Differences were then compared to spectral methods. The spectral methods were more expensive per time-step, but required a less-coarse grid than finite differences to yield the same error, showing that spectral methods are the most efficient overall.