AMATH 581 Final Report

Bose-Einstein Condensate in 3D

Daniel Leon

December 16, 2021

1 Introduction

Here we investigate the behaviour of the cubic non-linear Schrödnger equation under a lattice potential, a set-up known as the Gross-Pitaevskii system. It is our understanding, as someone only trained in chemical engineering and not quantum physics, that this set-up is used to model a periodic dilute gas known to be in the form of a Bose-Einstein condensate, which we think is some unique phase of matter where particles are synchronized at the atomic level, which can be described by obtaining the wave function $\psi(\vec{x}, t)$, where $\vec{x} = (x, y, z)^T$ in 3D. The wave function can be obtained by solving the aforementioned Gross-Pitaevskii system. The following is the dimensionless scaled version of the NLSE, which can be obtained using the Buckingham Π theorem.

$$i\psi_t = -\frac{1}{2}\nabla^2\psi + |\psi|^2\psi - [A_1 \sin^2(x) + B_1][A_2 \sin^2(y) + B_2][A_3 \sin^2(z) + B_3]\psi$$

The prescribed initial conditions are

$$\psi_0 = \sin(x)\sin(y)\sin(z)$$

$$\psi_0 = \cos(x)\cos(y)\cos(z)$$

Once we got the simulation running we decided to try different initial conditions, so we came up with two more initial conditions that also lead to interesting visualizations, along with all of the other manipulations to the conditions that were made.

$$\psi_0 = \sin(x^2)\sin(y^2)\sin(z^2)$$

$$\psi_0 = \cos(x^2)\cos(y^2)\cos(z^2)$$

We look at three-dimensional condensates with two sets of initial conditions outlined in the assignment, as well as two additional modifications to the given initial conditions. Additionally, we delved more deeply into modifications of the given parameters to investigate the behavior of the system at shorter time scales and with higher Fourier modes to obtain a level of detail much greater than required, (I realize that a big reason for limiting the resolution is so that the code can run on most systems at a reasonable time) and we looked at different relations between the constants A_i and B_i.

2 Solution Approach

A spectral solution approach using three-dimensional Fourier transform in the infinite spatial dimensions was used, which resulted in an ordinary differential equation in time, but in the Fourier

domain of $\vec{x}$. That solution then needs to be inverse transformed in order to view the solutions in $\mathbb{R}^3$, but the Fourier domain solutions are very worth viewing as well. Since our system was able to be put into the form

$$\frac{\partial u}{\partial t} = Lu + N(u)$$

where L is a linear, constant coefficient operator, and $N(u)$ contains non-linear and non-constant coefficient terms. We know how to solve these types of equations through the techniques of the course, specifically, Fourier transforming that sucker!

The visualization of all of the information contained in these data-sets, is a critical component of the problem. Since the NLSE has complex variabled input, we get to deal with both the real and imaginary components. We chose to display both the real and imaginary parts of both versions, both ψ and $\hat{\psi}$ using MATLAB's `subplotn` `isosurface()` and `imwrite()` commands, among many others to produce animated gifs. We also formatted the plots to display critical information, such at the governing equation and initial conditions, current time step, number of Fourier modes included in the 3D Fourier transform, and the value of the constants A_i and B_i.

3 Implementation

To solve this system numerically, we already mentioned the implementation of a spectral solution technique that, after applying the fast Fourier transform algorithm, we are left with an ordinary differential equation. That can then be solved using the standard numerical ODE solution techniques. In our case, we made use of the workhorse that is MATLAB's `ode45()`, which is a Runga-Kutta iteration scheme who's local and global error is $O(\Delta t^5)$ and $O(\Delta t^4)$ respectively.

As input and output, the solver takes vectors, so our 3D system needs to be reshaped for each iteration step of the time-stepper. However, for the evaluation and computation, we are generally working with 3D arrays, or tensors. So, within the RHS function for the ode solver, we include unpacking, inverse Fourier transforming, and reshaping steps before carrying out mathematical operations, the re-packed everything to be both in Fourier space and reshaped into a vector.

Upon successful integration by the time-stepper, we obtain a vector of $\hat{\psi}(\vec{k}, t)$ that has length $n^3 \times t_{steps}$, where n is the number of Fourier modes and t_{steps} is the number of time steps taken between t_0 and t_f. First that needs to be reshaped into an $n \times n \times n \times t_{steps}$ tensor. Then the solution can be inverse FT and stored into one $n \times n \times n \times t_{steps}$ tensor for viewing $\psi(\vec{x}, t)$. The same reshaped solution in Fourier space needs to be hit with `fftshift()` (along with the grid of k_x, k_y, k_z points) before the Fourier space solutions can be viewed.

To view this data in a cohesive manner, we chose to make use of MATLAB's `isosurface()` for visualizing all four (real and imaginary components in both real and Fourier space) components of the solutions for each time-step in the form of an animated gif. To include the context of each simulation, including initial conditions, current time-step, etc. we made extensive use of variables passed into the `animateBEC()` function that we set up, along with dynamic text boxes that set LaTeXand other strings onto the current frame of the figure. We made extensive use of the `sprintf()` function for the purpose just stated, as well as for dynamically generating file names that include the relevant parameters (this later format evolved over time).

One thing that we want to do, but ran out of time to figure it out, was to add a colormap to

the isosurface() plots, but alas...

4 Results

Now we will show a few screenshots of these gifs, but this is only a small sampling of the beautiful shapes, in both the spatial and Fourier spaces, created by this system. All of these following images are of simulations with 64 Fourier modes (the computer wasn't doing well with 128, I guess I found some limiting behavior there). To start with, here is the initial condition of sines with all of the stock values for A and B ($A_i = -1$, $B_i = 1$).

Here is a link to an album where we stored some of the most interesting gifs of a bunch of different conditions that were tried, many with 64 Fourier modes and smaller Δt. I am going to show several images in succession in the **F**igures section (after the **C**onclusions section). The figure descriptions attempt to encapsulate the main features of the data presented. Under the basic conditions, in the spatial domain, we generally see these equally spaced blobs that "breath" and flip orientation periodically. Movement in the Fourier space tends to correlate with movement in the spatial domain. On some occasions one of the data sets would just be a messy blob; this was more common in the Fourier space. On one occasion, that blob was observed on a 2π interval, but more delicate features were on a 4π domain, indicating to me that my domain may not contain all of the relevant features of the system under the given conditions, but that may not be true. I would need to investigate further. Another thing that I would change with my plots would be to increase the font size of the t that displays the current time step.

5 Conclusions

I wish that I could continue running simulations, improving my visualizations, and perfecting my analysis here, but I will note that we were supposed to talk about other systems or set-ups that we are now equipped to explore. I expanded these concepts beyond the original scope by looking at higher Fourier modes, investigating alternative initial conditions, time spans and step-size, and trying different conditions for the coefficients, all of which showed interesting results.

I could have gone for other visualization methods, such as contour plots at different planes across the domain, for instance. As far as other three-dimensional systems that we could investigate: there as so many, but the low hanging fruit would be 3D extensions to systems such as the wave equation (how do waves travel from a perturbation at the center of a body of fluid) and the heat equation (heat conduction through a 3D object) to name a few.

6 Figures

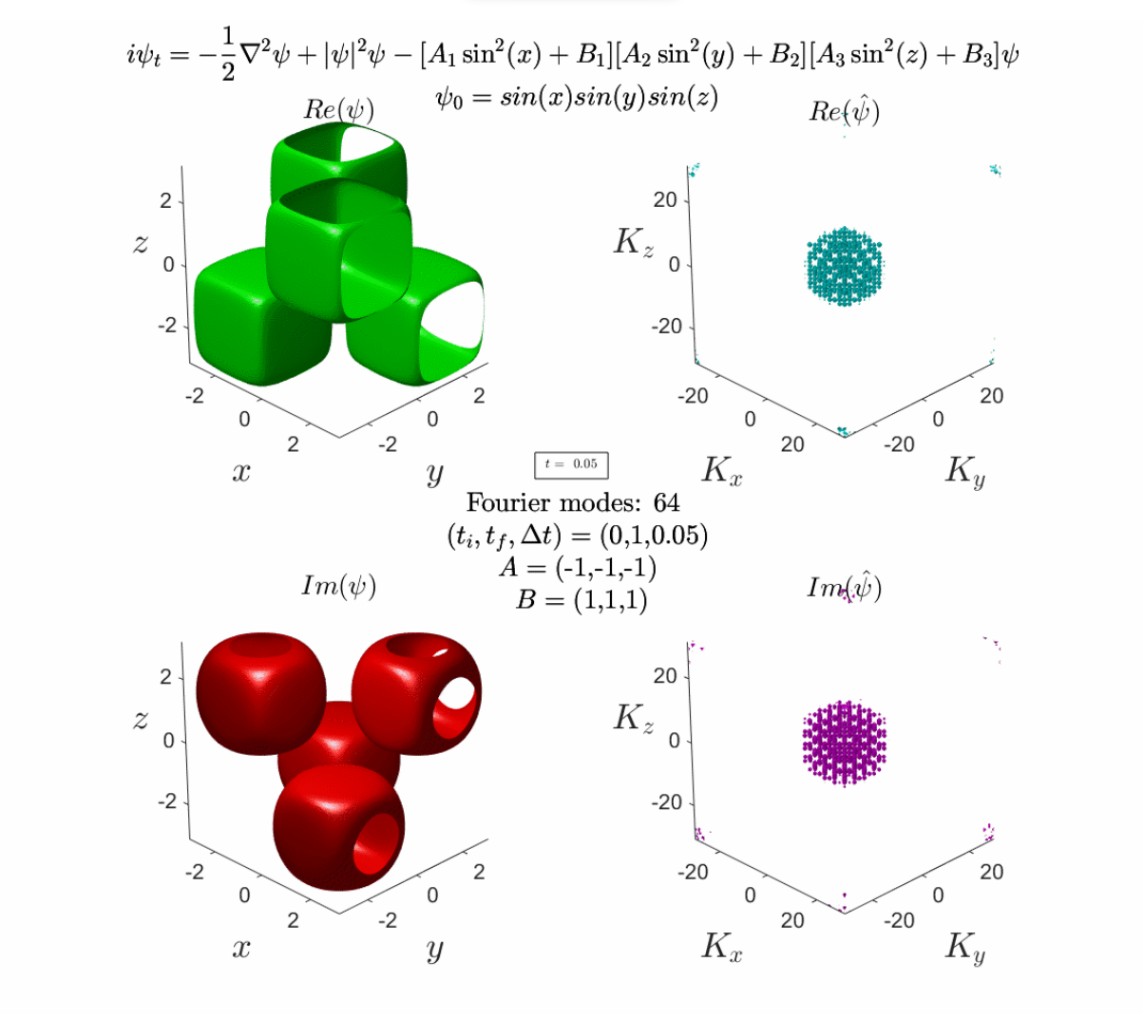

Figure 1: This is also close to the initial conditions of the system with initial conditions 2

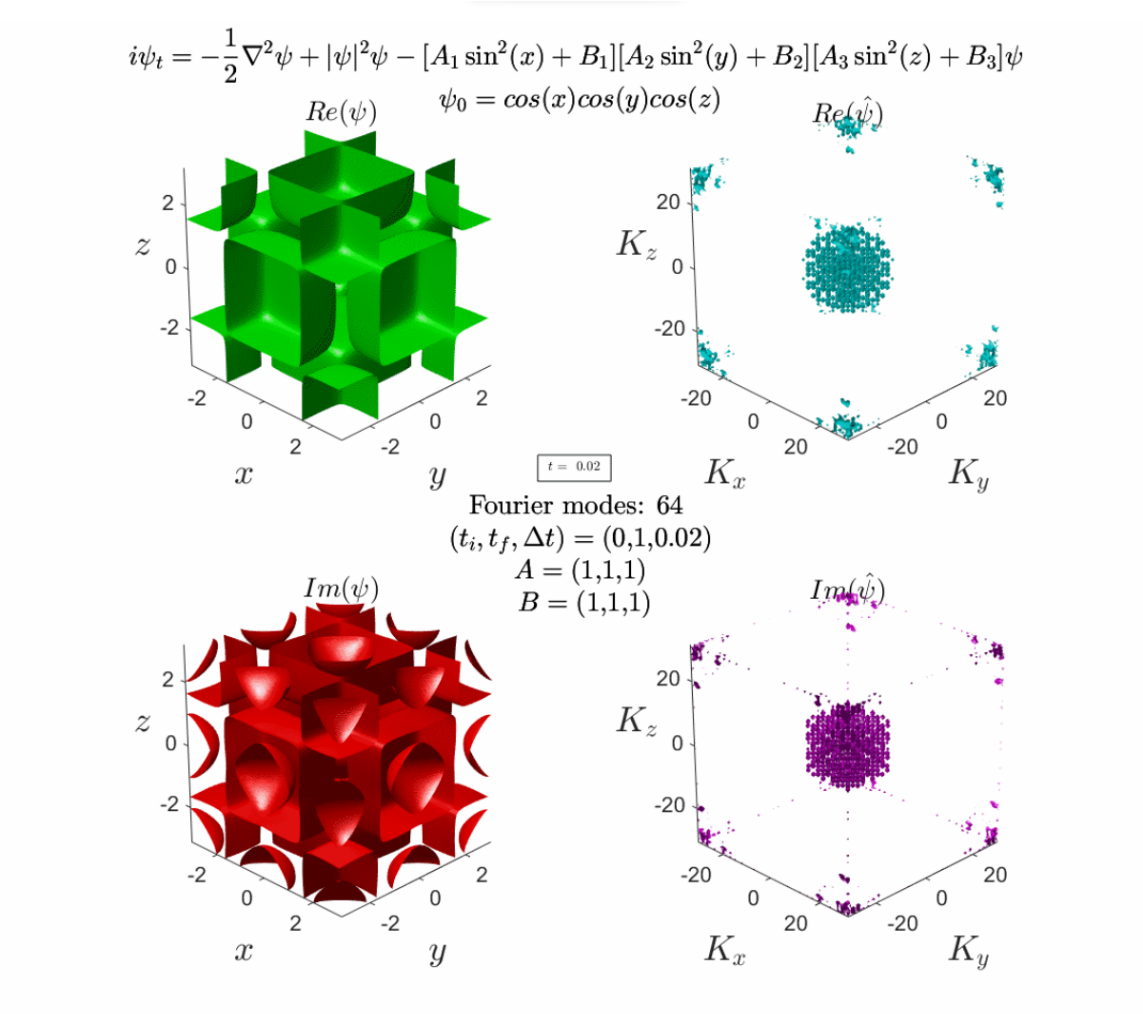

Figure 2: This is one step beyond the initial conditions of the system with initial conditions 1. One difference is that $A_i = B_i$. While the real solution in spatial domain appears the same as in the base prescribed conditions, there are differences elsewhere.

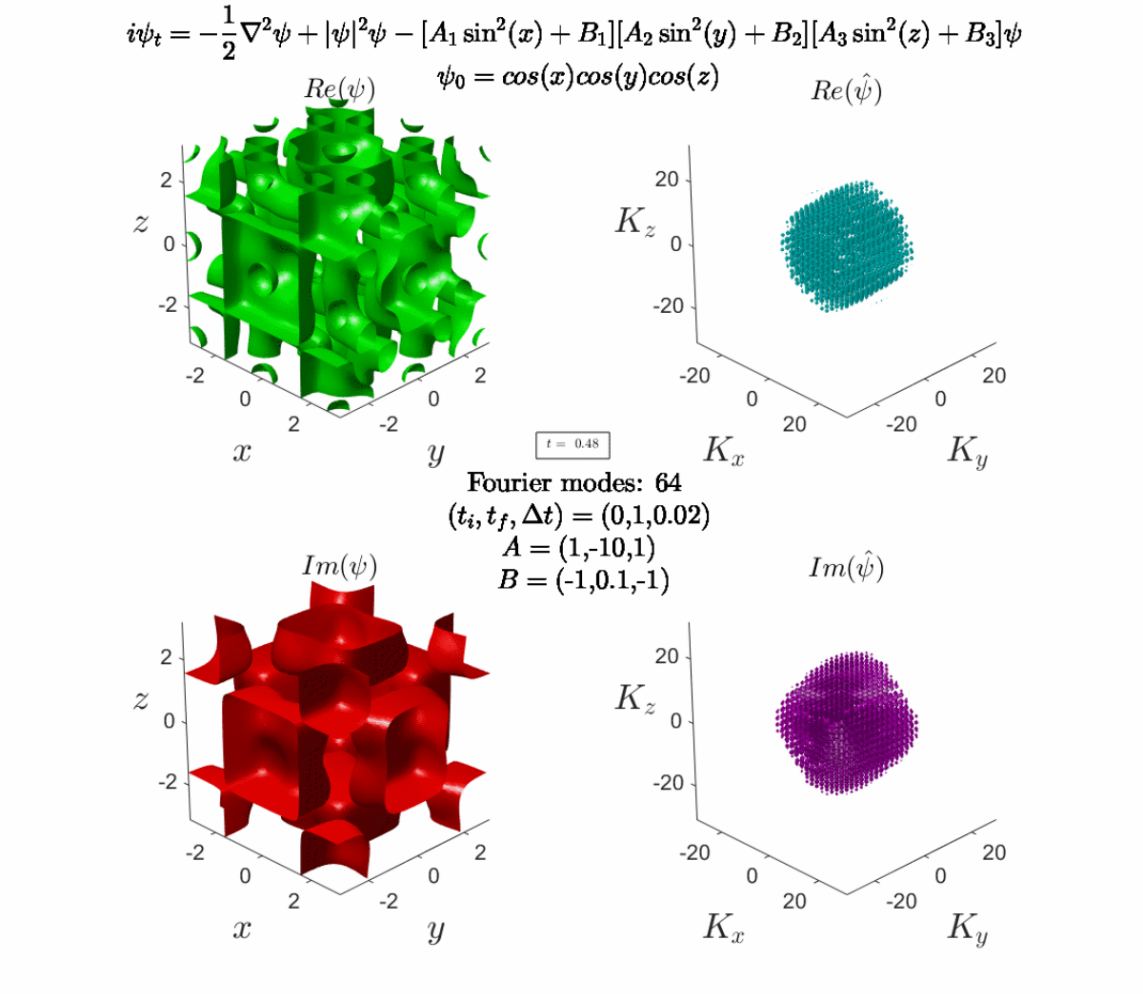

Figure 3: This is one step beyond the initial conditions of the system with initial conditions 1. One difference is the values of A_i and B_i are not all the same magnitude, as before. $t = 0.48$

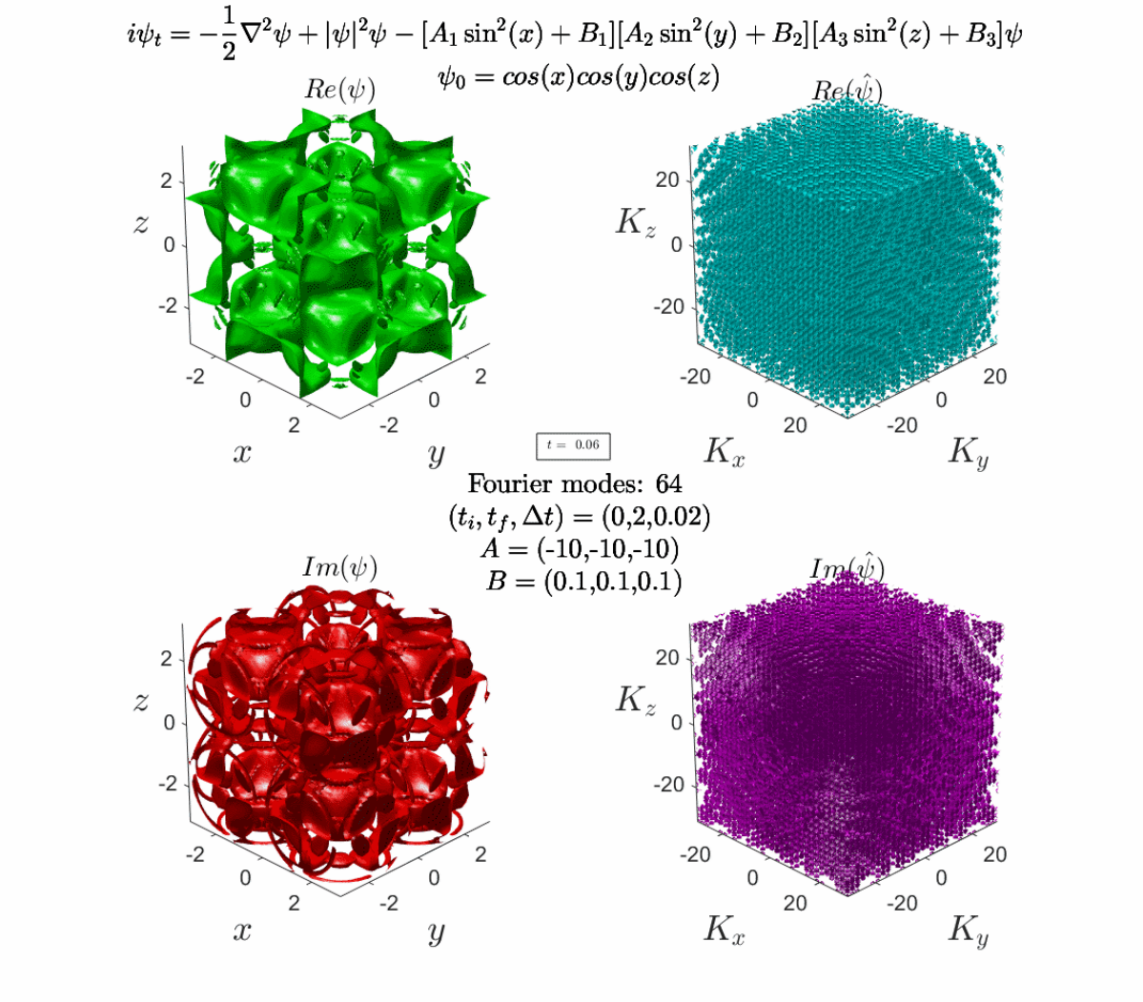

Figure 4: This is early after the initiation of the system with initial conditions 1. The spatial domain has really cool patterns very early after initiation, then they get big and blobby (apparently this isn't a word, according to Overleaf's spell-checker), like what's happening in the Fourier domain here. The Values of $|A_i| > |B_i|$. $t = 0.06$

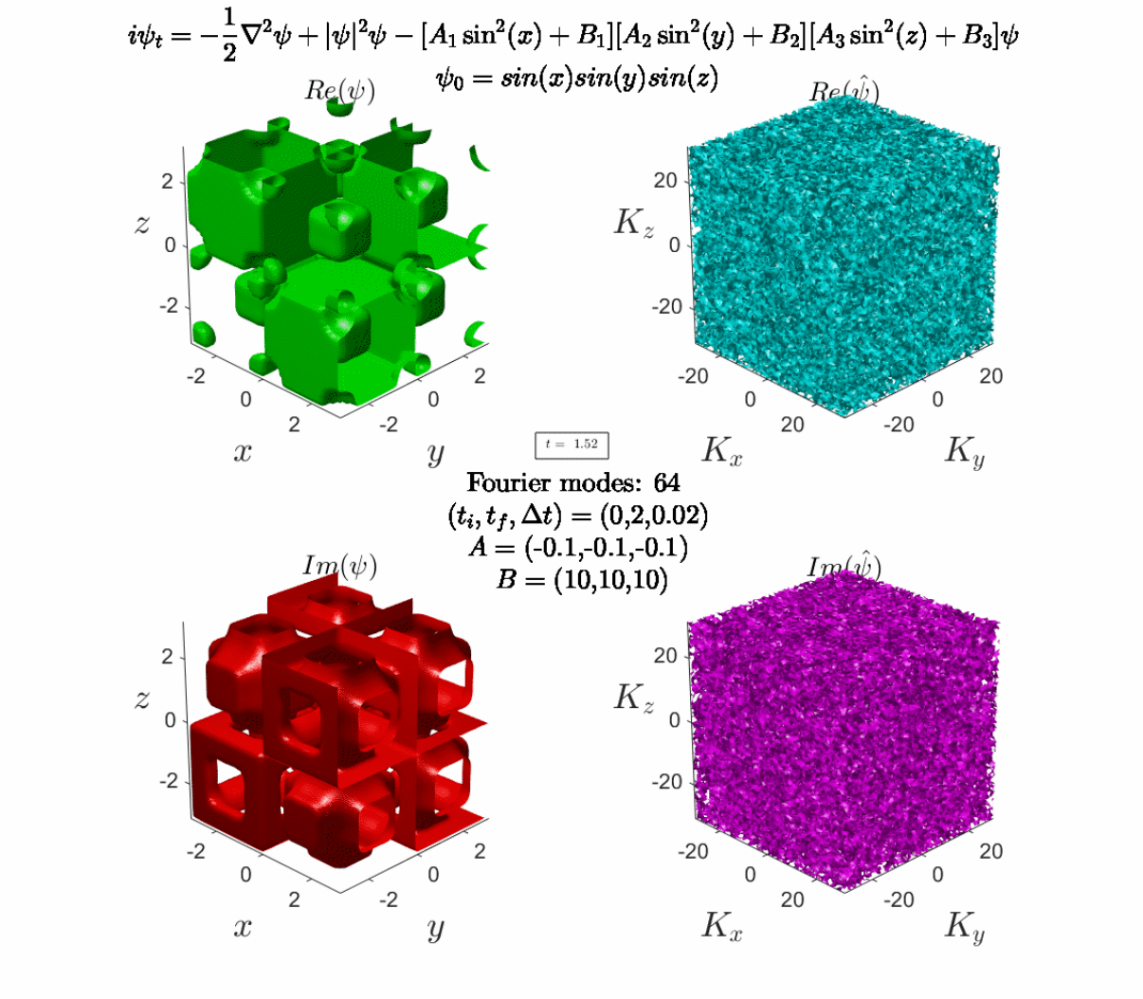

Figure 5: This image is pretty far into the system with initial conditions 2. The spatial domain has really cool patterns that cycle periodically to show these really neat looking geometric shapes that shrink/grow into/from themselves. The Fourier domain looks crazy like this pretty much the whole time. The Values of $|A_i| < |B_i|$. $t = 1.52$

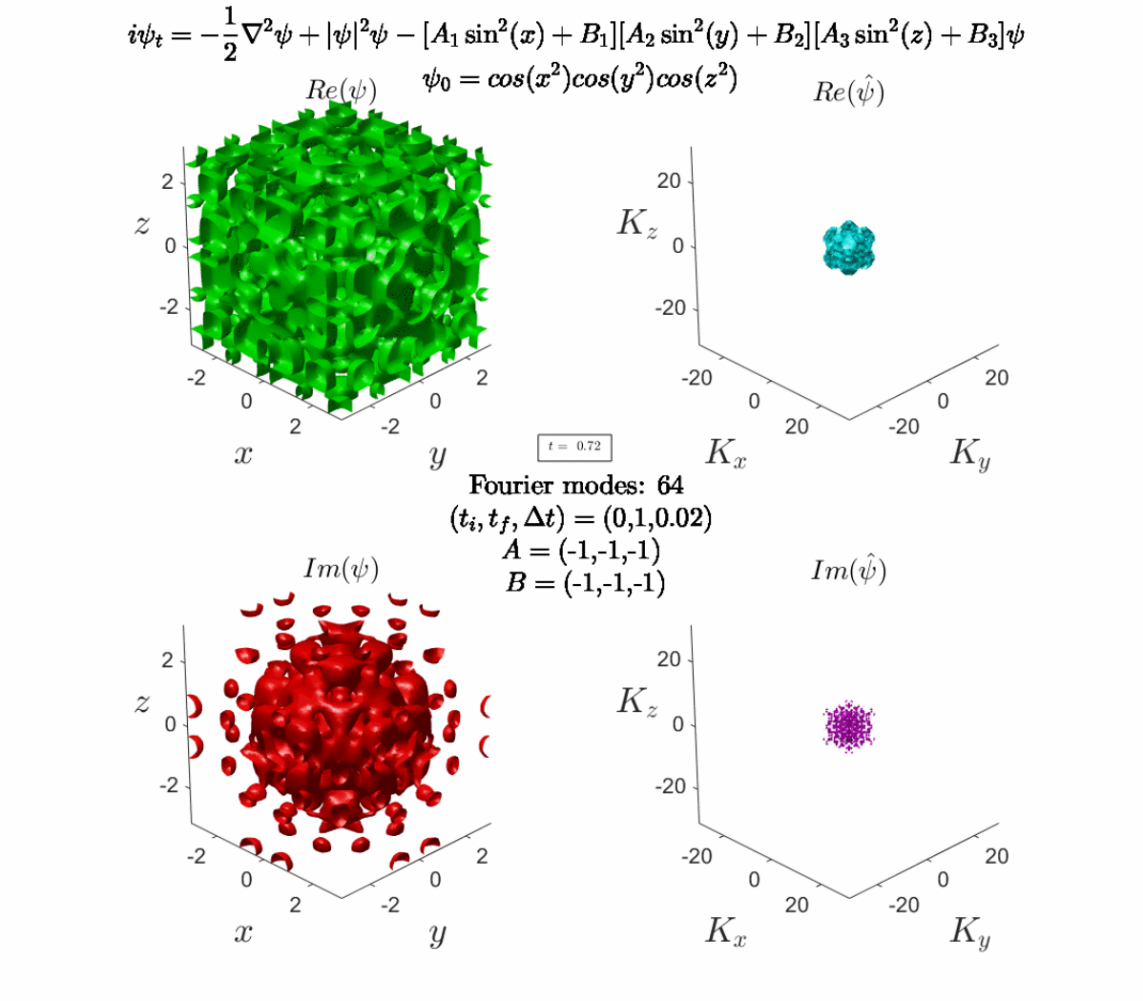

Figure 6: This is with initial conditions 3. The spatial domain has really cool patterns that cycle periodically to show these really neat looking geometric shapes that shrink/grow into/from themselves. The Fourier domain looks cool in this one too, shrinking and growing with time. The Values of $A_i = B_i$. $t = 0.72$

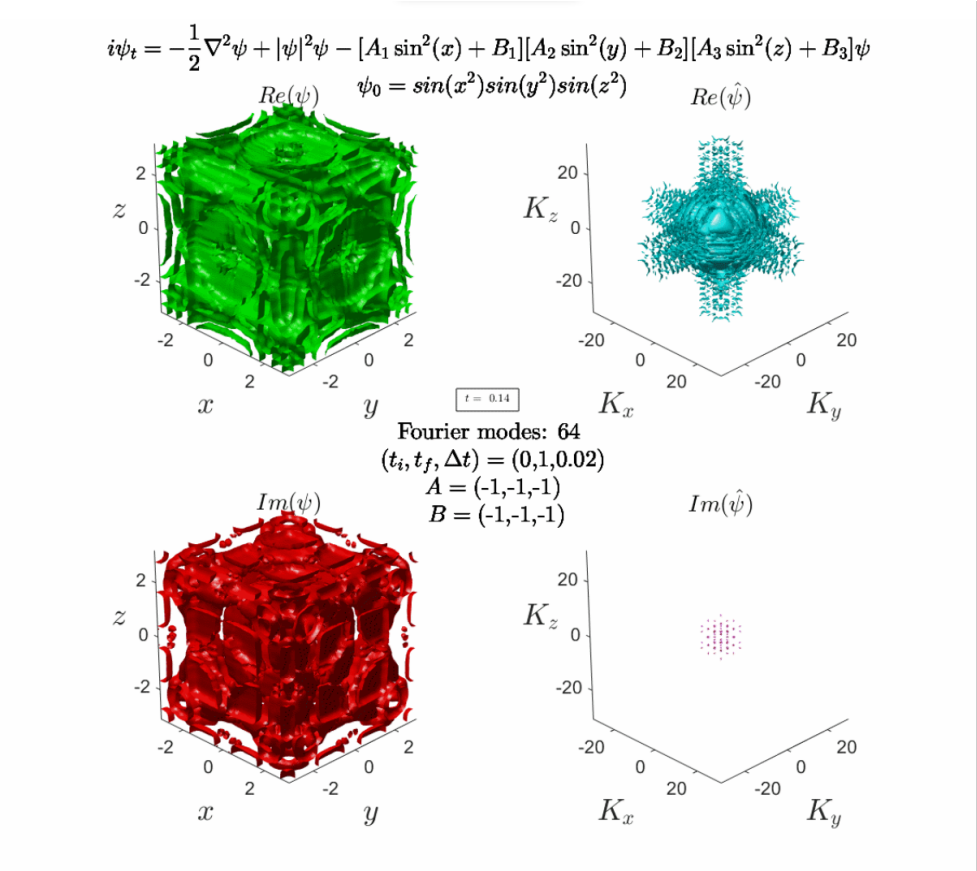

Figure 7: This is with initial conditions 4. The spatial domain has really cool filled-out patterns that cycle periodically to show these really neat looking geometric shapes that shrink/grow into/from themselves. The Fourier domain looks cool in this one too, shrinking and growing with time, occasionally with a nice spike, like shown here in the upper right. The Values of $A_i = B_i$. $t = 0.72$

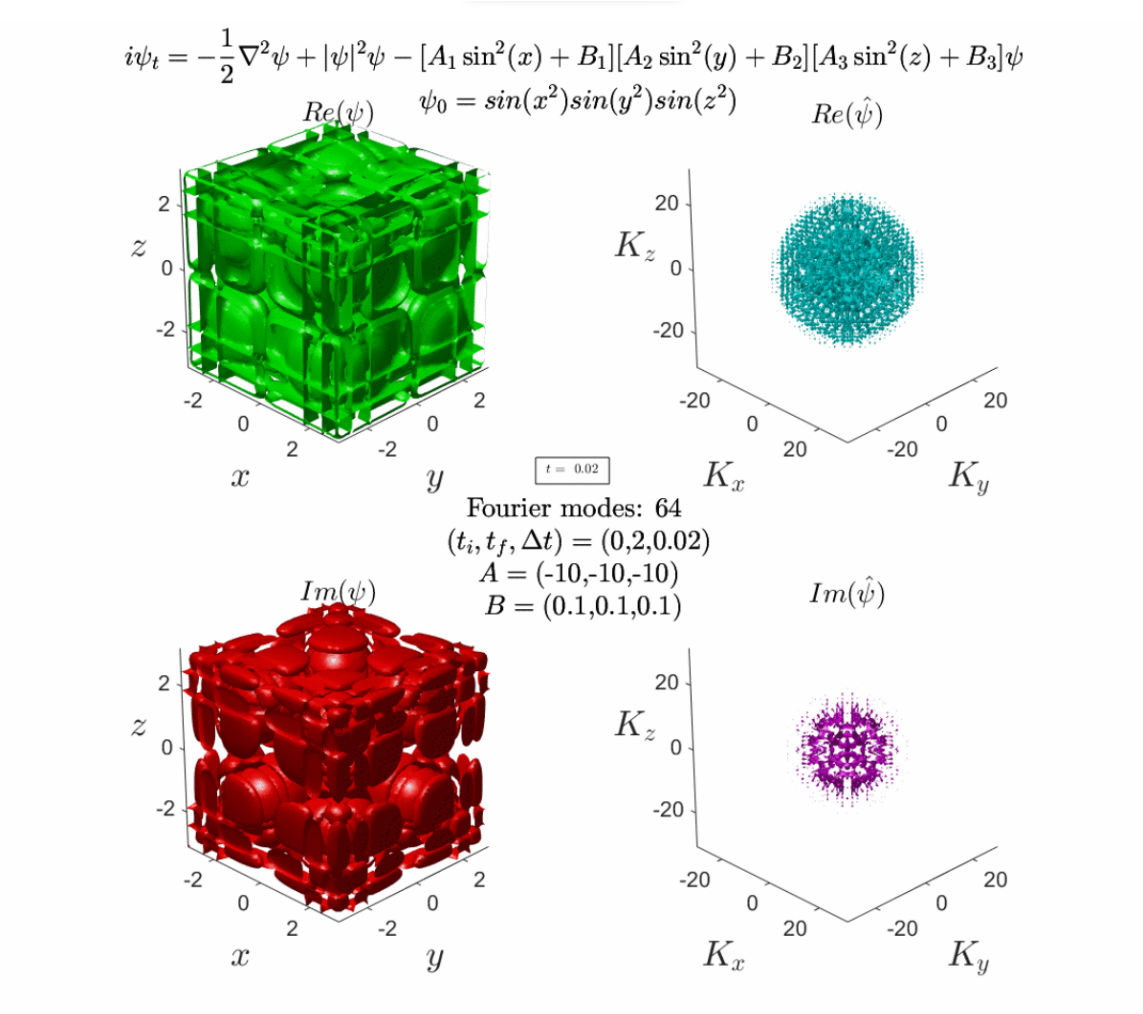

Figure 8: This is with initial conditions 4. The spatial domain has really cool patterns that cycle periodically early on, but they get messy looking quickly. The Fourier domain looks cool in this one, crystal looking balls. The Values of $|A_i| > |B_i|$. $t = 0.02$

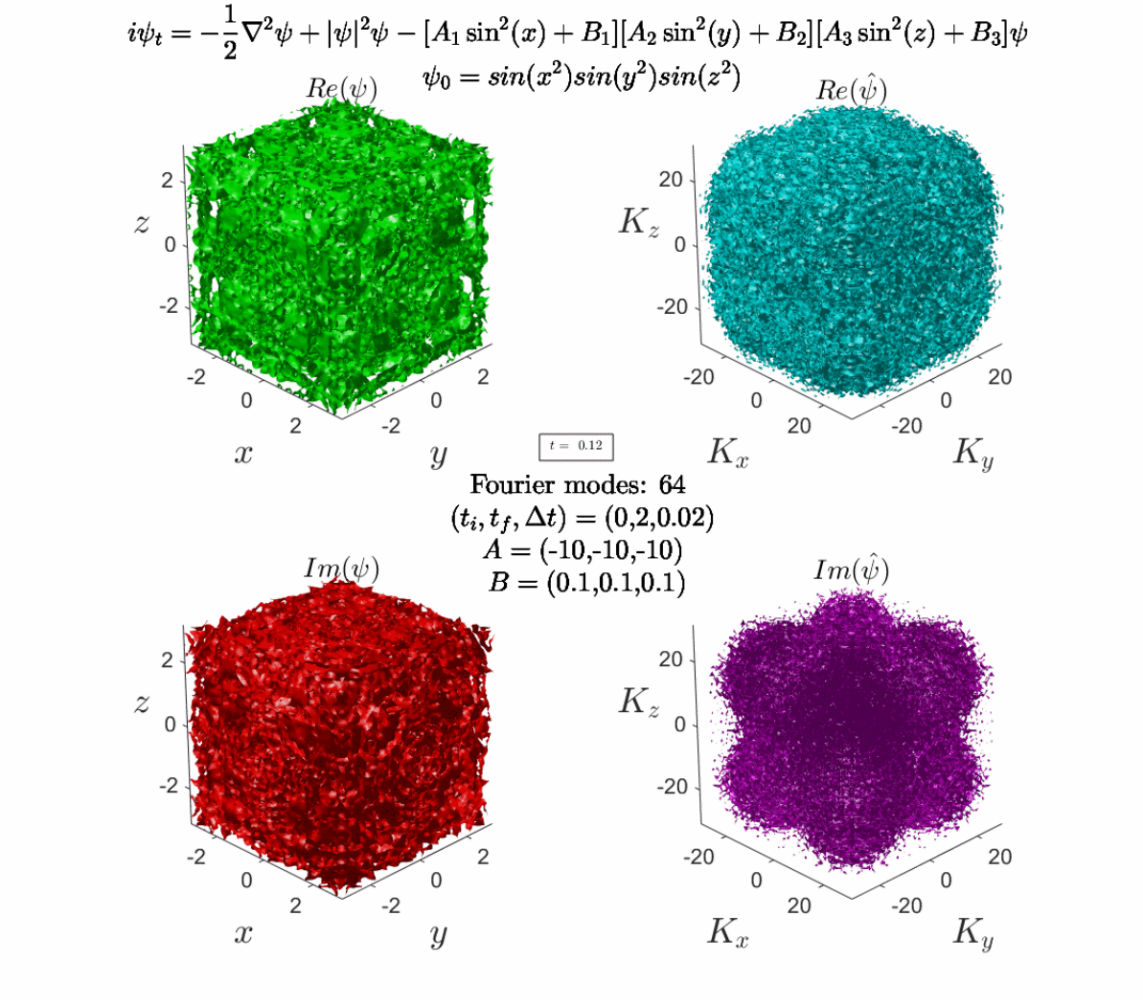

Figure 9: This is with initial conditions 4. This one is blobby . The Values of $|A_i| > |B_i|$. $t = 0.12$

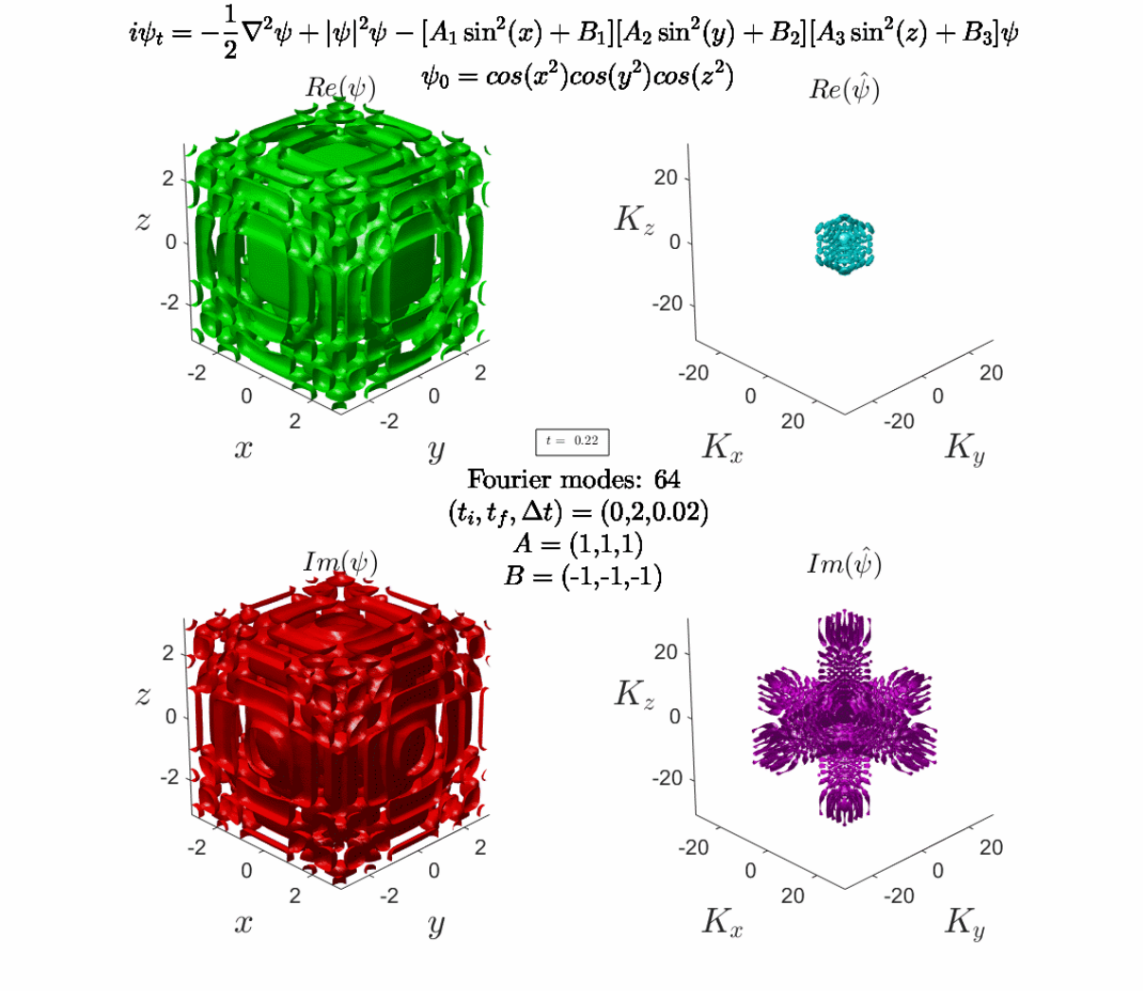

Figure 10: This last one with initial conditions 3 had some nice structure too. The Values of $|A_i| = |B_i|$. $t = 0.22$

7 MATLAB code

```
1
2  close all; clear all; clc
3  tic
4  A = -1*ones(3,1); B = -1.*A; % can set values for individual parameters
5  L = 2*pi;                        % define computational domain [-L/2 L/2]
6  n = 64;                          % define number of Fourier modes 2^n
7
8  tmax = 2;
9  dt = 0.02;
10 tspan = 0:dt:tmax;
11
12 x2 = linspace(-L/2,L/2,n+1);
13 x = x2(1:n); y = x; z = y;
14 [X, Y, Z] = meshgrid(x,y,z);
15
16 kx = (2*pi/L)*[0:(n/2-1) (-n/2):-1];       % rescale to 2pi domain on x
17 ky = kx; kz = ky;                          % rescale to 2pi domain on y and z
18 [Kx, Ky, Kz] = meshgrid(kx,ky,kz);
19 Lap = (Kx.^2 + Ky.^2 + Kz.^2)/2;
20
21 % Compute initial conditions in spatial domain
22 C = (A(1)*(sin(X)).^2 + B(1)).*(A(2)*(sin(Y).^2) + B(2)).*...
```

```
                                                (A(3)*(sin(Z).^2) + B(3));

% need shifted K terms for visualization
Kx_s = ifftshift(Kx);Ky_s = ifftshift(Ky);Kz_s = ifftshift(Kz);
Lk = max(max(max(Kz))); % find outer bound of k values (same for all directions)

% initial conditions
psi0_t1 = cos(X).*cos(Y).*cos(Z);
psi0_f1 = reshape((fftn(psi0_t1)),n^3,1);
% create string for labeling the plot
ICstr1 = '$$\\psi_0 = \\cos{(x)}\\cos{(y)}\\cos{(z)}$$';

psi0_t2 = sin(X).*sin(Y).*sin(Z);
psi0_f2 = reshape((fftn(psi0_t2)),n^3,1);
% create string for labeling the plot
ICstr2 = '$$ \\psi_0 = \\sin{(x)}\\sin{(y)}\\sin{(z)}$$';

% try experimental initial conditions
psi0_t3 = cos(X.^2).*cos(Y.^2).*cos(Z.^2);
psi0_f3 = reshape((fftn(psi0_t3)),n^3,1);
% create string for labeling the plot
ICstr3 = '$$\\psi_0 = \\cos{(x^2)}\\cos{(y^2)}\\cos{(z^2)}$$';

```

```
46 psi0_t4 = sin(X.^2).*sin(Y.^2).*sin(Z.^2);
47 psi0_f4 = reshape((fftn(psi0_t4)),n^3,1);
48 % create string for labeling the plot
49 ICstr4 = '$$ \\\psi_0 = \\sin{(x^2)}\\sin{(y^2)}\\sin{(z^2)}$$';
50
51
52 % %%
53 opts = odeset('AbsTol',1e-6,'RelTol',1e-6);
54
55 [~, psi1_f] = ode45(@(t,y) rhsNLSE(t,y,Lap,C,n),tspan,psi0_f1,opts);
56 [~, psi2_f] = ode45(@(t,y) rhsNLSE(t,y,Lap,C,n),tspan,psi0_f2,opts);
57
58 [~, psi3_f] = ode45(@(t,y) rhsNLSE(t,y,Lap,C,n),tspan,psi0_f3,opts);
59 [~, psi4_f] = ode45(@(t,y) rhsNLSE(t,y,Lap,C,n),tspan,psi0_f4,opts);
60
61 m = length(tspan);
62 % [m ~] = size(psi1_f); % get the number of time steps for unpacking
63 % [m ~] = size(psi2_f);
64 %
65 for j = 1:m
66     psi1_t(:,:,:,j) = ifftn((reshape(psi1_f(j,:),[n,n,n])));
67     psi1_fshift(:,:,:,j) = fftshift(reshape(psi1_f(j,:),[n,n,n]));
68
```

```
    psi2_t(:,:,:,j) = ifftn((reshape(psi2_f(j,:),[n,n,n])));
    psi2_fshift(:,:,:,j) = fftshift(reshape(psi2_f(j,:),[n,n,n]));

    psi3_t(:,:,:,j) = ifftn((reshape(psi3_f(j,:),[n,n,n])));
    psi3_fshift(:,:,:,j) = fftshift(reshape(psi3_f(j,:),[n,n,n]));

    psi4_t(:,:,:,j) = ifftn((reshape(psi4_f(j,:),[n,n,n])));
    psi4_fshift(:,:,:,j) = fftshift(reshape(psi4_f(j,:),[n,n,n]));
end

toc % Elapsed time is 1.819344 seconds.



% %%
tic
close all
% set up title to dynamically name files based on what they are
title1 =  sprintf(...
    'BECx_spatial_fourier_%d_L%dpi_t%G_dt%1g_A%G_B%G_exptAsign_Init1_1.gif',...
                                                    n,L/pi,tmax,dt,A(1),B(1));
animate581_BEC4(psi1_t, psi1_fshift,X,Y,Z,Kx_s,Ky_s,Kz_s,n,...
                                                tspan,L,Lk,title1,A,B,dt,ICstr1);
toc
```

```
92  tic
93  % %%
94  close all
95  title1 =  sprintf(...
96      'BECx_spatial_fourier_%d_L%dpi_t%G_dt%1g_A%G_B%G_exptAsign_Init2_1.gif',...
97                                                      n,L/pi,tmax,dt,A(1),B(1));
98  animate581_BEC4(psi2_t, psi2_fshift,X,Y,Z,Kx_s,Ky_s,Kz_s,n,...
99                                                  tspan,L,Lk,title1,A,B,dt,ICstr2);
100 toc
101 % %%
102 tic
103 close all
104 title1 =  sprintf(...
105     'BECx_spatial_fourier_%d_L%dpi_t%G_dt%1g_A%G_B%G_exptAsign_Init3_1.gif',...
106                                                     n,L/pi,tmax,dt,A(1),B(1));
107 animate581_BEC4(psi3_t, psi3_fshift,X,Y,Z,Kx_s,Ky_s,Kz_s,n,...
108                                                 tspan,L,Lk,title1,A,B,dt,ICstr3);
109 toc
110 % %%
111 tic
112 close all
113 title1 =  sprintf(...
114     'BECx_spatial_fourier_%d_L%dpi_t%G_dt%1g_A%G_B%G_exptAsign_Init4_1.gif',...
```

```
                                                    n,L/pi,tmax,dt,A(1),B(1));
animate581_BEC4(psi4_t, psi4_fshift,X,Y,Z,Kx_s,Ky_s,Kz_s,n,...
                                                tspan,L,Lk,title1,A,B,dt,ICstr4);
toc

```

Here is my RHS function for the time-stepper. Just a warning, that I didn't comment this one so well.

```
function rhs = rhsNLSE(t,psi_fL,Lap,C,n)


% psi_fL: psi in  Fourier space as an n^3 x 1 vector
psi_f = reshape(psi_fL,[n,n,n]); % Reshape into n x n x n array


% Inverse FT for some operations in spatial domain
psi_t = ifftn((psi_f));      % Reshape into n^2 x 1 vector
psi_tL = reshape(psi_t,n^3,1);  % Reshape into n x n x n array




% psi n^3x1 vec in Fourier space
Lpsi = Lap.*psi_f; % doing el. ws. mult as tensors


% Compute the non-linear terms separately, then combine
NL1 = fftn(reshape((abs(psi_tL).^2).*psi_tL,[n,n,n]));


% trig term, I originally called this non-linear, but I don't think that it is
NL2 = fftn(C.*psi_t);


% solve for psi
```

```
21 rhs = reshape((-1i).*Lpsi + (-1i).*NL1 - (-1i).*NL2,n^3,1);
```

Here is my function for producing the animations.

```
function [M,h,hf] = animate581_BEC4(psi,psi_f,X,Y,Z,Kx,Ky,Kz,n,tspan,L,Lk,...
                                                                title,A,B,dt,ICstr)


%     v = VideoWriter(title);
%     v.FrameRate = 30;
%     open(v);

    % determine the max and min value of omega, which change with each initial
    % condition, for plotting purposes
    wmax = ceil(max(max(max(max(psi)))));
    wmin = floor(min(min(min(min(psi)))));

    % make vector to rotate coordinates
%     rot = linspace(30,50,length(tspan));
    rot = 45.*ones(length(tspan),1); % choose for no rotation of axes

    % initiallize figure
    h = figure;
    % customize the figure axes
    axis tight manual
    ax = gca;
    ax.NextPlot = 'replaceChildren';
```

```
23
24      % preallocate array to store movie frames
25      M(length(tspan)) = struct('cdata',[],'colormap',[]);
26
27      % prepare figure properties, including setting the background to white
28      set(gcf,'units','normalized','outerposition',[0.2 -0.7 0.6 1],'color','w')
29      h.Visible = 'off';
30
31
32
33  %     % set the colormap
34  %     colormap(turbo);%lines);%jet)%hot)
35      subplot(2,2,1)
36      p1 = patch(isosurface(X,Y,Z,real(psi(:,:,:,1))),'facecolor','g',...
37                                                          'edgecolor','none'); %
38
39      colormap(turbo)
40      daspect([1 1 1]);
41      view(3);
42      axis tight;
43      camlight;
44      lighting gouraud;
45      camproj perspective;
```

```
46      set(gca,'xlim',[-L/2 L/2],'ylim',[-L/2 L/2],'zlim',[-L/2 L/2])
47      txt2a = sprintf('$Re(\\psi)$',n);
48      text(-L/2,L/2,L/2,txt2a,'fontsize',20,'fontweight','bold','interpreter',...
49                                              'latex','HorizontalAlignment','center');
50
51      subplot(2,2,2)
52      p2 = patch(isosurface(Kx,Ky,Kz,real(psi_f(:,:,:,1))),'facecolor','c',...
53                                                          'edgecolor','none'); %
54
55      colormap(turbo)
56      daspect([1 1 1]);
57      view(3);
58      axis tight;
59      camlight;
60      lighting gouraud;
61      camproj perspective;
62      set(gca,'xlim',[-Lk Lk],'ylim',[-Lk Lk],'zlim',[-Lk Lk])
63      txt3a = sprintf('$Re(\\hat{\\psi})$',n);
64      text(-Lk,Lk,Lk,txt3a,'fontsize',20,'fontweight','bold','interpreter',...
65                                            'latex','HorizontalAlignment','center');
66
67      subplot(2,2,3)
68      p3 = patch(isosurface(X,Y,Z,imag(psi(:,:,:,1))),'facecolor','r',...
```

```
                                                        'edgecolor','none'); %

    colormap(turbo)
    daspect([1 1 1]);
    view(3);
    axis tight;
    camlight;
    lighting gouraud;
    camproj perspective;
    set(gca,'xlim',[-L/2 L/2],'ylim',[-L/2 L/2],'zlim',[-L/2 L/2])
    txt4a = sprintf('$Im(\\psi)$',n);
    text(-L/2,L/2,L/2,txt4a,'fontsize',20,'fontweight','bold','interpreter',...
                                        'latex','HorizontalAlignment','center');

    subplot(2,2,4)
    p4 = patch(isosurface(Kx,Ky,Kz,imag(psi_f(:,:,:,1))),'facecolor','m',...
                                                    'edgecolor','none'); %

    colormap(turbo)
    daspect([1 1 1]);
    view(3);
    axis tight;
    camlight;
```

```
92      lighting gouraud;
93      camproj perspective;
94      set(gca,'xlim',[-Lk Lk],'ylim',[-Lk Lk],'zlim',[-Lk Lk])
95      txt5a = sprintf('$Im(\\hat{\\psi})$',n);
96      text(-Lk,Lk,Lk,txt5a,'fontsize',20,'fontweight','bold','interpreter',...
97                                              'latex','HorizontalAlignment','center');
98
99      r = 1; % relic variable from when I was rotating the axis
100     for j = 1:length(tspan)
101         %                                    display the current time
102         txt0 = sprintf('$t = \\hspace{3pt}$ %G',tspan(j));
103         ah = annotation('textbox',[0.2 0.37 0.6 0.2],'String',txt0,...
104         'FitBoxToText','on','interpreter','latex','HorizontalAlignment','center');
105         ah.BackgroundColor = [1 1 1];      % not purple background
106         %% If you want to run this code, remove ellipses breaking up string
107         txt1 = sprintf('Fourier modes: %d \n $(t_i,t_f,\\Delta t)= ...
108         ($%G,%G,%G$)$ \n $A = ($%G,%G,%G$)$ \n $B = ($%G,%G,%G$)$',...
109         n,tspan(1),tspan(end),dt,A(1),A(2),A(3),B(1),B(2),B(3));
110         %%
111         text(-0.35,1.1,txt1,'fontsize',20,'interpreter','latex','units',...
112                                     'normalized','HorizontalAlignment','center');
113
114         %% If you want to run this code, remove ellipses breaking up string
```

```
115         txtx = sprintf('$$i\\psi_t = -\\frac{1}{2}\\nabla^2\\psi + ...
116         |\\psi|^2\\psi -[A_1 \\sin^2(x) + B_1][A_2 \\sin^2(y) + B_2]...
117                                         [A_3 \\sin^2(z) + B_3]\\psi$$ \n');
118         %%
119         text(-0.35,2.6,txtx,'fontsize',20,'interpreter','latex','units',...
120                                     'normalized','HorizontalAlignment','center');
121         text(-0.35,2.5,ICstr,'fontsize',20,'interpreter','latex','units',...
122                                     'normalized','HorizontalAlignment','center');
123
124         view(rot(j),25);
125         subplot(2,2,1)
126         [faces_tr, vertices_tr] = isosurface(X,Y,Z,real(psi(:,:,:,j)));
127         set(p1, 'Faces',faces_tr,'Vertices',vertices_tr);
128         set(gca,'xlim',[-L/2 L/2],'ylim',[-L/2 L/2],'zlim',[-L/2 L/2])
129
130         % set axis labels and fonts
131         set(gca,'fontsize',16)
132         xlabel('$x$','interpreter','latex','fontsize', 26);
133         ylabel('$y$','interpreter','latex','fontsize',26);
134         zlabel('$z$ ','rotation',0,'interpreter','latex','fontsize',26);
135
136         view(rot(j),25);
137         subplot(2,2,2)
```

```
138         [faces_fr, vertices_fr] = isosurface(Kx,Ky,Kz,real(psi_f(:,:,:,j)));
139
140         set(p2, 'Faces',faces_fr,'Vertices',vertices_fr);
141         set(gca,'xlim',[-Lk Lk],'ylim',[-Lk Lk],'zlim',[-Lk Lk])
142
143         % set axis labels and fonts
144         set(gca,'fontsize',16)
145         xlabel('$K_x$','interpreter','latex','fontsize', 26);
146         ylabel('$K_y$','interpreter','latex','fontsize',26);
147         zlabel('$K_z$ ','rotation',0,'interpreter','latex','fontsize',26);
148
149         view(rot(j),25);
150         subplot(2,2,3)
151         [faces_ti, vertices_ti] = isosurface(X,Y,Z,imag(psi(:,:,:,j)));
152         set(p3, 'Faces',faces_ti,'Vertices',vertices_ti);
153          set(gca,'xlim',[-L/2 L/2],'ylim',[-L/2 L/2],'zlim',[-L/2 L/2])
154
155
156         % set axis labels and fonts
157         set(gca,'fontsize',16)
158         xlabel('$x$','interpreter','latex','fontsize', 26);
159         ylabel('$y$','interpreter','latex','fontsize',26);
160         zlabel('$z$ ','rotation',0,'interpreter','latex','fontsize',26);
```

```
161
162         view(rot(j),25);
163         subplot(2,2,4)
164         [faces_fi, vertices_fi] = isosurface(Kx,Ky,Kz,imag(psi_f(:,:,:,j)));
165
166         set(p4, 'Faces',faces_fi,'Vertices',vertices_fi);
167         set(gca,'xlim',[-Lk Lk],'ylim',[-Lk Lk],'zlim',[-Lk Lk])
168
169         % set axis labels and fonts
170         set(gca,'fontsize',16)
171         xlabel('$K_x$','interpreter','latex','fontsize', 26);
172         ylabel('$K_y$','interpreter','latex','fontsize',26);
173         zlabel('$K_z$ ','rotation',0,'interpreter','latex','fontsize',26);
174
175         drawnow
176         pause(0.1)
177         M(j) = getframe(h);
178
179         frame = getframe(gcf);
180         img =  frame2im(frame);
181         [img,cmap] = rgb2ind(img,64);
182
183         % Set the filename 1-4 based on which conditions are being used.
```

```
        if j == 1
            imwrite(img,cmap, title,'gif','LoopCount',Inf,'DelayTime',0.3);
        else
            imwrite(img,cmap, title,'gif','WriteMode','append','DelayTime',0.25);
        end
        pause(0.25);

%         writeVideo(v,M(j));

    end

    % this is needed to properly display the plots within the figure
    [ht w d] = size(M(1).cdata);
    hf = figure;
    set(hf,'position',[1 1 w ht]);
    axis off

%     close(v);

    % play video
    movie(hf,M);
%     movie(M);
end
```