M3N10-Computational PDES-Project 1

Matthew Cowley, CID 01059624

“The contexts of this report and associated codes are my own work unless otherwise stated” 08/01/2017

In the following project we are interested in the following problem:

$$\omega = \begin{cases} 1 & \text{for} \quad r \leq 1 \\ 0 & \text{for} \quad 2 > r > 1 \end{cases} \tag{1}$$

$$\phi(r,\theta,t) = a(r,t)cos(\theta) - b(r,t)sin(\theta) \tag{2}$$

where a(r,t) and b(r,t) satisfy the following linked diffusion equations given a constant η:

$$a_t - \omega b = \eta(a_{rr} + \frac{a_r}{r} - \frac{a}{r^2}) \tag{3}$$

$$b_t + \omega a = \eta(b_{rr} + \frac{b_r}{r} - \frac{b}{r^2}) \tag{4}$$

with given initial conditions a(r,0)=r and a(2,t)=2, a(0,t)=b(0,t)=b(2,t)=0 for all time t>0.

1. I have written some MATLAB code which solves the above coupled system using centered finite differences in terms of r and t. It uses uniform grid over r and t, with h and k being their step-lengths respectively.

2. We can find a theoretical limitation of k for a given value of h provided it is sufficiently small (we assume $h \leq 0.1$) to ensure numerical stability in the scheme. We shall find the limit by using the Maximum Principle Method shown in lectures. (shown in lecture 4, reference link provided below)

https://bb.imperial.ac.uk/bbcswebdav/pid-1259192-dt-content-rid-4137185_1/courses/COURSE-M345N10-17_18/lec4Fourier.pdf.

Let us first combine 3 and 4 into one equation.

Consider $(3) + i(4)$ with $y = a + ib$ we get the following system representing the above:

$$y_t - \omega y i = \eta(y_{rr} + \frac{y_r}{r} - \frac{y}{r^2}) \tag{5}$$

It has now become a 1D problem which is easier to analyse. Consider a PDE of the form given below:

$$Au_{rr} + Bu_{r\theta} + Cu_{\theta\theta} + Du_r + Eu_\theta + Fu = 0$$

A PDE is considered parabolic if $B^2 - 4AC = 0$ which is true for y in this case, if you ignore the fact that A,B,C might not be constants. It also bears a lot of similarities to the heat equation which is also parabolic. If you would like to know more about why this is the case, please look into the following reference http://how.gi.alaska.edu/ao/sim/chapters/chap3.pdf.

We can use Maximum Principle Method since our system is 1D and parabolic. In the context of the method our Φ is defined as below.

$$\Phi = y_t = \omega y i + \eta(y_{rr} + \frac{y_r}{r} - \frac{y}{r^2}) \tag{6}$$

For stability to occur we require that the coefficients in square brackets of (4.7) of lecture notes are positive. This is so that the maximum error between our solution and the real solution is of order $O(k, h^2) * t^j$ at a time t^j. Thus the error can be made as small as we like by choosing small enough steplengths, k and h. Thus we require that:

$$\frac{\delta\Phi}{\delta y_{rr}} \pm \frac{1}{2}\frac{\delta\Phi}{\delta y_r} \geq 0 \tag{7}$$

and

$$1 + k\frac{\delta\Phi}{\delta y} - 2\frac{k}{h^2}\frac{\delta\Phi}{\delta y_{rr}} \geq 0 \tag{8}$$

We can reduce these because $\frac{\delta\Phi}{\delta y_{rr}} = \eta$ and $\frac{\delta\Phi}{\delta y_r} = \frac{1}{r}$ along with $\frac{\delta\Phi}{\delta y} = \omega i - \frac{\eta}{r^2}$.

Now (7) just reduces to $r - \frac{1}{2}h \geq 0$ which is always true since we know the solution for r=0 (and r=2), so the smallest r can be is h. So $\frac{1}{2}h \geq 0$.

For (8) to be useful we need to find a bound for k. So we need to find the min($\frac{\delta \Phi}{\delta u}$) , to do this we use the triangle inequality.

$$|i\omega - \frac{\eta}{r^2}| \geq |i\omega| - |\frac{\eta}{r^2}|$$

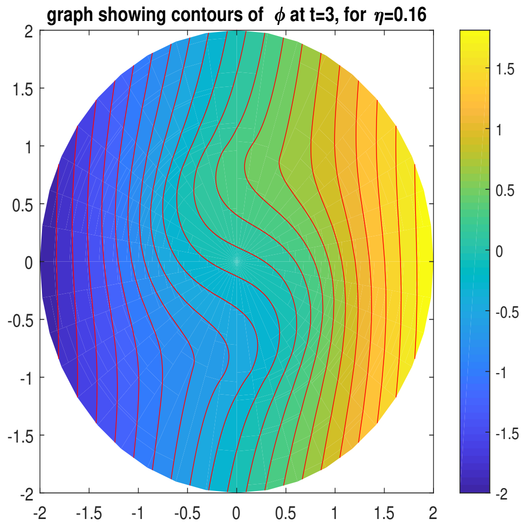
The minimum of the above given $h \leq 0.1$ occurs when r=h. Thus (8) reduces to:

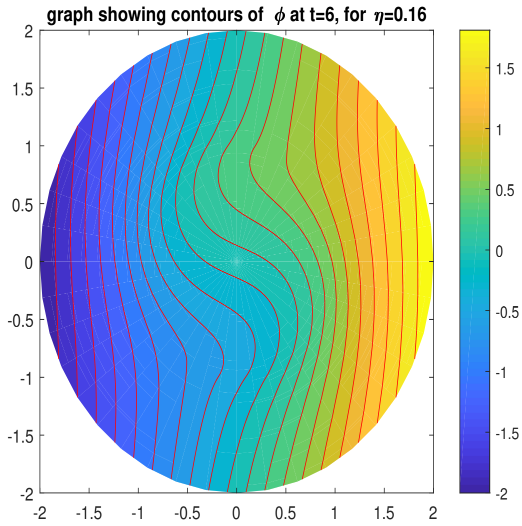
$$k \leq \frac{h^2}{3\eta - h^2} \tag{9}$$

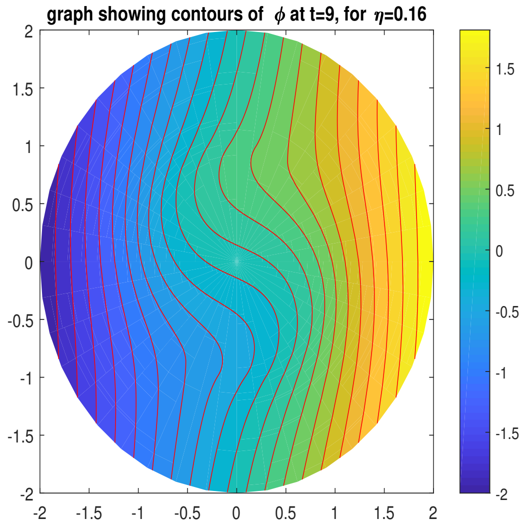
It appears that in practice k can be bigger then the above and still produce good results ($k = \frac{1}{2.5\eta}$ still gives good results). However the above bound does ensure in theory good results.

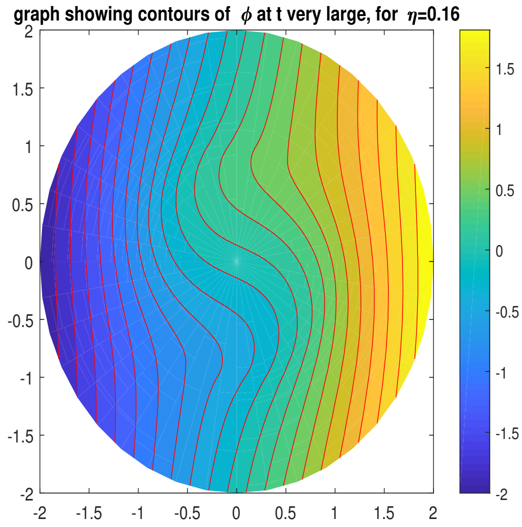
3. The MATLAB code in the appendix was used to create the following graphs, with the value of η being varied and k equal to bound (9).

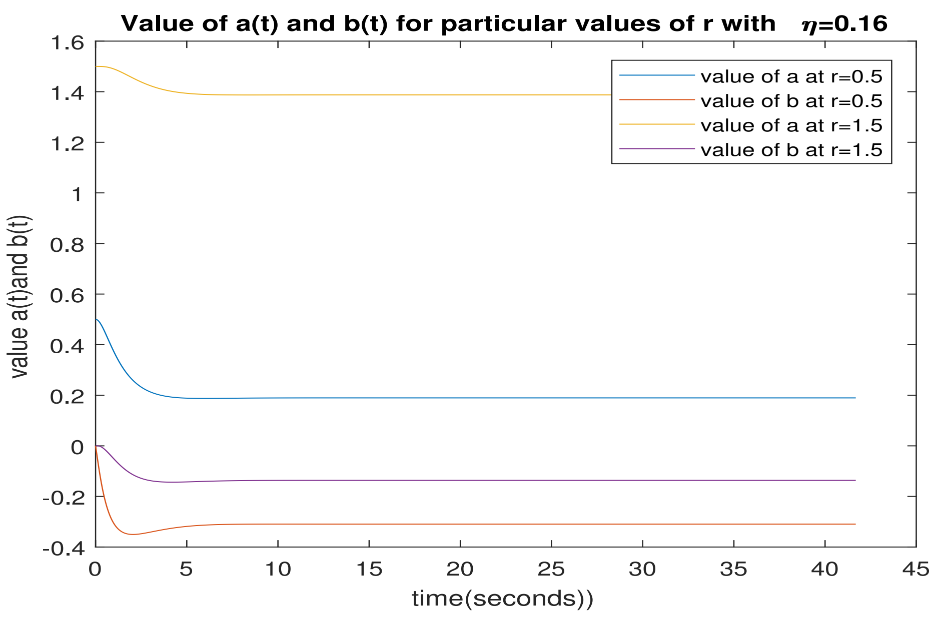
1 Graphs for $\eta = 0.16$



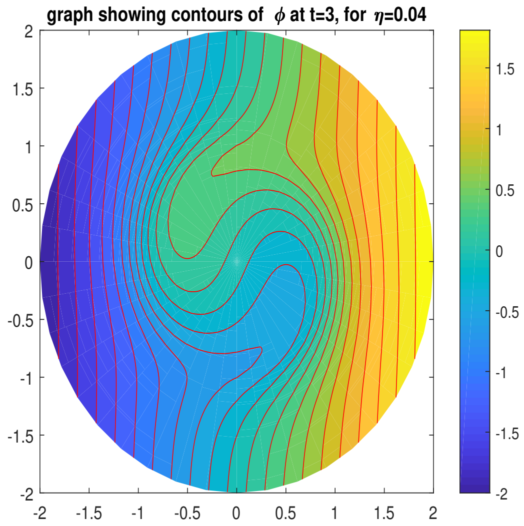


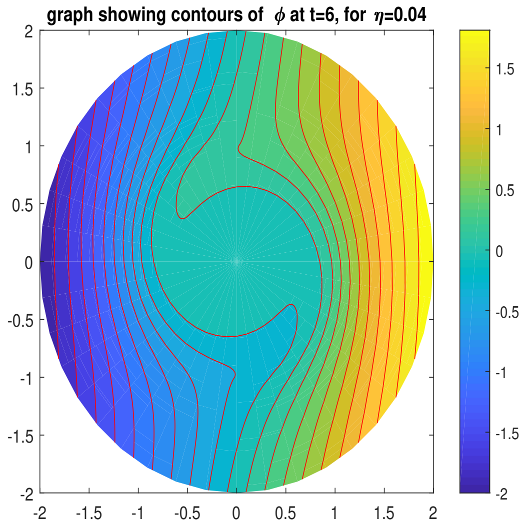


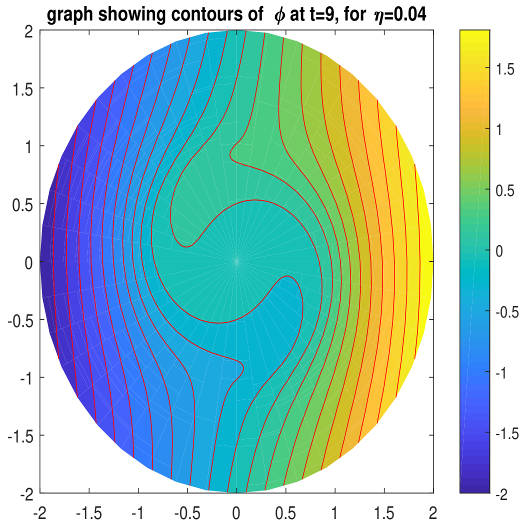


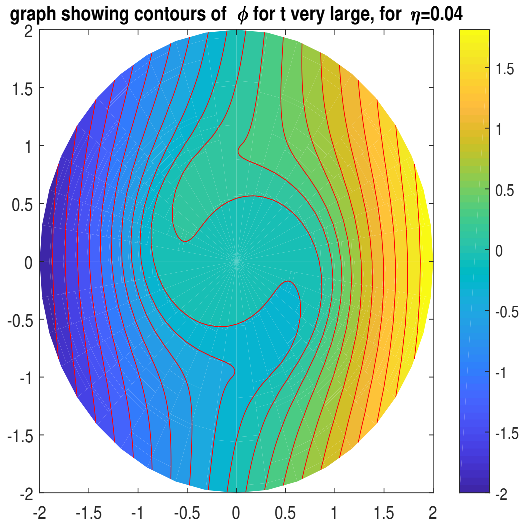


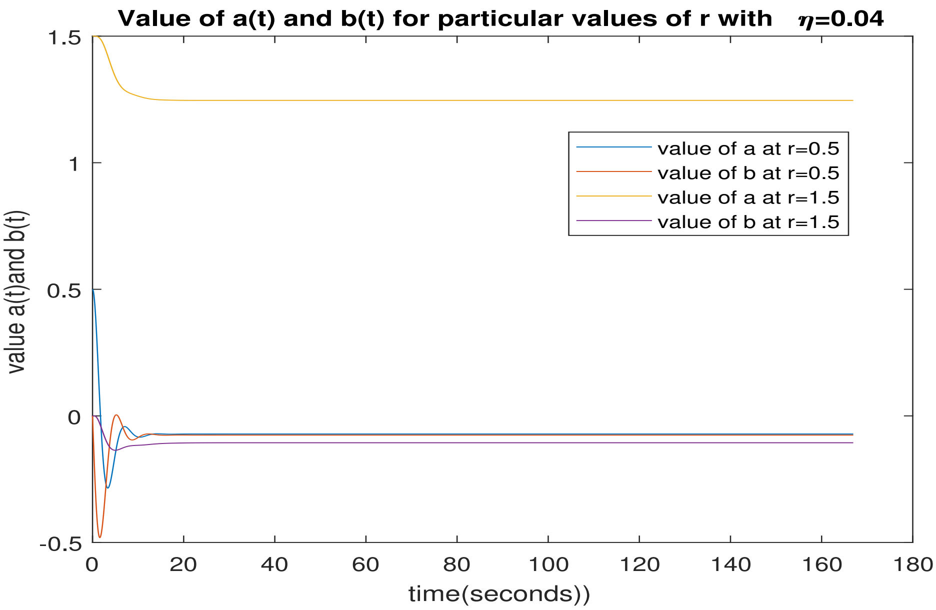
2 Graphs for $\eta = 0.04$



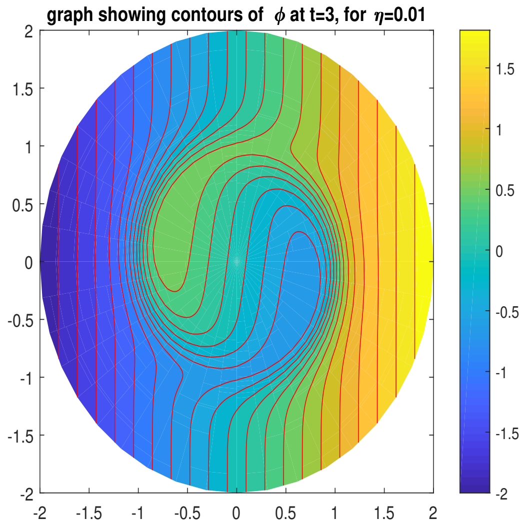


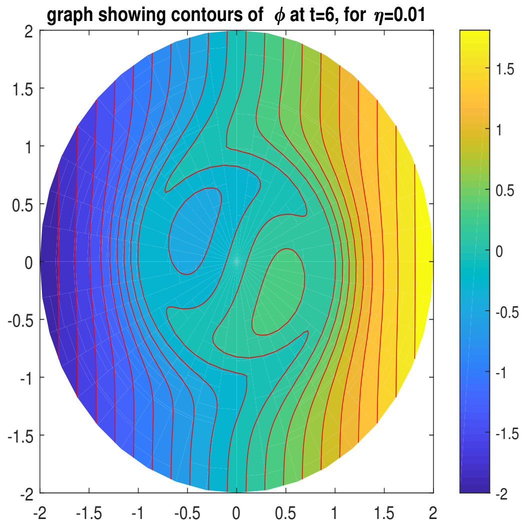


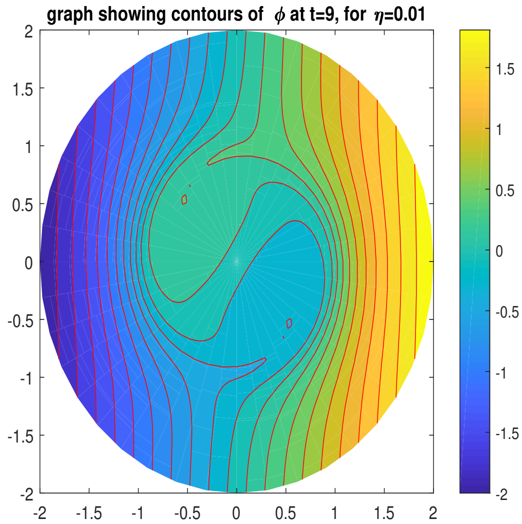


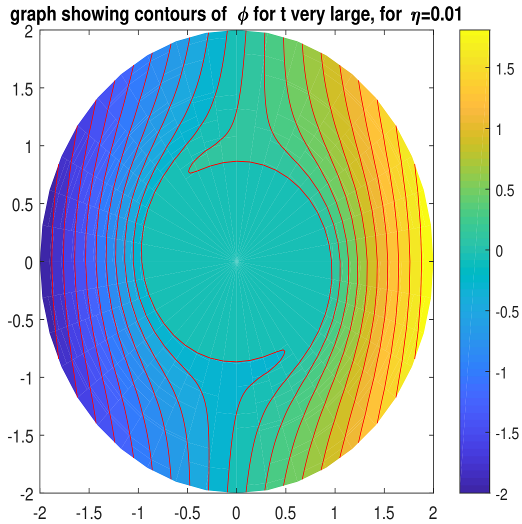


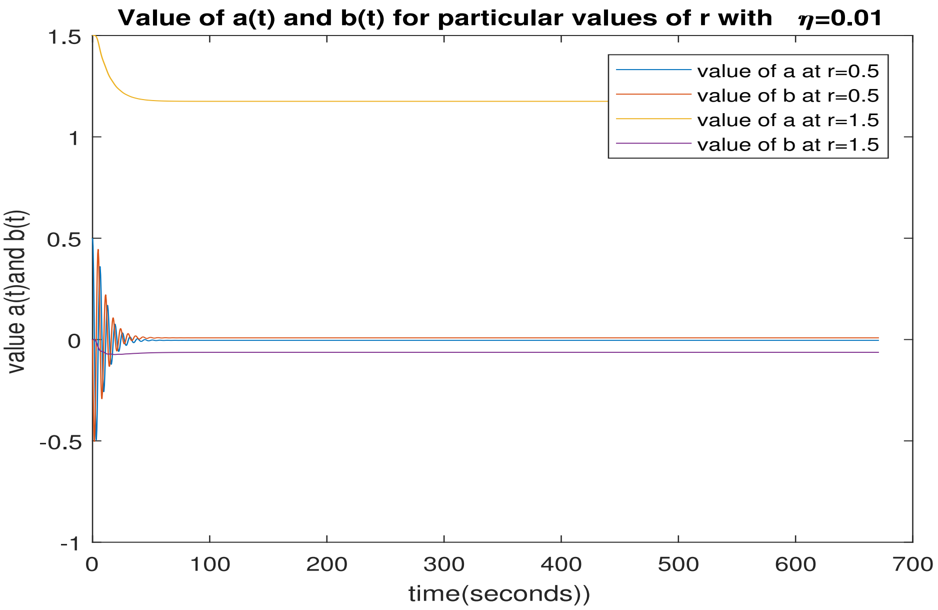
3 Graphs for $\eta = 0.01$







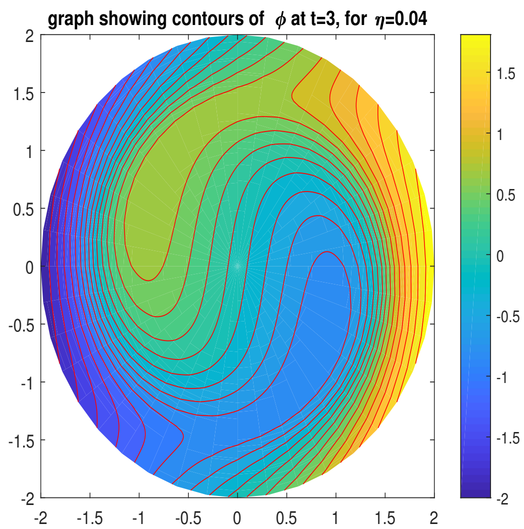


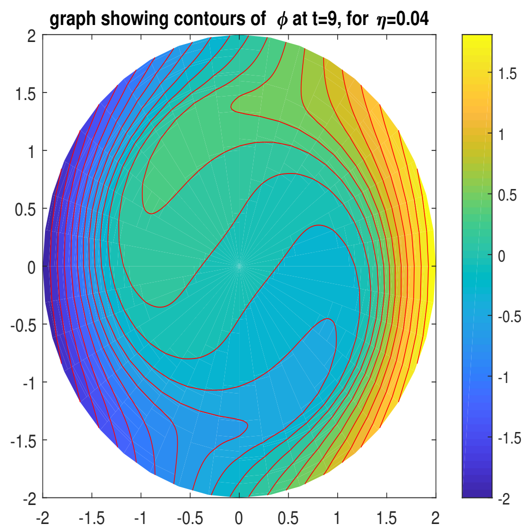


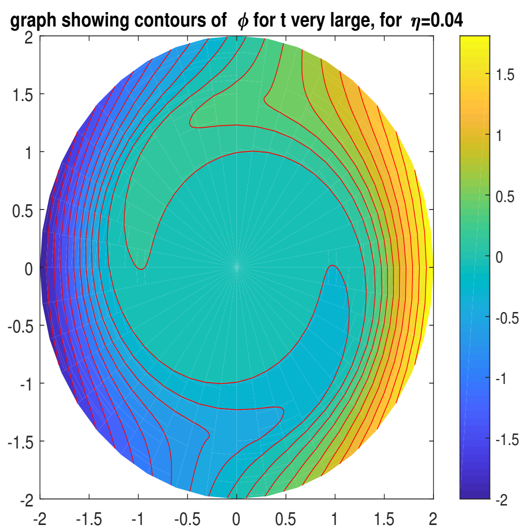
4. You know when the system is stable by observing that the contours stop changing. This is reinforced for particular values of r by looking at the a(t) and b(t) graph and observing that their values stop changing. When this happens for all r the system is stable. Oscillations do not seem to occur for $r > 1$ which is interesting.

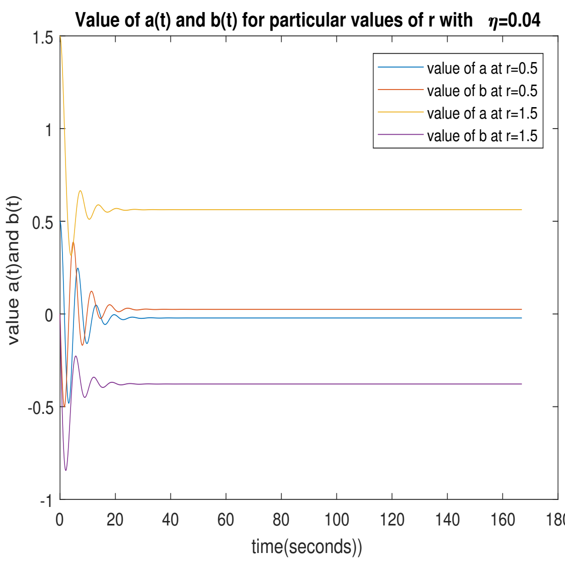
For all values of η the magnetic field lines become stable. They do this by rotating round the center and diffusing outwards. However for lower values of η there seem to be more of an effect on the field lines. By that I mean it takes longer for the system to become stable and the contours differing more from the initial conditions for all time. It appears that the time taken for stability to occur scales with η, this is because all graphs become stable around 17000 iterations with k chosen as it (9) which scales with η.

In this situation I believe η to be the viscosity of the conducting fluid. This is because it would move more for lower viscosities thus disturbing the magnetic field lines more and therefore it would take the field lines longer to recover to a stable state. Other evidence which supports this, is that when we alter $\omega = 1$ for $r \leq 1.5$, we see it disturbs the field lines in a greater area. Which represents a greater portion of fluid moving, thus effecting field lines more. Below are the graphs represented by this alternative function. As we can see it also takes longer to reacher a stable state. An interesting remark is how a(t) and b(t) values have changed now having more oscillations, before they become stable.









4 Appendix- MATLAB code

Code used to create raw data (formated as a function).

```
function [a,b]=rawdatafindiff(N,k,iter,eta)
r=0:(2/N):2;
a=zeros(iter+1,N+1);
b=zeros(iter+1,N+1);
a(1,1:N+1)=r;
a(2:iter+1,N+1)=2*ones(iter,1);


%s inictaes which entry of r for which r>1, producing q at which it happens
s=mod(N,2);
if s==0
    q=(N/2)+1;
else
    q=(N+1)/2;
end

%set up coefficents matrix
c=zeros(3,N+1);
for i=2:N
c(1:3,i)=[k*eta*((N/2)^2-(N/(4*r(1,i))));(1-k*eta*(0.5*(N^2)+(1/(r(1,i)^2)))); k*eta*
;
end

for j=1:iter
    for i=2:q
       a(j+1,i)=(a(j,i-1:i+1)*c(1:3,i))+k*b(j,i);
       b(j+1,i)=(b(j,i-1:i+1)*c(1:3,i))-k*a(j,i);
    end
    for i=q+1:N
        a(j+1,i)=(a(j,i-1:i+1)*c(1:3,i));
        b(j+1,i)=(b(j,i-1:i+1)*c(1:3,i));
    end
end

end
```

Code used to construct graphs from raw data

```
%inputs needed to create raw data, k being desired time step, N+1 being
%number of points used for r, iter being the number of time steps desired.
%eta relevant to intial equation


N=100;
iter=100000;
eta=0.01;
k=0.0002/(3*eta-0.0002);

[a,b]=rawdatafindiff(N,k,iter,eta);

%i is the time step at which we want the graph to be produced
```

```
[r,theta] = meshgrid(0:2/N:2,-pi:pi/20:pi);
x = r.*cos(theta);
y = r.*sin(theta);


i=3/k-mod(3/k,1)+1;
z=a(i,1:N+1).*cos(theta)-b(i,1:N+1).*sin(theta);
figure(1)
contourf(x,y,z,20,'r');
colorbar
title('graph showing contours of \phi at t=3, for \eta=0.01')
axis image

t=0:k:k*(iter);
t=t';
figure(5)
plot(t,a(1:iter+1,26),t,b(1:iter+1,26),t,a(1:iter+1,76),t,b(1:iter+1,76))
title('Value of a(t) and b(t) for particular values of r with \eta=0.01')
xlabel('time(seconds))')
ylabel('value a(t)and b(t)')
legend('value of a at r=0.5','value of b at r=0.5','value of a at r=1.5','value of b a
```