Thomas Hauch, Taylor Sipple

Math 53: Chaos!

Professor Barnett

December 2, 2011

Chaos in the Magnetic Pendulum

**I. Introduction**

The Random Oscillating Magnetic Pendulum (ROMP) is a popular desktop toy. It consists of a pendulum, with a small magnet attached to the end, which can swing freely over a metal base. On the base there are any number of strong, disc-shaped magnets, which act on the pendulum with either an attractive or repulsive force. When the pendulum is displaced and released, it tends to swing in an erratic, unpredictable pattern. Due to friction, the pendulum eventually comes to rest over one of the magnets (assuming they are attracting) or in some other region of the plane (if they are repulsive). If we attempt to release the pendulum from the same starting position, we see that the pendulum often ends up at a different position after traversing an entirely different orbit.

The dynamics of the magnetic pendulum appear to be chaotic. In particular, they exhibit strong sensitive dependence on initial conditions. And although the system gradually loses energy, the path of the pendulum does not seem to undergo any periodic motion until it is nearly at rest (when it tends to oscillate over a very short distance).

In this investigation, we devise a model in order to test some of these assumptions. Using several methods, we calculate Lyapunov exponents for an array of initial conditions in an undamped system. After allowing for damping, we examine how the final resting position of the pendulum is dependent on its initial conditions.

**II. Background and Methodology**

There have been a number of past experiments on the magnetic pendulum.1,2,4,5,6 In all the examples we came across, the authors have relied on several simplifications in their models.

In previous experiments, the magnetic pendulum has been represented using two parallel planes separated by a small distance *h*. The motion of the pendulum is restricted to one plane, while the fixed magnets are arranged on the other. As a result, the location of the pendulum is defined by its ­*x­-* and *y-*coordinates, and the forces exerted on the pendulum can be written in terms of their *x*-and *y-*components.

Furthermore, the pendulum is assumed to be very long relative to the height of the pendulum above the base, which means the force of gravity can be modeled as a linear restoring force using the small-angle approximation for the sine function.

In other words,  where *g* is a gravitational constant and  is the position vector in the *x-y* plane. For the purpose of this investigation, we scale the gravitational force by setting . Previous experiments have also included a damping force proportional to the velocity of the pendulum  where  is a coefficient of damping which we vary.

Lastly, these authors have assumed that the magnets are point charges interacting via an electrostatic force. By way of scaling, the authors reduce the electrostatic forces to a simple inverse-square law: .

The end result in each of these previous examples is a 2nd-order ordinary differential equation of the form:

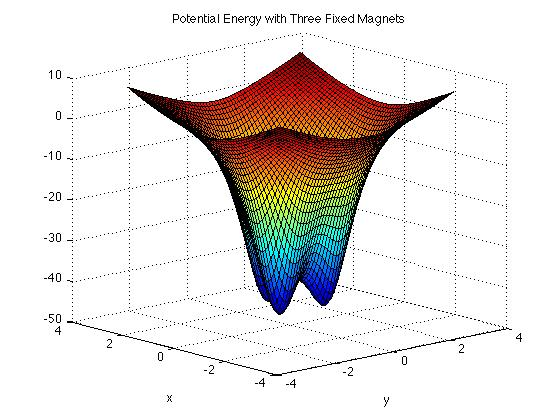


Equations of motion in *x-y­* plane. Previous models have treated gravity as a linear restoring force (here, with constant *G*). The magnetic interaction is an electrostatic force, which has been scaled with unit mass, and magnetic constant. Damping is proportional to the velocity of the pendulum.

Although we maintain several aspects of this model, including their approximations of the forces due to gravity and damping, we improve upon their estimation of the magnetic force. Magnets are not point charges, but rather dipoles, and the forces that govern their interaction evolve differently throughout space. According to Furlani3, the potential energy of the magnetic dipole-dipole interaction is given by:



Potential energy of magnetic dipole-dipole interaction, where **e12** is the unit vector between the centers of the magnetic dipoles. The vectors **m1** and **m2** denote the magnetic moments.

Since , it follows that the potential energy of gravity is defined by



Using the two equations above, we plot the potential field with three attracting magnets of equal strength. The magnets are arranged in an equilateral triangle centered at the origin. Not surprisingly, there are three potential wells centered above each fixed magnet. The plot of potential with three repulsive magnets is similar, although in that case the effects of gravity and magnetism are opposed. In either case, however, the motion of the pendulum is bounded by some square in the *x-y­ plane.*

**Figure 1**: Potential Plot with Three Attracting Magnets

Using the relationship above, it is possible to derive the magnetic dipole-dipole interaction force from the potential. Documenting the physics of magnetic dipoles is beyond the scope of this investigation, and so we rely on the result obtained by Furlani.3 The force exerted by one magnetic dipole on another is given by:



Force exerted by a fixed magnetic (with moment **m**1) on the pendulum magnet (with moment **m**2). We can eliminate components of the force along the z-axis, since our model restricts motion to the x-y plane.

Since we are restricting motion to the *x-y* plane, we can simplify our model by ignoring the component of the force in the *z-*axis. In the equation above, we eliminate the terms and , since the *x­-* and *y*-components of the magnetic moment vectors are zero. Furthermore, we note that the component of the vector in the direction of is simply *h*, the distance between the base and the pendulum at rest. Thus, , and similarly, . With these considerations, our model for the magnetic force reduces to:



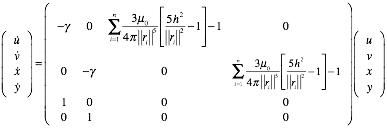


Taking the sum of our component forces, we arrive at the following system of 2nd-order ordinary differential equations:



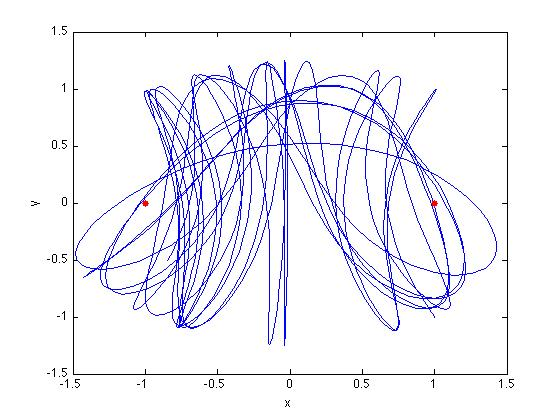
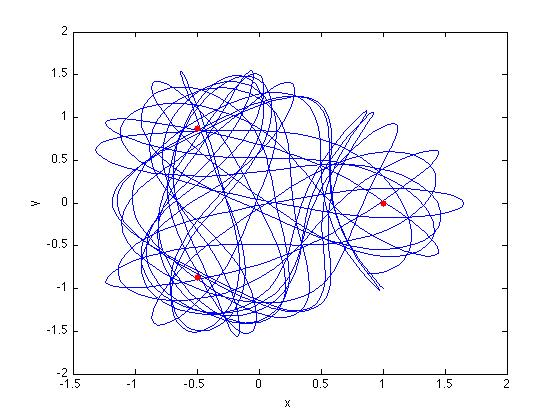
The total force is equal to the sum of the component forces. The index of the summation denotes the particular fixed magnet that we are considering. The magnetic force here is defined in the equation above.

In order to numerically solve an initial value problem, we derive a coupled system of first-order differential equations in terms of component forces, which is given in vector form by:



Coupled system of first-order differential equations, governing the motion of the pendulum

Given a set of initial conditions, we can solve the system numerically in MATLAB using ode45. To test our model, we plot the orbit of an undamped pendulum under the influence of either two or three fixed magnets. In each case, the pendulum begins at that point (1, -1) with zero initial velocity. We run ode45 over a time span [0,100] and record the results below.



**Figures 2, 3**: Undamped Motion With Three Attracting Magnets; With Two Attracting Magnets

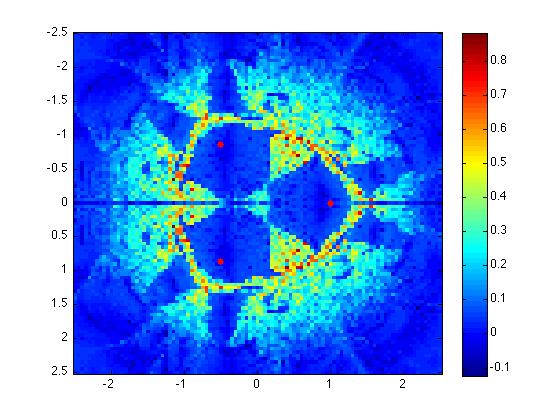
The motion in each case, but especially with three attracting magnets, *appears* chaotic. In order to quantify and corroborate this assumption, we run a series of tests measuring the Lyapunov exponents of the model.

**III. Undamped Motion**

In the case of undamped motion, we attempted to measure Lyapunov exponents for the flow. Two methods were used, and afterward we compared the results.

*A. Crude Method – Orbits initially displaced by distance ε*

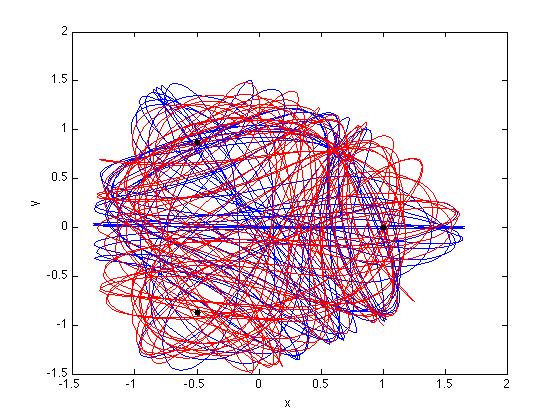
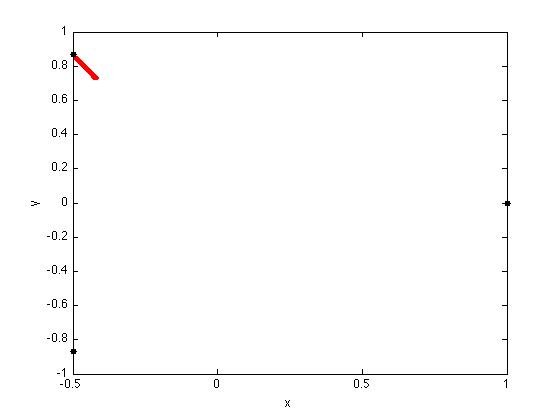
With this method, we measure the separation at each time-step of ode45. Because the orbit of the pendulum is bounded, the separation of divergent chaotic orbits stops growing upon reaching O(1). We use this property to measure the growth of the logarithm of this separation up to O(1), which gives us a crude estimate for h. We usedthis method to map the (crude) Lyapunov exponent at each point on a grid in R2.



**Figure 4**: Lyapunov Exponent “Heat Map” in R2

To verify that these numbers matched up with observed behavior, we found points with positive and negative Lyapunov exponents, respectively, and plotted the progress of two concurrent orbits. The point (-0.5, -0.85) corresponded to a Lyapunov exponent of

h = -0.0227. The orbits resulting from this initial condition, which do not diverge, are shown in Figure 5. The point (1, -1) resulted in an exponent h = 0.3349 and the divergent chaotic orbits in Figure 6.



**Figures 5, 6**: Non-Divergent Orbits for (-0.5, -0.85); Divergent Orbits for (1, -1)

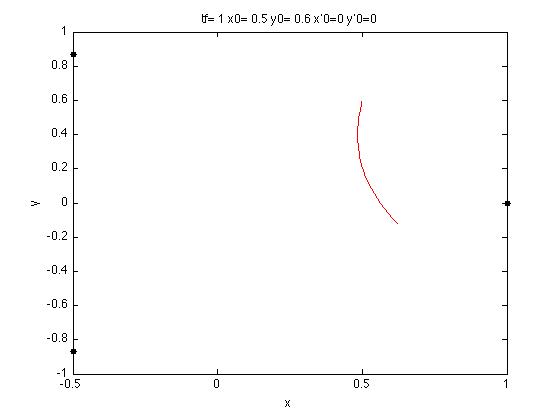
*B. Re-Orthogonalizing Version with Averaging Over Long Trajectory*

This method was adapted from Professor Alex Barnett’s script *lyapflow.m* and the function *lorenz\_time1map.m*. It computes an estimate of the Jacobian along each orthogonal vector using the following approximation for a given value of ε:



Estimates partial derivative of *f* with respect to ***x­****i* at the point ***z****i*.

For this method, we use two loops, the second nested within the first. The inner loop runs a series of time-1 maps and updates the Jacobian after each iteration and then re-orthogonalizes it. At the end of the inner loop the diagonal entries are extracted and converted into Lyapunov exponents. The outer loops keeps a weighted average of each of these runs, and at the end we obtain the Lyapunov exponents for this flow. This method has been shown to be within 1% accuracy for the Hénon map.

We determined that the time-1 map was appropriate by examining a few orbits. Figure 7 shows that over the time span [0,1], the pendulum swings approximately one arc, which makes our current timescale appropriate for the time-1 map.

While we did not have the time nor the computational power to compute the Lyapunov exponents over the grid, we were able to test it for the two values obtained from method A. During these measurements we made runs of 25 time-1 maps over 5 averaging iterations. For the initial point (1, -1), we obtained a largest Lyapunov exponent of h = 0.4605. For the initial point (-0.5, 0.85) we obtain h = -0.0148. These values vary significantly from the ones recorded by Method A. Nevertheless, the sign of each exponent is the same for each method, which suggests that Method A functions well as a rough, qualitative approximation.

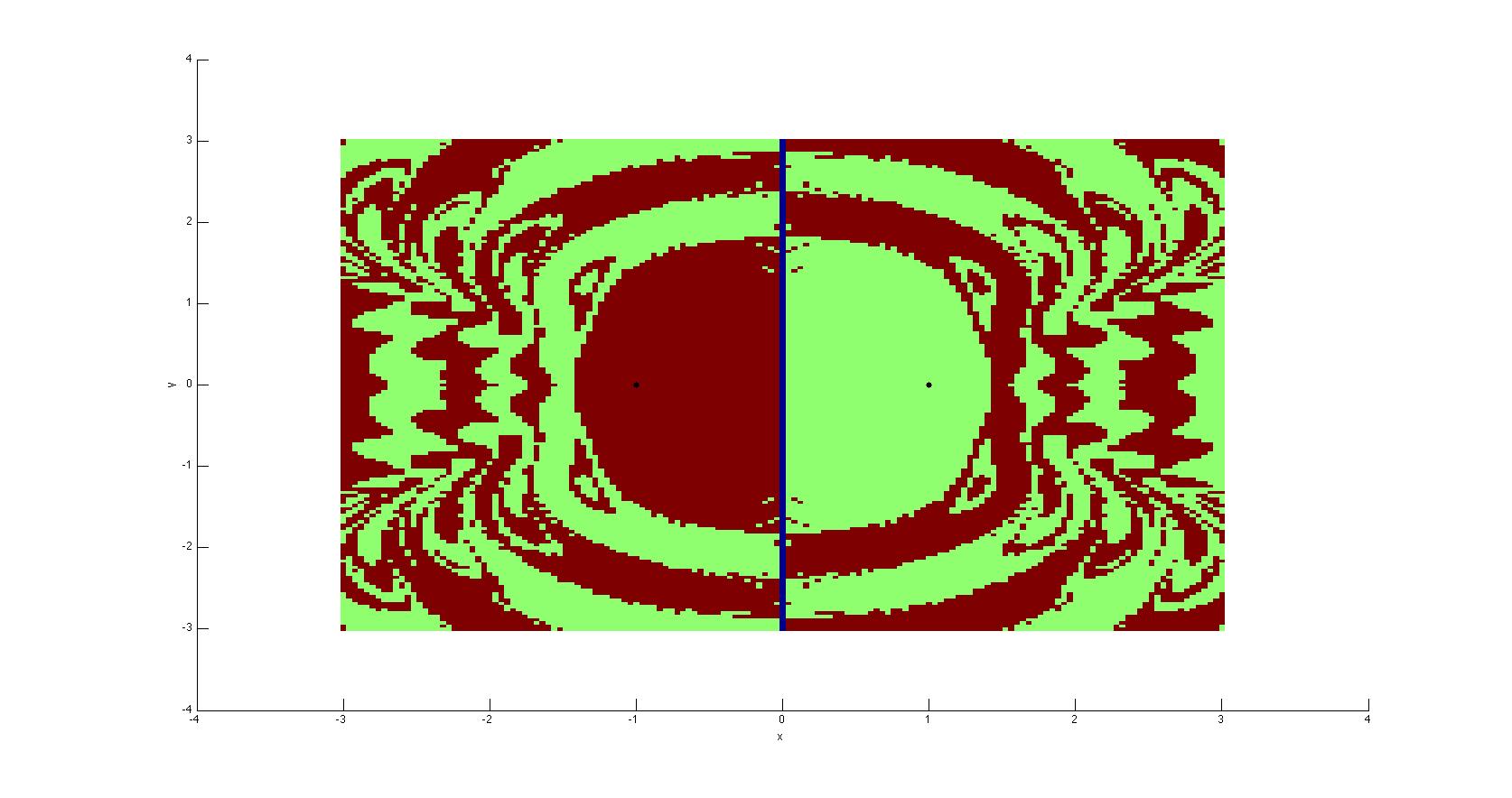
**Figure 7**: Time-1 Map

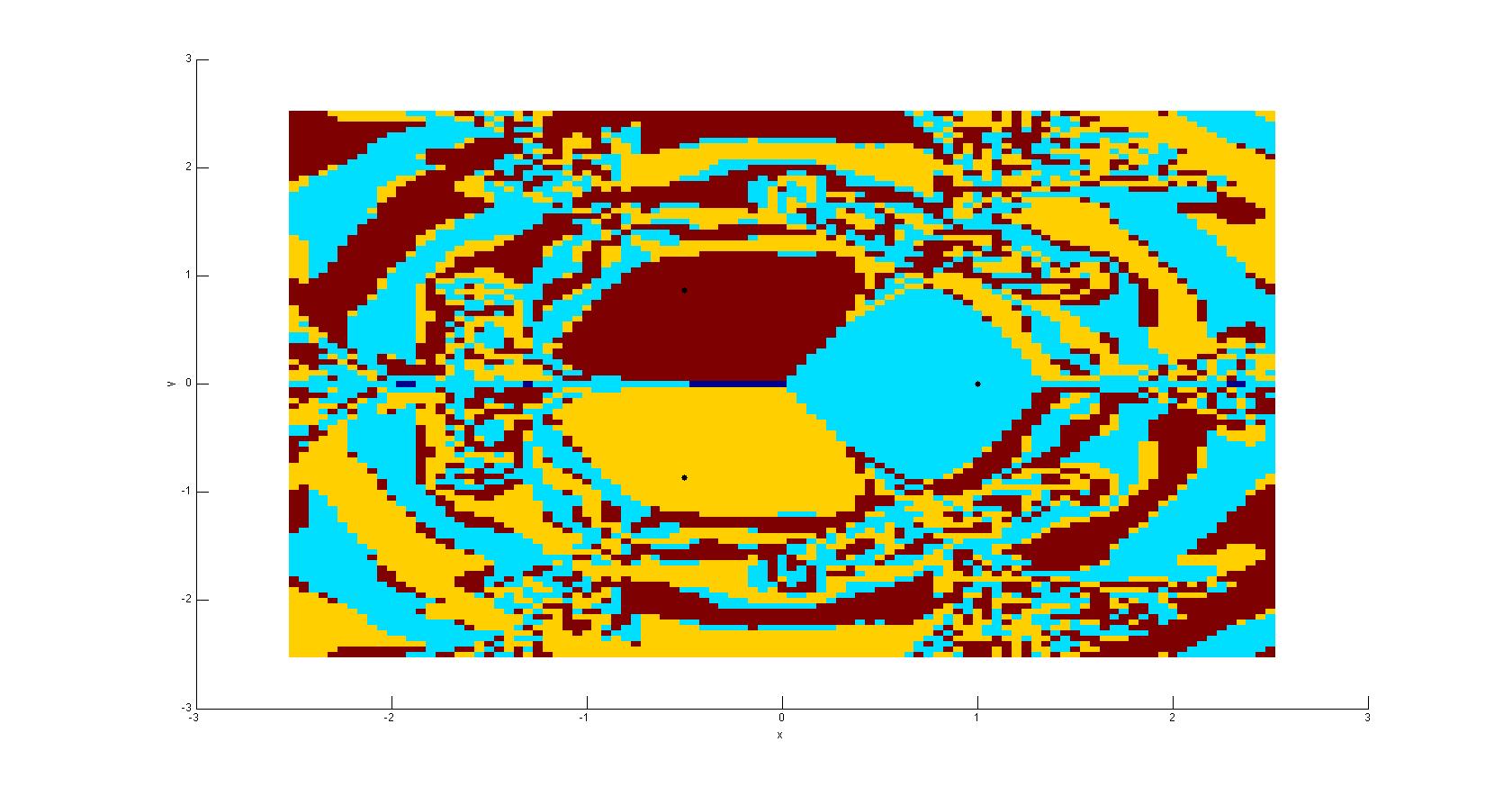
**IV. Damped Motion**

*A. Basins of Attraction*

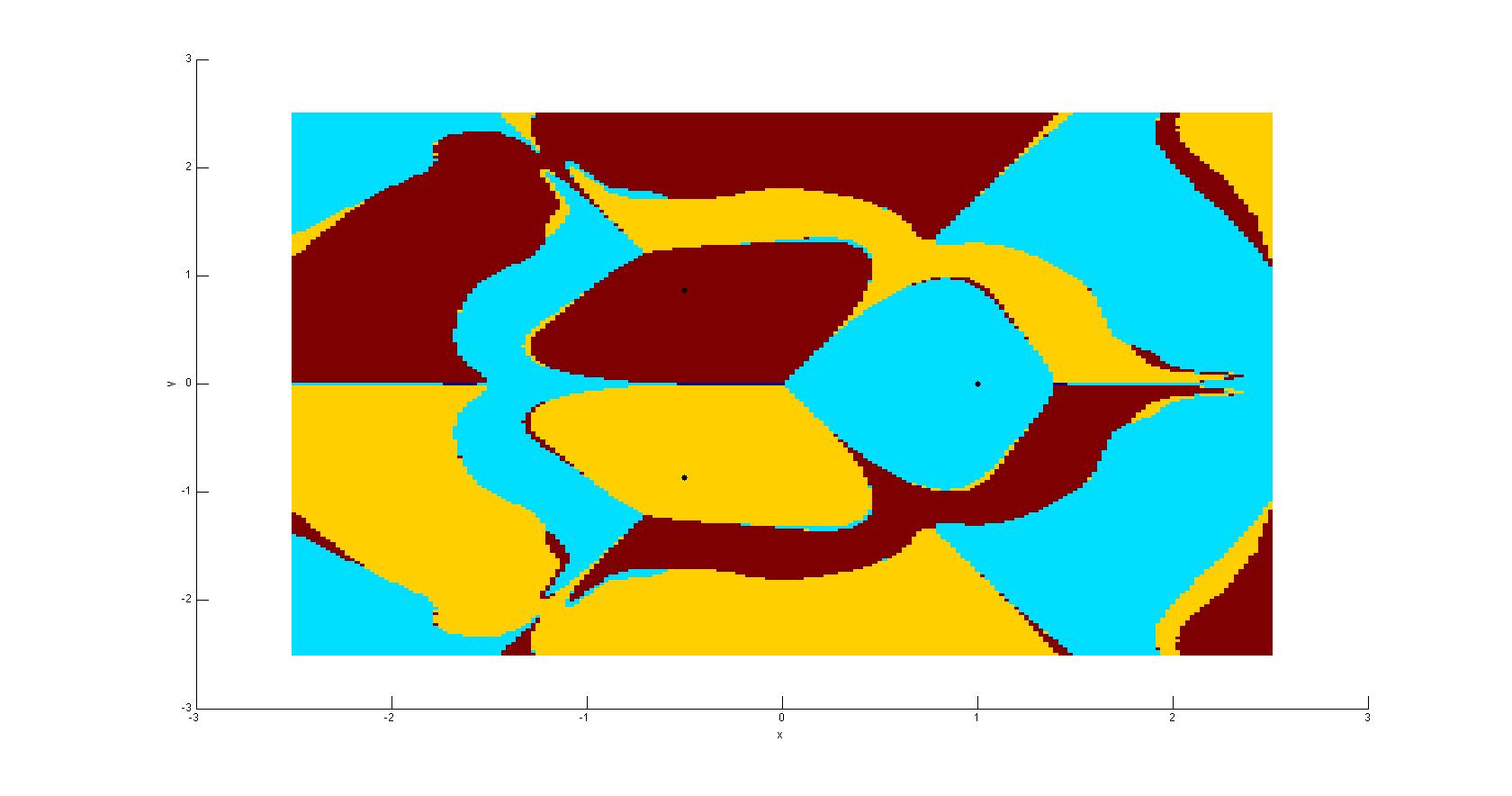
The dynamics of the system change significantly when we consider the damping force described in Part II. With damping present, the pendulum eventually comes to rest at one of several locations in the *x-y* plane. When the magnets are attracting, the pendulum will either come to rest above one of the magnets, or at some particular location between them such that the forces of gravity and magnetism balance. Previous investigations have sought to determine the basins of attraction for attracting magnets. In other words, they have investigated the end position of the pendulum given a set of initial conditions.

For two and three attracting magnets, these fixed points exist above each magnet and in the center of all of the magnets. This information allows us to map the basin of attraction for each magnet. As we vary the damping parameter γ, the map deforms in interesting ways. With two and the three magnets, and for γ = 0.15, we get the basins in Figures 7 and 8. In these plots, the magnets are denoted by black dots.

  
**Figure 8**: Basins of Attraction for Two Attracting Magnets, γ = 0.15

  
**Figure 9**: Basins of Attraction for Three Attracting Magnets, γ = 0.15

As we increase the value of γ, we see that this complicated image begins to simplify and the rate at which the pendulum settles to a fixed point increases. In the following image, we increase damping to γ = 0.40:

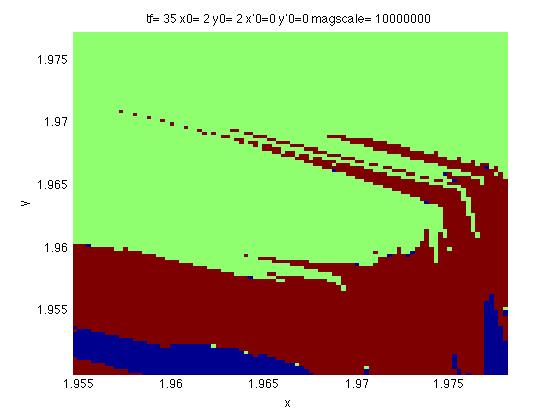


**Figure 10**: Three Magnet Basin of Attraction, γ = 0.40

*B. Fractal Basin Boundaries*

According to the authors of previous investigations, the boundaries between these regions are fractal basin boundaries. Peitgen, Jurgens, and Saupe describe a “very complicated structure which has similarity to a Cantor set. In other words, whenever two basins seem to meet, we discover upon closer examination that the third basin is there in between them, and so *ad infinitum*.” The complexity of these boundaries seems to be a product of the damping coefficient we apply to our model, as seen in the previous section.

These fractal basin boundaries have been observed and corroborated by others.1,2,4,6 We also carried out a numerical investigation of these boundaries using our improved model of the magnetic pendulum.

As before, we calculated orbits for an array of initial conditions and recorded the endpoint of the pendulum when it comes to rest. In this case, we tested a 200x200 array of initial conditions in the square . Although the resolution is limited by available computing power, the result nevertheless shows the signs fractal basin boundaries. In particular, we were able to resolve points in between two basins that were attracted to the third.

**Figure 11**: Fractal Basin Boundary, γ = 0.40



**V. Conclusions**

At times throughout the investigation, we were limited by the computing power available to us. This was particularly evident in the section above, as we attempted to examine the fractal basin boundaries of the damped system. With additional resources, it would be interesting to measure (using, for example, the correlation dimension) the fractals determined by the boundaries of these basins. Past experiments have not attempted to analyze these boundaries in any meaningful way, although doing so would provide a much deeper understanding of the pendulum’s dynamics.

Nevertheless, we believe our model represents an improvement over past investigations of the magnetic pendulum. It properly reflects the dipole-dipole interactions of the magnets, rather than simplifying the system with point charges. At the same time, our model continues to rely on several crude estimations, such as motion restricted to two dimensions and a linear gravitational force. Given more time and greater resources, we would extend this model into three dimensions to properly measure the dynamics of the magnetic pendulum.

**VI. Sources**

[1] Craven, Galen. “Chaotic Dynamics of a Magnetic Pendulum.” Wolfram Demonstrations.1 Dec. 2011. Web.

http://demonstrations.wolfram.com/ChaoticDynamicsOfAMagneticPendulum/

[2] Deacon, Christopher. “The Magnetic Pendulum.” Memorial University of Newfoundland, Department of Physics and Physical Oceanography. 30 Nov. 2011.Web.

<http://www.physics.mun.ca/~cdeacon/labs/3900/>

[3] Furlani, Edward. *Permanent Magnet and Electromechanical Devices.* San Diego, CA: Harcourt Academic Press. 2001. Print

[4] “The Magnetic Pendulum Fractal.” The Code Project. 22 Nov. 2006.

<http://www.codeproject.com/KB/recipes/MagneticPendulum.aspx>. Accessed 30 Nov. 2011. Web.

[5] Peitgen et al. *Chaos and Fractals: New Frontiers of Science*. New York, NY: Springer Publishing, 2004. Print.

[6] Tél, Támas and Márton Gruiz. *Chaotic Dynamics: An Introduction Based on Classical Mechanics*. Cambridge University Press. 2006. Print.

**VII. Appendix: MATLAB Codes by Report Section**

**II. Background and Methodology**

function [ uvxys ] = systemi( uvxy, gamma, rmags, coeffs, magscale, h, mu0 )

%SYSTEMI 'I' Magnet ROMP system

% UVXY is a vector that contains variables uvxy in that order for input

% GAMMA is the damping constant (positive)

% RMAGS is the 3xN matrix describing the position of each of the N

% magnets locations in R^3 in each column

% COEFFS is a vector of length N describing the relative weights of the

% magnets and their magnetic moments

% MAGSCALE is the scaling on magnetic force as compared to gravitational

% H is the height of the pendulum above the plane

% MU0 is a physical constant

nummags = length(coeffs);

u = uvxy(1); v = uvxy(2); x = uvxy(3); y = uvxy(4);

r = [x;y;h]\*ones(1,nummags) - rmags;

rs = zeros(1,nummags);

Fmag = zeros(3,nummags);

for i=1:nummags

rs(i) = norm(r(:,i)); % compute magnitude of separation between mags

Fmag(:,i) = ((3\*mu0)/(4\*pi\*rs(i)^5)) \* (-r(:,i) + 5\*h^2\*r(:,i)/rs(i)^2); % compute force for each mag

end

Fgrav = -[x;y;h];

uvxys = [ sum( magscale\*coeffs(:).\*Fmag(1,:)' ) + Fgrav(1) - gamma\*u;...

sum( magscale\*coeffs(:).\*Fmag(2,:)' ) + Fgrav(2) - gamma\*v;...

u;...

v];

end

function [ uvxys ] = system2i( uvxy, gamma, rmags, coeffs, magscale, h, mu0 )

%SYSTEM2I 2 'I' Magnet ROMP systems run at same time (to demo divergence)

% UVXY is a vector that contains variables uvxy (in that order) for

% input, storing initial conditions for two orbits

% GAMMA is the damping constant (positive)

% RMAGS is the 3xN matrix describing the position of each of the N

% magnets locations in R^3 in each column

% COEFFS is a vector of length N describing the relative weights of the

% magnets and their magnetic moments

% MAGSCALE is the scaling on magnetic force as compared to gravitational

% H is the height of the pendulum above the plane

% MU0 is a physical constant

% UVXYS is the output for the rate equation, giving the first

% derivative for each variable in two separate orbits

nummags = length(coeffs);

u1 = uvxy(1); v1 = uvxy(2); x1 = uvxy(3); y1 = uvxy(4);

u2 = uvxy(5); v2 = uvxy(6); x2 = uvxy(7); y2 = uvxy(8);

r1 = [x1;y1;h]\*ones(1,nummags) - rmags;

r2 = [x2;y2;h]\*ones(1,nummags) - rmags;

rs1 = zeros(1,nummags);

rs2 = zeros(1,nummags);

Fmag1 = zeros(3,nummags);

Fmag2 = zeros(3,nummags);

for i=1:nummags

rs1(i) = norm(r1(:,i)); % compute magnitude of separation between mags

rs2(i) = norm(r2(:,i));

Fmag1(:,i) = ((3\*mu0)/(4\*pi\*rs1(i)^5)) \* (-r1(:,i) + 5\*h^2\*r1(:,i)/rs1(i)^2); % compute force for each mag

Fmag2(:,i) = ((3\*mu0)/(4\*pi\*rs2(i)^5)) \* (-r2(:,i) + 5\*h^2\*r2(:,i)/rs2(i)^2);

end

Fgrav1 = -[x1;y1;h];

Fgrav2 = -[x2;y2;h];

uvxys = [ sum( magscale\*coeffs(:).\*Fmag1(1,:)' ) + Fgrav1(1) - gamma\*u1;...

sum( magscale\*coeffs(:).\*Fmag1(2,:)' ) + Fgrav1(2) - gamma\*v1;...

u1;...

v1;...

sum( magscale\*coeffs(:).\*Fmag2(1,:)' ) + Fgrav2(1) - gamma\*u2;...

sum( magscale\*coeffs(:).\*Fmag2(2,:)' ) + Fgrav2(2) - gamma\*v2;...

u2;...

v2];

end

**DAMPEDSYSTEM.M**

% Damped n-magnet system with variable parameters

% CONSTANTS

h = 1; % height of the pendulum stand above the z-plane at rest

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

x0 = .5;

y0 = .6;

xp0 = 0;

yp0 = 0;

tf = 1;

gamma = 0; % damping factor

eps = 1e-10; % for separating initial conditions

coeffs = [-1 -1 -1];

rmags = [-.5 -.5 1;sqrt(3)/2 -sqrt(3)/2 0;0 0 0];

% Our system

system = @(t,uvxy) systemi( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

profile on;

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0]);

%%%%%%%%

% For plotting two orbits separated by epsilon

%[tse,uvxyse] = ode45(system,[0 tf],[xp0;yp0;x0+eps;y0]);

%%%%%%%%

profile report;

profile off;

figure;

plot(uvxys(:,3),uvxys(:,4));

hold on;

%%%%%%%%

%plot(uvxyse(:,3),uvxyse(:,4),'r');

%%%%%%%%

plot(rmags(1,:),rmags(2,:),'.k','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale)]);

**POTENTIALPLOTFINAL.M**

xhat = [1;0;0];

yhat = [0;1;0];

zhat = [0;0;1];

height = 1\*zhat; % height of the pendulum stand above the plane at rest

len = 3;

mom = 1; % Magnetic moment for all magnets (magnitude);

mu0 = (4\*pi)\*10^-7;

mass = 1; % Mass of the pendulum

gravity = 9.8; % Acceleration due to gravity

moment1 = -mom .\* zhat; rmag1 = -0.5\*xhat + 0.866\*yhat;

moment2 = -mom .\* zhat; rmag2 = -0.5\*xhat - 0.866\*yhat;

moment3 = -mom .\* zhat; rmag3 = 1\*xhat;

momentp = mom .\* zhat;

coeff1 = -1;

coeff2 = -1;

coeff3 = -1;

gridley = -4:.1:4;

H = @(rp,pmom, rm,mmom) -norm(rp-rm).^(-3) \* (4\*pi\*3\*dot(mmom,rp-rm)\*dot(pmom,rp-rm) - dot(mmom,pmom));

grav = @(rp) (1/2)\*norm(rp).^2;

[x, y] = meshgrid(gridley,gridley);

z1 = zeros(length(x),length(y));

z2 = z1;

z3 = z1;

zg = z2;

for i = 1:length(x)

for j = 1:length(y)

z1(i,j) = H([x(1,i);y(j,1);1],momentp,rmag1,moment1);

z2(i,j) = H([x(1,i);y(j,1);1],momentp,rmag2,moment2);

z3(i,j) = H([x(1,i);y(j,1);1],momentp,rmag3,moment3);

zg(i,j) = grav([x(1,i);y(j,1);1]);

end

end

figure;

surf(x,y,z1+z2+z3+zg);

xlabel('y'); ylabel('x');

title('Potential Energy with Three Fixed Magnets');

**III. Undamped Motion**

*i. Crude Method – Orbits initially displaced by distance ε*

**SENSITIVEDEPENDENCE.M**

% CONSTANTS

h = 1; % height of the pendulum stand above the z-plane at rest

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

x0 = 2\*rand-1

y0 = 2\*rand-1

xp0 = 0;

yp0 = 0;

tf = 100;

gamma = 0; % damping factor

eps = 1e-10; % for separating initial conditions

coeffs = [-1 -1 -1];

rmags = [-.5 -.5 1;sqrt(3)/2 -sqrt(3)/2 0;0 0 0];

% Our system

system = @(t,uvxy) system2i( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0;xp0;yp0;x0+eps;y0]);

separation = [uvxys(:,3)-uvxys(:,7) uvxys(:,4)-uvxys(:,8)];

distance = zeros(max(size(separation)));

for i=1:max(size(separation))

distance(i) = norm(separation(i,:));

end

subplot(1,3,1);

semilogy(ts,distance);

xlabel('time'); ylabel('separation');

title(['Separation between orbits in same epsilon neighborhood for epsilon=',num2str(eps)]);

hs = 0\*ts;

for i=1:length(ts)

hs(i) = log(distance(i)/eps)/ts(i);

end

subplot(1,3,2);

plot(ts,hs);

xlabel('t'); ylabel('h');

title('Numerical Lyapunov Exponent vs. Time');

subplot(1,3,3);

plot(uvxys(1:end,3),uvxys(1:end,4));

hold on; plot(rmags(1,:),rmags(2,:),'.k','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale)]);

plot(uvxys(1:end,7),uvxys(1:end,8),'r');

hold off;

**LYAPSURF.M**

% CONSTANTS

h = 1; % height of the pendulum stand above the z-plane at rest

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

xp0 = 0;

yp0 = 0;

tf = 100;

gamma = 0; % damping factor

eps = 1e-10; % for separating initial conditions

% 2 MAGNETS

% coeffs = [-1 -1];

% rmags = [1 -1;0 0; 0 0];

% 3 MAGNETS

coeffs = [-1 -1 -1];

rmags = [-.5 -.5 1;sqrt(3)/2 -sqrt(3)/2 0;0 0 0];

% Our system

system = @(t,uvxy) system2i( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

% Now to run this over the whole grid [-2.5,2.5]x[-2.5,2.5] with resolution:

res = 1/4;

hs = zeros(2/res);

xx = -2.5:res:2.5;

profile on

row = 0;

for x0=-2.5:res:2.5

column = 0;

row = row+1;

for y0=-2.5:res:2.5

column = column+1;

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0;xp0;yp0;x0+eps;y0]);

separation = [uvxys(:,3)-uvxys(:,7) uvxys(:,4)-uvxys(:,8)];

distance = zeros(length(separation),1);

for i=1:length(separation)

distance(i) = norm(separation(i,:));

end

for m=1:numel(distance)

if distance(m) > 1/2

cutoff = m;

break;

else

cutoff = numel(distance);

end

end

hlocal = log(abs(distance(cutoff)-eps)/eps)/ts(cutoff);

hs(column,row) = hlocal;

end

end

profile report

profile off

figure;

xlabel('x'); ylabel('y'); zlabel('h');

imagesc(xx,xx,hs);

hold on; plot(rmags(1,:),rmags(2,:),'.k','MarkerSize',15);

*ii. Re-orthogonalizing Version with Averaging Over Long Trajectory*

function [ Df ] = crude\_jac( f, x, eps )

%CRUDE\_JAC Numerically computes the Jacobian of Coupled ODE System f(t,x)

% F is the coupled system f(t,x), which is autonomous and therefore

% requires no input for t

% X is the point

% EPS is the epsilon with which we approximate the derivative

len = length(x);

Df = zeros(len);

xeps = 0\*Df;

t = 0; % dummy variable for autonomous system

for i=1:len

xeps(:,i) = x;

xeps(i,i) = xeps(i,i) + eps;

Df(:,i) = (1/eps)\*(f(t,xeps(:,i))-f(t,x));

end

end

function [x, DFx] = lorenz\_time1mapmod(F, xo, eps)

% evolve Lorenz flow for 1 time unit, including finding the jacobean matrix

Df = @(y) crude\_jac(F, y, eps); % DF at y

J0 = eye(length(xo)); % initial Jac matrix

% 20-component ODE flow given by 4 components of solution and 16 components

% of the J matrix (J satisfies the ODE dJ/dt = Df.J)

G = @(t,z) [F(t,z(1:4,:)); Df(z(1:4,:))\*z(5:8,:); Df(z(1:4,:))\*z(9:12,:); ...

Df(z(1:4,:))\*z(13:16,:); Df(z(1:4,:))\*z(17:20,:) ];

[~, xs] = ode45(G, [0 1], [xo; J0(:)]); % numerically solve in t domain

x = reshape(xs(end,1:4), [4,1]); % extract the answer at the final time t=1

J = xs(end,5:end); % same for the J components

DFx = reshape(J,[4 4]); % send J out as a 4x4 matrix.

end

**LYAPFLOW\_MOD.M**

% lyapunov exponents in a flow in R^2

% CONSTANTS

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

% Magnet setup

h = 1; % height of the pendulum stand above the z-plane at rest

tf = 100;

gamma = 0; % damping factor

coeffs = [-1 -1 -1];

rmags = [-.5 -.5 1;sqrt(3)/2 -sqrt(3)/2 0;0 0 0];

% INITIAL CONDITIONS

x0 = -.5;

y0 = .85;

xp0 = 0;

yp0 = 0;

% Our system

system = @(t,uvxy) systemi( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

disp('Examining orbit');

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0]);

figure;

plot(uvxys(:,3),uvxys(:,4));

hold on; plot([-.5 -.5 1],[.866 -.866 0],'.r','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale), 'gamma=',num2str(gamma)]);

hold off; drawnow;

disp('Measuring Lyapunov exponents...'); eps = 1e-8;

profile on;

% ----- Re-orthogonalizing version, repeated averaging

M = 5; % how many averaging loops

N = 25; % how many its per meas step

x = [xp0;yp0;x0;y0];

h = zeros(4,1); % place to store averaged lyap exps

for m=1:M

J = eye(4); % Id is where Jacobean starts

for n=1:N

[x Jx] = lorenz\_time1mapmod(system, x, eps);

J = Jx\*J; % update Jacobean

[Q,R] = qr(J); % re-orthogonalize

J = Q\*diag(diag(R)); % but keep them correct lengths

end

rN = abs(diag(R)); % print out progress

h = h + log(rN)/N; disp('best lyap est so far:'); h/m

end

h = h/M % final answer

[~, ind] = max(abs(h));

fprintf('The largest Lyapunov exponent is %f\n',h(ind));

profile report;

**IV. Damped Motion**

**BASINS.M**

% CONSTANTS

h = 1; % height of the pendulum stand above the z-plane at rest

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

x0 = .5;

y0 = .6;

xp0 = 0;

yp0 = 0;

tf = 1;

gamma = 0; % damping factor

eps = 1e-10; % for separating initial conditions

coeffs = [-1 -1 -1];

rmags = [-.5 -.5 1;sqrt(3)/2 -sqrt(3)/2 0;0 0 0];

% Our system

system = @(t,uvxy) systemi( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

res = 1/25; %number of initial conditions tested per unit length

x = -3:res:3;

grid = zeros(length(x));

row = 0;

figure; hold on;

profile on

%tests initial conditions for each 1/25x1/25 square in the square

%[-3,3]x[-3x3]. For each initial condition, assigns a value based on the

%final resting position of the magnet

for x0 = -3:res:3;

column = 0;

row = row + 1;

for y0 = -3:res:3;

column = column + 1;

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0]);

if 0.4< uvxys(end,3) %ends up near magnet at (1,0)

grid(column,row) = 1;

elseif uvxys(end,4)<-0.5 %ends up near magnet at (-.5,-.866)

grid(column,row) = 2;

elseif uvxys(end,4)>0.5 %ends up near magnet at (.5,-.866)

grid(column,row) = 3;

else

grid(column,row) = 0; %ends up at center of the plane

end

end

end

profile report

%plot basins of attraction

imagesc(x,x,grid);

plot([-0.5 -0.5 1],[0.866 -0.866 0],'.k','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale)]);

figure;

plot(uvxys(:,3),uvxys(:,4));

hold on; plot([-0.5 -0.5 1],[0.866 -0.866 0],'.r','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale)]);

**TWOMAGBASINS.M**

% CONSTANTS

h = 1; % height of the pendulum stand above the z-plane at rest

mu0 = (4\*pi)\*10^-7;

magscale = 1e7; % scaling on magnetic force

xp0 = 0;

yp0 = 0;

tf = 1;

gamma = 0; % damping factor

eps = 1e-10; % for separating initial conditions

coeffs = [-1 -1];

rmags = [1 -1;0 0;0 0];

% Our system

system = @(t,uvxy) systemi( uvxy, gamma, rmags, coeffs, magscale, h, mu0 );

res = 1/25;

x = -3:1:3;

grid = zeros(length(x));

row = 0;

figure; hold on;

profile on

for x0 = -3:res:3;

column = 0;

row = row + 1;

for y0 = -3:res:3;

column = column + 1;

[ts,uvxys] = ode45(system,[0 tf],[xp0;yp0;x0;y0]);

if uvxys(end,3) > .5

grid(column,row) = 1;

elseif uvxys(end,3) < -.5

grid(column,row) = 2;

else

grid(column,row) = 0;

end

end

end

profile report

imagesc(x,x,grid);

plot([-1 1],[0 0],'.k','MarkerSize',15);

xlabel('x'); ylabel('y');

title(['tf= ',num2str(tf),' x0= ',num2str(x0),' y0= ',num2str(y0),' x''0=',num2str(xp0),' y''0=',num2str(yp0),' magscale= ',num2str(magscale)]);

hold off;