Chasing Chaos: Correlation of Experimental Results with Theoretical Simulation

Aubrey Overson, Austin Nelson, Dr. Marty Johnston

Summer 2005

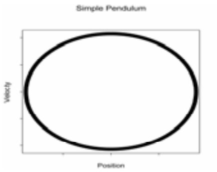
Introduction

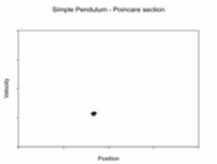
Many would scoff or snicker at the idea of predicting chaotic behavior since the common definition of the word conjures up synonyms like 'unpredictable' and 'disorderly'. However, with a bit of mathematical analysis and abstraction there are ways to see order and beauty in the mess. The goal of the project was develop a computer program written in Matlab to generate the same results as our experimental chaotic pendulum.

The equation of motion for the system is shown below:

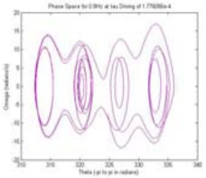
$$I\ddot{\theta} = -mgr\sin(\theta) - [b + \tau_F]\dot{\theta} - \tau_F\left[\frac{\dot{\theta}}{|\dot{\theta}| + \varepsilon}\right] + \tau_D\cos(\omega_D t + \delta)$$

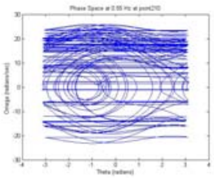
Poincare Sections

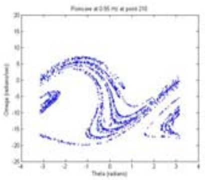




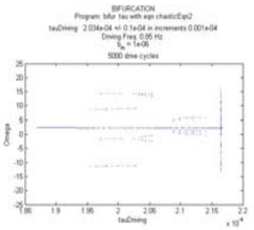
A Poincare section is effectively a sampling of the system's position and velocity (phase space) on a regular interval. If the system moving periodically like the pendulum of a grandfather clock, then the Poincare section would be a single dot. For a chaotic pendulum, if we look to see what it is doing every 2 seconds for example it will not have a predictable position and velocity. However it also can be just any where. The Poincare sections shown in the top right corner from our chaotic system. A closer look reveals a fractal nature and wispy patterns.

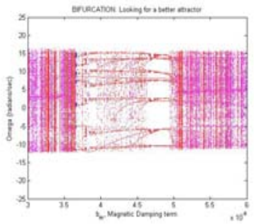






Bifurcation Diagrams





The bifurcation diagram was new to this summer's project. It serves as a ay to analyze the system's behavior. A bifurcation diagram steps through a certain system parameter (such as frequency) as a function of velocity. For example, if a system is periodic at a certain frequency there will be a single dot. As you vary the frequency the number of dots will correspond to the periodicity. Following this branching from one, two, three up to n periods is one way to find chaos in a system. Surprisingly, this has proven to be a great predictive tool when looking for finding chaotic regimes experimentally.

Coding Algorithm

The programs to make the Poincare Sections and Bifurcation Diagrams where written in Matlab using a 4th order Runga-Kutta Integration and a for loop.

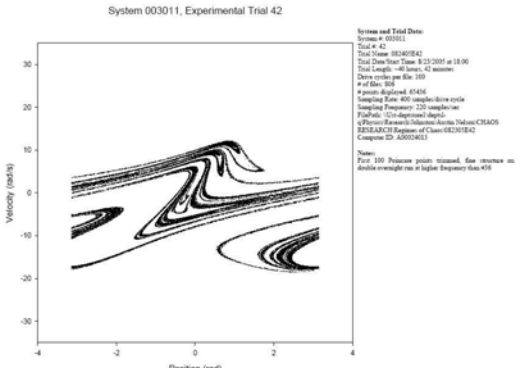
```
%--------------------Aubrey Overson c/o Marty Johnston--------------------%
%---------------------UST Physics Summer 2005 Research--------------------%
%
%This program uses a simple integration of our system's differential
%equation called chaoticEqTest_'run number'.

%--------------------MODIFICATIONS for integrator_fast--------------------%
%FIRST of all, this program is used with 'chaoticEqn' which has a
%magnetic damping term folded into the frictional coefficient.
%This program is used to make integration for our chaotic
%pendulum equation. It uses the tSpan array to feed the integration should
%be done over. It save the Poincare Point in an array and after a certain
%number of points have accumulated it saves it to a file. This is meant to
%hasten run time and lessen required memory.

%These can also be designated as user inputs
driving_freq_of_system = input('what freq would you like (in Hz)? \n');
phase_shift = input('Would you like a phase shift? \n');

% Pick a point to Poincare...make the t_eval array based on this index of
% poincare array. This can also be a user input.
point  = input('What point to Poincare? \n');

angular_freq = 2*pi*driving_freq_of_system;

%------------TIMING PARAMETERS--------------------%
t_driving = (2*pi)/angular_freq;
sample_step = t_driving/400;

%Effectively this is the number of points that get plotted per each driving
%torque...Could also be user specified

num_cycles = 100;
fileNumber = 0;

t_stop = num_cycles*t_driving;

z0 = [0 0];                                          % initial value of 'x'
options = odeset('reltol',1e-6);                     % set error tolerance for the 4th order
                                                       Runga

% Initialize theta and omega for poincare.
theta_2 = ones(1,num_cycles -1);
theta_pi_2 = ones(1,num_cycles -1);
omega_2 = ones(1,num_cycles -1);

t_poincare = [point*sample_step : t_driving : t_stop];

%start clock (just for kicks)
tic;

%displays what omega we have (just for a check)
disp(2*pi*driving_freq_of_system)

for j=1:1:(num_cycles-1)
    tSpan = [t_poincare(1,j) t_poincare(1,j+1)];
    %---------POINCARE FOR SUMMER EQN 2005 w/ damping term------------------%
    [t,z] = ode45(@chaoticEqTest_45, tSpan, z0, options, angular_freq);
    theta_2(j) = z(end,1);
    omega_2(j) = z(end,2);
    %Move along.... re-intialize
    z0 = [ z(end,1), z(end,2) ];
end
for j=1:1:(num_cycles-1)
    clear i
    %Poincare from -pi to pi
    c(j) = exp(i*(theta_2(j)+phase_shift));
    theta_pi_2(j) = atan2(imag(c(j)), real(c(j)));
        if mod(j,100)==0
            disp(theta_pi_2(j))
            disp(omega_2(j))
        end
end

%ends and displays clock
toc;

figure
plot(theta_pi_2, omega_2, 'b.', 'MarkerSize', 2)
xlabel('Theta (radians)')
ylabel('Omega (radians/sec) ')
```

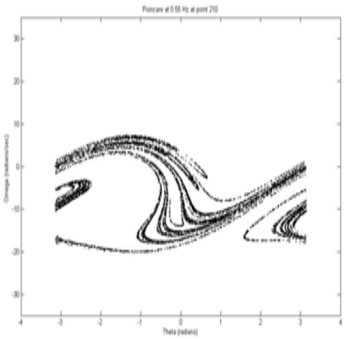
What's Next

Although the summer yielded many improvements, it also leaves us with more areas to explore. Looking ahead, we would like to add in magnetic field interactions into both the physical system and the computer model. We hope to not only maintain correspondence, but also explore synchronization of chaotic systems.

Promising Results

The figures below are the Poincare Sections for experimental chaos (see *Chasing Chaos: Finding Order in Disorder by Austin Nelson*) and the Poincare Section from my theoretical model. Note the strong correlation in scale and filamentary structure.





References

1) Moler, C. *Numerical Computing with Matlab.* MathWorks. 2004.
2) Shampine, L.F., Gladwell, I., Thompson, S. *Solving ODEs with Matlab.* Cambridge Press. Cambridge, UK. 2003.
3) Devaney, R. *A First Course in Chaotic Systems: Theory and Experiment.* Perseus Books. Boulder. 1992

Supported in part by NSF Grant DUE-0126849