Michael Cowan

Computational Physics

Problem Set 5

1. In this problem, we analyze the chaotic behavior of a damped, driven non-linear pendulum. In particular, we examine how the behavior changes as the parameters are changed. In each of the different run-throughs of the program, only one parameter value was changed from the initial system.

Code:

1. clear all;
2. theta0=input('Input the initial angle (degrees): ');
3. tau=input('Input the time step (s): ');
4. maxstep=input('Input the number of steps: ');
5. theta=theta0\*pi/180;
6. omega=0;
7. time=0;
8. g\_over\_L=1;
9. qd=.5;
10. fd=2/3;
11. Ad=1.2;
12. j=0;
13. for i=1:maxstep
14. theta\_bound = theta\*180/pi+180;
15. theta\_bound = mod(theta\_bound,360) - 180;
16. %to save space, only plot every 50th point
17. if (mod(i,50)==0)
18. thplot(i/50)=theta\_bound;
19. omegaplot(i/50)=omega\*180/pi;
20. end
21. accel = -g\_over\_L\*sin(theta) - qd \*omega + Ad\*sin(fd \* time);
22. omega = omega + tau\*accel;
23. theta = theta + tau\*omega;
24. time=time+tau;
25. %create the Poincare section (plot omega vs theta once per period)
26. if ( mod( fd\*time-pi/4 , 2\*pi ) < tau/2 )
27. j=j+1;
28. thpoin(j)=theta\_bound;
29. omegapoin(j)=omega\*180/pi;
30. end
31. end
32. figure(1); clf;
33. plot(thplot,omegaplot,'.');
34. xlabel('theta (degrees)');
35. ylabel('omega (degress/second)');
36. title('Phase Space');
37. pause(0.5);
38. figure(2); clf;
39. plot(thpoin,omegapoin,'.');
40. xlabel('theta (degrees)');
41. ylabel('omega (degress/second)');
42. title('Poincare Section');

Results:

Input the initial angle (degrees): 11.46

Input the time step (s): .01

Input the number of steps: 1000000

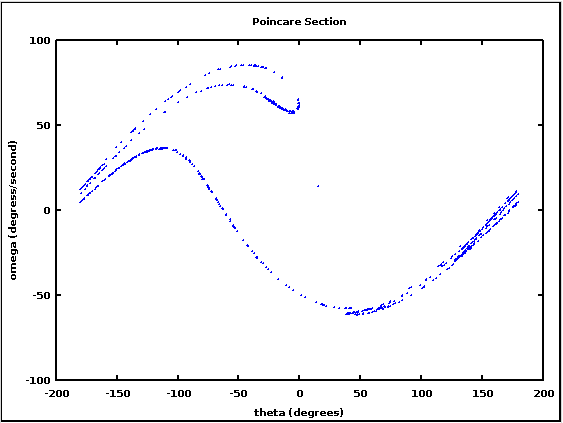
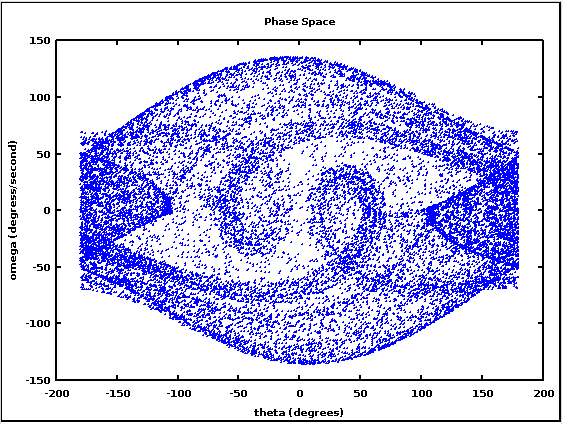


Figure 1: Phase space plot and Poincaré section for an initial angle of 11.46°; 1X106 steps at .01s per step

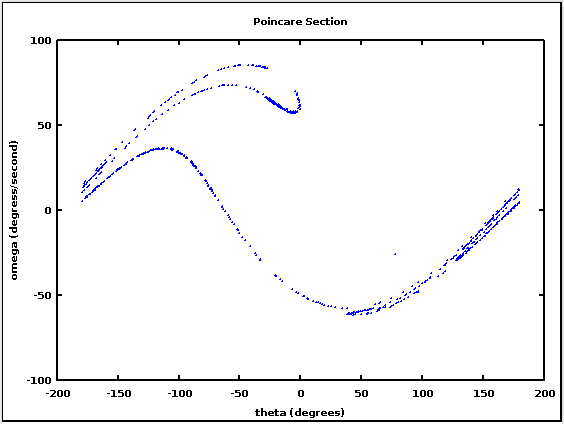
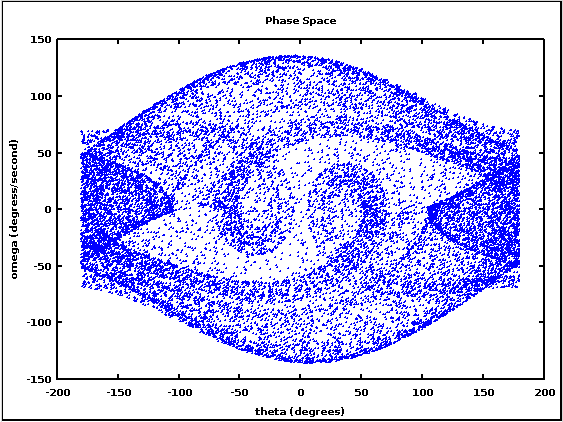


Figure 2: Phase space plot and Poincaré section for an initial angle of 100°; 1X106 steps at .01s per step

Note that the plots for an initial angle of 100° are almost identical to the plots for an initial angle of 11.46°. This is most likely because the motion is dominated by the driving and damping forces, the parameters of which were unchanged, and the initial angle only creates a small difference in the initial transient motion.

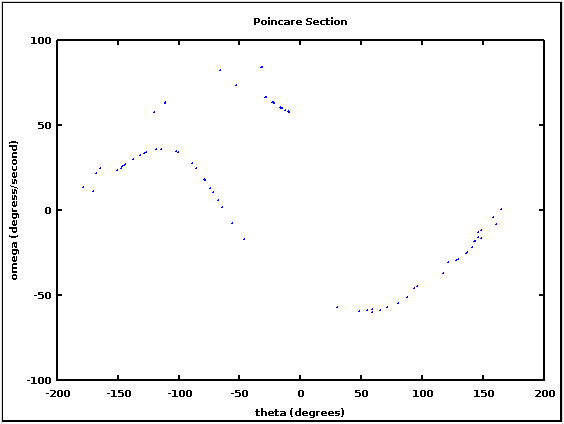
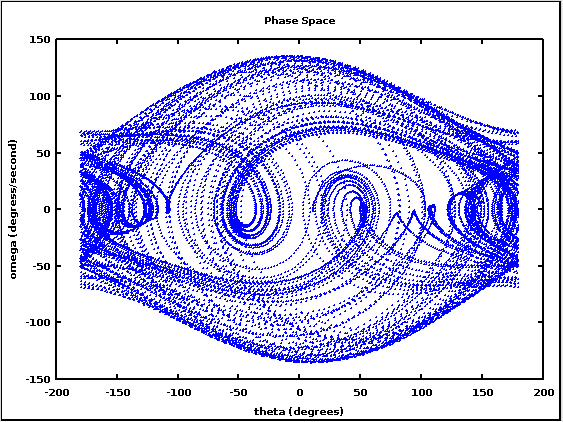


Figure 3: Phase space plot and Poincaré section for an initial angle of 11.46°; 1X106 steps at .001s per step

Here the size of the time step was decreased by an order of magnitude in order to verify that the attractor structure is not an artifact of the size of the time step chosen. While the plots are generally less filled in due to the decrease in the total simulated time, they do have the same shape as the previous plots, verifying that the time step size is not an influence on the resulting attractors.

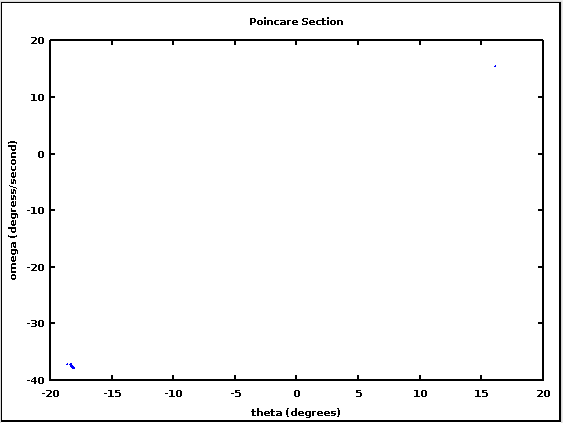
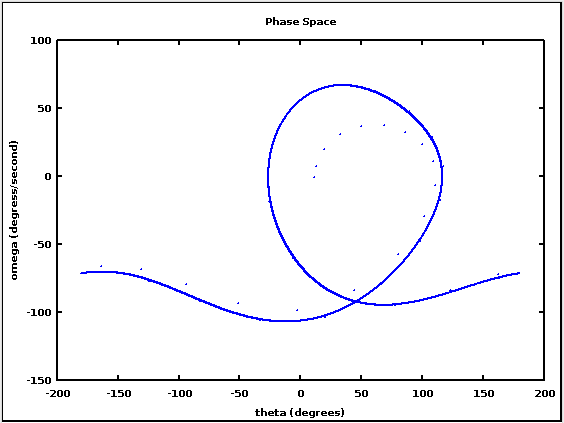


Figure 4: Phase space plot and Poincaré section for an initial angle of 11.46 degrees, but with the frequency of the driving force set to 0.6 instead of 2/3; 1X106 steps at .01s per step

Changing the driving frequency to 0.6 instead of 2/3 produces non-chaotic behavior. After a short transient, the phase space plot begins to follow a regular pattern. You can also see that all but one of the points in the Poincaré section are clustered in the bottom left corner of the plot.

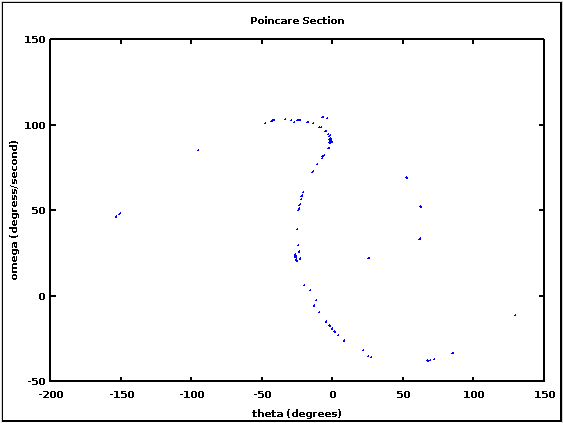
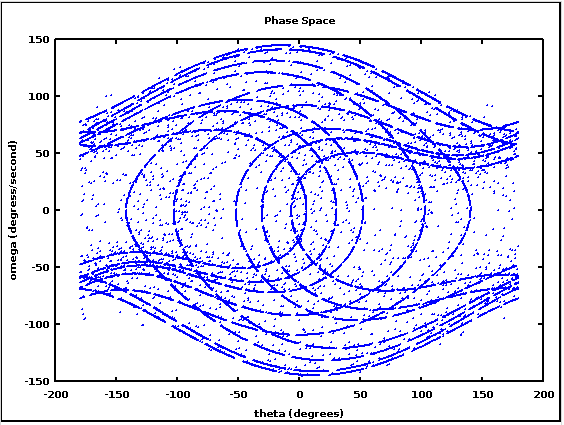


Figure 5: Phase space plot and Poincaré section for an initial angle of 11.46 degrees, but with the frequency of the driving force set to 1/3 instead of 2/3; 1X106 steps at .01s per step

Changing the frequency of the driving force to 1/3 brings back the chaotic behavior. The Poincaré section is entirely different from that in Figure 1, and there are significant differences in the Phase Space plot as well.

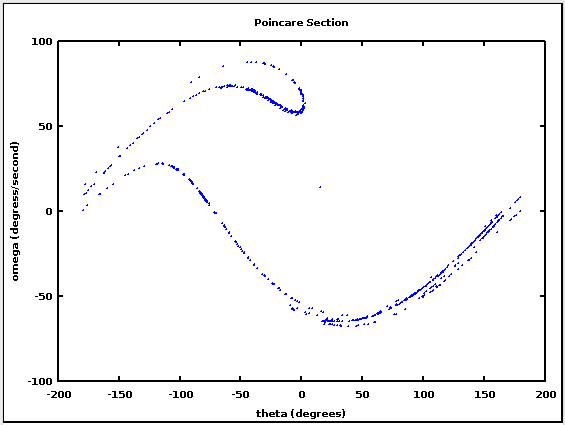
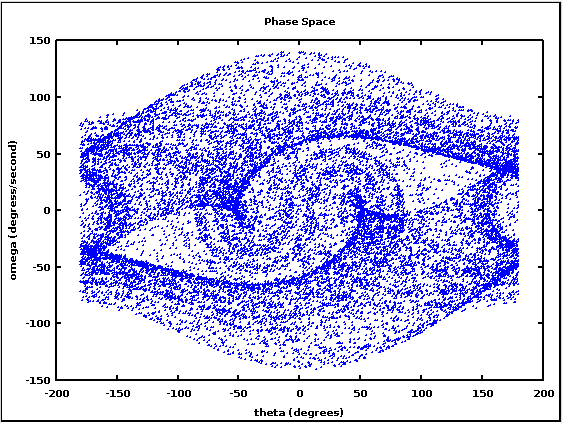


Figure 6: Phase space plot and Poincaré section for an initial angle of 11.46 degrees, but with the damping coefficient set to 0.45 instead of 0.5; 1X106 steps at .01s per step

Decreasing the magnitude of the damping force by 10% maintains the chaotic behavior. Interestingly, the Poincaré section is almost identical to that in Figure 1, but the phase space plot has significant differences.

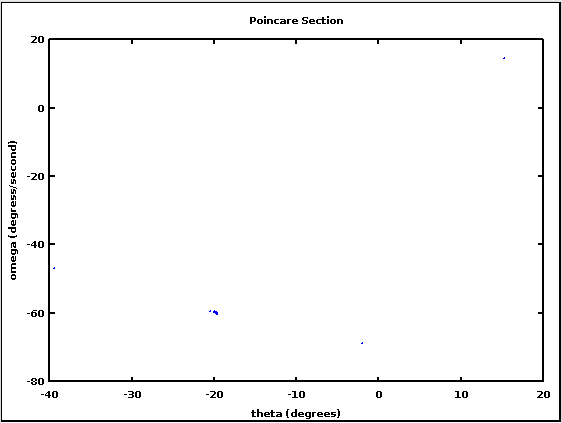
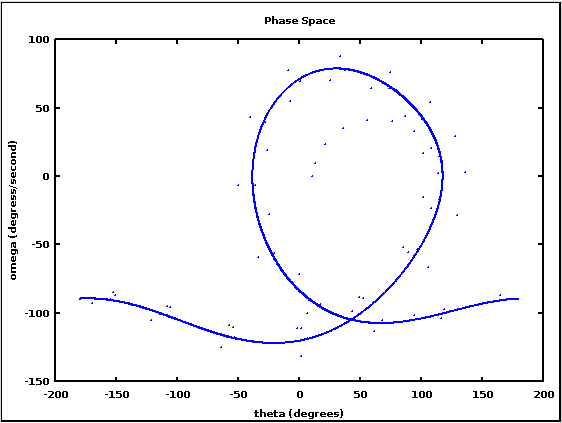


Figure 7: Phase space plot and Poincaré section for an initial angle of 11.46 degrees, but with the magnitude of the damping force set to 0.4 instead of 0.5; 1X106 steps at .01s per step

Changing the damping coefficient to 0.4 produces non-chaotic motion. Again, all of the points in the Poincaré section are clustered together, though here they are in a different place than they were in Figure 4. Also, the transient initial motion lasts longer, which makes since given that the damping force is smaller.

2. In this problem, we were asked to write a program to find the inverse of a matrix. An identity matrix is appended to the input matrix to produce an augmented matrix. A form of Gaussian elimination is then used to transform the input matrix into an identity matrix; the original identity matrix is then the inverse of the input matrix.

The program also handles several special cases: non-square input matrices (which are by definition not invertible), matrices that start with one or more elements equal to 0 on the diagonal, and matrices that start with elements <<1 on the diagonal. To handle the first situation, the program simply ends and outputs a message that the matrix is not square. For the second special case, rows are exchanged until there are no elements that =0 on the diagonal, then proceeds with the Gaussian elimination. In the third case, the row containing the element that is <<1 is multiplied by the reciprocal of that element, then the program proceeds with the Gaussian elimination.

Code:

1. clear all;
2. disp('Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:');
3. A=input('');
4. [m,n]=size(A);
5. if (m~=n) %check that the matrix is a square matrix
6. disp('The matrix is not a square matrix so it has no inverse.');
7. else
8. B=[A,eye(n,n)]; %create the augmented matrix by appending an identity matrix to A
9. %check to see if any of the main diagonal elements <<1
10. for i=1:n
11. if abs(B(i,i))<1e-4
12. %if the element is equal to 0, pivot the matrix
13. if abs(B(i,i))==0
14. if i==n
15. circshift(B,1);
16. i=1;
17. else
18. for j=i+1:n
19. if B(j,i)~=0
20. temp=B(j,:);
21. B(j,:)=B(i,:);
22. B(i,:)=temp;
23. end
24. end
25. end
26. else %if the element is small but not 0, multiply the row by its reciprocal
27. B(i,:)=(1/B(i,i)).\*B(i,:);
28. end
29. end
30. end
31. %Gaussian elimination
32. for i=1:n
33. B(i,:)=B(i,:)./B(i,i);
34. for j=1:n
35. if (j~=i) %skip the row we're working on
36. B(j,:)=B(j,:).-(B(j,i).\*B(i,:)); %change the rest of the column to 0's
37. end
38. end
39. end
40. %retreive the right half of the augmented matrix
41. for i=1:n
42. for j=1:n
43. C(i,j)=B(i,j+n);
44. end
45. end
46. disp('The inverse is: ');
47. disp(C);
48. end

Results:

For each of the given matrices, I ran the program, then checked the result by multiplying the output by the input.

Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:

[7,2,3;4,7,6;7,8,9]

The inverse is:

0.227273 0.090909 -0.136364

0.090909 0.636364 -0.454545

-0.257576 -0.636364 0.621212

>> A\*C

ans =

1.00000 0.00000 -0.00000

-0.00000 1.00000 -0.00000

0.00000 0.00000 1.00000

Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:

[4,2,3;4,5,6;7,8,9]

The inverse is:

0.33333 -0.66667 0.33333

-0.66667 -1.66667 1.33333

0.33333 2.00000 -1.33333

>> A\*C

ans =

1.00000 0.00000 -0.00000

0.00000 1.00000 0.00000

0.00000 -0.00000 1.00000

Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:

[1e-10,0,0;0,1,0;0,0,1]

The inverse is:

1.0000e+10 0.0000e+00 0.0000e+00

0.0000e+00 1.0000e+00 0.0000e+00

0.0000e+00 0.0000e+00 1.0000e+00

>> A\*C

ans =

1 0 0

0 1 0

0 0 1

Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:

[1,2,3;1,3,4;2,1,2]

The inverse is:

-2.00000 1.00000 1.00000

-6.00000 4.00000 1.00000

5.00000 -3.00000 -1.00000

>> A\*C

ans =

1.00000 0.00000 0.00000

0.00000 1.00000 0.00000

-0.00000 0.00000 1.00000

I also ran an additional matrix to check to see if the program could handle having an element of 0 along the diagonal in one or more rows. It was successful.

Input the matrix to be inverted in the form [a11,a12,a13,...;a21,a22,a23,...;a31,a32,a33...;...ann]:

[0,3,2;2,0,-1;-2,1,0]

The inverse is:

0.10000 0.20000 -0.30000

0.20000 0.40000 0.40000

0.20000 -0.60000 -0.60000

>> A\*C

ans =

1.00000 0.00000 0.00000

-0.00000 1.00000 0.00000

0.00000 0.00000 1.00000