Problem Set 1 #14

Logan Courtright

9/8/2019

1 Problem Description

For our example mechanical oscillator, with the value of α given by $\gamma/\omega_r = 1/10$, find the matrix A that corresponds to Eq. I.39. Find the eigenvalues and eigenfunctions of this matrix using MATLAB. Show the MATLAB output. Assume you have a driving force with an amplitude of 1 mN at the resonant frequency, find the matrix B, and use MATLAB to find $\tilde{u} = [\tilde{x}, \tilde{v}]^t$. Show the MATLAB output. Write the real expressions x and v and use a modified version of OscillatorI4 to draw a phase plot of the oscillation. How much energy is stored in the oscillator?

2 Solution

2.1 The A Matrix

Since Eq. I.39 came from Eq. I.17, we must find a similar two equations to construct A for the mechanical oscillator. Adding a driving force to Eq. I.10 gives the correct equations, which are

$$\frac{dx}{dt} - v = 0, \frac{dv}{dt} + \frac{k}{m}x + \alpha v = F' \tag{1}$$

where F' is the driving force into the mechanical oscillator. Putting this into an equation of the form I.38, A becomes

$$\mathrm{A} = \begin{bmatrix} 0 & 1 \\ -k/m & -\alpha \end{bmatrix} \tag{2}$$

where u in Eq. 1.38 is

$$\mathrm{u} = \begin{bmatrix} x \\ v \end{bmatrix} \tag{3}$$

where x is displacement and v is velocity. The MATLAB code to generate the matrix A with the given α value and an arbitrary spring constant k and mass m is shown in Figure 1 below.

To find the eigenvectors and eigenvalues of this new A matrix, the MATLAB command eig is used. When called on a matrix, eig returns two new matrices,

E and L, with the command given by: [E,L] = eig(A). L is a diagonal matrix with the same size as A that holds the eigenvalues of A in the diagonal. E is a matrix of the same size as A that holds the eigenvectors for each eigenvalue, so that A*E = E*L. Once the eigenvalues/vectors are generated, the eigenvector array E is normalized so that the first value in the matrix is 1. The code to generate this and the output are shown in Figures 2 and 3 below.

```
%Variable List
%alpha = damping coefficient
%m = mass of object in the end of the string in kg
%k = spring constant in N/m

alpha = 1/10;
m = 1;
k = 0.5;
A = [0 1; -k/m -alpha];
```

Figure 1: Code to Generate A Matrix

```
%Variable List
%E = diagonal matrix of eigenvectors
%E_norm = diagonal matrix of eigenvectors, normalized
%L = square matrix of eigenvectors

%find eigenvalues/vectors
[E,L] = eig(A);
%normalize eigenvectors
E_norm = E/E(1,1);
%disply results
disp('E:')
disp(E)
disp('E_norm:')
disp(E_norm)
disp('L:')
disp(L)
```

Figure 2: Code to Generate Eigenvalue/vector Matrices of A Matrix

```
E:
   0.8165 + 0.0000i   0.8165 + 0.0000i
  -0.0408 + 0.5759i  -0.0408 - 0.5759i

E_norm:
   1.0000 + 0.0000i   1.0000 + 0.0000i
  -0.0500 + 0.7053i  -0.0500 - 0.7053i

L:
  -0.0500 + 0.7053i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.0500 - 0.7053i
```

Figure 3: Output of Code from Figure 2

2.2 The B Matrix

To solve for $\boldsymbol{u}$, we want to create an equation of the form

$$\mathrm{B}\tilde{\mathrm{u}} = \tilde{\mathrm{D}} \tag{4}$$

To get the equation from Section 2.1 into the form of (4), we need to create the B matrix by combining together the $\mathrm{d}\tilde{\mathrm{u}}/\mathrm{dt}$ and A matrix terms. This is done by multiplying the identity matrix I by the $\mathrm{d}\tilde{\mathrm{u}}/\mathrm{dt}$ and subtracting A from that 2x2 matrix. Since the solution to $\tilde{\mathrm{u}}$ will be in phasor form, the derivative of $\tilde{\mathrm{u}} = i\omega$. However, in order to solve this numerically in MATLAB the value of ω has to be known. Since the problem says to assume the oscillator is operating as resonance frequency, we can assume that $\omega = \omega_r$. To calculate ω_r, we use the relationship given in the problem that $\alpha = \gamma/\omega_r = 1/10$. The eigenvalue matrix L created in Section 2.1 contains the values of γ in the real part of the eigenvalues, so we set $\gamma = 0.05$. From there, $\omega_r = \gamma/\alpha = 0.05/0.1 = 0.5$. From here, we can solve for the B matrix. The MATLAB code to do so is shown below in Figure 4, and the B matrix is shown in Figure 5.

```
%Variable List
%I = 2x2 identity matrix
%A = 2x2 A matrix calculated in Section 2.1
%L = eigenvalue matrix calculated in Section 2.1
%gamma = gamma value of spring equation
%omega = omega value of spring equation
%omega_r = omega_r value of spring equation
%alpha = damping coefficient, value set in Section 2.1

%find gamma
gamma = abs(real(L(1,1)));
%solve for omega_r using alpha
omega_r = gamma / alpha;
%assume omega = omega_r from problem
omega = omega_r;
%create identity matrix and multiply by i*omega
I = i*omega*eye(2);
%Calculate B
B = I - A
```

Figure 4: Code to Generate B Matrix

```
B =

   0.0000 + 0.5000i  -1.0000 + 0.0000i
   0.5000 + 0.0000i   0.1000 + 0.5000i
```

Figure 5: Output of Code from Figure 4

From here, solving for $\tilde{u}$ is fairly simple. Since the B matrix has been found, all we have to do is solve for the $\tilde{D}$ matrix then multiply the $\tilde{D}$ matrix by the inverse B matrix. Since the driving force, F', is represented by the time derivative of the velocity, the $\tilde{D}$ matrix can be represented as

$$\tilde{D} = \begin{bmatrix} 0 \\ i\omega\tilde{F} \end{bmatrix} \tag{5}$$

Using the given value of the diving force from the problem, $\tilde{u}$ can be solved for easily using MATLAB. The code to do so and the corresponding result are displayed below in Figures 6 and 7.

```
%Variable List
%F = Driving force of mechanical oscillator in mN
%D = Energy source matrix
%u = solution vector of distance and velocity

%use given value for F
F = 1;
%Calculate D
D = [0; i*omega*F]
%Calculate U using '\' function
u = B\D
```

Figure 6: Code to Generate D and u Matrices

```
D =

   0.0000 + 0.0000i
   0.0000 + 0.5000i


u =

   0.3846 + 1.9231i
  -0.9615 + 0.1923i
```

Figure 7: Output of Code from Figure 6

2.3 Visualising the Solution

To solve analytically for x and v, there are two separate parts that must be solved. Both distance and velocity have a transient solution, which dominates the domain response at the start of the oscillator's response but gradually goes to 0, and the driven solution that the transient solutions trend towards over time. The code to generate and plot these solutions is displayed in the MATLAB code below. Since the end result has a large difference in scale, two separate plots have been shown to better illustrate whats going on. The code for the second plot is not included, but it is just the first plot with different x axis limits.

```
%Variable List
%alpha, gamma, omega, omega_r = same values calculated previously
%FreqRat = ratio of frequency to resonant frequency, assumed to be 1
%omRat = ratio of resonant frequency to frequency, calculated, if they are
%FreqD = driving frequency
%not equal it will apear here instead of in FreqRat
%Fb, X0a, X0b = initial conditions of force driver and spring displacement
%gam = damping ratio of gamma to frequency
%gamRat = damping ratio of gamma to resonant frequency
%phoff = phase offset between displacement and velocity solutions
%ADc, CDc, SDc = factored in-phase components for driven solution
%XDc, XDs = cosine and sine components for driven solution
%Xc1, Xc2 = initial conditions for cosine component for transient solution
%Xs1, Xs2 = initial conditions for sine component for transient solution
%DisD, VelD = driven displacement and velocity complete solutions
%Dis1, Vel1 = transient displacement and velocity compelete solutions for
%initial condition 1 -> Displacement = 0
%Dis2, Vel2 = transient displacement and velocity compelete solutions for
%initial condition 2 -> Displacement = 1
%Dis1, Vel1 = total solutions for initial condition 1
%Dis2, Vel2 = total solutions for initial condition 2
```

Figure 8: Variable List

```
%frequency ratio assumed to be 1
FreqRat = 1;
% set initial values
X0b = [0 1];
Fb = 1e-3;
X0a = 1;
%calculate the phase offset
phoff = atan(gamma/omega);
%calculate the damping ratio to frequency
gam = gamma/omega;
%calculate the damping ratio to resonant frequency
gamRat = gamma/omega_r;
%calculate resonant frequency to frequency driving ratio
omRat = omega_r/omega;
%calculate driving frequency
FreqD = FreqRat*omRat;
%Calculate components for driven and transient solutions
ADc = 1/((1-FreqRat^2)^2 + alpha^2*FreqRat^2)^(1/2);
CDc = alpha*FreqRat*ADc;
SDc = (1-FreqRat^2)*ADc;
XDc = FreqRat*Fb*ADc*CDc;
XDs = FreqRat*Fb*ADc*SDc;
%calculate initial conditions for solutions
Xc1 = X0b(1) - XDc;  Xc2 = X0b(2) - XDc;
Xs1 = (1/FreqRat)*XDs - gam*Xc1;
Xs2 = (1/FreqRat)*XDs - gam*Xc2;
%set the time array plotting parameters
Nosc = 30; Npoints = 100; Ntot = Nosc*Npoints + 1;
%set up time array
time = 2*pi*(1:Ntot-1)/Npoints;
```

Figure 9: Code to Generate Initial Conditions and Solution Components

```
%Driven Solutions
DisD = XDc*cos(FreqD*time) - XDs*sin(FreqD*time);
VelD = -(omega_r/(omRat))*(XDs*cos(FreqD*time)...
    + XDc*sin(FreqD*time));
VelD = VelD*1e3;
%Initial Condition 1 Transient Solutions
DisT1 = (Xc1*cos(time) - Xs1*sin(time)).*exp(-gam*time);
VelT1 = -(omega_r/(omRat))*((Xc1-(gamma/omega_r)*Xs1)*sin(time)...
    +(Xs1+(gamma/omega_r)*Xc1)*cos(time)).*exp(-gam*time);
VelT1 = VelT1*1e3;
%Initial Condition 2 Transient Solutions
DisT2 = (Xc2*cos(time) - Xs2*sin(time)).*exp(-gam*time);
VelT2 = -(omega_r/(omRat))*((Xc2-(gamma/omega_r)*Xs2)*sin(time)...
    +(Xs2+(gamma/omega_r)*Xc2)*cos(time)).*exp(-gam*time);
VelT2 = VelT2*1e3;
%Adding Driven Solutions to Transient Solutions to get Total Solution
Dis1 = DisT1 + DisD;  Vel1 = VelT1 + VelD;
Dis2 = DisT2 + DisD;  Vel2 = VelT2 + VelD;
```

Figure 10: Code to Generate Complete Solutions

```
figure(1)
hold on
plot(.001*Vel1,Dis1)
plot(.001*Vel2,Dis2)
plot(.001*VelD,DisD,'g')
hold off
%set x and y limits
xlim([-0.5 0.5])
ylim([-1 1])
%set tick label interpreter to LaTeX
set(gca,'TickLabelInterpreter','LaTeX')
%label x and y axis and title
xlabel('Velocity (s)','interpreter','LaTeX')
ylabel('Displacement (m)','interpreter','LaTeX',...
    'fontname','Times New Roman')
title('Phase Plot: Displacement versus Velocity',...
    'interpreter','LaTex','fontname','Times New Roman')
%set x and y axis tick marks and tick labels
xticks([-0.5:0.25:0.5])
xticklabels({'$-0.5$','$-0.25$','$0$','$0.25$','$0.5$'});
yticks([-1:0.5:1])
yticklabels({'$-1$','$-0.5$','$0$','$0.5$','$1$'})
box on
%set legend and remove box around it
lgd = legend('Initial Displacement = 0','Initial Displacement = 1',...
    'Driven Solution');
set(lgd,'interpreter','LaTeX')
legend('boxoff')
%save as pdf
saveas(gcf,'14.pdf','pdf')
```

Figure 11: Code to Plot Solutions

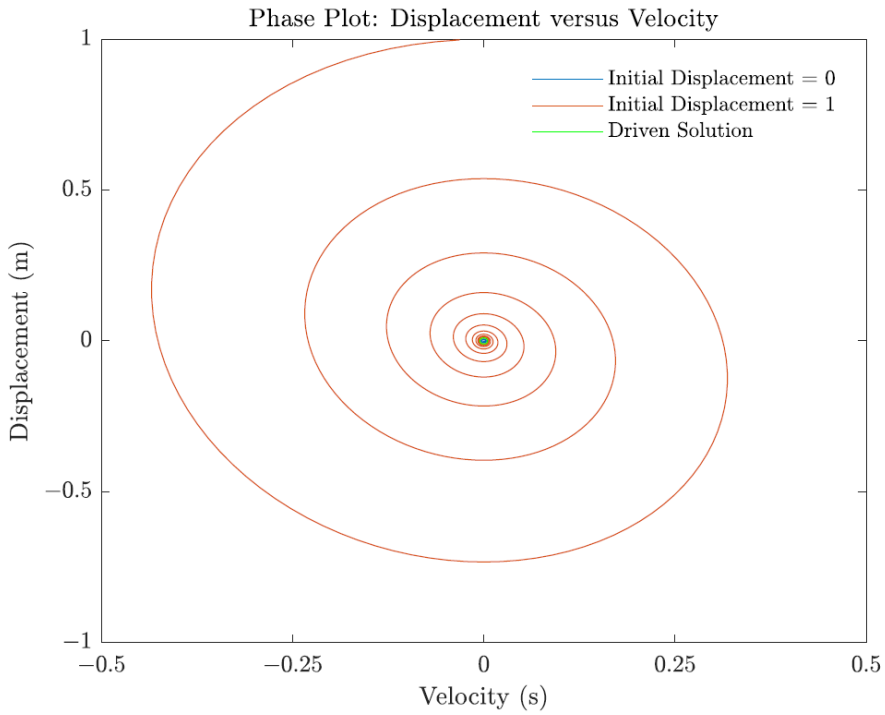

Figure 12: General Plot of Solutions

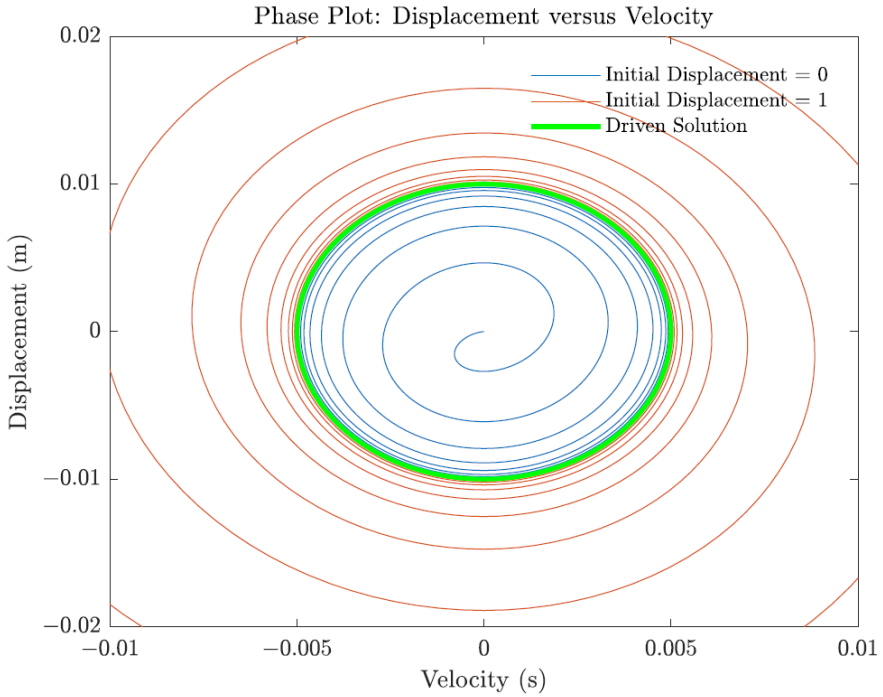

Figure 13: Focused Plot of Solutions

2.4 Energy

There are two types of energy to consider, potential energy and kinetic energy, where the total energy in the spring is the sum of both types. The potential energy is given by the equation

$$U_P = \frac{kx^2}{2} \tag{6}$$

and the kinetic energy is given by

$$U_K = \frac{mv^2}{2} \tag{7}$$

By substituting in the solutions for x and v the total energy can be determined. Focusing on the initial conditions case where the initial displacement is 1, the MATLAB plot for the the potential starting and kinetic energy is shown below.

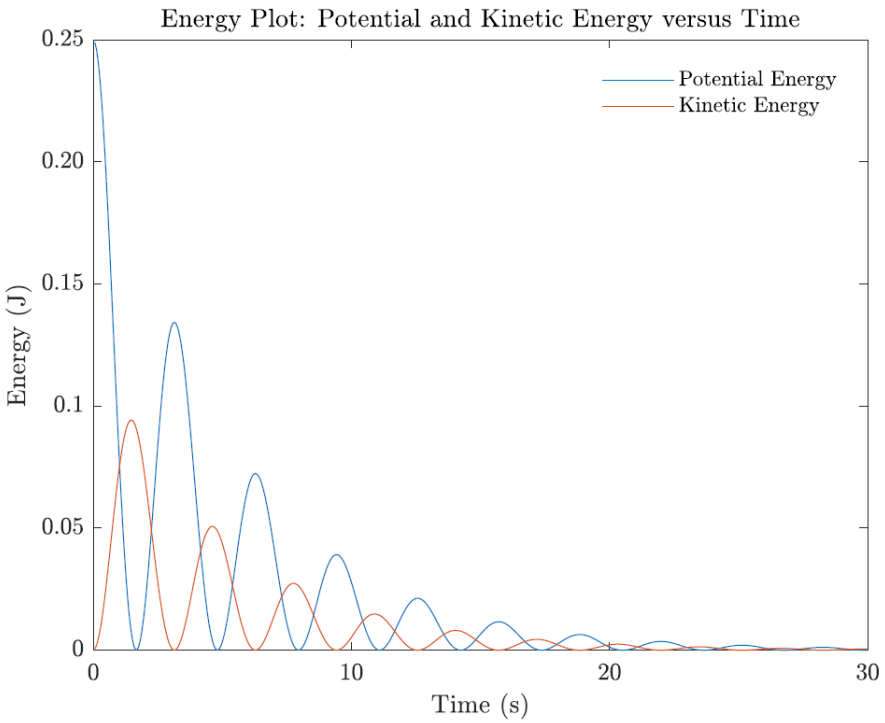

Figure 14: Energy Plot for Initial Condition Case Displacement = 1 m