CHAPTER 6

STATE SPACE:
FREQUENCY RESPONSE, TIME DOMAIN

6.1 Introduction – Frequency Response

This chapter will begin with the state space form of the equations of motion. We will use Laplace transforms to define the transfer function matrix. Next we will solve for the closed form transfer function matrix of the undamped tdof model using a symbolic algebra program and compare the answer with the solution presented in Chapter 2. MATLAB code will be used to set up frequency response calculations, using the full system matrix which allows the user to define damping values.

6.2 Solving for Transfer Functions in State Space Form Using Laplace Transforms

Starting with the complete set of state space equations:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathrm{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathrm{u} \end{aligned} \tag{6.1}$$

Ignoring initial conditions to solve for steady state frequency response, take the matrix Laplace transform of the state equation and solve for $\mathbf{x}(s)$ (Appendix 2):

$$s\mathbf{I}\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathrm{u}(s) \tag{6.2}$$

$$(s\mathbf{I} - \mathbf{A})\mathbf{x}(s) = \mathbf{B}\mathrm{u}(s) \tag{6.3}$$

$$\mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\ \mathbf{B}\mathrm{u}(s) \tag{6.4}$$

Substituting into the Laplace transform of the output equation:

$$\mathbf{y}(s) = \mathbf{C}\ (s\mathbf{I} - \mathbf{A})^{-1}\ \mathbf{B}\mathrm{u}(s) + \mathbf{D}\mathrm{u}(s) \tag{6.5}$$

Solving for the transfer function $\dfrac{\mathbf{y}(s)}{\mathrm{u}(s)}$:

$$\frac{\mathbf{y}(s)}{u(s)} = \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D} \tag{6.6}$$

Checking consistency of sizes

$$\begin{aligned} nx1 &= (nxn)x(nxn)x(nx1) + (nx1) \\ &= nx1 \end{aligned} \tag{6.7}$$

Letting $m_1 = m_2 = m_3 = m$, $k_1 = k_2 = k_3 = k$, $c_1 = c_2 = 0$ and rewriting the matrix equations of motion to match the original undamped problem used in Section 2.4.3 allows calculation of results by hand. The MATLAB code which follows, however, will allow **any** values to be used for the individual masses, dampers and stiffnesses.

$$(s\mathbf{I} - \mathbf{A}) = s\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$- \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-k_1}{m_1} & \frac{-c_1}{m_1} & \frac{k_1}{m_1} & \frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{c_1}{m_2} & \frac{-(k_1+k_2)}{m_2} & \frac{-(c_1+c_2)}{m_2} & \frac{k_2}{m_2} & \frac{c_2}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{m_3} & \frac{c_2}{m_3} & \frac{-k_2}{m_3} & \frac{-c_2}{m_3} \end{bmatrix}$$

$$= \begin{bmatrix} s & -1 & 0 & 0 & 0 & 0 \\ \frac{k_1}{m_1} & s+\frac{c_1}{m_1} & \frac{-k_1}{m_1} & \frac{-c_1}{m_1} & 0 & 0 \\ 0 & 0 & s & -1 & 0 & 0 \\ \frac{-k_1}{m_2} & \frac{-c_1}{m_2} & \frac{(k_1+k_2)}{m_2} & s+\frac{(c_1+c_2)}{m_2} & \frac{-k_2}{m_2} & \frac{-c_2}{m_2} \\ 0 & 0 & 0 & 0 & s & -1 \\ 0 & 0 & \frac{-k_2}{m_3} & \frac{-c_2}{m_3} & \frac{k_2}{m_3} & s+\frac{c_2}{m_3} \end{bmatrix}$$

$$= \begin{bmatrix} s & -1 & 0 & 0 & 0 & 0 \\ \frac{k}{m} & s & \frac{-k}{m} & 0 & 0 & 0 \\ 0 & 0 & s & -1 & 0 & 0 \\ \frac{-k}{m} & 0 & \frac{2k}{m} & s & \frac{-k}{m} & 0 \\ 0 & 0 & 0 & 0 & s & -1 \\ 0 & 0 & \frac{-k}{m} & 0 & \frac{k}{m} & s \end{bmatrix} \tag{6.8}$$

Here, in order to develop the entire 3x3 transfer function matrix, we will use a MIMO representation of **B** and **C**.

Taking **B** equal to the 6x3 matrix gives transfer functions for all three forces:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 1/m_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/m_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/m_3 \end{bmatrix} \tag{6.9}$$

Taking **C** equal to the 3x6 matrix below gives the three displacement transfer functions as outputs:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.10}$$

6.3 Transfer Function Matrix

Now that we have the terms required, we can substitute into the equation for the transfer function matrix:

$$\frac{\mathbf{y}(s)}{u(s)} = \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D} \tag{6.11}$$

We have an expression for $(s\mathbf{I} - \mathbf{A})$ above, but need to have its inverse. Using a symbolic algebra program to calculate the inverse even for this relatively small 3x3 problem yields a result which is too lengthy to be listed here in its entirety. To show that the calculation by hand really works, however, we will expand the equation above symbolically and then substitute the appropriate terms from the inverse to give the results for several of the transfer functions. We will refer to the $(s\mathbf{I} - \mathbf{A})^{-1}$ matrix by the notation "sia" and expand it as follows:

$$\frac{\mathbf{y}(s)}{u(s)} = \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} siai_{11} & siai_{12} & siai_{13} & siai_{14} & siai_{15} & siai_{16} \\ siai_{21} & siai_{22} & siai_{23} & siai_{24} & siai_{25} & siai_{26} \\ siai_{31} & siai_{32} & siai_{33} & siai_{34} & siai_{35} & siai_{36} \\ siai_{41} & siai_{42} & siai_{43} & siai_{44} & siai_{45} & siai_{46} \\ siai_{51} & siai_{52} & siai_{53} & siai_{54} & siai_{55} & siai_{56} \\ siai_{61} & siai_{62} & siai_{63} & siai_{64} & siai_{65} & siai_{66} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/m & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/m \end{bmatrix}$$

$$= \begin{bmatrix} siai_{11} & siai_{12} & siai_{13} & siai_{14} & siai_{15} & siai_{16} \\ siai_{31} & siai_{32} & siai_{33} & siai_{34} & siai_{35} & siai_{36} \\ siai_{51} & siai_{52} & siai_{53} & siai_{54} & siai_{55} & siai_{56} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/m & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/m \end{bmatrix}$$

$$= \begin{bmatrix} siai_{12}/m & siai_{14}/m & siai_{16}/m \\ siai_{32}/m & siai_{34}/m & siai_{36}/m \\ siai_{52}/m & siai_{54}/m & siai_{56}/m \end{bmatrix} \tag{6.12}$$

Listing the values for the $siai_{xx}$ terms used above from the symbolic algebra solution:

$$siai_{12} = siai_{56} = (m^3s^4 + 3m^2ks^2 + mk^2)/Den$$
$$siai_{32} = siai_{14} = siai_{54} = siai_{36} = (m^2ks^2 + mk^2)/Den$$
$$siai_{34} = (m^3s^4 + 2m^2ks^2 + mk^2)/Den$$
$$siai_{52} = siai_{16} = mk^2/Den$$

$$\text{where } Den = s^2(m^3s^4 + 4m^2ks^2 + 3mk^2)$$

(6.13a-e)

Dividing each of the above terms by "m" and presenting in the transfer function matrix form of (2.61):

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \frac{\begin{bmatrix} (m^2s^4 + 3mks^2 + k^2) & (mks^2 + k^2) & k^2 \\ (mks^2 + k^2) & (m^2s^4 + 2mks^2 + k^2) & (mks^2 + k^2) \\ k^2 & (mks^2 + k^2) & (m^2s^4 + 3mks^2 + k^2) \end{bmatrix}}{s^2\left(m^3s^4 + 4m^2ks^2 + 3mk^2\right)} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \tag{6.14}$$

The two derivations are identical.

6.4 MATLAB Code tdofss.m – Frequency Response Using State Space

6.4.1 Code Description, Plot

The four distinct transfer functions for the default values of m, k and c are plotted using MATLAB in **tdofss.m**, listed below. The four plots are displayed in Figure 6.1. The **A**, **B**, **C** and **D** matrices shown in (5.17a) are used as inputs to the program. A MIMO state space model is constructed and the MATLAB function bode.m is used to calculate the magnitude and phase of the resulting frequency responses. As described in the code, the resulting frequency response has dimensions of 6x3x200, where the "6" represents the 6 outputs in the output matrix **C**, the "3" represents the three columns of the input matrix **B** and the "200" represents the 200 frequency points in the frequency vector. The desired magnitude and phase can be extracted from the 6x3x200 matrix by defining the appropriate indices. The default values of c1 and c2 are zero.

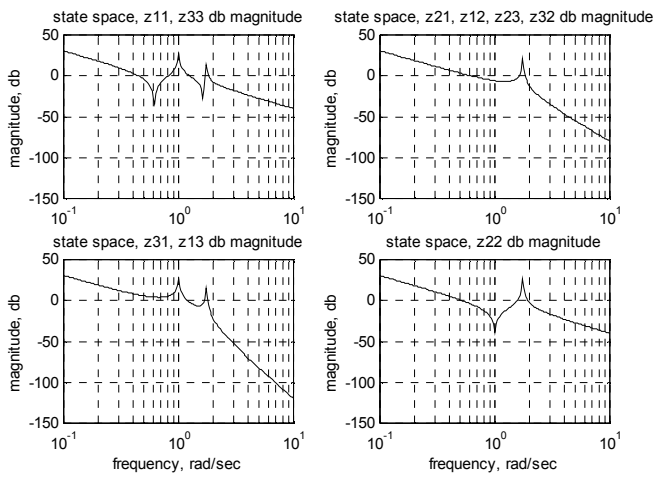

Figure 6.1: Four distinct frequency response amplitudes.

6.4.2 Code Listing

```
%       tdofss.m    state-space transfer function solution of tdof undamped model using
%       state-space matrices directly and the bode command

        clf;

        legend off;

        subplot(1,1,1);
```

```
            clear all;

%           define the values of masses, springs, dampers and Forces

            m1 = 1;
            m2 = 1;
            m3 = 1;

            c1 = input('input value for c1, default 0, ...  ');

            if  (isempty(c1))
                        c1 = 0;
            else
            end

            c2 = input('input value for c2, default 0, ...  ');

            if  (isempty(c2))
                        c2 = 0;
            else
            end

            k1 = 1;
            k2 = 1;

            F1 = 1;
            F2 = 1;
            F3 = 1;

%           define the system matrix, a

            a = [     0         1            0              0           0          0
                  -k1/m1   -c1/m1       k1/m1          c1/m1          0          0
                     0         0            0              1           0          0
                   k1/m2     c1/m2   -(k1+k2)/m2   -(c1+c2)/m2     k2/m2      c2/m2
                     0         0            0              0           0          1
                     0         0          k2/m3          c2/m3     -k2/m3     -c2/m3];

%           define the input matrix, b, a 6x3 matrix

            b = [     0         0          0
                    F1/m1       0          0
                      0         0          0
                      0       F2/m2        0
                      0         0          0
                      0         0        F3/m3];

%           define the output matrix, c, the 6x6 identify matrix

            c = eye(6,6);

%           define the direct transmission matrix

            d = 0;

%           solve for the eigenvalues of the system matrix
```

```
        [xm,omega] = eig(a);

%       Define a vector of frequencies to use, radians/sec.  The logspace command uses
%       the log10 value as limits, i.e. -1 is 10^-1 = 0.1 rad/sec, and 1 is
%       10^1 = 10 rad/sec.  The 200 defines 200 frequency points.

        w = logspace(-1,1,200);

%       use the "ss" function to define state space system for three inputs, forces at
%       masses 1, 2 and 3 and for all 6 states, three displacements and three velocities

        sssys = ss(a,b,c,d);

%       use the bode command with left hand magnitude and phase vector arguments
%       to provide values for further analysis/plotting

%       the mag and phs matrices below will be 6x3x200 in size
%       the appropriate magnitude and phase to plot for each transfer function
%       are called by appropriate indexing

%       first index 1-6:  z1 z1dot z2 z2dot z3 z3dot
%       second index 1-3:  F1  F2  F3
%       third index 1-200:  all frequency points, use ":"

        [mag,phs] = bode(sssys,w);

        z11mag = mag(1,1,:);
        z11phs = phs(1,1,:);

        z21mag = mag(3,1,:);
        z21phs = phs(3,1,:);

        z31mag = mag(5,1,:);
        z31phs = phs(5,1,:);

        z22mag = mag(3,2,:);
        z22phs = phs(3,2,:);

%       calculate the magnitude in decibels, db

        z11magdb = 20*log10(z11mag);

        z21magdb = 20*log10(z21mag);

        z31magdb = 20*log10(z31mag);

        z22magdb = 20*log10(z22mag);

%       plot the four transfer functions separately, in a 2x2 subplot form

        subplot(2,2,1)
        semilogx(w,z11magdb(1,:),'k-')
        title('state space, z11, z33 db magnitude')
        ylabel('magnitude, db')
        axis([.1 10 -150 50])
        grid
```

```
subplot(2,2,2)
semilogx(w,z21magdb(1,:),'k-')
title('state space, z21, z12, z23, z32 db magnitude')
ylabel('magnitude, db')
axis([.1 10 -150 50])
grid

subplot(2,2,3)
semilogx(w,z31magdb(1,:),'k-')
title('state space, z31, z13 db magnitude')
xlabel('frequency, rad/sec')
ylabel('magnitude, db')
axis([.1 10 -150 50])
grid

subplot(2,2,4)
semilogx(w,z22magdb(1,:),'k-')
title('state space, z22 db magnitude')
xlabel('frequency, rad/sec')
ylabel('magnitude, db')
axis([.1 10 -150 50])
grid

disp('execution paused to display figure, "enter" to continue'); pause

subplot(2,2,1)
semilogx(w,z11phs(1,:),'k-')
title('state space, z11, z33 phase')
ylabel('phase, deg')
%axis([.1 10 -400 -150])
grid

subplot(2,2,2)
semilogx(w,z21phs(1,:),'k-')
title('state space, z21, z12, z23, z32 phase')
ylabel('phase, deg')
%axis([.1 10 -400 -150])
grid

subplot(2,2,3)
semilogx(w,z31phs(1,:),'k-')
title('state space, z31, z13 phase')
xlabel('frequency, rad/sec')
ylabel('phase, deg')
%axis([.1 10 -400 -150])
grid

subplot(2,2,4)
semilogx(w,z22phs(1,:),'k-')
title('state space, z22 phase')
xlabel('frequency, rad/sec')
ylabel('phase, deg')
%axis([.1 10 -400 -150])
grid

disp('execution paused to display figure, "enter" to continue'); pause
```

6.5 Introduction – Time Domain

Starting with the equations of motion in state space, we will use Laplace transforms to discuss the theoretical solution to the time domain problem. We will define and discuss two methods of calculating the matrix exponential. Then we will use a sdof forced system with position and velocity initial conditions to illustrate the technique. The closed form solution for our tdof example problem with step forces applied to all three masses and with different initial conditions for each mass is too complicated to be shown so we will use only MATLAB for its solution.

6.6 Matrix Laplace Transform – with Initial Conditions

We start with the state equations in general form, (6.1). Taking the matrix Laplace transform of a first order differential equation (DE) with initial conditions (Appendix 2):

$$\begin{aligned} \mathcal{L}\{\dot{\mathbf{x}}(t)\} &= s\mathbf{x}(s) - \mathbf{x}(0) \\ \mathcal{L}\{\mathbf{x}(t)\} &= \mathbf{x}(s) \end{aligned} \tag{6.15}$$

Taking the matrix Laplace transform of (6.1) and solving for $\mathbf{x}(s)$:

$$\begin{aligned} s\mathbf{x}(s) - \mathbf{x}(0) &= \mathbf{A}\mathbf{x}(s) + \mathbf{B}\,u(s) \\ (s\mathbf{I} - \mathbf{A})\mathbf{x}(s) &= \mathbf{x}(0) + \mathbf{B}\,u(s) \\ \mathbf{x}(s) &= (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,u(s) \end{aligned} \tag{6.16a,b,c}$$

Solving for the output vector $\mathbf{y}(s)$:

$$\begin{aligned} \mathbf{y}(s) &= \mathbf{C}\,\mathbf{x}(s) \\ &= \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,u(s) \end{aligned} \tag{6.17}$$

The input matrix $\mathbf{B}$ and output matrix $\mathbf{C}$ are familiar from earlier state space presentations. There is a new term in the equation for the Laplace transform of $\mathbf{y}(s)$, the term $(s\mathbf{I} - \mathbf{A})^{-1}$.

There are many methods of calculating the inverse $(s\mathbf{I} - \mathbf{A})^{-1}$ (Chen 1999). If the problem is small, for example 2x2, the inverse can be handled in closed form. Then $\mathbf{y}(s)$ can be back-transformed term by term to get the solution in the time domain, as we shall see in the example in the next section.

For another solution method it is useful to recall the geometric series expansion below, for $|r| < 1$:

$$\frac{1}{1-r} = 1 + r + r^2 + r^3 + \ldots \tag{6.18}$$

Expanding $(s\mathbf{I} - \mathbf{A})^{-1}$ with the series expansion analogy above, the inverse results in the infinite series in (6.19).

$$\begin{aligned}(s\mathbf{I} - \mathbf{A})^{-1} &= \frac{1}{s\mathbf{I} - \mathbf{A}} = \frac{\frac{1}{s}}{\mathbf{I} - \frac{\mathbf{A}}{s}} = \frac{1}{s}\left[\mathbf{I} + \frac{\mathbf{A}}{s} + \frac{\mathbf{A}^2}{s^2} + \frac{\mathbf{A}^3}{s^3} + \ldots\right] \\ &= \frac{\mathbf{I}}{s} + \frac{\mathbf{A}}{s^2} + \frac{\mathbf{A}^2}{s^3} + \frac{\mathbf{A}^3}{s^4} + \ldots\end{aligned} \tag{6.19}$$

6.7 Inverse Matrix Laplace Transform, Matrix Exponential

Now that we have the inverse in series form, it is easy to back-transform to the time domain, term by term. We introduce two new terms, $\mathbf{\Phi}(t)$, the **inverse Laplace transform** of $(s\mathbf{I} - \mathbf{A})^{-1}$ which equals $e^{\mathbf{A}t}$, the **matrix exponential**.

$$\begin{aligned}\mathbf{\Phi}(t) &= \mathcal{L}^{-1}\left\{(s\mathbf{I} - \mathbf{A})^{-1}\right\} \\ &= \mathcal{L}^{-1}\left\{\frac{\mathbf{I}}{s} + \frac{\mathbf{A}}{s^2} + \frac{\mathbf{A}^2}{s^3} + \frac{\mathbf{A}^3}{s^4} + \ldots\right\} \\ &= \mathbf{I} + \mathbf{A}t + \frac{(\mathbf{A}t)^2}{2!} + \frac{(\mathbf{A}t)^3}{3!} + \ldots \\ &= e^{\mathbf{A}t}\end{aligned} \tag{6.20}$$

6.8 Back-Transforming to Time Domain

Now that the form of the matrix exponential is known, we can back-transform the entire equation of motion, from (6.16c):

$$\mathcal{L}^{-1}(\mathbf{x}(s)) = \mathcal{L}^{-1}\left[(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,u(s)\right] \tag{6.21}$$

The result is:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\,\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\,\mathbf{B}\,u(\tau)\,d\tau \qquad (6.22)$$

The first term in (6.22) is the response due to the initial condition of the state and the second term is the response due to the forcing function. The second term is the **convolution integral**, or **Duhamel integral**, and results from back-transforming the product of two Laplace transforms.

6.9 Single Degree of Freedom System – Calculating Matrix Exponential in Closed Form

Calculating the matrix exponential in closed form for greater than a 2x2 matrix is difficult without the aid of a symbolic algebra program. Even with the program the result can be quite complicated.

A simple, rigid body example will be used to demonstrate how a matrix exponential and transient response are calculated.

We will use the system in Figure 6.2, a mass with position and velocity initial conditions and a step force applied.

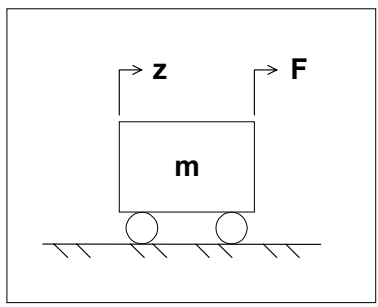

Figure 6.2: sdof system with initial conditions and step force applied.

6.9.1 Equations of Motion, Laplace Transform

Start with the equation of motion:

$$m\ddot{z} = F \qquad (6.23)$$

Defining the states:

$$\begin{aligned} x_1 &= z \\ x_2 &= \dot{z} \end{aligned} \qquad (6.24)$$

Defining derivatives and inserting the value for acceleration:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{F}{m} \end{aligned} \tag{6.25}$$

The above can be written in matrix form, recognizing that F/m is the acceleration and applying a unity magnitude step:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \left(\frac{F}{m}\right) \end{bmatrix} (1) \tag{6.26}$$

Defining the system matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{6.27}$$

Taking the inverse of the $(s\mathbf{I} - \mathbf{A})^{-1}$ term:

$$(s\mathbf{I} - \mathbf{A})^{-1} = \left(\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \right)^{-1} = \begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s^2} \\ 0 & \frac{1}{s} \end{bmatrix} \tag{6.28}$$

6.9.2 Defining the Matrix Exponential – Taking Inverse Laplace Transform

Using the table of inverse Laplace transforms from Appendix 2 yields the matrix exponential.

$$e^{\mathbf{A}t} = \mathcal{L}^{-1} \left\{ \begin{bmatrix} \frac{1}{s} & \frac{1}{s^2} \\ 0 & \frac{1}{s} \end{bmatrix} \right\} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \tag{6.29}$$

6.9.3 Defining the Matrix Exponential – Using Series Expansion

A Power Series Expansion can also be used to find the matrix exponential for this simple example because higher powers of $\mathbf{A}t$ go to zero quickly:

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{(\mathbf{A}t)^2}{2!} + \frac{(\mathbf{A}t)^3}{3!} + \ldots$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + (\text{all other terms zero}) \qquad (6.30)$$

$$= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

This is the same solution as (6.29).

6.9.4 Solving for Time Domain Response

Thus, the general solution for x(t) as a function of time becomes:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\,\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\,\mathbf{B}\,u(t)\,d\tau$$

$$= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} + \int_0^t \begin{bmatrix} 1 & t-\tau \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ \left(\frac{F}{m}\right) \end{bmatrix}(1)\ d\tau$$

$$= \begin{bmatrix} x_1(0) + t\,x_2(0) \\ x_2(0) \end{bmatrix} + \int_0^t \begin{bmatrix} (t-\tau)\left(\frac{F}{m}\right) \\ \left(\frac{F}{m}\right) \end{bmatrix} d\tau$$

$$= \begin{bmatrix} x_1(0) + t\,x_2(0) \\ x_2(0) \end{bmatrix} + \left\{ \begin{bmatrix} \left(t\tau - \frac{\tau^2}{2}\right)\left(\frac{F}{m}\right) \\ \left(\frac{F}{m}\right)\tau \end{bmatrix} \right\}\Bigg|_0^t$$

$$= \begin{bmatrix} x_1(0) + t\,x_2(0) \\ x_2(0) \end{bmatrix} + \begin{bmatrix} \left(t^2 - \frac{t^2}{2}\right)\left(\frac{F}{m}\right) \\ t\left(\frac{F}{m}\right) \end{bmatrix}$$

$$= \begin{bmatrix} x_1(0) + t\,x_2(0) \\ x_2(0) \end{bmatrix} + \begin{bmatrix} \left(\frac{t^2}{2}\right)\left(\frac{F}{m}\right) \\ t\left(\frac{F}{m}\right) \end{bmatrix} \tag{6.31}$$

This result is the same as the familiar equations for the position and velocity of a mass undergoing a constant acceleration:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \text{initial position} + \text{time} \times (\text{initial velocity}) + \frac{(\text{acceleration}) \times (\text{time}^2)}{2} \\ \text{initial velocity} + (\text{acceleration}) \times (\text{time}) \end{bmatrix} \tag{6.32}$$

6.10 MATLAB Code tdof_ss_time_ode45_slnk.m – Time Domain Response of tdof Model

6.10.1 Equations of Motion Review

There are several ways to numerically solve for transient responses using MATLAB. One method uses numerical integration, calling the integration routine from a command line and defining the state equation in a separate MATLAB function. Another method uses Simulink, a linear/nonlinear graphical block diagram model building tool linked to MATLAB.

We will solve for the transient response of our tdof model using both methods and compare the results with the closed form solution calculated using the modal transient response method in Chapter 9.

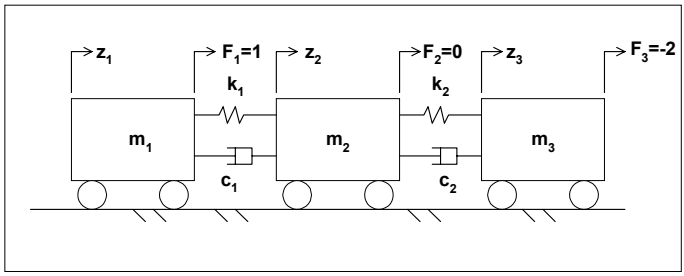

Figure 6.3: tdof model with damping for use in MATLAB/Simulink models.

$z_1(0) = x_1(0) = 0$	$z_2(0) = x_3(0) = -1$	$z_3(0) = x_5(0) = 1$
$\dot{z}_1(0) = x_2(0) = -1$	$\dot{z}_2(0) = x_4(0) = 2$	$\dot{z}_3(0) = x_6(0) = -2$

Table 6.1: Initial conditions for tdof model in Figure 6.3.

Step function forces of amplitudes indicated in Figure 6.3 are applied to masses 1 and 3; mass 2 has no force applied. Initial conditions of position and velocity for each mass are shown in Table 6.1.

The equations of motion in state space are then:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-k_1}{m_1} & \frac{-c_1}{m_1} & \frac{k_1}{m_1} & \frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{c_1}{m_2} & \frac{-(k_1+k_2)}{m_2} & \frac{-(c_1+c_2)}{m_2} & \frac{k_2}{m_2} & \frac{c_2}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{m_3} & \frac{c_2}{m_3} & \frac{-k_2}{m_3} & \frac{-c_2}{m_3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ \frac{0}{m_2} \\ 0 \\ \frac{-2}{m_3} \end{bmatrix} \quad (1)$$

(6.33)

The initial condition vector, x(0) is:

$$\mathbf{x}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \\ x_5(0) \\ x_6(0) \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \dot{z}_1(0) \\ z_2(0) \\ \dot{z}_2(0) \\ z_3(0) \\ \dot{z}_3(0) \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -1 \\ 2 \\ 1 \\ -2 \end{bmatrix} \tag{6.34}$$

The output equation for the displacement outputs (no velocities included) with no feedthrough term is:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + (0)(1) \tag{6.35}$$

These are the system matrices that are used in the MATLAB code below.

6.10.2 Code Description

Two methods will be used to solve for the time domain response. The MATLAB code **tdof_ss_time_ode45_slnk.m** is used for both methods, prompting the user to define which solution technique is desired.

The first method uses the MATLAB Runge Kutta method ODE45 and calls the function file **tdofssfun.m**, which contains the state equations. The results are then plotted. To use the ODE45 solver, type "tdof_ss_time_ode45_slnk" from the MATLAB prompt and use the default selection.

The second solution uses the Simulink model **tdof_ss_simulink.mdl** and the plotting file **tdof_ss_time_slnk_plot.m**.

To use the Simulink solver:

1) Type "tdof_ss_time_ode45_slnk" and choose the Simulink solver.

2) The program will prompt the reader to type "tdof_ss_simulink" at the MATLAB command prompt. This will bring up the Simulink model on the screen.

3) Click on the "simulation" choice in the model screen and then choose "start." The Simulink model will then run.

4) To see the plotted results, type "tdof_ss_time_slnk_plot."

6.10.3 Code Results – Time Domain Responses

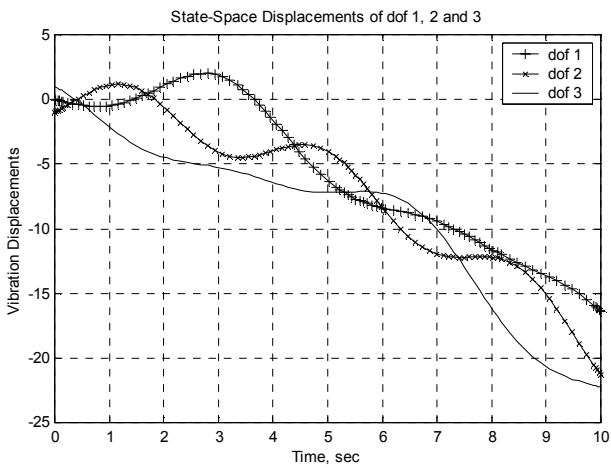

Figure 6.4: ODE45 simulation motion of tdof model.

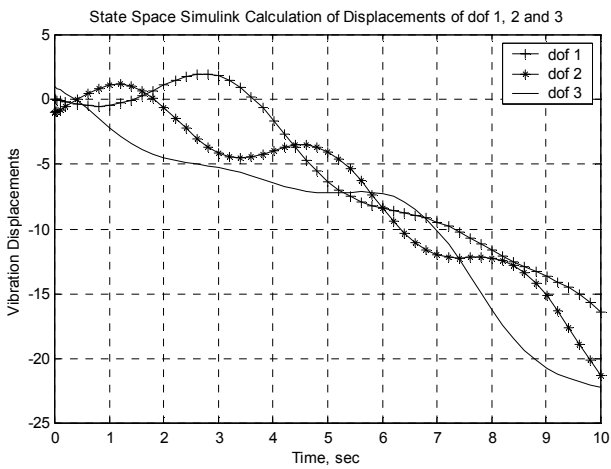

Figure 6.5: Simulink simulation motion of tdof model.

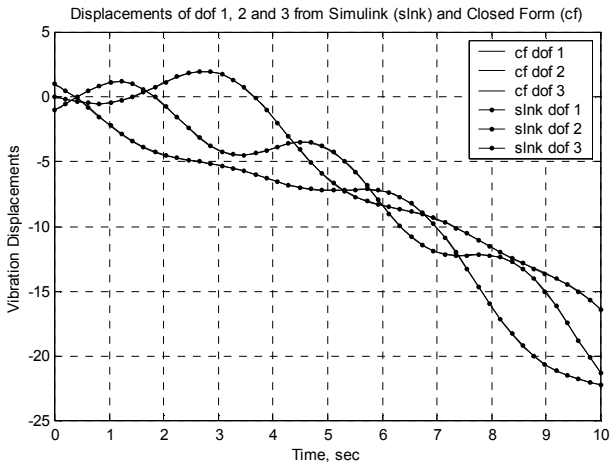

Figure 6.6: Overlay of closed form solution from Chapter 9, Figure 9.4, with Simulink solution.

6.10.4 Code Listing

```
%       tdof_ss_time_ode45_slnk.m       state-space solution of tdof model with
%       initial conditions, step function forcing function and displacement outputs
%       using the ode45 solver or Simulink, user is prompted for damping values

        clear all;

        global a b u            %       this is required to have the parameters available
                                %       for the function

        which_run = input('enter "1" for Simulink or "enter" for ode45 run ...  ');

        if  isempty(which_run)
                which_run = 0
        end

%       define the values of masses, springs, dampers and Forces

        m1 = 1;
        m2 = 1;
        m3 = 1;

        c1 = input('input value for c1, default 0.0, ...  ');

        if  (isempty(c1))
                c1 = 0.0;
        else
        end

        c2 = input('input value for c2, default 0.0, ...  ');
```

```
        if  (isempty(c2))
                c2 = 0.0;
        else
        end

        k1 = 1;
        k2 = 1;

        F1 = 1;
        F2 = 0;
        F3 = -2;

%       define the system matrix, a

        a = [  0          1          0          0          0          0
          -k1/m1     -c1/m1      k1/m1      c1/m1         0          0
               0          0          0          1          0          0
           k1/m2      c1/m2   -(k1+k2)/m2  -(c1+c2)/m2   k2/m2      c2/m2
               0          0          0          0          0          1
               0          0       k2/m3      c2/m3     -k2/m3     -c2/m3];

%       define the input matrix, b

        b = [  0
             F1/m1
               0
             F2/m2
               0
            F3/m3];

%       define the output matrix for transient response, c, displacements only

        c = [1 0 0 0 0 0
             0 0 1 0 0 0
             0 0 0 0 1 0];

%       define the direct transmission matrix for transient response, d, the same number of
        rows as c and the same number of columns as b

        d = zeros(3,1);

        if  which_run == 0      %           transient response using the ode45 command


        u = 1;

        ttotal = input('Input total time for Simulation, default = 10 sec, ...  ');

        if (isempty(ttotal))
        ttotal = 10;
        else
        end

        tspan = [0 ttotal];

        x0 = [0 -1 -1 2 1 -2]';                 %  initial condition vector, note transpose
```

```
        options = [];                       %  no options specified for ode45 command

        [t,x] = ode45('tdofssfun',tspan,x0,options);

        y = c*x';                   % note transpose, x is calculated as a column vector in time

        plot(t,y(1,:),'k+-',t,y(2,:),'kx-',t,y(3,:),'k-')
        title('State-Space Displacements of dof 1, 2 and 3')
        xlabel('Time, sec')
        ylabel('Vibration Displacements')
        legend('dof 1','dof 2','dof 3')
        grid

        else        %  setup Simulink run

%       define the direct transmission matrix for transient response, d, the same number of
        rows as c and the same number of columns as b

%       define time for simulink model

        ttotal = input('Input total time for Simulation, default = 10 sec, ...  ');

        if (isempty(ttotal))
        ttotal = 10;
        else
        end

        disp('  ');
        disp('  ');
        disp('  ');
        disp('  ');
        disp('  ');
        disp('  ');
        disp('Run the Simulink model "tdof_ss_simulink.mdl" and then');
        disp('run the plotting file "tdof_ss_time_slnk_plot.m"');

        end
```

6.10.5 MATLAB Function tdofssfun.m – Called by tdof_ss_time_ode45_slnk.m

```
        function xprime = tdofssfun(t,x)

%       function for calculating the transient response of tdof_ss_time_ode45.m

        global  a b u

        xprime = a*x + b*u;
```

6.10.6 Simulink Model tdofss_simulink.mdl

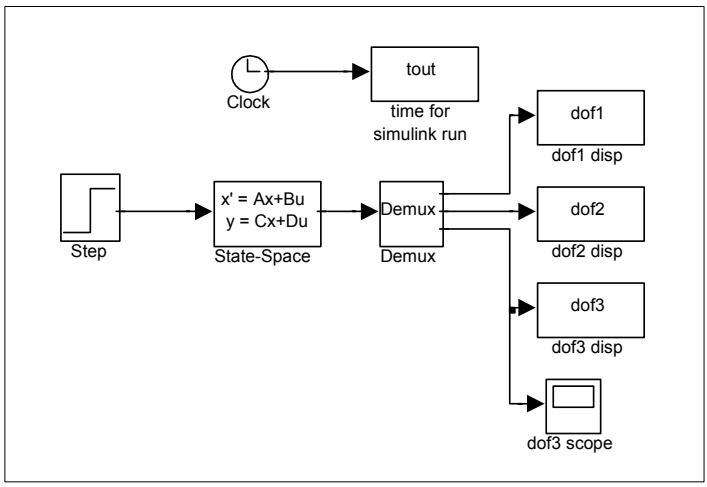

Figure 6.7: Block diagram of Simulink model tdofss_simulink.mdl.

The block diagram was constructed by dragging and dropping blocks from the appropriate Simulink block library and connecting the blocks. The input is the step block. The clock block is used to output time to the tout block for plotting in MATLAB. The model is defined in the state space block, reading in values for the a, b, c and d matrices from the MATLAB workspace, created during execution of **tdof_ss_time_ode45_slnk.m**. The demux block separates the vector output of the state space block and sends the displacements of the three masses to three blocks for storing for plotting in MATLAB. The scope block brings up a scope screen and shows the position of dof3 versus time as the program executes. This example is so small that the screen displays instantly for the default 10 sec time period, but for a longer time period the scope traces the progress of the simulation.

Problems

Note: All the problems refer to the two dof system shown in Figure P2.2.

P6.1 Set $m_1 = m_2 = m = 1$, $k_1 = k_2 = k = 1$, $c_1 = c_2 = 0$ and define the state space matrices for a step force applied to mass 1 and for output of position of mass 2. Write out by hand the equation for the transfer functions matrix as shown in (6.11). Extra credit: use a symbolic algebra program to take the inverse of the $(s\mathbf{I} - \mathbf{A})$ term and then multiply out the equations to see that they match the results of P2.2.

P6.2 (MATLAB) Modify the code **tdofss.m** for the two dof system and plot the distinct frequency responses.

P6.3 (MATLAB) Modify the code **tdof_ss_time_ode45_slnk.m** for the two dof system with $m_1 = m_2 = m = 1$, $k_1 = k_2 = k = 1$ and $c_1 = c_2 = 0$ for the following step forces and initial conditions:

a) $F_1 = 0, F_2 = -3$

b) $z_1 = 0, \dot{z}_1 = -2, z_2 = -1, \dot{z}_2 = 2$

Plot the time domain responses using both MATLAB and Simulink.