i6.M APPENDIX: TRANSIENT RESPONSES USING MATLAB

The simplest way to generate a step response with MATLAB is to use the step function, `step(NUM,DEN)`. Here `NUM` and `DEN` are vectors as discussed in Section 5.M but this time they are part of the closed-loop transfer functions, *H(s)* or 1-*H(s)*, whereas they were part of *G(s)* in Section 5.M.

6.M.1 `step` Program with Simple Vectors

If, in Eq. (6.4), we let $\zeta = 0.5$, $\omega_n = 1$, and set α either to zero or one, then all of the coefficients of *H(s)* and 1-*H(s)* are either 0 or 1. This is of no fundamental importance but it does lead to the simple vectors used in the scripts for error and output response with $\alpha = 0$, `ErTrn0a` and `OutTrn0a`, and with $\alpha = 1$, `ErTrn1a` and `OutTrn1a,` as illustrated in the following program.

`ErTrn1a`

```
% ERROR TRANSIENT RESPONSE FOR
% alpha = 1, zeta = 0.5, Wn = 1
% Step, Ramp, or Parabolic inputs
% Choose response type by changing "den" below

num = [1 0 0];
denstep = [1 1 1];
denramp = [1 1 1 0];
denparab = [1 1 1 0 0];
t = linspace(0,10,100);
den = denstep;   %Set den to denstep, denramp, or denparab
step(num,den,t);
grid;
```

If we set $\alpha = 1$ in Eq. (6.4b), only the s^2 term exists in the numerator, corresponding to `num` from `ErTrn1a`. The denominator coefficients are all 1, as in `denstep` above. When we execute this file, with `den = denstep`, we obtain results as shown in Fig. 6.3 for $\alpha = 1$, $\zeta = 0.5$, $\omega_n = 1$.

If we integrate the step input to produce a ramp, the resulting output will be the integral of the step response. We can find it by multiplying 1-*H(s)* by 1/*s*, corresponding to a shifting of all the 1s in `denstep` to the left, producing `denramp`. When we execute the file with `den = denramp`, therefore, we obtain the ramp error response of Fig. 6.7 with $\alpha = 1$, $\zeta = 0.5$, $\omega_n = 1$.

Similarly, if we integrate again, the output will be the response to a parabola $t^2/2$, as shown in Fig. 6.10 with $\alpha = 1$, $\zeta = 0.5$, $\omega_n = 1$. Since the function `step` is automatically producing the plot, it will be labeled "Step Response," even when we are displaying the response to a ramp or parabola.

Of course we can obtain responses for other values of α, ζ and ω_n by changing NUM and DEN appropriately to represent Eq. (6.4).

6.M.2 Effect of Added Pole

The curves of Fig. 6.14 can be generated with a slight modification of the program given above. The open-loop transfer function for $\alpha = 1$ is Eq. (4.23). We are using $\omega_n = 1$ for simplicity and to normalize the time ($\omega_n t \Rightarrow t$). Since $\zeta = 0.5$ also, by Eq. 4.16 $\omega_z = 1$ and the open loop gain is

$$G(s) = \frac{1}{s^2}(1 + s). \tag{6.M.1}$$

With an added pole at

$$\omega_+ = \omega_n / r_n , \tag{6.M.2}$$

the open-loop transfer function becomes

$$G_+(s) = \frac{1}{s^2}\frac{1 + s}{1 + r_n s} , \tag{6.M.3}$$

leading to a close-loop transfer function of

$$H_+(s) = \frac{1}{1 + G_+(s)} = \frac{r_n s^3 + s^2}{r_n s^3 + s^2 + s + 1} . \tag{6.M.4}$$

The corresponding NUM and DEN vectors can be seen in the program that follows.

```
ExtrPole

% ERROR STEP RESPONSE WITH ADDED POLE
% Compare to Fig. 6.14
% alpha = 1, zeta = 0.5, Wn = 1
% rn: ratio of natural frequency to added pole

t = linspace(0,10,100);

clf; rn = 0
while (rn >= 0)
num = [rn 1 0 0];
den = [rn 1 1 1];
step(num,den,t);
grid;
```

```
rn = input('enter ratio of Wn to added pole (negative to
quit)')
end % while
```

6.M.3 Transient Response Program with Dynamic Entry

`Transnts` is a program that uses the techniques above to plot error responses to step, ramp, and parabolic inputs to a second-order loop while permitting the user to select parameters dynamically, that is the parameters (except for ω_n) or the input can be altered and a new plot obtained without starting to program over. Results will be as shown in the responses in Fig. 6.2 to 6.4 and 6.7 to 6.11 but α and ζ can be selected arbitrarily, not limited to the common values in the plots. Again, these plots will be labeled "Step Response" because of they are being produced by MATLAB's `step` function.

6.M.4 State Space

`SSstep` produces the same step response as does `ErTrn1a` above but employs MATLAB's ordinary state-space matrix method that is hidden by the `step` function used in `ErTrn1a`.

```
SSstep

%Develop error step response by standard matrix procedure
% alpha = 1, zeta = 0.5, Wn = 1
% Step Input

% Generate vector of computation times:
inc = .05; %sampling period
ending = 10;
last = 1 + ending/inc; % index of last data points
t = linspace(0,ending,last); % linear sequence of times
dt = t(2) - t(1); % time increment

% Generate input step vector:
u = ones(1,last); % unit step input; row vector with input
                    at each time

clf; % clear previous graphs

NUM = [1 0 0]; % for error response, as before
DEN = [1 1 1]; % denstep from ErTrn1a
[a,b,c,d] = tf2ss(NUM,DEN); % get elements of dynamic
              matrix equation (c is bottom row of C and d
              is bottom row of D in Eq. (6.44)
[Phi, Gamma] = C2D(a,b,dt); % Continuous-to-Discrete, gets
              values for use in solution equation
```

```
x = ltitr(Phi,Gamma,u'); % state variable solution for each
                value of t
y = c*x'+d*u;  % outputs. x' is the transpose of X in Eq.
                (6.33);  y is the lower element in Eq. (6.45)
                % x', y and u have a column for each time
                increment

plot(t,y)
grid
title('Time Response, Error, Step Input')
xlabel('seconds')
ylabel('radians')
```

To generate a ramp response we could change `DEN`, as we did in "`ErTrn1a`" but the more usual way is to change the driving function, which is enclosed in a box above. In `SSramp,` we replace that box by

```
for i = 1:last,
     u(i) = (i-1)*inc;
end % generation of input vector
```

This generates a vector whose elements are proportional to time. However, it represents a stepped (sampled and held) approximation to a ramp whereas, previously (e.g., in `ErTrn1a`), a stepped version of a step, which is still a step, was integrated within the response function to produce a time-continuous ramp, so the previous method is more accurate.

The reader might wish to generate a vector representing a parabolic input, $t^2/2$.

6.M.5 Adding Phase-Plane Output

`SSaddedP` performs the same function as `ExtrPole` but using the basic state space equations. We have also added a phase-plane plot, a plot of frequency versus phase. To get frequency, we must output the derivative of the phase so **Y** has two elements, φ and $\dot{\varphi}$. To accomplish this we give NUM two rows, first the numerator that will produce φ_{out} and second its derivative. Since taking the derivative is equivalent to multiplying by *s*, the orders of all of the powers of *s* are increased by one. This amounts to shifting the coefficients in the second row to the left, each to apply to a one-higher order of derivative (or of *s*). If we had retained the numerator for 1-*H*, as in the previous programs, the left-most coefficient, which was 1, would have shifted left, increasing the order of the vector and making it of higher order than DEN. This is not permitted so we changed from representing the error transfer function, 1-*H*, to the output transfer function, *H*. We did this by subtracting the previous NUM from DEN, equivalent to subtracting 1-*H* from 1. We will obtain the error response later by subtracting the output from the input.

```
SSaddedP

%ERROR STEP RESPONSE WITH ADDED POLE
% Develop output time response and
% linear phase plane by ordinary matrix procedure
% Compare to Fig. 6.14
% alpha = 1, zeta = 0.5, Wn = 1
% rn: ratio of natural frequency to added pole

% Generate vector of computation times
inc = .05; %sampling period
ending = 10;
last = 1 + ending/inc; % index of last data points
t = linspace(0,ending,last); % linear sequence of times

u = ones(1,last); % unit step input

rn = 0
while(rn>=0)
   NUM = [0 0 1 1   % first row is output (phase)
          0 1 1 0]; % second row is derivative of
                    output (frequency)
   DEN = [rn 1 1 1];
   [a,b,c,d] = tf2ss(NUM,DEN); % elements of dynamic matrix
                    equation
   [Phi, Gamma] = C2D(a,b,inc); % values for use in
                    solution eqaution
   x = ltitr(Phi,Gamma,u'); % solutions for each value of t
   y = c*x'+d*u; % outputs

 clf;
 er = u - y(1,:);
   subplot(211), plot(t,er,'g')
   grid
   title('Time Response, Error (linear)')
   xlabel('seconds')
   ylabel('radians')
   subplot(212), plot(y(1,:),y(2,:),'r')
   grid
   title('Phase Plane, Output (linear)')
   xlabel('phase (radians)')
   ylabel('freq (rad/sec)')

   rn = input('r: ratio of Wn to added pole (negative to
                    quit)')
end %while
```

The resulting graph for $r = 0.5$ is shown below. Note how the damped oscillations of the time response appear in the phase-plane plot. The phase plane-plot with $r = 1$ is especially interesting.

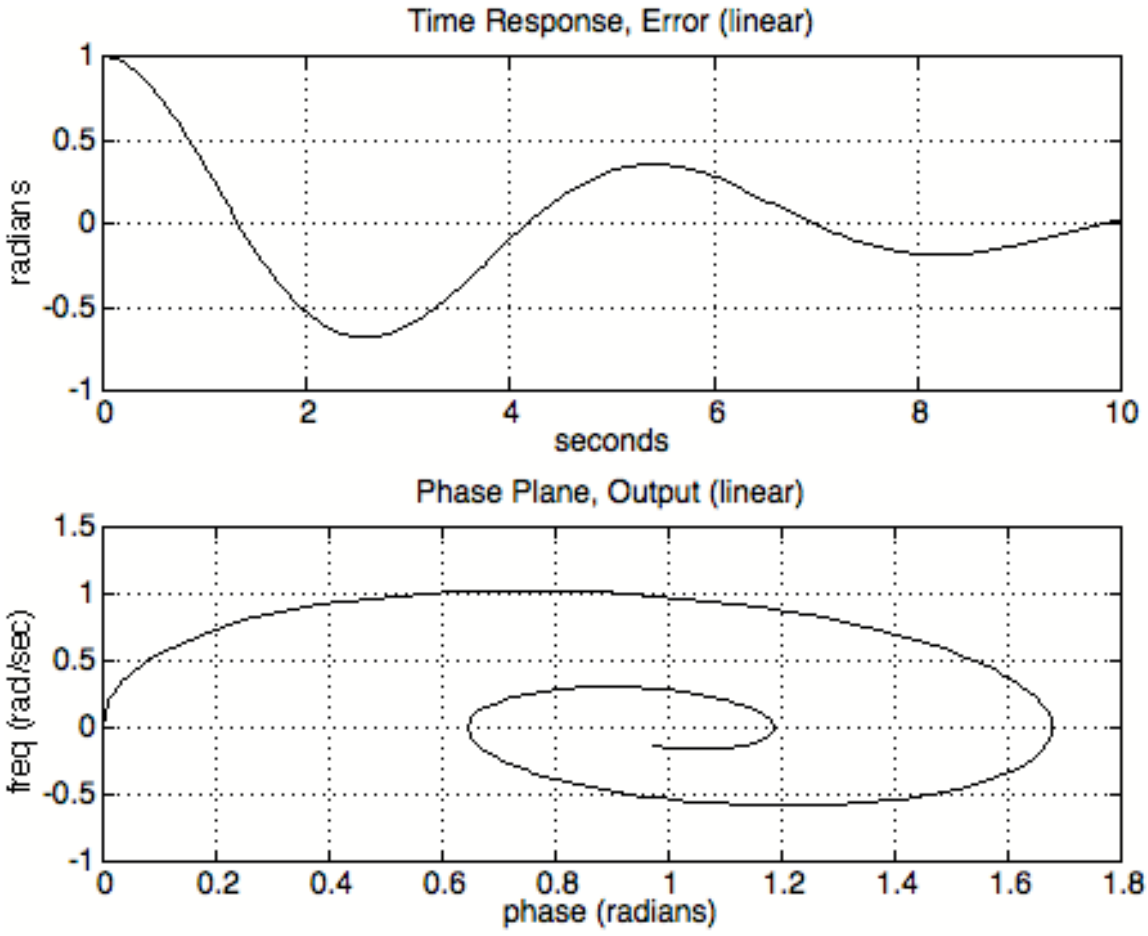

Fig. 6.M.1 Time response and phase-plane plot for step input with added pole, $r = 0.5$.

6.M.6 Adding Initial Conditions

`LMatPhPn` is a step-response program for second-order loops that shows both the error response versus time and the phase plane and permits entry of damping factor and alpha, converting these values to the inputs required for NUM and DEN. It also allows the specification of initial phase and frequency.

To obtain an ordinary step response, set `SR` to 1 and provide a value for *A*, the input phase step amplitude. In that case the state variables are set to zero (initial steady state). Otherwise the transient starts with the specified initial output phase `Phinit` and radian frequency `Winit`, the state variables being set to whatever is necessary to produce them. The presence of a phase with magnitude *A* at the input still influences the results however — it establishes the final value.

Enter parameter changes before executing the program. Try several values of α with other parameters constant and initial conditions specified. Note how the response is independent of α under these conditions.

A second program `LMPhPnHz` differs only in that frequency is given in Hz.