or, using the *translation in Laplace domain* property given by Eq. 2.65, we can write the output finally as

$$y(t) = 2y_0 e^{-t} - y_0 e^{-2t}; \quad (t \geq 0) \tag{2.101}$$

In Example 2.11 we have seen how we can evaluate a single-input, single-output system's response if we know its transfer function, applied input and initial conditions, by using a partial fraction expansion of $Y(s)$. For a system with complex poles (such as Example 2.8), finding partial fraction expansion can be very difficult. Fortunately, the MATLAB intrinsic command *residue* makes finding partial fraction expansion a simple affair. All one has to do is to specify the numerator and denominator polynomials of the *rational function* in s – such as Eq. (2.98) – for which a partial fraction expansion is desired. For example, if the rational function is $N(s)/D(s)$, then the coefficients of the polynomials $N(s)$ and $D(s)$ *in decreasing powers of* s are specified in two vectors, say, n and d. Then the *residue* command is used as follows to give the terms of the partial fraction expansion:

```
>>[k,p,c] = residue(n,d) <enter>
```

where p is a vector containing the *poles* of $N(s)/D(s)$, k is a vector containing the corresponding *residues*, and c is the *direct constant*. In terms of the elements of p and k, the partial fraction expansion is given by

$$N(s)/D(s) = c + k_1/(s - p_1) + \cdots + k_n/(s - p_n) \tag{2.102}$$

where all the poles, p_j, are *distinct* (i.e. they appear *only once* – as in Example 2.11). If a pole, say p_m, is *repeated q times*, then the partial fraction expansion obtained from the *residue* command is given by

$$\begin{aligned} N(s)/D(s) = c + k_1/(s - p_1) + \cdots + k_m/(s - p_m) + k_{m+1}/(s - p_m)^2 \\ + k_{m+2}/(s - p_m)^3 + \cdots + k_{m+q-1}/(s - p_m)^q + \cdots + k_n/(s - p_n) \end{aligned} \tag{2.103}$$

Now we are well equipped to talk about a linear system's response to singularity functions.

2.5 Response to Singularity Functions

In the previous two sections, we saw how frequency response, Laplace transform, and transfer function can be used to evaluate a linear system's characteristics, and its response to initial conditions (Example 2.11). Here we will apply a similar approach to find out a linear system's response to singularity functions, and extend the method for the case of arbitrary inputs. We had ended Section 2.2 with a remark on the special place held by the unit impulse function in control theory. To understand why this is so, let us define *impulse response*, $g(t)$, as the response of a system to a unit impulse, $\delta(t)$, applied as input at time $t = 0$. Furthermore, it is assumed that the system is at rest at $t = 0$, i.e. all

initial conditions (in terms of the output, $y(t)$, and its time derivatives) are zero. We know from Table 2.1 that the Laplace transform of $\delta(t)$ is unity. Also, from the definition of the transfer function for a single-input, single-output system (Eq. (2.72)) $Y(s) = G(s)U(s)$. Since, in this case, $y(t) = g(t)$ and Laplace transform of the input, U(s)=1, it implies that the following must be true:

$$g(t) = \mathcal{L}^{-1}Y(s) = \mathcal{L}^{-1}[G(s)U(s)] = \mathcal{L}^{-1}G(s) \tag{2.104}$$

Equation (2.104) denotes a very important property of the impulse response, namely that the impulse response of a linear, time-invariant system with *zero initial conditions* is equal to the *inverse Laplace transform* of the system's *transfer function*. Hence, the symbol $g(t)$ for the impulse response! One can thus obtain $G(s)$ from $g(t)$ by applying the Laplace transform, or $g(t)$ from $G(s)$ by applying the inverse Laplace transform. Since the transfer function contains information about a linear system's characteristics, we can now understand why impulse response (and the unit impulse function) deserve a special place in control theory. In a manner similar to the impulse response, we can define the *step response*, $s(t)$, as a linear, time-invariant system's response to unit step input, $u_s(t)$, applied at time $t = 0$ with *zero initial conditions*. Again, using Table 2.1, we note that Laplace transform of the unit step function is given by $U(s) = 1/s$, and the step response can be expressed as

$$s(t) = \mathcal{L}^{-1}[G(s)U(s)] = \mathcal{L}^{-1}[G(s)/s] \tag{2.105}$$

which shows that the step response is also intimately related with a system's transfer function, and hence with its characteristics.

Example 2.12

Let us revisit the second order system consisting of the electrical network of Examples 2.8 and 2.9, and evaluate the impulse response, $g(t)$, and step response, $s(t)$, for this system with zero initial conditions. Equation (2.80), gives the system's transfer function as $G(s) = 0.5s/(s^2 + 30s + 10^6)$. Using the partial fractions expansion of $G(s)$, we can write

$$G(s) = k_1/(s - p_1) + k_2/(s - p_2) \tag{2.106}$$

where p_1 and p_2 are the two *poles* (roots of the denominator polynomial of $G(s)$), and the *residues* k_1 and k_2 are evaluated as follows:

$$k_1 = [(s - p_1)G(s)]|_{s=p1} = 0.5p_1/(p_1 - p_2) \tag{2.107}$$

$$k_2 = [(s - p_2)G(s)]|_{s=p2} = 0.5p_2/(p_2 - p_1) \tag{2.108}$$

Again, from Example 2.9 we know that $p_1 = -15 - 999.9i$ and $p_2 = -15 + 999.9i$. Then from Eqs. (2.107) and (2.108), $k_1 = 0.25 - 0.00375i$ and $k_2 = 0.25 + 0.00375i$. The residues can be verified by using the MATLAB intrinsic command *residue* as follows:

```
>>N=[0.5 0]; D=[1 30 1e6]; [k,p,c]=residue(N,D) <enter>
```

```
k =
 0.25000000000000+0.00375042194620i
 0.25000000000000-0.00375042194620i

p =
   1.0e+002*
 -0.15000000000000+9.99887493671163i
 -0.15000000000000-9.99887493671163i

c =
   [ ]
```

Taking the inverse Laplace transform of Eq. (2.106) with the use of Table 2.1, we get the following expression for the impulse response, $g(t)$:

$$g(t) = k_1 \exp(p_1 t) + k_2 \exp(p_2 t);\ (t \geq 0) \tag{2.109}$$

Using the fact that p_1 and p_2 are *complex conjugates* (and k_1 and k_2 are also complex conjugates), we can simplify Eq. (2.109) to give

$$g(t) = e^{-15t}[0.5\cos(999.9t) - 0.0075\sin(999.9t)];\ (t \geq 0) \tag{2.110}$$

Note that the impulse response, $g(t)$, given by Eq. (2.110) has an amplitude which decreases *exponentially* with time due to the term e^{-15t}. This is a characteristic of a underdamped, *stable* system, as seen in Figure 2.24.

Since the poles can also be represented in terms of their natural frequency, ω_n, and damping-ratio, ς, as $p_{2,1} = -\varsigma\omega_n \pm i\omega_n(\varsigma^2 - 1)^{1/2}$, (see Eqs. (2.83) and (2.84)) we can also write

$$g(t) = y_0 \exp(-\varsigma\omega_n t)\sin[\omega_n t\ (1 - \varsigma^2)^{1/2} + \theta]/(1 - \varsigma^2)^{1/2};\quad (t \geq 0) \tag{2.111}$$

where $\theta = \cos^{-1}(\varsigma)$ and

$$y_0 = 2k_1[(1 - \varsigma^2)^{1/2}]/e^{i(\theta - \pi/2)} = 2k_2[(1 - \varsigma^2)^{1/2}]/e^{-i(\theta - \pi/2)} \tag{2.112}$$

You can verify Eqs. (2.111) and (2.112) using complex algebra, but don't worry if you don't feel like doing so, because Eq. (2.109) can be directly obtained using MATLAB to get the impulse response, $g(t)$, as follows:

```
>>p1=-15-999.9i; p2=conj(p1); k1=0.25-0.00375i; k2=conj(k1); <enter>

>>t=0:0.001:1; g=k1*exp(p1*t)+k2*exp(p2*t); plot(t,g),
  xlabel('Time (s)'), ylabel('g(t)') <enter>
```

Here *conj( )* is the intrinsic MATLAB operator that calculates the complex conjugate, *exp( )* is the exponential function, and *plot(x,y)* is the MATLAB command for plotting the vector x against the vector y. Note that both t and g are vectors of the same size. Also, note the ease by which complex vector calculations have been made using MATLAB. The same computation in a low-level language – such as

Fortran, Basic, or C – would require many lines of programming. For more information on the usage of a MATLAB command type *help < name of command> <enter>* at the MATLAB prompt.

We know from Example 2.9 that for the example electrical network, the natural frequency and damping-ratio are, $\omega_n = 1000$ rad/s and $\varsigma = 0.015$, respectively. Substituting these numerical values in Eq. (2.112), we get $y_0 = 0.5$ amperes.

We can also evaluate the step response, $s(t)$, of the electrical network by taking the inverse Laplace transform of $G(s)/s$ as follows:

$$\begin{aligned} s(t) &= \mathcal{L}^{-1}[G(s)/s] = \mathcal{L}^{-1}[0.5/(s^2 + 30s + 10^6)] \\ &= \mathcal{L}^{-1}[2.5 \times 10^{-4}i/(s - p_1) - 2.5 \times 10^{-4}i/(s - p_2)] \end{aligned} \tag{2.113}$$

or

$$\begin{aligned} s(t) &= 2.5 \times 10^{-4}i[\exp(p_1 t) - \exp(p_2 t)] \\ &= 5 \times 10^{-4}e^{-15t} \sin(999.9t); \quad (t \geq 0) \end{aligned} \tag{2.114}$$

Note that $s(t)$ given by Eq. (2.114) also indicates a stable system due to the oscillatory step response with a decaying amplitude of $5 \times 10^{-4}e^{-15t}$.

The calculation of step and impulse responses can be generalized by using the partial fraction expansion of $G(s)/s$ and $G(s)$, respectively. Taking the inverse Laplace transform of Eq. (2.104), we can express the impulse response for a unit impulse input applied at $t = 0$ as follows:

$$\begin{aligned} g(t) = c\delta(t) &+ k_1 \exp(p_1 t) + \cdots + k_m \exp(p_m t) + k_{m+1} t \exp(p_m t) \\ &+ k_{m+2} t^2 \exp(p_m t)/2 + \cdots + k_{m+q-1} t^{q-1} \exp(p_m t)/(q-1)! \\ &+ \cdots + k_n \exp(p_n t) \quad (t \geq 0) \end{aligned} \tag{2.115}$$

Note that in deriving Eq. (2.115), we have used the *translation in Laplace domain* property of the Laplace transform (Eq. (2.65)), and have written the inverse Laplace transform of $1/(s - p)^k$ as $t^{k-1}e^{pt}/(k-1)!$. If the impulse is applied at $t = t_0$, the time t in Eq. (2.115) should be replaced by $(t - t_0)$, since the Laplace transform of $g(t - t_0)$ is $G(s)\exp(-st_0)$. If the system is *strictly proper* (i.e. the degree of the numerator polynomial of the transfer function, $G(s)$, is *less than* that of the denominator polynomial), then the direct constant, c, in the partial fraction expansion of $G(s)$ is zero, and the impulse response does not go to infinity at $t = 0$ (Eq. (2.115)). Hence, for strictly proper transfer functions, we can write a computer program using MATLAB to evaluate the impulse response using Eq. (2.115) with $c = 0$. Such a program is the M-file named *impresp.m*, which is listed in Table 2.2, and can be called as follows:

```
>>[g,t] = impresp(num,den,t0,dt,tf) <enter>
```

Table 2.2 Listing of the M-file *impresp.m*, which calculates the impulse response of a strictly proper, single-input, single-output system

impresp.m

```
function [y,t]=impresp(num,den,t0,dt,tf);
%Program for calculation of impulse response of strictly proper SISO
   systems
%num = numerator polynomial coefficients of transfer function
%den = denominator polynomial coefficients of transfer function
%(Coefficients of 'num' and 'den' are specified as a row vector, in
%decreasing powers of 's')
%t0 = time at which unit impulse input is applied
%dt = time-step (should be smaller than 1/(largest natural freq.))
%tf = final time for impulse response calculation
%y = impulse response; t= vector of time points
%copyright(c)2000 by Ashish Tewari
%
%Find a partial fraction expansion of num/(den):-
[r,p,k]=residue(num,den);
%Calculate the time points for impulse response:-
t=t0:dt:tf;
%Find the multiplicity of each pole, p(j):-
for j=1:size(p)
n=1;
          for i=1:size(p)
                    if p(j)==p(i)
                              if(i~=j)
                              n=n+1;
                              end
                    end
          end
mult(:,j)=n;
end
%Calculate the impulse response by inverse Laplace transform of
%partial-fraction expansion:-
y=zeros(size(t));
j=1;
while j<=size(p,1)
         for i=1:mult(:,j)
         y=y+r(j+i-1)*((t-t0).^(i-1)).*exp(p(j)*(t-t0))/factorial(i-1);
         end
         j=j+i;
end
```

where *num* and *den* are row vectors containing numerator and denominator polynomial coefficients, respectively, of the transfer function, $G(s)$, in decreasing powers of s, *t0* is the time at which the unit impulse input is applied, *dt* is the *time-step* size, *tf* is the *final time* for the response, g is the returned impulse response, and t is the returned vector of time points at which $g(t)$ is calculated. Instead of having to do inverse Laplace transformation by hand, we can easily use *impresp* to quickly get the impulse response of a strictly proper

Table 2.3 Listing of the M-file *stepresp.m*, which calculates the step response of a proper, single-input, single-output system

stepresp.m

```
function [y,t]=stepresp(num,den,t0,dt,tf);
%Program for calculation of step response of proper SISO systems
%num = numerator polynomial coefficients of transfer function
%den = denominator polynomial coefficients of transfer function
%(Coefficients of 'num' and 'den' are specified as a row vector, in
%decreasing powers of 's')
%t0 = time at which unit step input is applied
%dt = time-step (should be smaller than 1/(largest natural freq.))
%tf = final time for step response calculation
%y = step response; t= vector of time points
%copyright(c)2000 by Ashish Tewari
%
%Find a partial fraction expansion of num/(den.s):-
[r,p,k]=residue(num,conv(den,[1 0]));
%Calculate the time points for step response:-
t=t0:dt:tf;
%Find the multiplicity of each pole, p(j):-
for j=1:size(p)
n=1;
          for i=1:size(p)
                    if p(j)==p(i)
                              if(i~=j)
                              n=n+1;
                              end
                    end
          end
mult(:,j)=n;
end
%Calculate the step response by inverse Laplace transform of
%partial-fraction expansion:-
y=zeros(size(t));
j=1;
while j<=size(p,1)
         for i=1:mult(:,j)
         y=y+r(j+i-1)*((t-t0).^(i-1)).*exp(p(j)*(t-t0))/factorial(i-1);
         end
         j=j+i;
end
```

plant. The M-file *impresp.m* uses only the intrinsic MATLAB functions, and is useful for those who do not have Control System Toolbox (CST). Usage of CST command *impulse* yields the same result. (We postpone the discussion of the CST command *impulse* until Chapter 4, as it uses a *state-space* model of the system to calculate impulse response.) Note the programming steps required in *impresp* to identify the multiplicity of each pole of $G(s)$. For increased accuracy, the time-step, dt, should be as small as possible, and, in any case, should *not exceed* the reciprocal of the *largest natural frequency* of the system.

In a manner similar to the impulse response, the step response calculation can be generalized by taking the inverse Laplace transform of the partial fraction expansion of

$G(s)/s$ as follows:

$$s(t) = k_1 + k_2 \exp(p_2 t) + \cdots + k_m \exp(p_m t) + k_{m+1} t \exp(p_m t) + k_{m+2} t^2 \exp(p_m t)/2 + \cdots + k_{m+q-1} t^{q-1} \exp(p_m t)/(q-1)! + \cdots + k_n \exp(p_n t)(t \geq 0) \quad (2.116)$$

where k_1 is the residue corresponding to the pole at $s = 0$, i.e. $p_1 = 0$, and $k_2 \ldots k_n$ are the residues corresponding to the poles of $G(s)$, $p_1 \ldots p_n$, in the partial fraction expansion of $G(s)/s$. Note that if $G(s)$ is a *proper* transfer function (i.e. numerator polynomial is of *lesser* or *equal degree* than the denominator polynomial), then $G(s)/s$ is *strictly proper*, and the direct term, c, is zero in the partial fraction expansion of $G(s)/s$. Thus, we can evaluate the step response of a proper system using Eq. (2.116), which should be modified for a unit step input applied at $t = t_0$ by replacing t in Eq. (2.116) by $(t - t_0)$. A MATLAB program called *stepresp.m*, which evaluates the step response of proper system by Eq. (2.116), is listed in Table 2.3, and can be used as follows:

```
>>[s,t] = stepresp(num,den,t0,dt,tf) <enter>
```

where *num* and *den* are row vectors containing numerator and denominator polynomial coefficients, respectively, of the transfer function, $G(s)$, in decreasing powers of s, *t0* is the time at which the unit step input is applied, *dt* is the *time-step* size, *tf* is the *final time* for the response, s is the returned step response, and t is a vector containing time points at which $s(t)$ is calculated. The M-file *stepresp.m* uses only intrinsic MATLAB functions, and is useful in case you do not have access to Control System Toolbox (CST). The CST command *step* is a quick way of calculating the step response. The GUI tool associated with the command *step* also lets you get the values of $s(t)$ and t at any point on the step response curve by merely clicking at that point.

Example 2.13

Let us compute and plot the step and impulse responses of the aircraft transfer function, $\theta(s)/\delta(s)$, of Example 2.10, given by Eq. (2.88). We must begin with the specification of the transfer function as follows:

```
>>a=[1 0.02]; b=[1 0.4]; num=-1.4*conv(a,b) <enter>
num =
  -1.4000   -0.5880   -0.0112

>>a=[1 0.005 0.006]; b=[1 1 1.4]; den=conv(a,b) <enter>
den =
  1.0000   1.0050   1.4110   0.0130   0.0084
```

Note that the transfer function is *strictly proper* (the numerator polynomial is of *second degree*, while the denominator polynomial is of *fourth degree*). Hence, we can use *impresp.m* to compute the impulse response, $g(t)$, and plot the result as follows:

```
>>[g,t] = impresp(num,den,0,0.5,250); plot(t,g) <enter>
```