Examples 8.5 and 8.6 show that finding the pulse transfer functions of sampled-data analog systems requires converting the system into an equivalent system in which both output and input appear to be digital signals by appropriate placement of samplers. In doing so, care must be exercised that the characteristics of the system are unchanged. It is clear that the *pulse transfer function* of a general *sampled-data analog* system consisting of several *analog* sub-systems cannot be handled in the *same* manner as the *transfer function* of an *analog system*. For example, the pulse transfer function, $G(z)$, of a sampled-data analog system consisting of two analog sub-systems with transfer functions $G_1(s)$ and $G_2(s)$ in series *cannot* be written as $G(z) = G_1(z)G_2(z)$, but only as $G(z) = z\{G_1(s)G_2(s)\}$. This is due to the fact that, generally, $z\{G_1(s)G_2(s)\} \neq G_1(z)G_2(z)$. However, if we are dealing with a *digital* system consisting of *digital* sub-systems, the *pulse transfer functions* of the sub-systems are handled in precisely the same manner as the *transfer functions* of *analog* sub-systems systems. For example, the pulse transfer function of a digital system consisting of *two* digital sub-systems with pulse transfer functions $G_1(z)$ and $G_2(z)$ in *series* is merely $G(z) = G_1(z)G_2(z)$. All of this shows us that we must be extremely careful in deriving the pulse transfer functions. It always helps to write down the input-output relationships (such as in Example 8.6) of the various sub-systems as *separate* equations, and then derive the overall input-output relationship from those equations.

8.4 Frequency Response of Single-Input, Single-Output Digital Systems

In a manner similar to an analog system with transfer function, $G(s)$, whose frequency response is the value of $G(s)$ when $s = i\omega$, we can define the *frequency response* of a *digital* system with a *pulse transfer function*, $G(z)$, as the value of $G(z)$ when $z = e^{i\omega T}$, or $G(e^{i\omega T})$. In so doing, we can plot the gain and phase of the frequency response, $G(e^{i\omega T})$, as functions of the frequency, ω, somewhat like the Bode plots of an analog system. However, the *digital Bode plots* crucially depend upon the sampling interval, T. For instance, the gain, $|G(e^{i\omega T})|$, would become *infinite* for some values of ωT. The frequency response of a digital system is related to the steady-state response to a harmonic input, provided the system is stable (i.e. its harmonic response at large time exists and is finite). The sampling rate of a harmonic output is crucial in obtaining the digital system's frequency response. If a high-frequency signal is sampled at a rate smaller than the signal frequency, then a large distortion and *ambiguity* occur in the sampled data. For example, if we sample two very different harmonic signals, say $\sin(0.25\pi t/T)$ and $\sin(1.75\pi t/T)$, with the same sampling interval, T, then the two sampled data would be *identical*. In other words, we lose information about the higher frequency signal by sampling it at a lower rate. This important phenomenon of sampled-data analog systems is known as *aliasing*. To avoid aliasing, the sampling rate must be at least *twice* the signal's *bandwidth*, ω_b (i.e. the *largest* frequency contained in the signal). The minimum acceptable sampling rate is called the *Nyquist frequency*. The Nyquist frequency in rad/s is thus given by *half* the required *sampling rate*, i.e. $2\pi/T = 2\omega_b$, or $\pi/T = \omega_b$, where T is the sampling interval. Hence, the frequency response of a digital system with a given sampling interval, T, is usually calculated only for $\omega \leq \pi/T$.

MATLAB's Control System Toolbox (CST) provides the command *dbode* for computing the frequency response of a digital system, and is used as follows:

```
>>[mag,phase,w] = dbode(num,den,T) <enter>
```

where *num* and *den* are the numerator and denominator polynomials of the system's pulse transfer function, $G(z)$, in decreasing powers of z, T is the sampling interval, *mag* and *phase* are the returned magnitude and phase (degrees), respectively, of the computed frequency response, $G(e^{i\omega T})$, and w is the vector of discrete frequency points (upto the Nyquist frequency) at which the frequency response is computed. The user can optionally specify the desired frequency points as the *fourth* input argument of the command *dbode*. When used without the output arguments on the left-hand side, *dbode* produces the digital Bode plots on the screen.

Example 8.7

Let us find the frequency response of a digital system with pulse transfer function, $G(z) = z/(z^2 + 2)$, and sampling interval, $T = 0.1$ second. To get the magnitude and phase of the frequency response, we can use the command *dbode* as follows:

```
>>num = [1 0];den = [1 0 2];dbode(num,den,0.1) <enter>
```

The resulting digital Bode magnitude and phase plots are shown in Figure 8.5. Note that the Nyquist frequency is $\pi/T = 31.4159$ rad/s, indicated as the highest frequency in Figure 8.5 at which the frequency response is plotted. At the Nyquist frequency, the phase is seen to approach 180°. There is a peak in the gain plot at $\omega = 15.7$ rad/s at which the gain is 0 dB and the phase is 90°. A gain of 0 dB

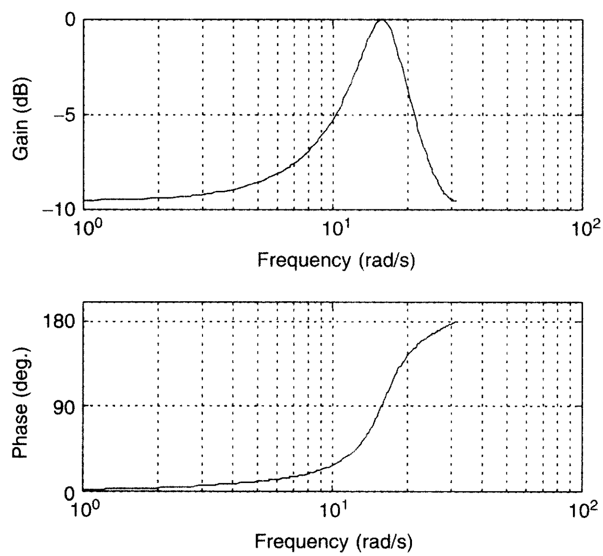

Figure 8.5 Digital Bode plot of the pulse transfer function, $G(z) = z/(z^2 + 2)$ with a sampling interval, $T = 0.1$ second