Lecture 23 The Zero Order Hold as an Algorithm or a D/A Circuit, and the A/D Circuit (4/18/20)

1. The ZOH as an Method of Matching the Step Response

In the last lecture we addressed a discrete approximation, $G(z)$, of a continuous-time transfer function $G(s)$ such that the impulse response $g(kT)$ matched the impulse response $g(t)$. This is the appropriate strategy if one knows a priori that the inputs to $G(s)$ are impulsive in nature. By that, we mean that they are of short duration relative to the system dominant time constant, τ, and that they are spaced far apart relative to τ.

Suppose now that we know a priori that the inputs to $G(s)$ are relatively smooth inputs; as opposed to impulsive ones. The simplest example of a smooth input is a unit step. It is so smooth that it is a constant for all time. In this situation it would be appropriate to arrive at a transfer function $\breve{G}(z)$, such that the unit step response, call it $h(kT)$, matched the continuous-time step response $h(t)$.

So, let's define the transfer function $H(s)\overset{\Delta}{=}G(s)/s$ with corresponding impulse response $h(t)$. The appropriate approximation $H(z)$ will then simply be the z-transform counterpart of $H(s)\overset{\Delta}{=}G(s)/s$. Once we have $H(z)$, we need to then use it to arrive at our goal, which is $\breve{G}(z)$. To this end, consider the approximate derivative of $h(t)$ that is

$$\frac{h(kT)-h[(k-1)T]}{T}\overset{\Delta}{=}\breve{g}(kT). \tag{1.1}$$

The z-transform of (1) is:

$$\breve{G}(z)=\left(\frac{1-z^{-1}}{T}\right)H(z)=\left(\frac{1-z^{-1}}{T}\right)\mathbb{Z}\{G(s)/s)\}. \tag{1.2}$$

The quantity $\mathbb{Z}\{F(s)\}$ is simply notation for the z-transform counterpart of $F(s)$ that can be obtained from a table of z-transforms.

Example 1 In order to compare results here to those of the last lecture, we will let

$$G(s)=0.5\left(\frac{1}{s+5}\right)=0.1\left(\frac{5}{s+5}\right). \tag{E1.1}$$

From Table 8.1, the z-transform counterpart to $G(s)/s$ is most closely related to entry #12:

$$\frac{a}{s(s+a)}\quad\leftrightarrow\quad\frac{Tz(1-\alpha)}{(z-1)(z-\alpha)}. \tag{E1.2}$$

Recall that $\alpha\overset{\Delta}{=}e^{-aT}$ for notational convenience. Hence, we have

$$\mathbb{Z}\{G(s)/s\} = 0.1\left(\frac{Tz(1-\alpha)}{(z-1)(z-\alpha)}\right). \tag{E1.3}$$

From (2) we arrive at our goal:

$$\breve{G}(z) = \left(\frac{1-z^{-1}}{T}\right)\mathbb{Z}\{G(s)/s)\} = 0.1\left(\frac{1-\alpha}{z-\alpha}\right). \tag{E1.4}$$

Write (E1.4) as:

$$\breve{G}(z) = \frac{0.1(1-\alpha)z^{-1}}{1-\alpha z^{-1}} = \frac{Y(z)}{F(z)}. \tag{E1.5}$$

From (E1.5) we have:

$$Y(z) - \alpha z^{-1}Y(z) = 0.1(1-\alpha)z^{-1}F(z). \tag{E1.6}$$

The difference equation corresponding to (E1.6) is:

$$y_k - \alpha y_{k-1} = 0.1(1-\alpha)\,f_{k-1}. \tag{E1.7}$$

<u>Numerical values</u>: For $T = 0.02$ we have $\alpha = e^{-5(0.02)} = e^{-0.1} = 0.9048$. Inserting these into (E1.4) gives:

$$\breve{G}(z) = \frac{0.0095}{z-0.9048}. \tag{E1.8}$$

From Matlab we have:

$$\text{c2d(G,.02,'zoh') = } 0.009516/(\text{ z - 0.9048}). \tag{E1.9}$$

Comparing (E1.8) and (E1.9) shows that the Matlab c2d that includes the 'zoh' flag (which, by the way, is the default flag) gives the discrete approximation $\breve{G}(z)$ such that the discrete-time step response $h(kT)$ will match $h(t)$. This matching is illustrated in the figure at right.

As an approximate integration method, we see that the zoh is a Riemann approximation that under-estimates the area beneath the step response. □

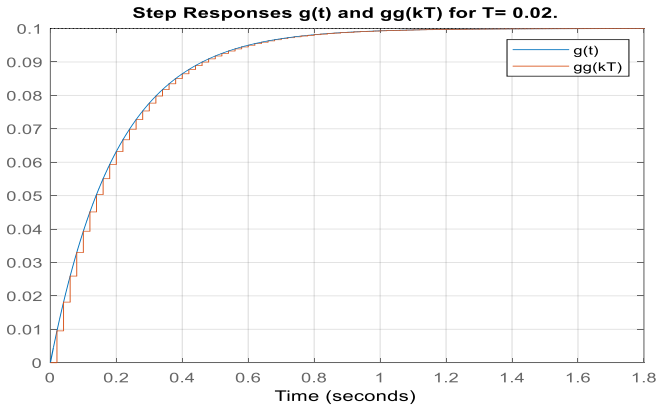

Figure E1.1 Plots of $h(t)$ and $h(kT)$.

CONCLUSION 1: From the above example we can conclude that the Matlab command 'c2d' that uses the 'zoh' flag will arrive at the transfer function $\breve{G}(z)$, such that the discrete-time step response matches the continuous-time one. For this reason, one could also label it as the 'step-invariant' method of approximating $G(s)$.

2. The ZOH as a Circuit

The ZOH can be used to arrive at a discrete-time approximation of $G(s)$ that matches the step response. However, it is also a circuit that transforms numbers computed from a digital control algorithm into continuous-time voltage that is needed to control the plant. It works in the following way.

Suppose that the output of the digital controller is the number 1.0. This number triggers a voltage, say 1 volt that is held for T seconds. If the next number is a 0.0 then a voltage of 0 volts is held for the next T seconds. This is illustrated at right.

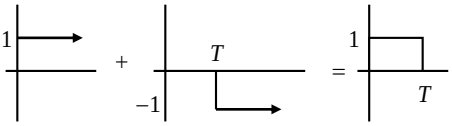

Figure 1 Behavior of a ZOH.

More generally, the continuous-time voltage to the plant will have a staircase structure. In words, a discrete sequence $\{u(kT)\}_{k=0}^{\infty}$ is converted into an analog (i.e. continuous-time) staircase function. It is for this reason that the ZOH is called a ***digital-to-analog converter*** (ADC). It is the most basic type of ADC, albeit, also the most popular.

How to mathematically model the ZOH circuit is tricky because half of it is in the discrete-time domain and the other half is in the continuous-time domain. To arrive at a model, consider viewing the number 1.0 coming into the circuit at time $t=0$ as a continuous-time Dirac delta function, $\delta(t)$. The immediate response to $\delta(t)$ is a unit step function, $1(t)$, as shown in the left plot in Figure 1. At time $t=T$ a second negative delta function $-\delta(t-T)$ goes into the circuit, thereby generating $-1(t-T)$. This is shown in the middle plot of Figure 1. The output of the circuit is the sum of $1(t)$ and $-1(t-T)$, as shown in the right plot in Figure 1. Hence, we can model the circuit output as

$$\breve{u}(t) = 1(t) - 1(t-T). \tag{2.1}$$

The response (2.1) is viewed as the impulse response of the ZOH. The input was a single impulse followed T seconds later by a zero, and the output is (2.1). Note that we used the symbol $\breve{u}(t)$. The reason is that we will reserve the symbol $u(t)$ to denote the output of the analog controller that the digital one is emulating.

Now recall that for any $f(t)$ with Laplace transform $F(s)$, the Laplace transform of $f(t-t_0)$ is $e^{st_0}F(s)$. Recall also that $z \overset{\Delta}{=} e^{sT}$. Hence, taking the Laplace transform of (1) gives the transfer function of the ZOH:

$$\breve{U}(s) = \frac{1}{s} - \frac{e^{-sT}}{s} = \frac{1-z^{-1}}{s} = ZOH(s). \tag{2.2}$$

The bridge between the digital world of the controller and the analog world of the plant is evident in (2.2), in the sense that it includes both *z* and *s*. Even so , it a continuous-time transfer function because we modeled the number 1.0 at time zero as $\delta(t)$.

More generally, the discrete-time sequence $\{u(kT)\}_{k=0}^{\infty}$ going into the ZOH can be modeled as

$$v(t)=\sum_{k=0}^{\infty}u(kT)\delta(t-kT)\cdot \tag{2.3}$$

The Laplace transform of the input (2.3) is

$$V(s)=\sum_{k=0}^{\infty}u(kT)z^{-k}\cdot \tag{2.4}$$

For the input (2.4) to the ZOH the output in the s-domain is:

$$\breve{U}(s)=ZOH(s)V(s)=\left(\frac{1-z^{-1}}{s}\right)\sum_{k=0}^{\infty}u(kT)z^{-k}\cdot \tag{2.5}$$

However, recall that the z-transform of the discrete-time function $\{u(kT)\}_{k=0}^{\infty}$ is, by definition, $U(z)$. Hence, for a general discrete-time sequence $\{u(kT)\}_{k=0}^{\infty}$ going into the ZOH, the output in the s-domain is

$$\breve{U}(s)=\left(\frac{1-z^{-1}}{s}\right)U(z)\cdot \tag{2.6}$$

The FRF of the ZOH Circuit

In (2) we arrived at a continuous-time model for the transfer function of a ZOH circuit. The associated FRF is:

$$ZOH(i\omega)=\frac{1-e^{-i\omega T}}{i\omega}\cdot \tag{2.7}$$

As with any transfer function, it is important to understand how it behaves as a filter. In particular, one should always be concerned with the filter bandwidth. For this reason, we will now compute the magnitude and phase functions associated with (2.7). Write (2.7) as

$$ZOH(i\omega)=\frac{1-e^{-i\omega T}}{i\omega}=\frac{1-(\cos\omega T-i\sin\omega T)}{i\omega}=\frac{(1-\cos\omega T)+i\sin\omega T}{i\omega}. \tag{2.8}$$

Then the magnitude is: $$M(\omega)=\frac{(1-\cos\omega T)^2+(\sin\omega T)^2}{\omega^2}=\frac{\sqrt{2(1-\cos\omega T)}}{\omega}. \tag{2.9}$$

The phase is:
$$\theta(\omega) = \tan^{-1}\left(\frac{\sin \omega T}{1-\cos \omega T}\right) - \frac{\pi}{2}. \tag{2.10}$$

For small values of ωT we have $\cos \omega T \cong 1 - \frac{(\omega T)^2}{2}$. This results in

$$M(\omega) \cong T \text{ and } \theta(\omega) \cong 0 \quad \text{for} \quad \omega T << 1 \tag{2.13}$$

Recall that at the outset we assumed that the number 1.0 resulted in 1.0 volts. This was arbitrary. Let's now assume that the number 1.0 results in $1/T$ volts. Then the ZOH transfer function becomes

$$ZOH(s) = \frac{1-z^{-1}}{Ts}. \tag{2.14}$$

The phase remains unchanged, but the magnitude becomes

$$M(\omega) = \frac{(1-\cos \omega T)^2 + (\sin \omega T)^2}{\omega^2} = \frac{\sqrt{2(1-\cos \omega T)}}{\omega}. \tag{2.15}$$

Hence,
$$M(\omega) \cong 1 \quad \text{for} \quad \omega T << 1. \tag{2.16}$$

The Bode plot for the ZOH is shown at right for a Nyquist frequency $\omega_N = 15$ (resulting in $T = \pi / \omega_N = \pi$). We see that the ZOH -3dB BW is close to ω_N. At frequencies below $\omega_N / 10$ it behaves almost ideally. It has constant gain, and very little phase drop. Near ω_N the phase becomes significant. At ω_N it is -90°. This could have a destabilizing influence on a lead controller having a center frequency in this region. Hopefully, the filter designer would realize that designing a filter with a center frequency anywhere near ω_N is asking for problems. □

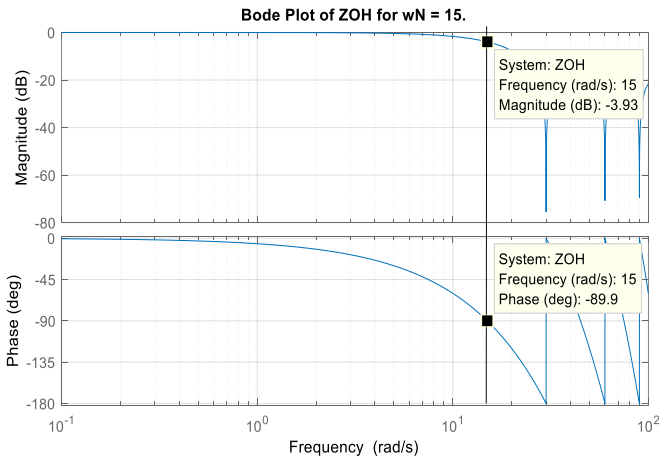

Figure 2 Bode plot of $ZOH(s)$ for $\omega_N = 15$.

3. The Analog-to-Digital (A/D) Converter Circuit

Let $y(t)$ be a measurement process that is described by

$$b_n y^{(n)} + \cdots + b_1 y^{(1)} + b_0 y = a_m f^{(m)} + \cdots + a_1 x^{(1)} + a_0 x \,. \tag{3.1}$$

Assume that the roots of the characteristic polynomial of (3.1) are all in the proper LHP. We do not measure $f(t)$. However, we do require that it's Laplace transform, $F(s)$, exists. We then have:

$$Y(s) = \left(\frac{a_m s^m + \cdots + a_1 s + a_0}{b_n s^n + \cdots + b_1 s + b_0} \right) F(s) \stackrel{\Delta}{=} G(s)F(s) \,. \tag{3.2}$$

Now let $y(kT) \stackrel{\Delta}{=} y_k$ denote the sampled measurement process. Note again that we are <u>not explicitly</u> sampling $f(t)$. Even so, in view of (3.1) we <u>are, effectively</u>, sampling $f(t)$, since (3.1) becomes a difference equation of the form:

$$\beta_n y_k + \cdots + \beta_1 y_{k-n+1} + \beta_0 y_{k-n} = \alpha_m f_k + \cdots + \alpha_1 f_{k-m+1} + \alpha_0 f_{k-m} \,. \tag{3.3}$$

Taking the z-transform of (3.3) gives:

$$Y(z) = \left(\frac{\alpha_m + \cdots + \alpha_1 z^{-m+1} + \alpha_0 z^{-m}}{\beta_n + \cdots + \beta_1 z^{-n+1} + \beta_0 z^{-n}} \right) F(z) \stackrel{\Delta}{=} G(z)F(z) \,. \tag{3.4}$$

This is illustrated in the figure at right. The block diagram (i) illustrates (3.2). Both (ii) and (iii) illustrate (3.4). We see that the effect of the A/D circuit that is used to sample $y(t)$ in (ii) is equivalent to sampling $f(t)$, and then replacing $G(s)$ by its z-transform counterpart, $G(z)$.

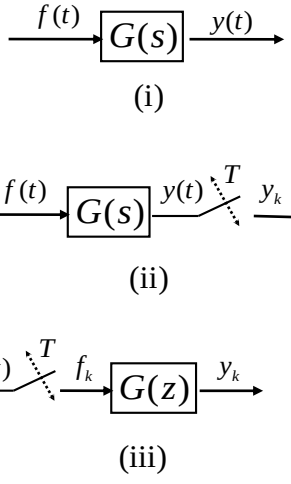

Figure 3.1

The discrete-time transfer function $G(z)$ is that which can be obtained from a standard table of z-transforms. It is not obtained from other Matlab methods of going from $G(s)$ to $G(z)$, such as the 'zoh' method or the 'tus' method. Those methods are algorithms that one can choose to use to approximate a transfer function in the s-domain by one in the z-domain. They are not hardware. The Matlab method that results in $G(z)$ given by (3.4) as a result of the A/D circuit must be obtained using the 'impulse' method given in Matlab. Just as the 'zoh' method can reflect the use of a D/A circuit or an algorithm for approximating $G(s)$, the 'impulse' method can be used to reflect the use of an A/D circuit or an algorithm for approximating $G(s)$.

Example 3.1 In this example we will investigate how an A/D circuit followed by a ZOH D/A circuit influences the FRF associated with $G(s)=1/(s+1)$. The block diagram is shown at right. The transfer function between $f(t)$ and $\hat{y}(t)$ is $\hat{G}(s)=G(z)\,ZOH(s)$. In particular, we will use sampling periods associated with the Nyquist frequencies (i) $\omega_N=15$.

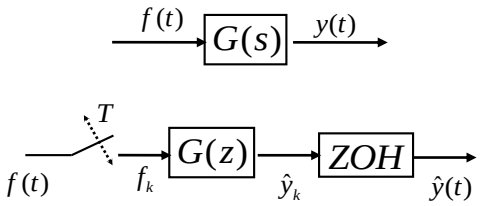

Figure E3.1 Original & approximate systems.

```
%Example 3.1
G=tf(1,[1,1]);
wN=15; %Choose Nyquist frequency
T=pi/wN; %Sampling period
Gz=c2d(G,T,'impulse');
[nGz,dGz]=tfdata(Gz,'v');
z1=exp(-s*T);
Gz1=nGz(1)/(dGz(1)+dGz(2)*z1);
ZOH=(1-z1)/(T*s);
%Ghat=d2c(Gz,'zoh'); THIS IS BS!
Ghat=ZOH*Gz1;
figure(60)
bode(G,Gz,Ghat)
title(['Bode Plots of G(s), G(z),and Ghat(s)for
wN = ',num2str(wN),' r/s'])
legend('G','Gz','Ghat')
grid
```

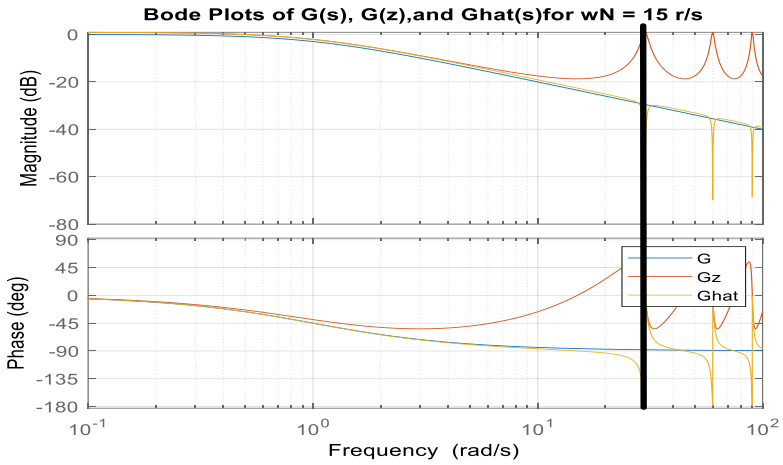

Figure E3.2 Overlaid Bode plots of $G(s)$, $G(z)$, and $\hat{G}(s)$ for $\omega_N=15$.

Before we discuss Figure E3.2 we need to discuss the code. I had been under the impression that the command in BLUE would amount to placing a ZOH circuit after $G(z)$. I was wrong. I know of no Matlab commands that do this. Simulink has a ZOH D/A circuit object, but does not allow access to the underlying Matlab code. Hence, I had to create the continuous-time $ZOH(s)$ transfer function. I also had to construct the continuous-time periodic version of $G(z)$. Only then was I able to compute $\hat{G}(s)$. [**Note**: I just discovered this on 4/18/20! For years I had assumed theat the 'c2d' command using the 'zoh' flag was equivalent to the language these acronyms stand for: convert from a discrete-time to a continuous-time transfer function using a ZOH.] We will now discuss the magnitude and phase approximations of $G(s)$ using $\hat{G}(s)$.

On p.617 of the book the authors state "… best performance of a digital controller is obtained when the sample rate is greater than 25 times the bandwidth." The transfer function $G(s)=1/(s+1)$ has a -3dB BW $\omega_{BW}=1$. The left Bode plots in Figure E3.2 were obtained for a Nyquist frequency $\omega_N=15$, which corresponds to a sampling frequency of 30. This is close to the 25 times recommended by the authors. the BW. From Figure E3.2 we see that both the magnitude and phase approximations are reasonably good up to ω_N.

The Bode plots of $G(z)$ are included to show how the A/D circuit influences the magnitude and phase approximation. We already know that up to ω_N, the ZOH D/A circuit has nearly constant magnitude, and that the phase is nearly zero up to $\omega_N/10$, but that it drops notably thereafter. The reason that the phase of $\hat{G}(s)$ is so accurate near ω_N is a matter of

coincidence. The phase of $G(z)$ returns to 0°, while the phase of $ZOH(s)$ reaches its limit of -90°. It just so happens that the limiting phase of $G(s)$ is also -90°.

At frequencies just above ω_N we see that the accuracy of the magnitude of $\hat{G}(s)$ is good. However, the accuracy of the phase deteriorates. □

4. Summary

In this lecture we first addressed the ZOH as an algorithm. The 'impulse' flag mathematically converts $G(s)$ to $G(z)$ using a standard table of z-transforms. This method is called the impulse-matching method, since the impulse response associated with $G(z)$ matches that associated with $G(s)$ at the sample times. The 'zoh' flag mathematically converts $G(s)$ to $G(z)$ using a method, whereby the step response associated with $G(z)$ matches that associated with $G(s)$ at the sample times. For this reason, I would prefer that it be called 'step', since it is a step-matching method.

We then addressed the ZOH as a D/A zero-order-hold circuit. I had assumed that the command 'd2c(Gz,'zoh') wouold place a ZOH after Gz, which is where the circuit is placed. I discovered that I was wrong when I noticed that the phase of $G(z)$ went back to zero at ω_N, but that the phase of $\hat{G}(s) = G(z)ZOH(s)$ went up toward zero; not toward -90° as I knew it should. It turns out that the 'd2c(Gz,'zoh') command places the inverse of $ZOH(s)$ prior to $G(z)$. It is an algorithm that converts $G(z)$=c2d(G,T,'zoh') back to the original G.

In fact, there is no Matlab function that models a ZOH D/A circuit. Simulink has a block that does this, but gives no access to the underlying Matlab code. This is, arguably, the most important takeaway from this lecture. The code to emulate the ZOH D/A circuit is trivial:

```
z1=exp(-s*T);    ZOH=(1-z1)/(T*s);
```

However, since ZOH is a continuous-time transfer function, it is necessary to convert the discrete-time transfer function $G(z)$ to a continuous-time transfer function where $G(z = e^{i\omega T})$ is periodic in ω. Matlab does not allow discrete-time transfer functions to be multiplied by continuous-time ones. The needed cammands are:

```
Gz=c2d(G,T,'impulse');  [nGz,dGz]=tfdata(Gz,'v'); Gz1=nGz(1)/(dGz(1)+dGz(2)*z1);
```

Conclusion

I can very much appreciate that more than a few students might be 'lost' after reading these notes. I chose to spend time on digital systems because I believe that many of you will encounter things like A/D and D/A circuits. A few of you might even decide that you want to build your own autopilot digital controller. Please do not hesitate to email me if you have confusion after reading these notes at least twice.

Matlab Code

```
%PROGRAM NAME: lec23.m
G=tf(1,[2,10]);
T=0.02;
GGz=c2d(G,T,'zoh');
figure(40)
step(G,GGz)
title(['Step Responses g(t) and gg(kT) for T= ',num2str(T),'.'])
legend('g(t)','gg(kT)')
grid
%------------------------
%Bode Plot of ZOH
s=tf('s');
wN=1;
T=pi/wN;
z1=exp(-s*T);
ZOH=(1-z1)/(T*s);
figure(50)
bode(ZOH)
title(['Bode Plot of ZOH for wN = ',num2str(wN),'.'])
grid
%==========================
%Example 3.1
G=tf(1,[1,1]);
wN=30; %Choose Nyquist frequency
T=pi/wN; %Sampling period
Gz=c2d(G,T,'impulse');
Ghat=d2c(Gz,'zoh');
figure(60)
bode(G,Gz,Ghat)
title(['Bode Plots of G(s), G(z),and Ghat(s)for wN = ',num2str(wN),' r/s'])
legend('G','Gz','Ghat')
grid
```