Lecture 22 The Nuts and Bolts of the z-Transform (4/17/20)

1. Motivation

The concept of the z-transform is analogous to the concept of the Laplace transform; the difference being that the former is couched in discrete time, while the latter is couched in continuous time. Often, a first course in feedback control will not address the z-transform. There is already more than enough material to cover. I have chosen to cover it in this course due to the fact that in this day and age practically all development- both hardware and software- is carried out in the digital world. Moreover, the costs have reduced to the point where even an engineering student can afford to purchase the materials needed to build a sophisticated systems, such as a hobby drone flight control unit, or a robotic manipulator position control unit.

So, before we get into the nuts and bolts of the z-transform, let's briefly review the salient features of the Laplace transform, and how those features carry over to the z-transform.

Solving an Approximate Differential Equation

The motivation for the Laplace transform was introduced early in the course as a method for both solving and recovering a differential equation that describes the dynamics between an input and an output. Consider the following example.

Example 1.1 Suppose we want to solve the following differential equation:

$$2\dot{y}+10y=u_s(t) \quad \text{where } y(0_-)=0. \tag{E1.1}$$

The input $u_s(t)$ in (E1.1) is the unit step. Taking the Laplace transform of (1.1) gives:

$$2sY(s)+10Y(s)=\frac{1}{s}. \tag{E1.2}$$

Solving (E1.2) for $Y(s)$ gives:

$$Y(s)=\frac{1}{s(2s+10)}. \tag{E1.3}$$

Equation (E1.3) is the solution; however, it is a solution in the s-domain. To obtain the solution in the time domain, we need only identify an entry in a table of Laplace transform pairs that is suitable for that purpose. In the table on the inside of the cover of the book we have entry #11:

$$F(s)=\frac{a}{a(s+a)} \qquad \leftrightarrow \qquad f(t)=1-e^{-at}. \tag{E1.4}$$

Rewrite (E1.3) as:

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

Then from (E1.4) we arrive at:

$$y(t) = 0.1\left(1 - e^{-5t}\right). \tag{E1.6}$$

Now let's approximate $\dot{y}(t)$ by simply

$$\frac{y(kT) - y[(k-1)T]}{T} \stackrel{\Delta}{=} \frac{y_k - y_{k-1}}{T}. \tag{E1.7}$$

There are many ways to approximate $\dot{y}(t)$. The approximation (E1.7) is called a 1st order backward difference approximation. It is the simplest way. Substituting (E1.7) into the discrete-time version of (E1.1) gives:

$$2\left(\frac{y_k - y_{k-1}}{T}\right) + 10y_k = 1_k \quad \text{for all } k \geq 0. \tag{E1.8}$$

This can be rearranged to give:

$$\left(\frac{2}{T} + 10\right)y_k - \left(\frac{2}{T}\right)y_{k-1} = 1_k. \tag{E1.9}$$

Equation (E1.9) is called a 1st order difference equation. It is the discrete-time counterpart to the 1st order differential equation (E1.1). A key conceptual difference between these two equations lies in the fact that (E1.9) can be expressed as

$$y_k = \left(\frac{2}{2+10T}\right)y_{k-1} + \left(\frac{T}{2+10T}\right)1_k. \tag{E1.10}$$

Equation (E1.10) is the recursive form of the solution of the difference equation (E1.8). Were we content with this form, there would be no need to go any further. In fact, recursive forms such as (E1.10) are the preferred forms for real-time implementation of digital controllers.

In order to obtain an explicit expression for y_k, we will need to use a table of z-transforms. To this end, we must first define the z-transform. This was given at the end of the last lecture. We repeat it here for convenience.

Definition 1.1 The z-transform of $\{f_k\}_{k=0}^{\infty}$ is $F(z) = \sum_{k=0}^{\infty} f_k z^{-k}$.

Now consider the sequence $\{g_k = f_{k-1}\}_{k=0}^{\infty}$. In expanded form it is simply $\{0, f_0, f_1, \cdots\}$. In words, it is a shifted version of $\{f_k\}_{k=0}^{\infty}$. Then, per Definition 1.1 we have:

$$G(z) = \sum_{k=0}^{\infty} g_k z^{-k} = \sum_{k=0}^{\infty} f_{k-1} z^{-k} = z^{-1} \sum_{k=0}^{\infty} f_{k-1} z^{-(k-1)} = z^{-1} \sum_{k=0}^{\infty} f_k z^{-k} = z^{-1} F(z).$$

Hence, we see that a single backward shift of $\{f_k\}_{k=0}^{\infty}$ is equivalent to multiplication of $F(z)$ by z^{-1}. More generally, we have the following Laplace transform and z-transform counterparts:

$$f(t) \leftrightarrow F(s) \quad \Rightarrow \quad f^{(n)}(t) \leftrightarrow s^n F(s) \qquad \text{and} \qquad f_k \leftrightarrow F(z) \quad \Rightarrow \quad f_{k-n} \leftrightarrow z^{-n} F(z).$$

Just as the left property is central to solving differential equations using the Laplace transform, the right property is central to solving difference equations using the *z*-transform.

Now, From Table 8.1 on p.620 we have entry #3:

$$1_k \leftrightarrow \frac{z}{z-1}. \tag{E1.11}$$

Hence, the z-transform of (E1.9) is:

$$\left(\frac{2}{T}+10\right)Y(z)-\left(\frac{2}{T}\right)z^{-1}Y(z)=\frac{z}{z-1}. \tag{E1.11}$$

Solving (E1.11) for $Y(z)$ gives:

$$Y(z)=\frac{Tz^2}{[(2+10T)z-2](z-1)}. \tag{E1.12}$$

The closest expression to (E1.12) in Table 8.1 is entry #12:

$$1-\alpha^k \leftrightarrow \frac{z(1-\alpha)}{(z-\alpha)(z-1)}. \tag{E1.13}$$

Note that for notational convenience I have used $\alpha \overset{\Delta}{=} e^{-aT}$. So we must now manipulate (E1.12) into a similar form. Write (E1.12) as

$$Y(z)=z\left(\frac{T}{2+10T}\right)\frac{z}{[z-2/(2+10T)](z-1)}. \tag{E1.14}$$

Let $\alpha = 2/(2+10T)$. Then (E1.14) becomes

$$Y(z)=z\left(\frac{T}{(2+10T)(1-\alpha)}\right)\frac{z(1-\alpha)}{(z-\alpha)(z-1)}=z(0.1)\frac{z(1-\alpha)}{(z-\alpha)(z-1)}. \tag{E1.15}$$

Just as z^{-1} is equivalent to a 1-shift backward in time, z is equivalent to a 1-shif forward in time. Hence, we arrive at

$$y_k = 0.1(1-\alpha^{k+1}) \quad \text{where} \quad \alpha = 2/(2+10T). \tag{E1.16}$$

It should be clear that the accuracy of (E1.16) in approximating (E1.6) will depend on the chosen value for *T*. To get a rough idea of what values might be acceptable, consider that the time constant associated with (E1.1) is $\tau = 0.2$. It is reasonable to suspect that *T* will have to be a fraction of this value. Choosing $T = 0.02$ resulted in Figure E1.

```
%Example 1.1
G=tf(1,[2,10]);
figure(10)
tmax=1.2;
step(G,tmax)
T=0.02;
a=2/(2+10*T);
t=0:T:tmax;
nt=length(t);
k=1:nt;
y=0.1*(1-a.^k);
hold on
plot(t,y,'r')
title(['Step Responses y(t) and y(kT) for T=',num2str(T),' sec.'])
grid
legend('y(t)','y(kT)')
%Recursive Solution:
y=zeros(1,nt);
d=10+2/T;
for k=2:nt
    y(k)=((2/T)*y(k-1)+1)/d;
end
```

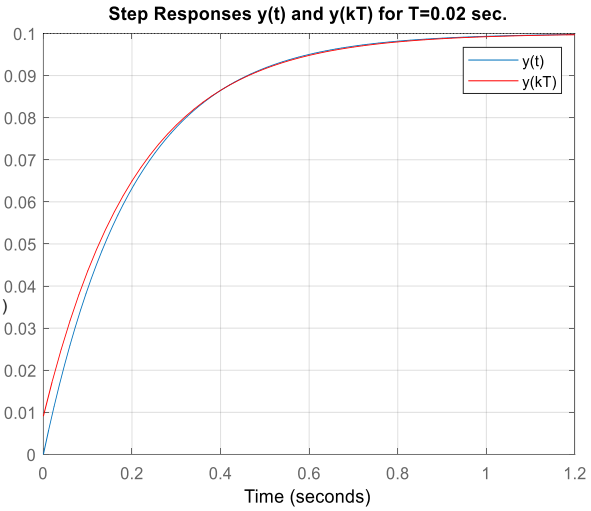

Figure E1 Plots of $y(t)$ and $y(kT)$ for $T = 0.02$.

Recovering an Approximate Differential Equation

Just as one can recover the differential equation from the associated s-domain transfer function, one can recover a finite-difference equation from the associated z-domain transfer function. This will be illustrated in the following example.

Example 2 Consider the transfer function associated with (E1.1):

$$\frac{Y(s)}{F(s)} = G(s) = \frac{1}{2s+10} = 0.5\left(\frac{1}{s+5}\right). \tag{E2.1}$$

From Table 8.1 we have entry #11:

$$\frac{1}{s+a} \leftrightarrow \frac{z}{z-\alpha}. \tag{E2.2}$$

As in Example 1, I have used $\alpha \overset{\Delta}{=} e^{-aT}$ for notational convenience. From (E2.2) we have the following relation:

$$\frac{Y(s)}{F(s)} = G(s) = 0.5\left(\frac{1}{s+5}\right) \quad \leftrightarrow \quad \frac{Y(z)}{F(z)} = G(z) = 0.5\left(\frac{z}{z-\alpha}\right). \tag{E2.3}$$

Cross-multiplying the rightmost relation in (E2.3) gives:

$$(z-\alpha)Y(z) = 0.5zF(z). \tag{E2.4}$$

Multiplying both sides of (E2.4) by z^{-1} gives:

$$Y(z) - \alpha z^{-1}Y(z) = 0.5F(z). \tag{E2.5}$$

Noting that multiplication by z^{-1} corresponds to a 1-shift backwards in time gives:

$$y_k - \alpha y_{k-1} = 0.5 f_k . \tag{E2.6}$$

Hence, (E2.6) is the discrete-time counterpart of the differential equation (E1.1) obtained using (E2.3).

From (E2.6) we have the recursive solution

$$y_k = \alpha y_{k-1} + 0.5 f_k . \tag{E2.7}$$

We will now compare (E2.7) to the recursive form associated with Example 1. In (E1.10) the input was chosen to be $f_k = 1_k$. Hence, the general recursive solution was:

$$y_k = \left(\frac{2}{2+10T} \right) y_{k-1} + \left(\frac{T}{2+10T} \right) f_k . \tag{E2.8}$$

Even though (E2.7) and (E2.8) have the same structure, clearly they do not have the same constants. This is because of the different way in which they were arrived at. Equation (E2.8) was arrived at using a first order backward-difference approximation of $\dot{y}(t)$, while (E2.7) was arrived at using (E2.3). Both forms are correct, in so far as they are both valid approximate solutions on the differential equation (E1.1). The question of which one is better will depend on two things. First is the choice of T. If T is chosen to be suitably small enough, then one should expect that they will be comparably accurate. Second is the nature of $f(t)$.

For $f(t) = 1_t$ the approximate responses (E2.7) and (E2.8) are shown at right.

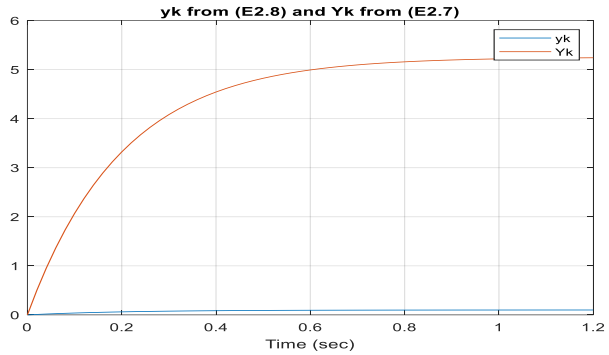

Figure E2.1 Step responses for (E2.7) and (E2.8).

We see that (E2.7) has a problem, since we know that $G(0) = 0.1$. The first thing to check in such a situation is the factor of T. For the value T=0.2 used here, Were (E2.7) to be multiplied by it, that could solve the problem.

Doing just that resulted in the figure at right. One can conclude that the table entry (E2.2) is missing a factor of T. [See Remark 2.2 in Lecture 21 for warning of such issues concerning T.] As to the question 'which one is better'? Personally, I would argue that (E2.8) is better. It is notably more accurate in approaching the value 0.1.

```
y=zeros(1,nt);
d=10+2/T;
for k=2:nt
    y(k)=((2/T)*y(k-1)+1)/d;
end
Y=zeros(1,nt);
f=ones(1,nt);
aa=exp(-5*T);
for k=2:nt
    %Y(k)=aa*Y(k-1)+0.5*f(k);
    Y(k)=aa*Y(k-1)+T*0.5*f(k);
end
```

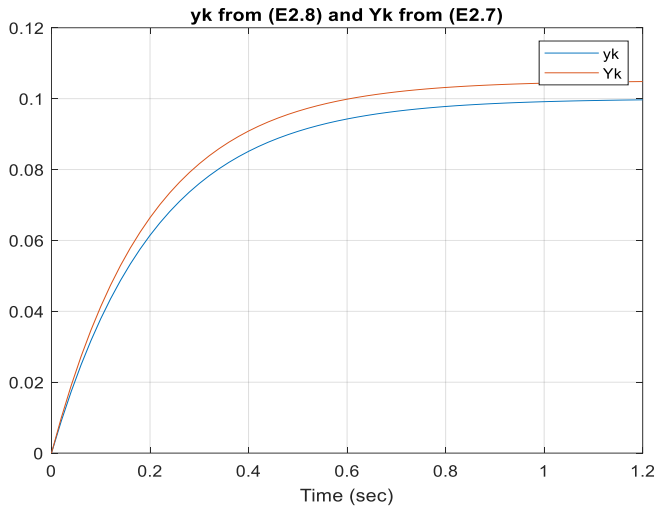

Figure E2.2 Step responses for (E2.7) multiplied by T and (E2.8).

It should be noted that when T=0.005 was used, the plots associated with Figure E2.2 were identical. □

2. Using Matlab's e-Table of z-Transforms

The last two examples illustrated a couple of points in relation to Table 8.1 in the book. First, it is by no means comprehensive. Second, it omits a factor of *T*. It turns out that Matlab has an e-table; though it is not promoted as such. In this section we will address some of the nuts and bolts associated with z-transforms; specifically, in relation to the Matlab command 'c2d'. The arguments of this command are c2d(G,T,'method'). The acronym 'c2d' refers to continuous-to-discrete'. The first argument 'G' refers to the continuous-time transfer function $G(s)$ for which a discrete-time approximation $G(z)$ is desired. The second argument T is the specified sampling period (in seconds). The third argument 'method' might be 'impulse', 'zoh', or 'tus'. These are three of many methods of computing a z-transform.

It is worth repeating that almost every textbook in feedback controls includes a table of Laplace to z-transforms. Moreover, that table is based on only one method; namely the 'impulse' method. If one desires to use another method, then one must work to arrive at the desired *z*-transform. The beauty of Matlab is that it does the work for you. It allows us to spend more time trying to understand the methods, as opposed to how to carry them out.

The 'impulse' method of Computing a z-transform

For a transfer function $G(s)$, denote the corresponding impulse response function as $g(t)$, and denote its sampled version as $g(kT) \overset{\Delta}{=} g_k$. The z-transform of this sampled impulse response is

$$G(z) = \sum_{k=0}^{\infty} g_k z^{-k} \,. \tag{1}$$

Note that we have not included the factor of T in (1). This is because most tables, including Table 8.8 in the book do not. Even so, we *will* need to eventually include it. It is also worth recalling that the variable z is a dummy variable. In fact, $z = e^{sT}$. Yes, *z* is a function of *s* and *T*. If one can keep this in mind, then the z-transform (1) can be viewed for what it is:

$$G(z) = \sum_{k=0}^{\infty} g_k z^{-k} = \sum_{k=0}^{\infty} g(kT) e^{-s(kT)} = (1/T) \sum_{k=0}^{\infty} g(kT) e^{-s(kT)} T \cong (1/T) \int_0^{\infty} g(t) e^{-st} dt = G(s) \,. \tag{2}$$

In words, (1) is a Riemann-sum approximation of the Laplace transform, scaled by 1/*T*. Because it is based on the sampled system impulse response, it is called the ***impulse-invariant*** *z*-transform. This is what the flag 'impulse' refers to. The reason for this name is that the impulse response associated with $G(z)$ is $g(kT)$. In word, the sampled system impulse response lies along the continuous system impulse response $g(t)$. Another way of saying this is that the sampled system impulse response matches the continuous one at the sample times.

Example 3 Consider once again the transfer function

$$\frac{Y(s)}{F(s)} = G(s) = \frac{1}{2s+10} = 0.5\left(\frac{1}{s+5}\right). \tag{E3.1}$$

In Example 2 we concluded that

$$G(s) = 0.5\left(\frac{1}{s+5}\right) \quad \leftrightarrow \quad G(z) = 0.5T\left(\frac{z}{z-\alpha}\right). \tag{E3.2}$$

For $T = 0.02$ we have $\alpha = e^{-aT} = e^{-5(0.02)} = 0.9048$. It follows from (E3.2) that

$$G(z) = 0.5T\left(\frac{z}{z-\alpha}\right) = \frac{0.01z}{z-0.9048}. \qquad \text{(E3.3)}$$

Consider the following Matlab commands:

T=0.02; Gz=c2d(G,T,'impulse') Gz = 0.01 z / (z - 0.9048). (E3.4)

We see that the c2d command that uses the flag 'impulse') gives the same result as using the book table (including the factor if *T*).

Before leaving this example, let's compare plots of the impulse responses for $G(s)$ and $G(z)$.

A comparison of the impulse responses $g(t)$ and $g(kT)$ is shown at right.

```
G=tf(1,[2,10]);
T=0.02;
Gz=c2d(G,T,'impulse');
figure(30)
impulse(G,Gz,'r')
title(['Impulse Responses g(t) and g(kT) for T= ',num2str(T),'.'])
legend('g(t)','g(kT)')
grid
```

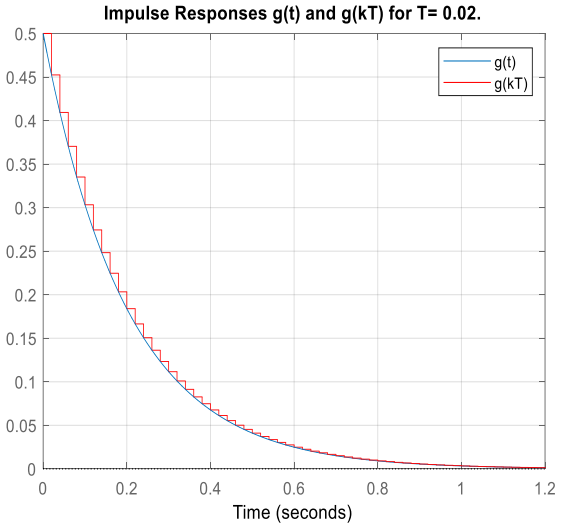

Figure 3.1 Impulse responses $g(t)$ and $g(kT)$.

What you should find strange, to say the least, is that $g(kT)$ is plotted as a step function not points. $G(z)$ is $\{g(kT)\}_{k=0}^{\infty}$. It is only defined at the sampling times. This can be disturbing because even of one acknowledges this fact, it suggests two possible values for $g(kT)$ for any $k > 0$; the one at the top of the step, and the one at the bottom of the step.

One solution to any confusion is to write one's own plotting code:

```
[g,t]=impulse(G);
[gg,tt]=impulse(Gz);
figure(31)
plot(t,g)
hold on
plot(tt,gg,'*r')
title(['Impulse Responses g(t) and g(kT) for T= ',num2str(T),'.'])
legend('g(t)','g(kT)')
grid
```

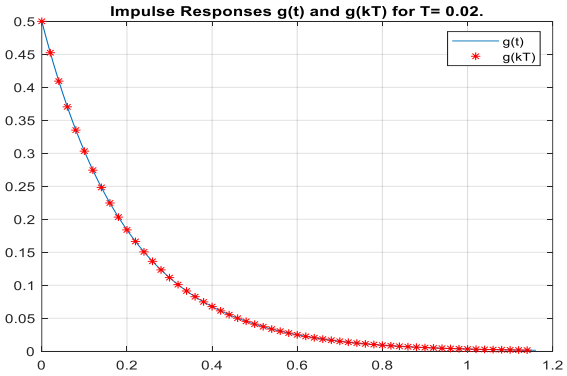

Figure 3.2 Impulse responses $g(t)$ and $g(kT)$.

Which is the preferred plot? The answer to this question depends on the reason for computing the z-transform. If it is to arrive at the sampled system transfer function $G(z)$ using the impulse-invariant method, then Figure 3.2 is the appropriate figure for the sampled system impulse response. On the other hand, if the goal is to arrive at a Riemann sum approximation to the integral in (2) for $s = 0$, then Figure 3.1 is the appropriate on. It clearly show the nature of a Riemann sum approximation. It is a method that uses the left endpoint of each step as the value over the step width, *T*. In

Figure 3.1 we see that the approximation over-estimates the value of $G(s=0)$. In other words, it over-estimates the static gain. This is illustrated in the following Bode plots.

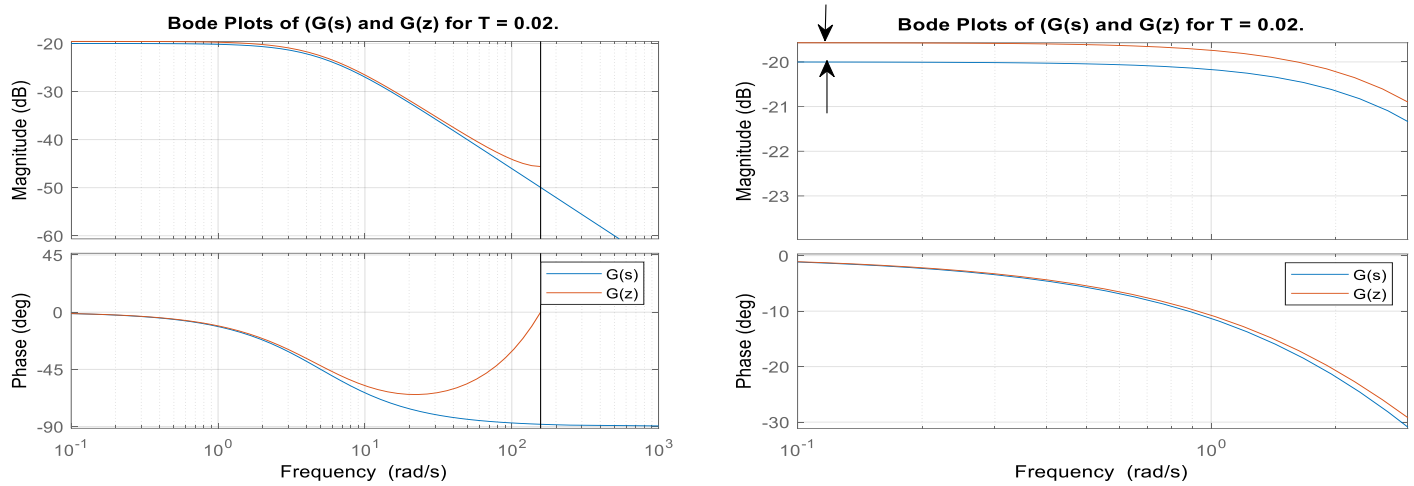

Figure E3.3 Bode plots for $G(s)$ and $G(z)$.

We see that the static gain for $G(z)$ is ~0.5dB higher than for $G(s)$. □

In view of Example 3, we can make the following conclusion.

CONCLUSION 1. The Matlab command c2d(G,T,'impulse') computes the z-transform in the manner computed in the tables of most textbooks. However, unlike most textbook tables, it includes the factor of T.

Remark 1. Even though we have explained it before, it doesn't hurt to explain it again. One of the reasons that most table do not include T is because of the assumption that $G(z)$ is desired for the purpose of matching the discrete and continuous time impulse responses. The discrete-time approximation of the unit impulse $\delta(t)$ that has infinite height, zero width, but area equal to one, is $\delta(kT)=1/T$ for $k=0$, and is zero for $k\neq 0$. This approximation also has area equal to one. Including the factor of T in $G(z)$, and then computing the impulse response cancels that factor out.

More generally, one might ask: How is an input $f(t)$ then modeled in discrete time? The answer is: If can be modeled as a discrete-time pulse train with values $f(kT)/T$. Here again, we see that the T's cancel out. Hence, one can simply use the input $f(kT)$ and not include the factor of T in $G(z)$. □

Matlab Code

```
%PROGRAM NAME: lec22.m     4?16/20
%Example 1.1
G=tf(1,[2,10]);
figure(10)
tmax=1.2;
step(G,tmax)
T=0.02;
a=2/(2+10*T);
t=0:T:tmax;
nt=length(t);
k=1:nt;
y=0.1*(1-a.^k);
hold on
plot(t,y,'r')
title(['Step Responses y(t) and y(kT) for T=',num2str(T),' sec.'])
grid
legend('y(t)','y(kT)')
%Recursive Solution:
y=zeros(1,nt);
d=10+2/T;
for k=2:nt
    y(k)=((2/T)*y(k-1)+1)/d;
end
Y=zeros(1,nt);
f=ones(1,nt);
aa=exp(-5*T);
for k=2:nt
    %Y(k)=aa*Y(k-1)+0.5*f(k);
    Y(k)=aa*Y(k-1)+T*0.5*f(k);
end
figure(20)
plot(t,[y;Y])
title('yk from (E2.8) and Yk from (E2.7)')
legend('yk','Yk')
grid
xlabel('Time (sec)')
%==========================================
%Example 3
G=tf(1,[2,10]);
T=0.02;
Gz=c2d(G,T,'impulse');
figure(30)
impulse(G,Gz,'r')
title(['Impulse Responses g(t) and g(kT) for T= ',num2str(T),'.'])
legend('g(t)','g(kT)')
grid
%------------------------
figure(31)
[g,t]=impulse(G);
[gg,tt]=impulse(Gz);
figure(31)
plot(t,g)
hold on
plot(tt,gg,'*r')
title(['Impulse Responses g(t) and g(kT) for T= ',num2str(T),'.'])
legend('g(t)','g(kT)')
grid
%------------------------
figure(32)
bode(G,Gz)
title(['Bode Plots of (G(s) and G(z) for T = ',num2str(T),'.'])
legend('G(s)','G(z)')
grid
```