Digital Representations of Analog Systems for Control System Applications

Peter Way
May 21, 2009

This tutorial shows how digital controllers can be created from an analog (or continuous) model. This is especially useful when a processor is used to control an analog system. Note that this process works equally well for converting filters or compensators from analog to digital form. Many methods are available to convert between analog and digital systems. I have tried several others and selected two approaches that model the analog system well: Tustin’s Method and Hold Equivalence. These are evaluated to show how they compare with an analog system. These methods were tedious to implement in the past, and are now easily managed by tools for doing algebra symbolically, or solving the equations in matrix form. This makes the job much easier and more accurate. Just a modest amount of theory is presented to help understanding of the methods. If you want to see the full details and derivations of formulas, consult a digital control textbook\(^3\). This application note is somewhat introductory but the techniques can also be applied for advanced solutions.

Digital representation of systems

![Figure 1](image-url)
Processors sample analog (continuous) system signals and convert the values to numbers they can work with. Typically an Analog-to-Digital converter (ADC) takes a reading of the signal at each time \( T \) (the sample time). A series of these values are held in the processor. Another common input device is a rotary or linear encoder. In this case pulses are counted as the encoder moves. At each time \( T \), the present count is added/subtracted to get a new position depending on direction of rotation. Velocity is proportional to the number of counts received during each time \( T \). Once the desired calculations are made in the processor the numbers are reconstructed into a signal that the rest of the system can use. If an analog output is desired, the simplest solution is to output the current value through a Digital to Analog converter (DAC) and hold it until a new one arrives. This implies a Zero Order Hold (ZOH) and is the most common reconstructor. Another popular way to reconstruct the signal is through a Pulse Width Modulator (PWM). This modulator makes a constant frequency square wave and varies the width of the pulse. The benefit of a PWM is that it can efficiently drive power transistors to move motors or actuators. Many processors have built-in PWM and DAC outputs and ADC converters. SimApp\(^4\) is a simulation program that was used to make all the models in this paper.

There are two main options for designing a digital controller:

**Method 1**: Start by designing an analog controller. Then convert it to digital form including the effects of sampling the analog plant and holding the output at the DAC. This approach is most intuitive for the designer since you are working in the familiar s-domain (Laplace domain). The key is to make a digital equivalent of the controller (green box in Figure 1) that really represents the analog controller well.

**Method 2**: Start by converting the Plant, DAC/ZOH and ADC to an effective digital form. Then design a digital controller that makes this system perform as desired. This method may have some advantages when the plant contains a delay element since this is naturally modeled in digital form. This model is not as intuitive since the dynamic specifications for the system need to be defined in the discrete domain (z-domain). This usually begins with the s-domain specifications and then the poles/zeros of the solution are mapped to the z-domain.

We will assume Method 1 in this application note. You may have done a compensator design, either as a PID or a more sophisticated one. Since compensators are essentially filters, the discussion will show how to convert an example analog filter to a digital one. Then the whole system can be simulated to ensure the right results are achieved.

**Discrete Equivalents - the Z Transform - simpler that it would seem**

The job of the processor is to read in inputs (at the ADC) at each sample time \( T \):

\[
 u_0, u_1, \ldots, u_{n-1}, u_n ,
\]

and then to convert them to outputs over time (to the DAC):

\[
 y_0, y_1, \ldots, y_{n-1}, y_n .
\]

The subscript \( n \) represents the current time and \( n-1, n-2 \) etc. others happen earlier at time intervals \( T \). The processor only keeps the last few values required to make the calculation of the output.

In the processor we may want to make an output from the inputs by a difference equation like this, where \( a \) and \( b \) values are constants:

\[
 y_n = a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} + \ldots + b_0 u_n + b_1 u_{n-1} + b_2 u_{n-2} + \ldots
\]
This is a general form for a digital filter with a single input/output. It is in the form of an equation that can be easily implemented in a processor. Note that if we make the \( a \) values 0, and \( b_n = b_{n-1} = b_{n-2} = 1/3 \), the result is a moving average of three values (a very simple digital filter).

Two important points for implementation:
1. The difference equation is assumed to be calculated instantly when the timer goes off and \( u_n \) is measured. The output \( y_n \) goes immediately to the DAC. Naturally the calculations take some time. You can precalculate items that depend on past values during the last time sample and then just add the new information \( b_0u_n \) once the timer interrupt occurs. If the calculation delay \( T_d \) between measurement and output is significant, you can model an extra small delay \( e^{-sT_d} \) delay into the model of the analog plant being controlled.
2. Many of the calculations involved in digital control should be done with floating point. While integer or byte-level math can be done, it can take a lot of effort to implement the equations for numerical accuracy. Why not use a faster processor if possible?

Understanding the Z transform
If you remember the basics of Laplace transforms (frequency domain), a delay of time \( T \) is represented by multiplication with \( e^{-sT} \). Please review a control system text if you are interested in knowing how this works. So, a way of dealing with the inputs and outputs above and providing the correct relationship between them is a number of fixed time \( T \) delays. Note that \( u_0 \) is \( T \) earlier than \( u_1 \) and so on to higher subscripts. A general way to say it: \( u_{n-k} \) is \( kT \) earlier than \( u_n \). If the current time is \( u_n \), the delay from a previous time is written in the frequency domain is: \( e^{-skT} \). This is the same as \( (e^{-sT})^k \). Now to simplify this notation, we will use: \( z^{-k} = e^{-skT} \). (Who wants to write that exponential all the time?). \( e^{-skT} \) is then equal to \( z^{-k} \). You will also see \( z = e^{sT} \) in books since this is mathematically correct (but would require you to have time machine to travel to the future). So the equations always have to be rearranged so that the exponents are negative or 0 before implementation).

Now you can see how to convert any filter written in the time domain to a Z transform version:

\[
Y(z) = (a_1z^{-1} + a_2z^{-2} + a_3z^{-3} \ldots)Y(z) + (b_0 + b_1z^{-1} + b_2z^{-2} \ldots)U(z)
\]

It is very easy to convert between the time representation and the Z-transform. You just replace each delay subscript \( -k \) with the appropriate \( z^{-k} \). Variables \( U \) and \( Y \) are capitalized to denote z transformed variables. A transform from the time domain to the z domain can be understood (at least by me) as: Make me a function that defines a new relationship between the variables \( y \) and \( u \) so that all the timing between the values are properly maintained - and simplifies the analysis to be an algebra problem. This is analogous to the role of the Laplace transform for continuous systems.

Rearranging this:

\[
\frac{Y(z)}{U(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2} \ldots}{1 - a_1z^{-1} - a_2z^{-2} - a_3z^{-3} \ldots}
\]
This looks a lot like a transfer function in the s-domain. It has many of the same properties. For example, you can make the Z transform of an input signal \( U(z) \), multiply it by the transfer function, and get the output. Then you can make the output signal by converting it back to a time series of values. Why go through all this? This would have been a much more complicated exercise if you did it in the time domain. The polynomials can also be manipulated to understand the dynamics of the system.

Note that the coefficients of the Z transform are NOT the same as the ones for an equivalent Laplace transform representation. It is this conversion process that we're going to discuss next. The Laplace transform of the controller in Figure 1 needs to be converted to a difference equation in the controller. This needs to include the effect of sampling the signal from the analog system and the reconstruction of the signal at the output of the DAC. The entire green box in Figure 1 needs to be converted to digital form.

Z-transform equivalents - Tustin’s Rule

Let's consider how you might perform an integration. (This is like accumulating velocity to make position so we will use these variables for the example). The frequency domain version of this is: \( 1/s \)

Trapezoidal integration is shown below - we are averaging the Velocity from the last time to the current time, and multiplying by \( T \). Then we add it to the last position we had. Intuitively it makes sense. If you travel an average velocity for a time \( T \), your position will change as shown below.

Velocity \( V_n \)
Position \( P_n \)
\[ P_n = P_{n-1} + T \frac{(V_n + V_{n-1})}{2} \]

Taking the Z transform of this:
\[ P(z) = P(z)z^{-1} + \frac{T}{2} V(z)(1 + z^{-1}) \]

Rearranging:
\[ \frac{P(z)}{V(z)} = \frac{T}{2} \frac{(1 + z^{-1})}{(1 - z^{-1})} \]

since this is an approximation of \(1/s\) integration,
Tustin probably thought: why don't we replace \(s\) in transfer functions as follows?
\[ s \approx \frac{2}{T} \frac{(z - 1)}{(z + 1)} \]

It turns out that this has wonderful properties and is a great way to make conversions provided the sample time is chosen carefully (more on this later). The algebra can be tedious, however, and it is easy to make mistakes. Fortunately modern tools that can solve symbolic equations are available to do the work. Scientific Notebook\(^1\) was used to make this Application Note and rearrange all these equations.

**Bandpass Filter Example**
The following is a model of a bandpass filter. This could just as well have been an analog compensator. It is of the form:
\[
H(s) = \frac{s}{s^2 + 2\zeta\omega_n s + \omega_n^2}
\]
\[
H(s) = \frac{s}{s^2 + s + 25}
\]

This is a very underdamped example with damping \(\zeta = 0.1\) and natural frequency of \(\omega_n = 5\text{rad/s}\)

In the present example of a bandpass filter, substitute Tustin’s Rule. \(T\) is chosen to match a textbook example with sampling at 3Hz \(\rightarrow T = 1/(2\pi 3) = 0.053\)

Let’s begin with the generic equation for a second order bandpass filter, replace \(s\) with Tustin’s formula and rearrange collecting terms. Then the equation is rearranged as a ratio of two polynomials.

\[
H(z) = \left[ \frac{s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \right]_{s = \frac{2(1-z^{-1})}{T(1+z^{-1})}} = \frac{1}{T} \left( \frac{2z - 2}{(z + 1)} \left( \frac{\omega_n^2 + \frac{1}{T^2} \frac{(2z-2)^2}{(z+1)^2} + \frac{2}{T}\zeta\omega_n\frac{2z-2}{z+1}}{(T^2 \omega_n^2 + 4\zeta T\omega_n + 4)z^2 + (2T^2 \omega_n^2 - 8)z + (T^2 \omega_n^2 - 4\zeta T\omega_n + 4)} \right)
\]

Now assign values
\(\zeta = 0.1\)
\[ \omega_n = 5 \]
\[ T = 0.053 \]
\[ H(z) = \frac{z^2 - 1.0}{39.398z^2 - 74.147z + 37.398} \]

Remember to convert the equation before implementation to something with only negative exponents by diving numerator and denominator by \( z^2 \) (so you don’t need a time machine).

**A word about sample time T**

T defines the sampling frequency \( \omega_s = 1/T \).
It is important to make \( \omega_s/\omega_n > 2 \) to avoid aliasing. \( \omega_n \) in this example is the natural frequency of the system. In other systems it can represent the bandwidth or the highest frequency for which the dynamics need to be rendered accurately. The current example was developed at a \( \omega_s/\omega_n = 3.8 \). In practice \( 6 < \omega_s/\omega_n < 15 \) is a good range. It is generally good to make \( \omega_s \) as small as possible relative to the dynamics of the system being converted to digital form. As \( \omega_s \) increases, Tustin’s rule makes a more precise rendition of the analog system. Beyond a certain point, however, the system will exhibit numerical problems in the computer simulation, not to mention the implementation in a processor. Lower \( \omega_s \) provides coarser control input, lower accuracy of the approximation and the processor time can be used for other tasks.

A great benefit of simulating the system is that you can see exactly what the effects of sampling will be. The frequency plots accurately model the effect of the sampling delays and plot the entire system as if it were a continuous system. This way you can easily compare the original analog system with its digital representation. The time simulations show the quantization and delays of the digital output.

Tustin’s rule may be all you need to convert analog to digital systems. It is important to realize that this method is quite sensitive to selection of the sample period. If you want a more comprehensive approach especially for more complex systems, consider the more advanced methods below.

**Discrete equivalent by Zero Order Hold Equivalence**
The idea behind this method is to attempt to make the outputs of the continuous system and the digital representation equivalent during the sample points. Then the digital system holds the value until the next output by using the Zero-Order-Hold (ZOH).

We know that any stairstep input waveform \( u_n, u_{n-1}, u_{n-2}, \ldots \) can be decomposed into a series of delayed step functions (the steps are not equal to the \( u_n \) values but that does not matter for what follows.) Since we are assuming a linear system, the output of the system is the sum of the output of all these steps acting individually on the system. So... in order to make the systems equivalent, the step response of both systems should match. Then every other delayed step response will also match.

In the s-domain, a step input is represented by \( 1/s \). In the z-domain, a step is represented by \( z^{-1} \). Remember that for Laplace and Z transforms, the output is equal to the transform...
of the input times the transfer function. If we want the outputs to match, we would write:

\[ \frac{z}{z-1} H(z) = \frac{1}{s} H(s) \rightarrow H(z) = \frac{z-1}{z} Z[H(s)/s] \]

\( Z[ \cdot ] \) means "take the Z transform"

In the bandpass example

\[ H(z) = \frac{z-1}{z} Z\left[ \frac{1}{s^2 + s + 25} \right] \]

First put the polynomial in \( s \) in a form that fits a formula available in the Z-transform table. There is one that matches:

\[ H(s) = \frac{b}{(s + a)^2 + b^2} = \frac{b}{s^2 + 2as + (a^2 + b^2)} \]

\[ a^2 + b^2 = 25 \]
\[ 2a = 1 \]

The solutions of these equations are: \( [a = \frac{1}{2}, b = \frac{3}{2} \sqrt{11}] \), \( [a = \frac{1}{2}, b = -\frac{3}{2} \sqrt{11}] \)

\[ a = 0.5 \]
\[ b = 4.9749 \]
\[ T = 0.053 \]

The Z transform for \( H(s) \) above is found in the table:

\[ H(z) = \frac{z-1}{b^2} \frac{ze^{-aT} \sin(bT)}{z^2 - 2e^{-aT} \cos(bT) + e^{-2aT}} = \frac{1.0z - 1.0}{19.601z^2 - 36.858z + 18.589} \]

This method is fine for simple calculations, but can be very time consuming for larger systems. In that case, you have to make expansions of all the terms and find suitable z-transforms. This example was included in order to illustrate the technique and to make the key point that the systems are made equivalent by matching their output to one step. Fortunately, there is a method that is more computer-friendly below.

**Discrete Equivalent by State Space Methods**

This method is the same as the Zero Order Hold above, but is more general. First, we have to represent a system in a matrix format. This is also called the state-space representation, and is a cornerstone of modern control methods. The state-space method is especially good for complex systems with many inputs and outputs or coupling between the equations.

First, lets make descriptions of transfer functions in State-Space form. There are many ways to do this for a general system. One easy choice for a polynomial transfer function is
the control canonical form. See a modern control textbook for details.

A continuous transfer function we want to convert is:

\[
H(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + b_2 s^{n-2} \ldots b_n}{s^n - a_1 s^{n-1} - a_2 s^{n-2} - a_3 s^{n-3} \ldots a_n}
\]

This function can be represented as follows (note that state space equations are written in time with \(x\) as a state vector and \(x'\) the derivative of the state vector. We have made a little leap here. We were starting with an s-domain version and jumped to a time version. The time equations are easily converted to the frequency domain and back again if you replace \(x' \rightarrow sX(s), y \rightarrow Y(s), u \rightarrow U(s)\)

<table>
<thead>
<tr>
<th>Time version</th>
<th>Frequency version</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x' = Ax + Bu)</td>
<td>(sX(s) = AX(s) + BU(s))</td>
</tr>
<tr>
<td>(y = Cx + Du)</td>
<td>(sY(s) = CX(s) + DU(s))</td>
</tr>
</tbody>
</table>

\(x, u, y\) are vectors, and \(A, B, C, D\) are matrices.

The control canonical form is easy to write from the polynomial above (and a lot of work to derive, so let Dr. Ogata\(^2\) do the math for you). The A matrix coefficients are just the denominator polynomial coefficients. The C matrix takes a little more effort to calculate according to the formulas below.

\[
A = \begin{bmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ldots & 1 \\
-a_n & -a_{n-1} & -a_{n-2} & \ldots & -a_1
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 \\
0 \\
0 \\
\vdots \\
1
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
b_n - a_n b_0 & b_{n-1} - a_{n-1} b_0 & \ldots & b_1 - a_1 b_0
\end{bmatrix}
\]

\[D = b_0\]

SimApp has polynomial and state space blocks available, so you can use simulation to check that your state space model is equal to the original transfer function.

A similar description can be developed for a matrix difference equation:

\[
x_{n+1} = \Phi x_n + \Gamma u_n
\]

\[
y_n = C x_n + D u_n
\]

\(x, u, y\) are state vectors, and \(\Phi, \Gamma, C, D\) are matrices. Note that \(C, D\) are the same as for the original continuous system. The new dynamics are wrapped up in \(\Phi\) and \(\Gamma\). Would it not be great if you could make a Zero Order Hold version of the continuous system just by calculating new matrices? That is what we will do next - again leaving the math to the controls textbooks.

\(\Phi, \Gamma\) can be calculated from the continuous system as follows\(^3\):
Let 
\[ \Psi \approx I + \sum_{n=1}^{7} \frac{A^n T^n}{(n+1)!} \]
This is a series approximation and should be accurate enough with 7 terms. For better numerical properties you can calculate it as follows:

\[ \Psi \approx I + \frac{A T}{2} \left( I + \frac{A T}{3} \left( \ldots I + \frac{A T}{N} (I + \frac{A T}{N}) \right) \right) \]
\[ \Phi = I + A T \Psi \]
\[ \Gamma = \Psi T B \]
\[ I \quad \text{is the identity matrix} \]

Although these equations look complicated, they are very easy to implement with any matrix package or Scientific Notebook.

Now evaluate the bandpass example. First write the state-space continuous system in control canonical form.

\[ A = \begin{bmatrix} 0 & 1 \\ -25 & -1 \end{bmatrix} \]
\[ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]
\[ C = \begin{bmatrix} 0 & 1 \end{bmatrix} \]
\[ I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

Calculate the matrices.
\[ T = 0.053 \]
\[ \Psi = I + \sum_{n=1}^{7} \frac{A^n T^n}{(n+1)!} = \]
\[ \Psi = \begin{bmatrix} 0.98849 & 2.5886 \times 10^{-2} \\ -0.64716 & 0.9626 \end{bmatrix} \]
\[ \Phi = I + F T \Psi = \begin{bmatrix} 0.9657 & 5.1018 \times 10^{-2} \\ -1.2754 & 0.91468 \end{bmatrix} = \]
\[ \Gamma = \Psi T G = \begin{bmatrix} 1.3720 \times 10^{-3} \\ 5.1018 \times 10^{-2} \end{bmatrix} \]
This difference equation is the ZOH equivalent system. Each of these state space results can be entered into state space blocks for discrete and continuous systems. Note that SimApp uses a block construction that neatly puts all the information into one matrix.

The model:

**Original Bandpass Filter** $H(s) = \frac{s}{(s^2+s+25)}$

Below are the frequency responses of the bandpass filter with all these methods in one plot.
Comparison of Frequency Responses

<table>
<thead>
<tr>
<th>Digital SS</th>
<th>Amplitude [dB]</th>
<th>Phase</th>
<th>ZOH transform</th>
<th>Amplitude [dB]</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-17.53</td>
<td>-97.734</td>
<td></td>
<td>-17.526</td>
<td>-97.73</td>
</tr>
<tr>
<td>Analog SS</td>
<td>Amplitude [dB]</td>
<td>Phase</td>
<td>Analog1</td>
<td>Amplitude [dB]</td>
<td>Phase</td>
</tr>
<tr>
<td></td>
<td>-17.578</td>
<td>-82.405</td>
<td></td>
<td>-17.578</td>
<td>-82.405</td>
</tr>
<tr>
<td>Tustin</td>
<td>Amplitude [dB]</td>
<td>Phase</td>
<td>Tustin</td>
<td>Amplitude [dB]</td>
<td>Phase</td>
</tr>
<tr>
<td></td>
<td>-17.908</td>
<td>-82.691</td>
<td></td>
<td>-17.908</td>
<td>-82.691</td>
</tr>
</tbody>
</table>

Magnitudes agree well with G(s)

Tustin phase response is more accurate than ZOH

Tustin breaks down at $\omega T = \pi$
Comparison of Time Responses

Note that ZOH is exact at the sample points and then holds

Tustin responds immediately to prior sample

References

1 Notebook is a program that can be used to write technical papers and solve equations in a symbolic or numerical form. It is available from MacKichan software at www.mackichan.com
2 Modern Control Engineering (4th Edition), Katsuhiko Ogata
3 Digital Control of Dynamic Systems (1980), Gene Franklin and David Powell
4 SimApp is a block diagram based dynamic simulation program. It is available at www.simapp.com

© VentiMar, LLC
www.ventimar.com