z-Transforms

Introduction:

Anything you can do in software you can do in hardware, and visa versa.

In Circuits II and Electronics II, we deisgn filtes in the s-plane. These include

- RC low-pass filters,
- RLC band-pass filters, and
- Active filters

To describe these filters, differential equations are used. This results in transfer functions in 's' where

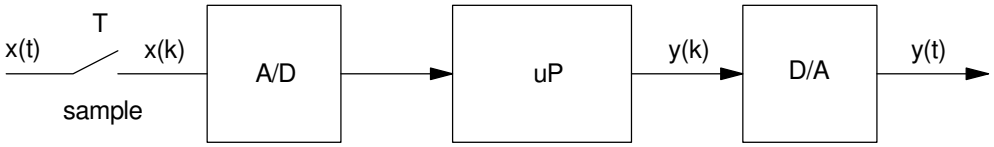
sY

means

the derivative of y(t)

In contrast, with a microcontroller software is used to produce the output of the digital filter. Here, the microcontroller

- Samples the error every T seconds
- The sampled signal is sent through an analog to digital (A/D) converter
- A program on the microcontroller computes the output (software), and then
- The output of the microcontroller is sent to a digital to analog (D/A) converter, producing y(t)



Using a microcontroller has several advantages:

- Code that ran yesterday should also run today.
- DC offsetts don't exist in software. Zero plus zero is zero.
- If you want a more complex controller, you just add lines of code.
- If you want to change the controller, you just download a new program.

The problem with using a microcontroller, however, is that LaPlace transforms don't work well when dealing with coding. For example, a typical digital filter will look like the following:

```
while(1) {

    k = k + 1;                          // k = iteration number

    x1 = x0;                            // x(k-1)
    x0 = A2D_Read(0);                   // read in x(k) from the A/D

    y1 = y0;                            // y(k-1)
    y0 = y1 + 0.2*(x0 - 0.9*x1);

    Wait_10ms();

    }
```

Note that this program implements a difference equation: the value of y at iteration k is equal to

$$y(k) = y(k-1) + 0.2(x(k) - 0.9x(k-1))$$

This is typical of any program: the value of the variables at any time are a function of the previous values. The reason we don't use LaPlace transforms for coding is that LaPlace transforms are designed for solving differential equations. To solve a difference equation, we need a different tool: that tool is the z-transform.

Recall that the reason we use LaPlace transforms when solving differential equations is they turn differential equations into algebraic equations in 's', and algebra is easier than calculus. The basic assumption behind LaPlace transforms is that all functions are in the form of

$$y = e^{st}$$

This results in differentiation becoming multiplication by 's'

$$\frac{dy}{dt} = s \cdot e^{st} = sY$$

Here 'sY' means "the derivative of Y". This allows us to convert differential equations into algebraic equations in s. For example, the transfer function

$$Y = \left(\frac{8s+3}{s^2+7s+12}\right)X$$

means the differential equation which describes the relationship between X and Y is

$$(s^2 + 7s + 12)Y = (8s + 3)X$$

or

$$\frac{d^2y}{dt^2} + 7\frac{dy}{dt} + 12y = 8\frac{dx}{dt} + 3x\,.$$

In contrast, the z-transform converts difference equations in to algebraic equations in 'z'. The basic assumption behind the z-transform is that all functions are in the form of

$$y = z^k$$

where time is equal to kT and T is the sampling rate. (Think of it as a microcontroller with an interrupt being called every T seconds. 'k' is how many times the interrupt has been called.) With this assumption, the next value of y will be

$$y(k+1) = z^{k+1} = z \cdot z^k = z \cdot y(k)$$

Here, 'zY' means "the next value of Y." This in turn converts difference equations into algebraic equations in z.

Implementing K(z) in Software

In the s-domain, sY means "the derivative of Y"

In the z-domain, zY means "the next value of Y" or "y(k+1)"

If you have a filter

$$Y = K(z)X$$

or

$$Y = \left(\frac{a_2 z^2 + a_1 z + a_0}{z^3 + b_2 z^2 + b_1 z + b_0} \right) X$$

with a sampling rate of T seconds, you can implement this filter in code as follows.

i) Cross multiply:

$$(z^3 + b_2 z^2 + b_1 z + b_0)Y = (a_2 z^2 + a_1 z + a_0)X$$

ii) Convert back to the time domain, noting that zY means y(k+1):

$$y(k+3) + b_2 y(k+2) + b_1 y(k+1) + b_0 y(k) = a_2 x(k+2) + a_1 x(k+1) + a_0 x(k)$$

This is the difference equation which relates X and Y.

One problem with this answer is it uses future values of each variable and the future is rather hard to program. To solve this, do a change of variable (or delay everything by 3), resulting in

$$y(k) + b_2 y(k-1) + b_1 y(k-2) + b_0 y(k-3) = a_2 x(k-1) + a_1 x(k-2) + a_0 x(k-3)$$

This is also the difference equation which relates X and Y (both are correct). This one is easier to program, however, since it uses past values of X and Y rather than future values.

iii) Solve for y(k)

$$y(k) = -b_2 y(k-1) - b_1 y(k-2) - b_0 y(k-3) + a_2 x(k-1) + a_1 x(k-2) + a_0 x(k-3)$$

iv) Write this in code:

That's essentially your program. Note that you have to remember the previous outputs and inputs. One way to do this is

```
while(1) {

    x3 = x2;              // x(k-3)
    x2 = x1;              // x(k-2)
    x1 = x0;              // x(k-1)
    x0 = A2D_Read(0);     // read x(k) from the A/D

    y3 = y2;              // y(k-3)
    y2 = y1;              // y(k-2)
    y1 = y0;              // y(k-1)

    y0 = -b2*y1 - b1*y2 - b0*y3 + a2*x1 + a1*x2 + a0*x3;

    D2A(y0);              // output y(k) to the D/A converter
    Wait_10ms();

    }
```

Example 2: Implement the following filter. Assume a sampling rate of 10ms.

$$Y = \left(\frac{0.2z(z-0.9)}{(z-1)(z-0.5)} \right) X$$

Solution: Multiply it out

$$Y = \left(\frac{0.2(z^2-0.9z)}{z^2-1.5z+0.5} \right) X$$

Cross multiply and solve for the highest power of zY

$$(z^2 - 1.5z + 0.5)Y = 0.2(z^2 - 0.9z)X$$

$$z^2 Y = (1.5z - 0.5Y + 0.2(z^2 - 0.9z)X$$

meaning

$$y(k) = 1.5y(k-1) - 0.5y(k-2) + 0.2(x(k) - 0.9x(k-1))$$

In code, only one line changes

```
while(1) {

   x2 = x1;              // x(k-2)
   x1 = x0;              // x(k-1)
   x0 = A2D_Read(0);     // read in x(k) from the A/D

   y2 = y1;              // y(k-2)
   y1 = y0;              // y(k-1)

   y0 = -1.5*y1 +0.5*y2 + 0.2*(x0 - 0.9*x1);

   Wait_10ms();

   }
```

There are several things to note here:

- Filters in the z-domain can be implemented exactly in software. That isn't true in the LaPlace domain.
- To change the filter, you just change one line of code. That's much easier than building a new op-amp filter.
- Complex poles and zeros are not a problem in the z-domain. All you care about are the coefficients in the numerator and denominator polynomials.
- If you have a 3rd-order filter, you need to remember the 3 previous values of the inputs and outputs. A 4th-order filter remembers the 4 previous values.

One other important thing to note:

- In the s-domain, we don't like to have more zeros than poles. More zeros than poles means you're differentiating the input. This tends to create a noise amplifier.
- In the z-domain, you cannot have more zeros than poles. More zeros than poles means you're using future values of the input - which I don't know how to do.

Also also

- You have to have integer powers of s. $s^{1/2}Y$ means "the half-derivative of Y". I have no idea what a half-derivative is. $s^{1/2}Y$ doesn't make sense.
- You have to have integer powers of z. $z^{1/2}Y$ means "the value of Y next time you half-call the subroutine." I know how to call a subroutine one time. I know how to call it two times. I don't know how to call a subroutine half a time. $z^{1/2}Y$ doesn't make sense either.

Find the response of K(z) for a sinusoidal input

To solve for the output with a sinusoidal input, you use phasor analysis. In the s-domain, you let

$$s = j\omega.$$

In the z-plane, you need a conversion from the s-plane to the z-plane.

The basic assumption behind LaPlace transforms is that all functions are in the form of

$$y(t) = e^{st}$$

Assume a fixed sampling rate of T seconds so that

$$t = kT$$

and k is the sample number. Then

$$y(kT) = e^{skT}$$

or

$$y(k) = (e^{sT})^k$$

This is identical to the assumption behind z-transforms.

$$z = e^{sT}$$

Hence, to find the output for a filter with a sinusoidal input, you let

$$z = e^{sT} = e^{j\omega T}$$

(note: TI calculators need to be in radian mode for this to work.)

Example: Find the output of the following system

$$Y = \left(\frac{20}{(s+1)(s+5)}\right) X$$

where

$$x(t) = 3\sin(4t)$$

Solution: Evaluate at s = j4

$$Y = \left(\frac{20}{(s+1)(s+5)}\right)_{s=j4} (0 - j3) = -2.066 + j0.947$$

giving

$$y(t) = -2.066\cos(4t) - 0.947\sin(4t)$$

Example: Find the output of the following system with T = 0.01s (10ms)

$$Y = \left(\frac{0.02z}{(z-0.9)(z-0.8)}\right)X$$

$$x(t) = 3\sin(4t)$$

Solution: Evaluate at

$$s = j4$$

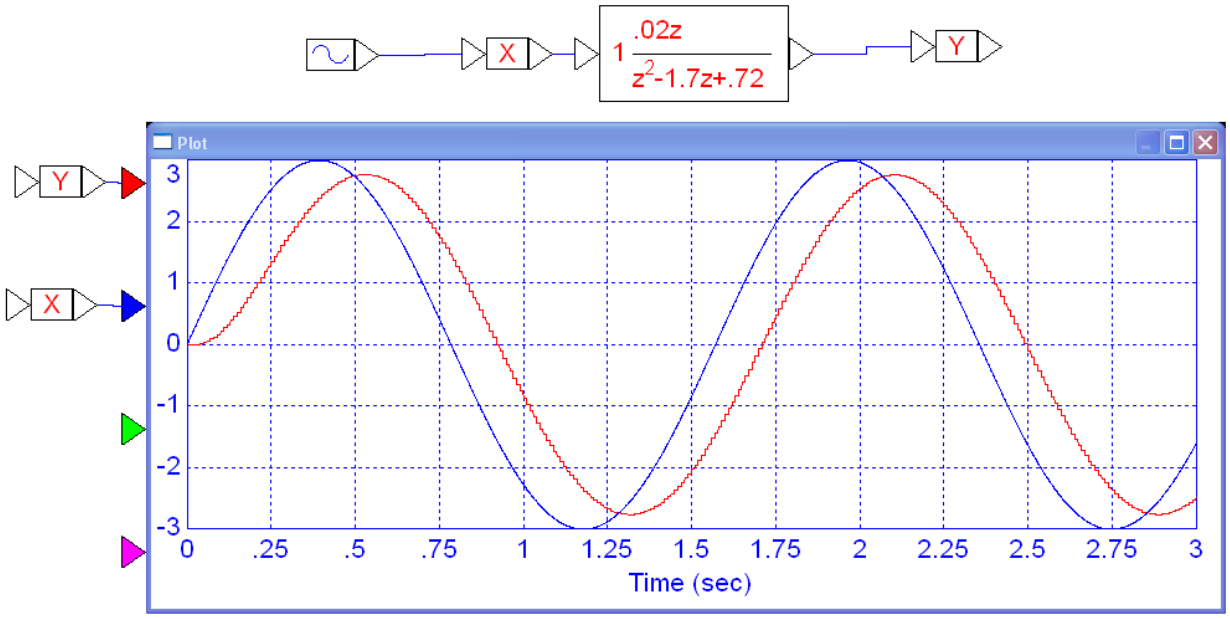
$$z = e^{sT} = e^{j0.04} = 1\angle 2.291^0$$

$$Y = \left(\frac{0.02z}{(z-0.9)(z-0.8)}\right)_{z=1\angle 2.291^0}(0-j3) = -1.423 - j2.366$$

meaning

$$y(t) = -1.423\cos(4t) + 2.366\sin(4t)$$

You can verify this in VisSim:



Response of K(z) to a sinusoidal input. Note that the output is a sine wave with a gain and phase shift, just like we had in the s-domain. Also note that you can see the sampling rate on the output (the stair-steps on the red curve, y(k))

Table of z-transforms

If you want to find the output of a filter G(s) with a step input, you use LaPlace transforms along with a table of LaPlace transforms and partial fraction expansion.

Similarly, if you want to find the output of a filter G(z) with a step iput, you use z-transforms along with a table of z-transforms and partial fraction expansion.

Here we derive that table of z-transforms.

i) Delta Function $\delta(k)$. The discrete-time delta function is

$$\delta(k) = \begin{cases} 1 & k = 0 \\ 0 & otherwise \end{cases}$$

k	0	1	2	3	4	5	6	7
delta(k)	1	0	0	0	0	0	0	0

The z-transform of a delta function is '1', just like the s-domain.

ii) Unit Step: The unit step is

$$u(k) = \begin{cases} 1 & k \geq 0 \\ 0 & otherwise \end{cases}$$

It's z-transform can be derives as follows. The unit step is:

k	0	1	2	3	4	5	6	7
u(k)	1	1	1	1	1	1	1	1
(1/z)*u(k)	0	1	1	1	1	1	1	1
Subtract								
(1-1/z)u(k)	1	0	0	0	0	0	0	0

So,

$$\left(1 - \tfrac{1}{z}\right) u(k) = 1$$

$$\left(\tfrac{z-1}{z}\right) u(k) = 1$$

$$u(k) = \tfrac{z}{z-1}$$

iii) Decaying Exponential. Let

$$x(k) = a^k u(k)$$

k	0	1	2	3	4	5	6	7
x(k)	1	a	a^2	a^3	a^4	a^5	a^6	a^7
a*(1/z)*x	0	a	a^2	a^3	a^4	a^5	a^6	a^7
Subtract								
(1-a/z)x	1	0	0	0	0	0	0	0

so

$$(1 - \tfrac{a}{z})X = 1$$

$$(\tfrac{z-a}{z})X = 1$$

$$X = (\tfrac{z}{z-a})$$

These let you create a table of z-transforms like we had in the s-domain:

function	y(k)	Y(z)
delta	$\delta(k)$	1
unit step	$u(k)$	$\frac{z}{z-1}$
decaying exponential	$a^k u(k)$	$\frac{z}{z-a}$
damped sinewave	$2b \cdot a^k \cdot \cos(k\theta + \phi) \cdot u(k)$	$\left(\frac{(b\angle\phi)z}{z-(a\angle\theta)}\right) + \left(\frac{(b\angle-\phi)z}{z-(a\angle-\theta)}\right)$

For the last entry, assume you have a complex conjugate pair:

$$Y(z) = \left(\frac{(b\angle\phi)z}{z-(a\angle\theta)}\right) + \left(\frac{(b\angle-\phi)z}{z-(a\angle-\theta)}\right)$$

Take the inverse z-transform

$$y(k) = \left((b\angle\phi)(a\angle\theta)^k + (b\angle-\phi)(a\angle-\theta)^k\right)u(k)$$

Simplifying

$$y(k) = b(a^k)\left(e^{j(k\theta+\phi)} + e^{-j(k\theta+\phi)}\right)u(k$$

Using Euler's identity

$$y(k) = 2b(a^k)\left(\frac{e^{j(k\theta+\phi)}+e^{-j(k\theta+\phi)}}{2}\right)u(k)$$

$$y(k) = 2b \cdot a^k \cdot \cos(k\theta + \phi) \cdot u(k)$$

Solving Functions in the z-Domain

Problem 1: Find the step response of

$$Y = \left(\frac{0.2z}{(z-0.9)(z-0.5)}\right)X$$

i) Replace X(z) with the z-transform of a step

$$Y = \left(\frac{0.2z}{(z-0.9)(z-0.5)}\right)\left(\frac{z}{z-1}\right)$$

ii) Use partial fractions to expand this. Note, however, that the table entries have a 'z' in the numerator. So, factor this out first then take the partial fraction expansion

$$Y = \left(\frac{0.2z}{(z-1)(z-0.9)(z-0.5)}\right)z$$

$$Y = \left(\left(\frac{4}{z-1}\right) + \left(\frac{-4.5}{z-0.9}\right) + \left(\frac{0.5}{z-0.5}\right)\right)z$$

Multiply through by z

$$Y = \left(\left(\frac{4z}{z-1}\right) + \left(\frac{-4.5z}{z-0.9}\right) + \left(\frac{0.5z}{z-0.5}\right)\right)$$

iii) Now apply the table entries

$$y(k) = 4 - 4.5 \cdot (0.9)^k + 0.5 \cdot (0.5)^k \qquad k >= 0$$

Problem 2: Find the step response of a system with complex poles:

$$Y = \left(\frac{0.2z}{\left(z-0.9\angle 10^0\right)\left(z-0.9\angle -10^0\right)}\right)X$$

i) Replace X with its z-transforrm (a unit step)

$$Y = \left(\frac{0.2z}{\left(z-0.9\angle 10^0\right)\left(z-0.9\angle -10^0\right)}\right)\left(\frac{z}{z-1}\right)$$

ii) Factor our a z and use partial fractions

$$Y = \left(\left(\frac{5.355}{z-1}\right) + \left(\frac{2.98\angle 153.97^0}{z-0.9\angle 10^0}\right) + \left(\frac{2.98\angle -153.97^0}{z-0.9\angle -10^0}\right)\right)z$$

iii) Convert back to time using the table of z-transforms

$$y(k) = 5.355 + 4.859 \cdot (0.9)^k \cdot \cos\left(10^0 \cdot k - 153.97^0\right) \qquad k >= 0$$

Note that in the s-domain, rectangular coordinates are more convenient

- The real part of s tells you the rate at which the exponential decays
- The complex part of s tells you the frequency of oscillations.

In the z-domain, polar coordinates are more convenient

- The amplitude of z tells you the rate at which the signal decays
- The angle of z tells you the frequency of oscillation.

In this case,

- The signal decays by 10% each sample (0.9)k
- The phase changes by 10 degrees each sample, or requires 36 samples per cycle

Time Value of Money

As a sidelight, you can also solve time-value of money problems using z-transforms.

Assume you borrow $100,000 for a house. How much do you have to pay each month to pay off the loan in 10 years? Assume 6% interest per year (0.5% per month).

Solution: Let x(k) be how much money you owe today. The amount you owe next month, x(k+1), is

$$x(k+1) = 1.005x(k) - p + X(0)\cdot\delta(k)$$

where 'p' is your monthly payment. Converting to the z-domain

$$zX - X(0) = 1.005X - p\left(\frac{z}{z-1}\right)$$

$$(z-1.005)X = X(0) - p\left(\frac{z}{z-1}\right)$$

$$X = \left(\frac{X(0)}{z-1.005}\right) - p\left(\frac{z}{(z-1)(z-1.005)}\right)$$

Using partial fractions

$$X = \left(\frac{X(0)}{z-1.005}\right) + pz\left(\left(\frac{200}{z-1}\right) - \left(\frac{200}{z-1.005}\right)\right)$$

Converting back to the time domain

$$x(k) = 1.005^k X(0) - 200p(1.005^k - 1)u(k)$$

After 10 years (k=120 payments), x(k) should be zero

$$x(120) = 0 = \$181,939 - 200p(0.8194)$$

$$p = \$1110.20$$

Your monthly payments are $1,110.20.

If you stretch this out to 30 years (k = 360 payments), the monthly payment becomes

$$x(360) = 0 = \$602,257 - 200p(5.0226)$$

$$p = \$599.55$$

Paying off the loan over a time span 3 times longer doesn't reduce the payments by 3 times. It's actually only 46% less. The total amount you'll pay on the loan, however, increases from $133,224 to $215,838.

note: That's pretty much all a business calculator is: a calculator which does z-transforms where the keys are renamed "interest rate", "initial loan value" and "number of payments."