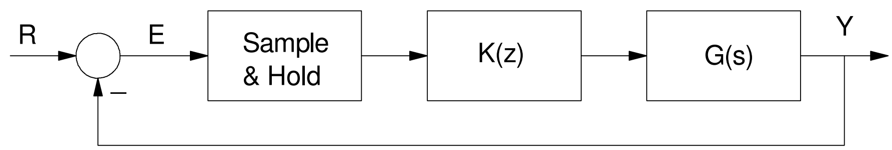
Root Locus in the z-Domain

Discussion:

Relative to the microcontroller, a feedback system looks like it's a discrete-time system. It looks like it's in the z-domain. The goal is to find the compensator, K(z), which gives a 'good' response.



Mathematically, the open-loop transfer function is

$$Y = H(z)G(z)K(z)E$$

where H(z) is the transfer function of the sample and hold. This is approximately a 1/2 sample delay which is often ignored or lumped into G():

$$H() = e^{-sT/2}$$

If you ignore H, the closed-loop transfer function is

$$Y = \left(\frac{GK}{1+GK}\right)R$$

If GK has zeros and poles:

$$GK = k\frac{z}{p}$$

the closed-loop transfer function becomes

$$Y = \left(\frac{kz}{p+kz}\right)E$$

The roots of the closed-loop system are then just as they were in the s-plane:

$$p(z) + kz(z) = 0$$

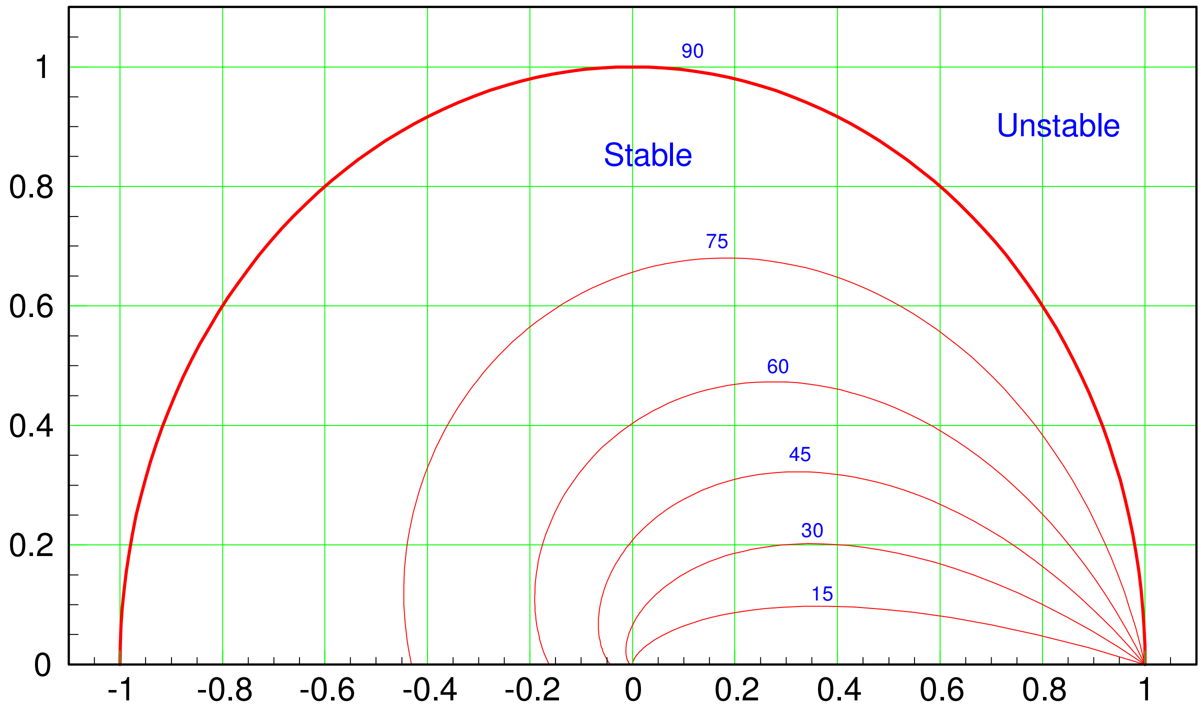
Mathematically, the roots to this polynomial don't care if the variable is 's', 'z', or anything else. The roots (and hence root locus plot) behave just the same in the s-plans as they do in the z-plane.

The only difference in the z-plane is how you interpret the meaning of the pole locations.

Recall the relationship between the s-plane and the z-plane is

$$z = e^{sT}$$

This causes the damping lines in the s-plane to spiral in the z-plane as follows:



The different points on this plane result in different decay rates.

Like the s-plane, where you place the dominant pole in the z-plane determines the response of the closed-loop system. For example, if the dominant pole is on the real axis between (0, 1), the system decays exponentially with a 2% settling time of

$$z^k = 0.02$$

$$k = \frac{\ln(0.02)}{\ln(z)}$$

For example, a pole at z = 0.95 has a 2% settling time of 73 samples (round up)

$$k = \frac{\ln(0.02)}{\ln(0.95)} = 72.26 \text{ samples}$$

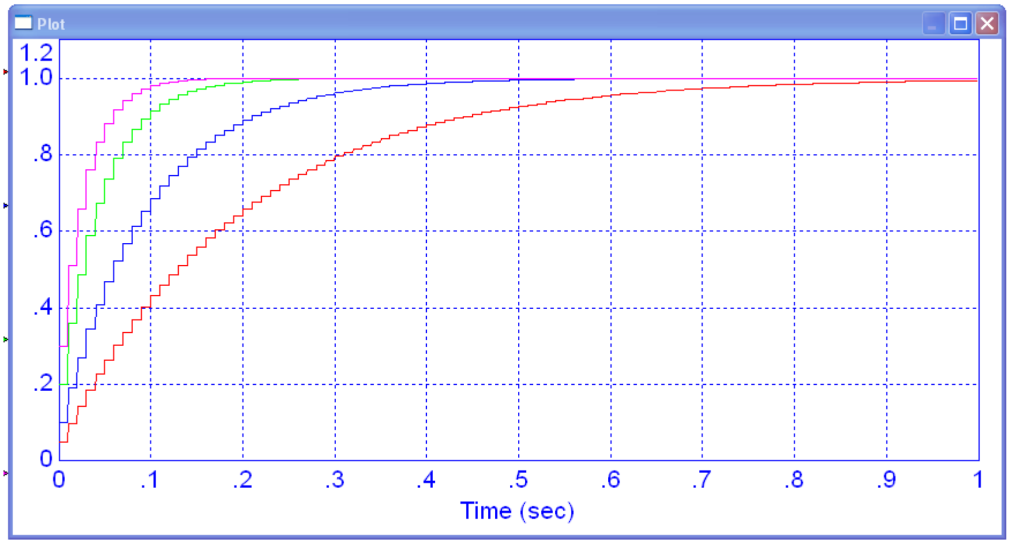
A pole at z = 0.8 has a 2% settling time of 18 samples

$$k = \frac{\ln(0.02)}{\ln(0.8)} = 17.53 \text{ samples}$$

Similarly, for any pole, the amplitude tells you the settling time as

$$|z|^k = 0.02$$

$$k = \frac{\ln(0.02)}{\ln(|z|)}$$



Step Response of a Discrete-Time System with a Pole at { 0.95 (red), 0.9 (blue), 0.8 (green), and 0.7 (pink). T = 0.01 second

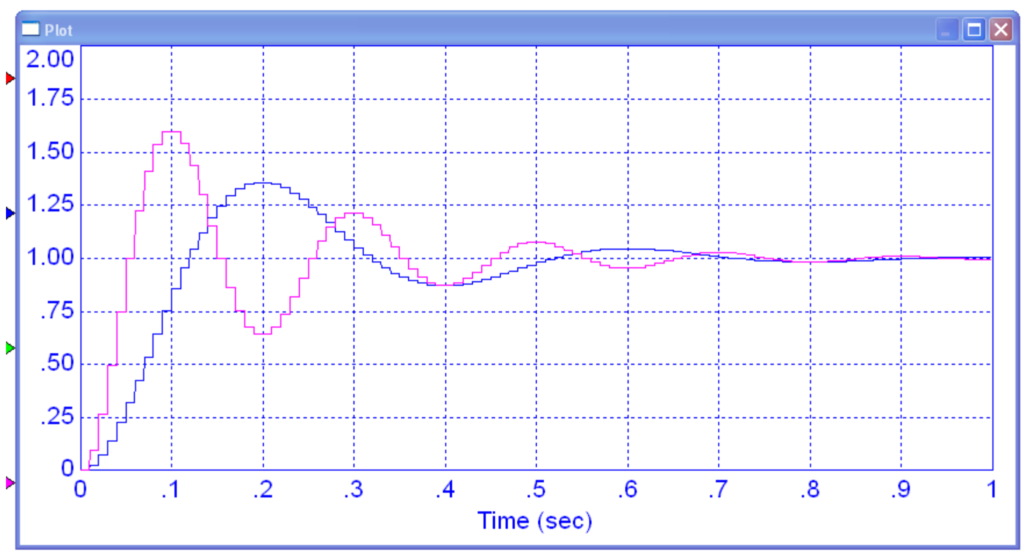
If the amplitude of the pole is 0.95 (meaning a 2% settling time of 73 samples) and you vary the angle of the pole, the frequency of oscillation will be

$$\text{period} = \frac{360^0}{angle} \text{ samples}$$

Poles at

$0.95\angle\pm 9^0$ takes 40 samples for a period

$0.95\angle\pm 18^0$ takes 20 samples for a period



Step Response for Poles at 0.95 / 9 degrees (blue) and 18 degrees (pink). The angle of the pole tells you the frequency of oscillation

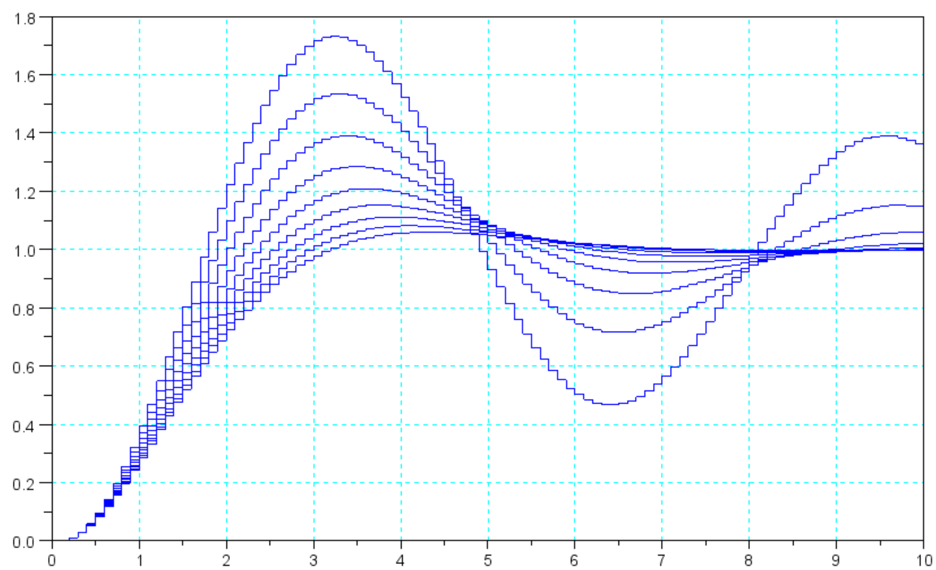
The damping ratio is sort of the angle of the pole - skewed as a log spiral. My preference for determining the damping ratio is to convert to and from the s-plane as

$$z = e^{sT}$$

For example, with a sampling rate of T = 0.1 second

Damping Ratio	Pole in the s-plane	Poles in the z-plane
0	$-1\angle\pm 90^0$	0.9950 + j0.0998
0.2	$-1\angle\pm 78.46^0$	0.9755 + j0.0959
0.4	$-1\angle\pm 66.42^0$	0.9568 + j0.0879
0.6	$-1\angle\pm 53.13^0$	0.9388 + j0.0753
0.8	$-1\angle\pm 36.86^0$	0.9214 + j0.0553
1	$-1\angle\pm 0^0$	0.9048 + j0

The resulting step response as you vary the damping angle is identical to what you get in the s-plane



Step Response of G(z) with the damping ratio varying from 0.1 (max overshoot) to 1.0 in steps of 0.1

Just like in the s-plane, the root-locus in the z-plane tells you what kind of responses you can get by varying a gain, k. The 'best' spot to select is usually

- The largest gain (which results in a fast system with low error),
- That meets your design criteria (such as no more than 20% overshoot in the step response)

Example: Assume the system to be controlled is

$$G(s) = \left(\frac{1000}{(s+5)(s+10)(s+20)} \right)$$

Design a digital compensator, K(z) = k, with a sampling rate of T = 10ms for

- No overshoot,
- 20% overshoot, and
- The maximum gain for stability.

Solution: First, convert G(s) to G(z). With T = 0.01 second

$$G(z) \approx \left(\frac{0.0008413z}{(z-0.9512)(z-0.9048)(z-0.8187)} \right)$$

Now, draw the root locus of G(z) and add in the damping lines

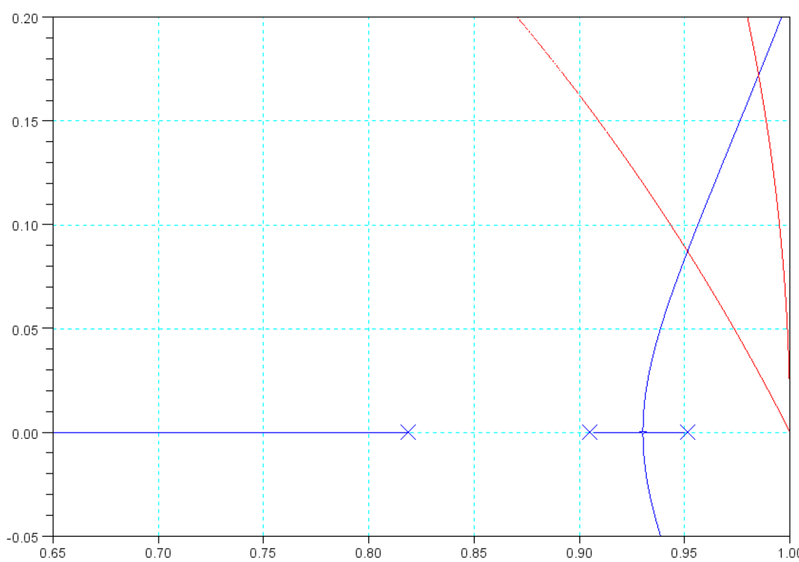
In Matlab:

```
T = 0.01;
Gz = zpk(0, [0.9512, 0.9048, 0.8187], 0.0008413);
k = logspace(-2,2,1000)';
R = rlocus(G,k);

% draw the damping lines on this graph

hold on
s = [0:0.01:100] * (-1+j*2);
z = exp(s*T);
plot(real(z),imag(z),'r')

s = [0:0.01:100] * (j*1);
z = exp(s*T);
plot(real(z),imag(z),'r')
```



Root-Locus of G(z) with 0.4559 and 0.0 damping lines

Note that

- The damping lines depart from z = 1 (which is DC in the z-plane)
- The damping lines are slightly bent. This is the log-spiral resulting from mapping s to z as e^{sT}.

Once draw, find the spot on the root locus

a) No overshoot. This is the breakaway point on the root-locus

$$z = 0.9305$$

At this point

$$\left(\frac{0.0008413z}{(z-0.9512)(z-0.9048)(z-0.8187)} \right)_{z=0.9305} = 13.1620\angle 180^0$$

The compensator gain is then

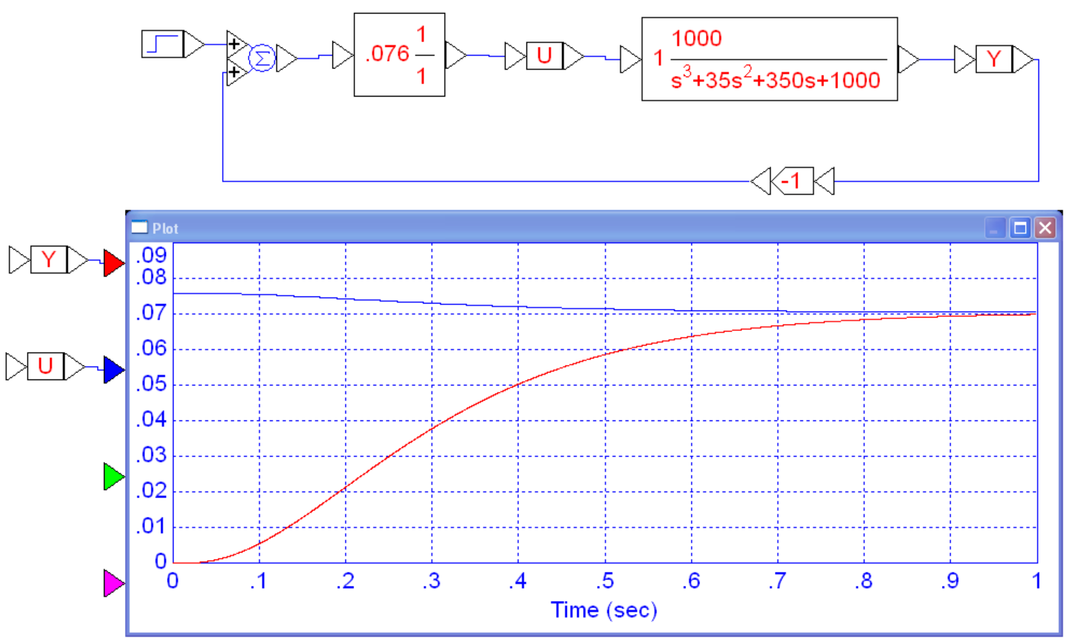
$$K = \frac{1}{13.1620} = 0.0760$$

This results in a 2% settling time of

$$t_{2\%} = \frac{\ln(0.02)}{\ln(0.9305)} = 54.3 \text{ samples} \quad (0.543 \text{ seconds})$$

The error constant, Kp, is the DC gain of G*K. Since G(s) and G(z) at DC is one

$$Kp = 0.0760$$



Step Response with K(z) chosen to place the poles at the breakaway point (no overshoot).

b) 20% overshoot. This point on the root locus is

z = 0.9513 + j0.0873

At this point

$$\left(\frac{0.0008413z}{(z-0.9512)(z-0.9048)(z-0.8187)}\right)_{z=0.9513+j0.0873} = 0.5867\angle 180^0$$

K(z) is then

$$K = \frac{1}{0.5867} = 1.7044$$

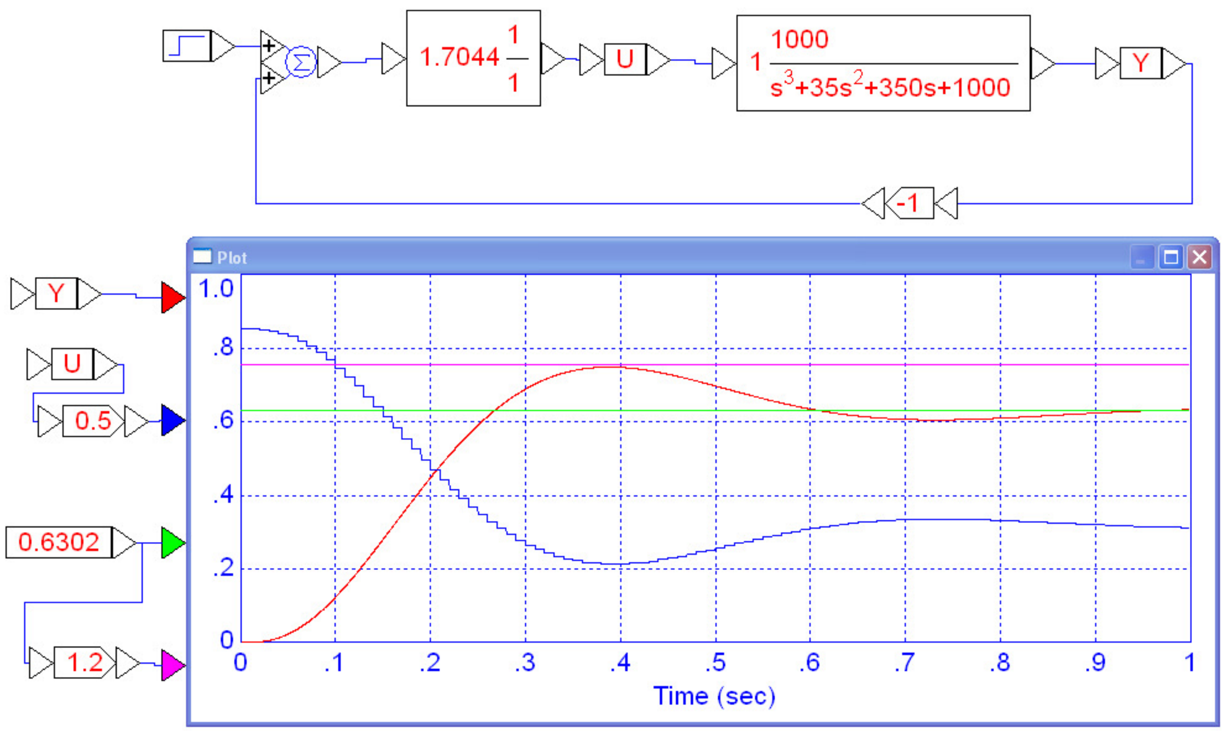
The resulting 2% settling time is

$$t_{2\%} = \frac{\ln(0.02)}{\ln(|0.9513+j0.0873|)} = 85.52 \text{ samples}$$

Since the DC gain of G(s) and G(z) is one, the error constant Kp is K*1 or

Kp = 1.7044

Checking in VisSim (using the actual analog system and a digital compensator with a sampling rate of 10ms)



Step Response with K(z) chosen for 20% overshoot

Note that the overshoot is almost 20%, as desired.

c) Max Gain for Stability (the jw crossing). The point that intersects with the unit circle is

$$z = 0.9850 + j0.1723 = 1\angle 9.9215^0$$

At this point

$$\left(\frac{0.0008413z}{(z-0.9512)(z-0.9048)(z-0.8187)} \right)_{z=0.9850+j0.1723} = 0.1053\angle 180^0$$

meaning

$$K(z) = \frac{1}{0.1053} = 9.5008$$

With this K(z), the 2% settling time is infinite.

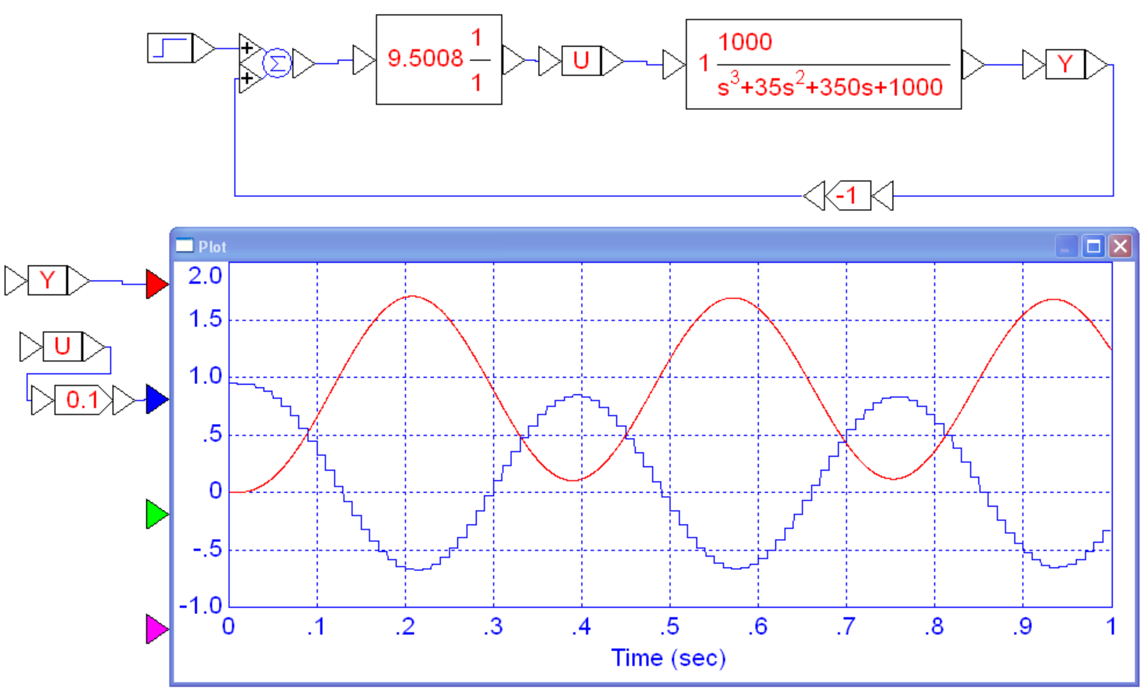
The frequency of oscillation is

$$\angle z = 9.9215^0$$

$$\text{period} = \frac{360^0}{9.9215^0} = 36.28 \text{ samples} = 0.3628 \text{ seconds}$$

$$f = \frac{1}{period} = 2.75 \text{ Hz}$$

Checking in VisSim



Step Response with K(z) chosen to place the poles on the unit circle. The period of oscillations = 0.3628 seconds

Moral: Root-locus works in the z-plane just like it does in the s-plane. The only difference is how you interpret the results.

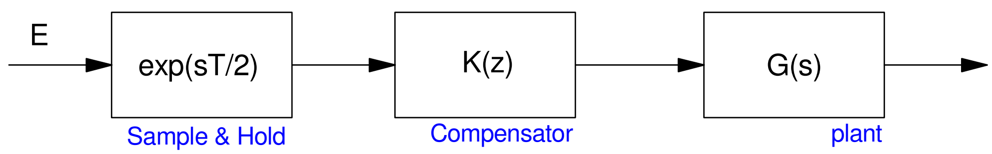
Alternate Method:

Note that when using root-locus techniques, you really only care about one point: the one where the angles add up to 180 degrees. At this point

$$G \cdot K = 1\angle 180^0$$

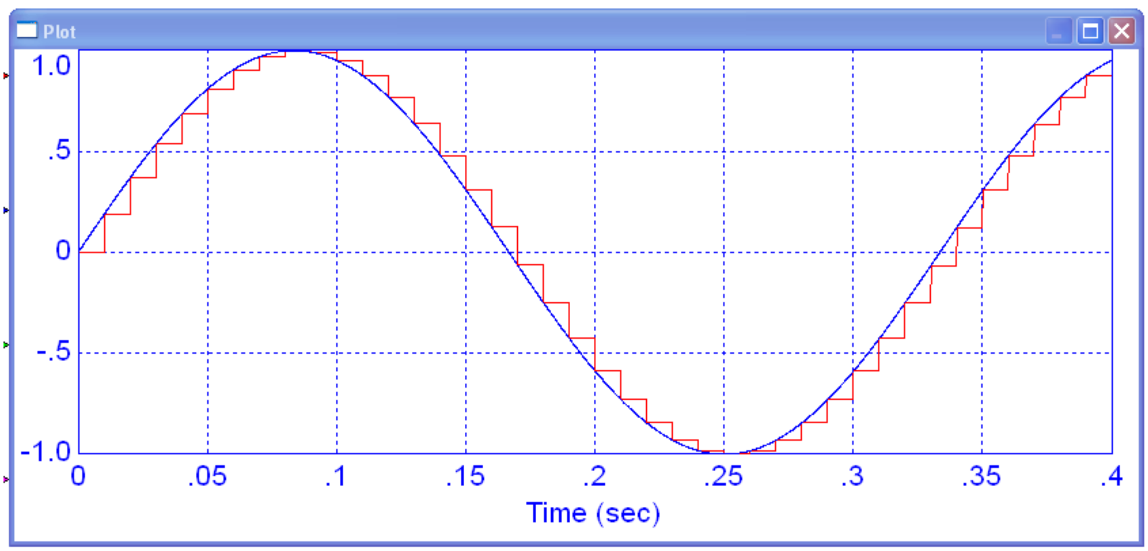
When you have a digital compensator, you really have three terms:

G(s) * K(z) * sample and hold



The open-loop system has three terms:

The sample and hold adds a delay of 1/2 sample. For example, if you sample a 3Hz sine wave with T = 0.01 second, you get a 3Hz sine wave, delayed by 5ms



3Hz sine wave (blue) sampled at 10ms (red) results in a 3Hz sine wave delayed by 1/2 of a sample (5ms)

Likewise, you can solve the previous root-locus problems using numerical methods to find the point on the damping line where

$$angle(G(s) \cdot K(z) \cdot e^{-sT/2}) = 180^0$$

You're kind-of mixing planes with this approach. Since you only care about one point, however, you don't care. You just

- Guess the point, s
- Compute the corresponding point in the z-plane as $z = e^{sT}$
- Evaluate the above function, and
- Repeat until the angles add up to 180 degrees

Example: Find the gain, k, that results in 20% overshoot in the step response.

In Matlab, start with a guess of s = -5 + j10. Evaluate G(s)*delay

```
-->s = 5*( -1 + j*2);
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.5009862 + 0.0882455i
```

The angle isn't zero (the complex part is non-zero), so try a different s, such as 10% bigger

```
-->s = s*1.1;
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.4037178 + 0.1524678i
```

That made the complex part worse. If 10% bigger is bad, try 10% smaller

```
-->s = s*0.9;
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.5110062 + 0.0796359i

-->s = s*0.9;
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.6083330 - 0.0321254i
```

Too far (there was a sign flip on the complex portion). Try 1% larger now.

```
-->s = s*1.01;
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.5999484 - 0.0199126i
```

Too far (sign flip on the complex portion). Try 0.1% smaller

```
-->s = s*0.999;
-->1000 / ( (s+5)*(s+10)*(s+20))  *  exp(-s*T/2)
  - 0.5843619 + 0.0012012i
```

Close enough. This results in

```
-->k = 1/abs(ans)

    1.7086736
```

which is about the same result we got with root locus techniques. It's a little more accurate, however, since you don't have to approximate G(s) with G(z) with this method. Either way works - the lecture notes and homework solutions use this latter method, however. (My calculator has a solver function).