Digital Control of a DC Servo Motor

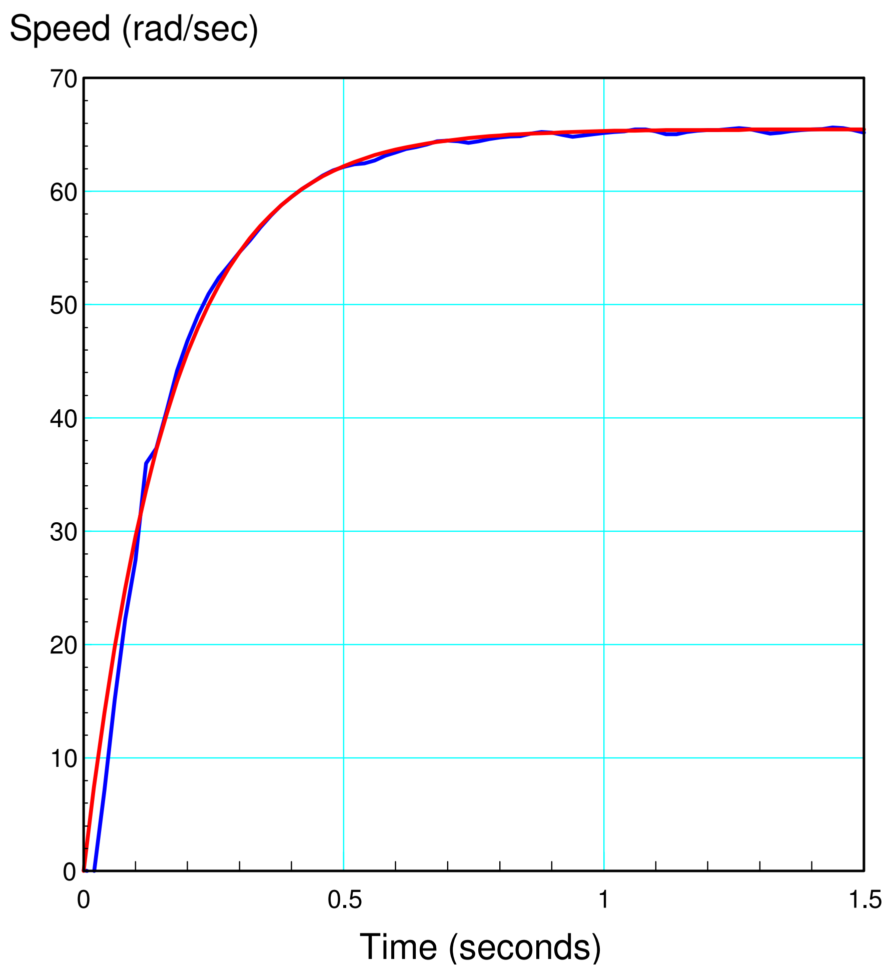
ECE 461/661 Controls Systems

Jake Glower - Lecture #34

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Problem:

- Control the speed of a DC servo motor
- Control the position of a DC servo motor



The mathematical model from before:

- Clifton 000-053479-002

$$\omega \approx \left(\frac{39.28}{s+6}\right) V_a$$

$$\theta \approx \left(\frac{39.28}{s(s+6)}\right) V_a$$

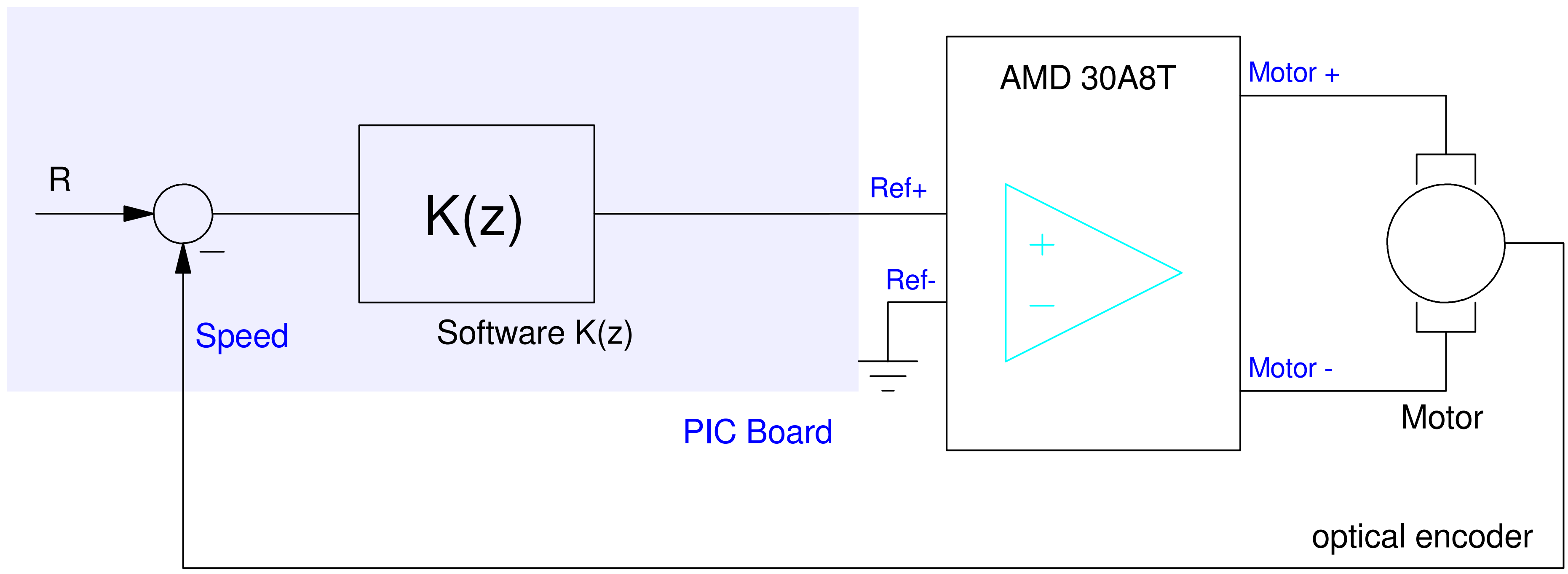
With T = 20ms

$$\omega \approx \left(\frac{0.7404}{z-0.8869}\right) V_a$$

$$\theta \approx \left(\frac{0.02z}{z-1}\right)\left(\frac{0.7404}{z-0.8869}\right) V_a$$

Hardware Setup

- Use a microcontroller to compute speed (or angle),
- Also use it to implement K(z)



Speed Control: Gain Compensation

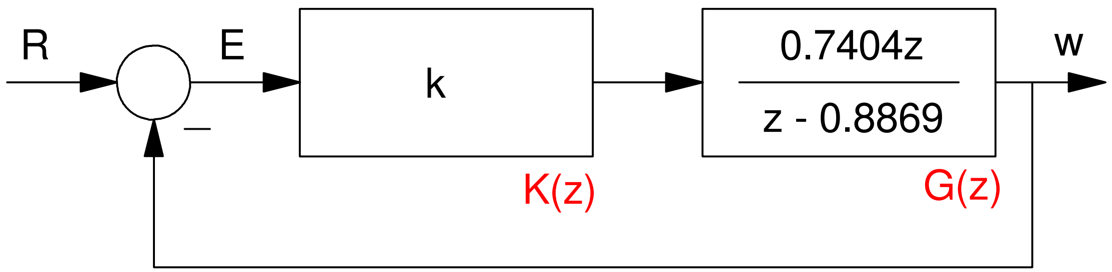
$$G(s) = \left(\frac{39.28}{s+6}\right)$$

$$T = 20ms$$

$$G(z) = \left(\frac{0.7404}{z-0.8869}\right)$$

$$K(s) = k$$

$$GK = \left(\frac{0.7404k}{z-0.8869}\right)$$



Predicted Response

$$(GK)_s = -1$$

z =0.6

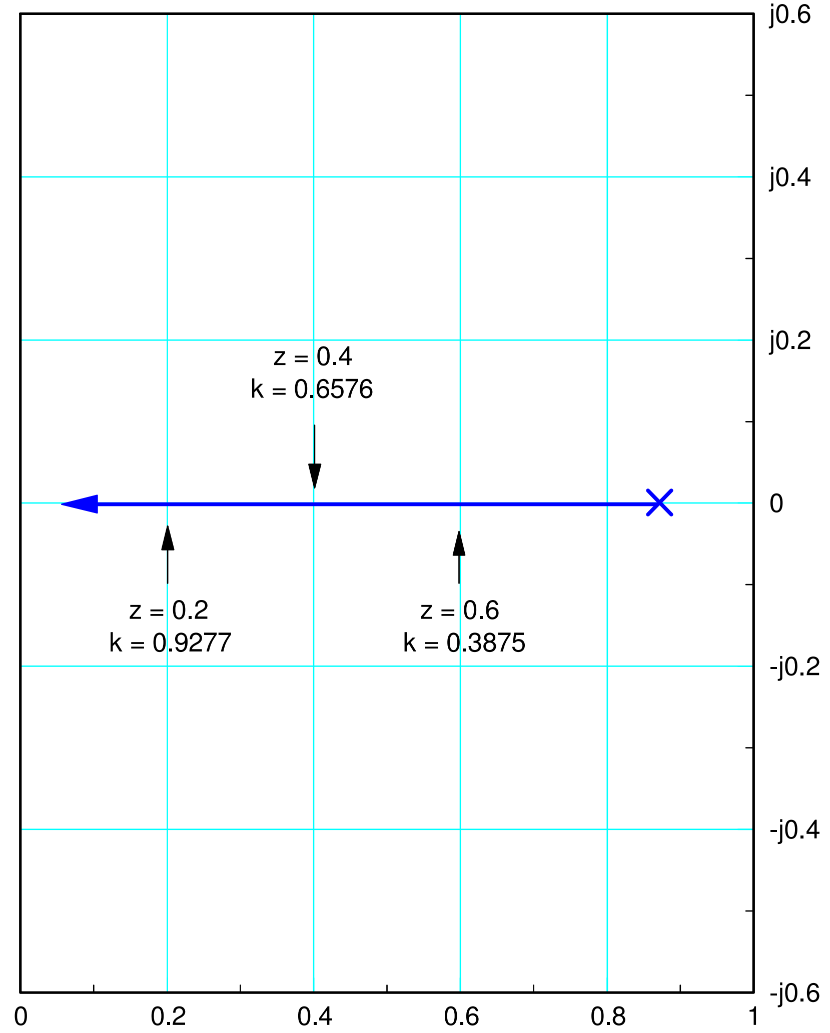
- k = 0.3875

z = 0.4

- k = 0.6576

z = 0.2

- k = 0.9277



Experimental Results:

z =0.6

- k = 0.3875

z = 0.4

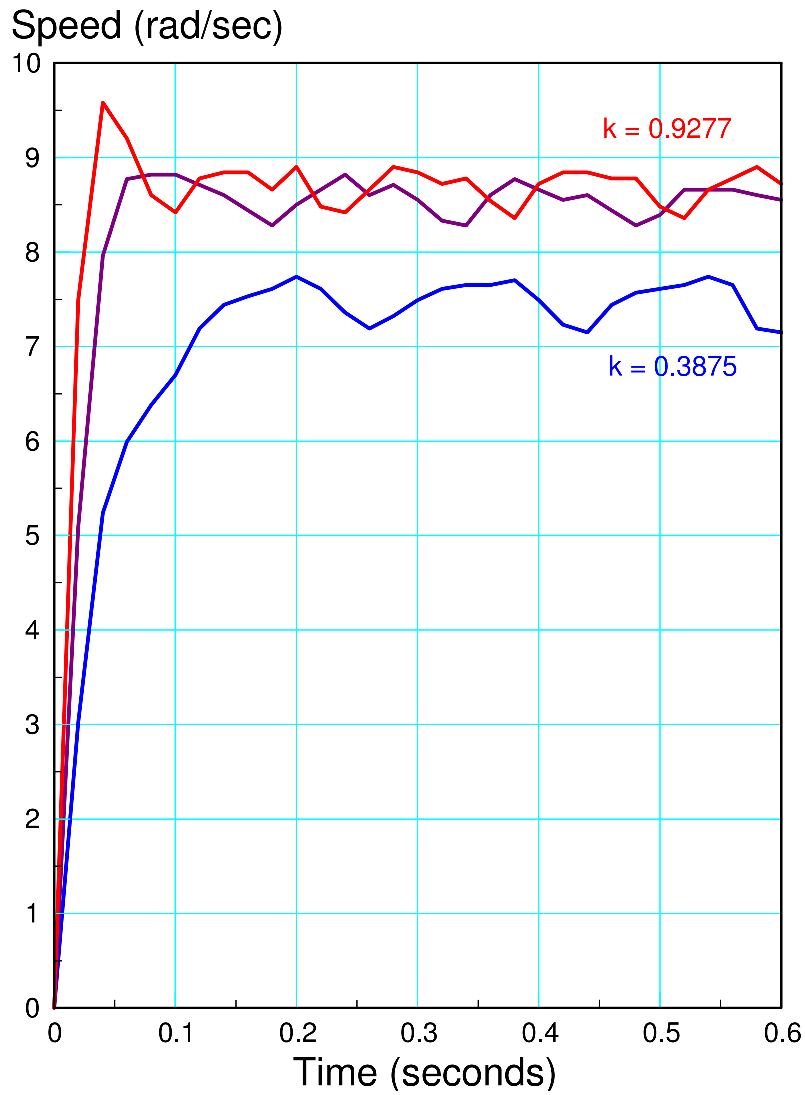
- k = 0.6576

z = 0.2

- k = 0.9277

Code:

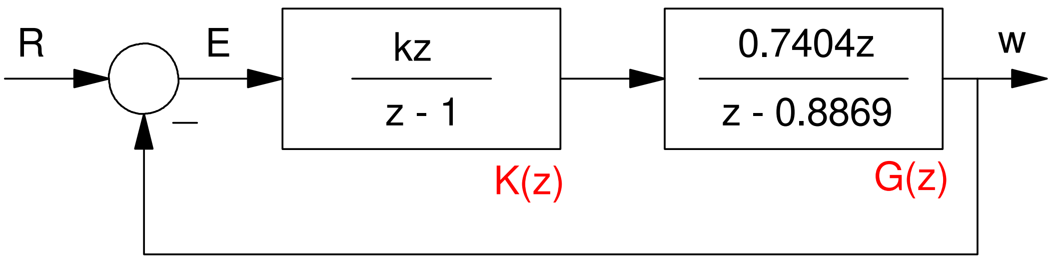
```
while(1) {
    :
    :
    E = REF - SPEED;
    U = 0.9722 * E;
    D2A(U);
    Wait_20ms();
    }
```



Speed Control: I Compensation

$$K(z) = \left(\frac{kz}{z-1}\right)$$

$$GK = \left(\frac{0.7404k\cdot z}{(z-1)(z-0.8869)}\right)$$



Type-1 System

- No Error for a Step Input

Code:

```
while(1) {
     :
     :
     E = REF - SPEED;
     U = U + k * E;
     D2A(U);
     Wait_20ms();
     }
```

I Compensation

z = 0.99

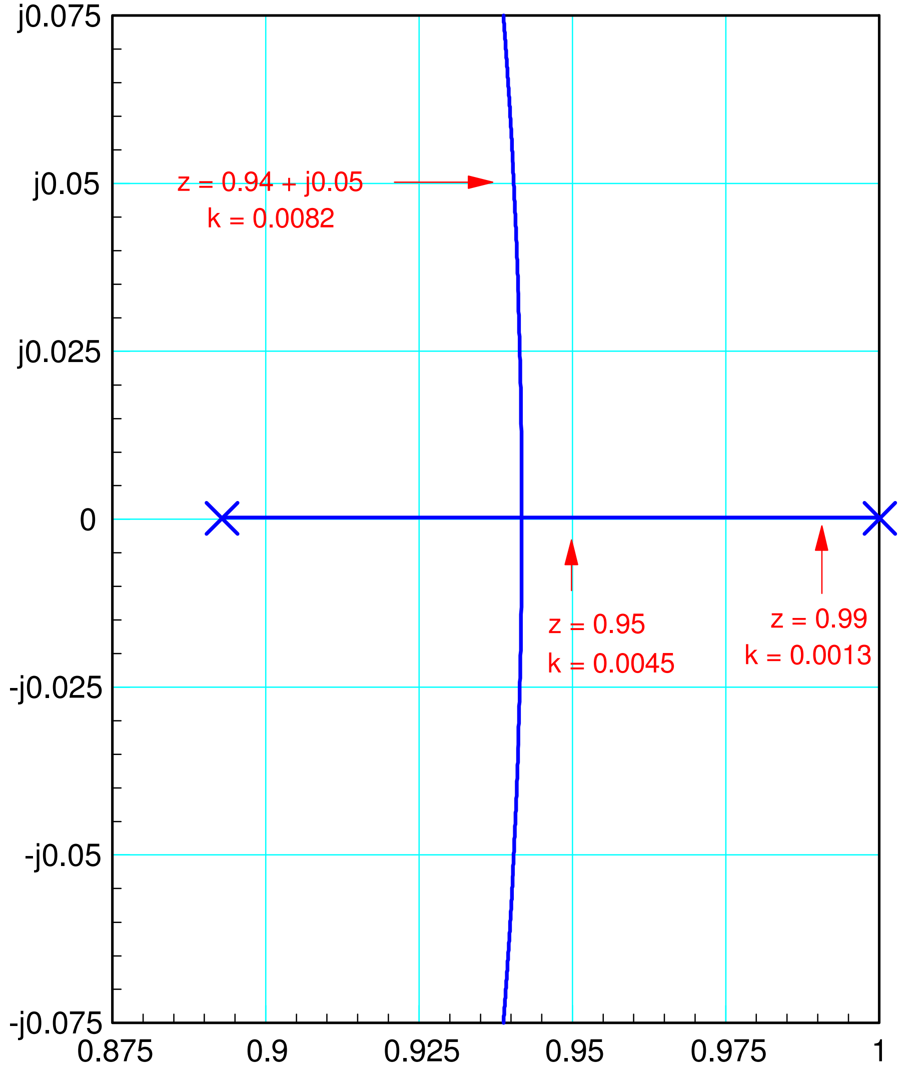
- $\left(\frac{0.7404k \cdot z}{(z-1)(z-0.8869)}\right)_{z=0.99} = -1$
- k = 0.0013

z = 0.95

- $\left(\frac{0.7404k \cdot z}{(z-1)(z-0.8869)}\right)_{z=0.95} = -1$
- k = 0.0045

z = 0.9434 + j0.0535

- $\left(\frac{0.7404k \cdot z}{(z-1)(z-0.8869)}\right)_{z} = -1$
- k = 0.0082



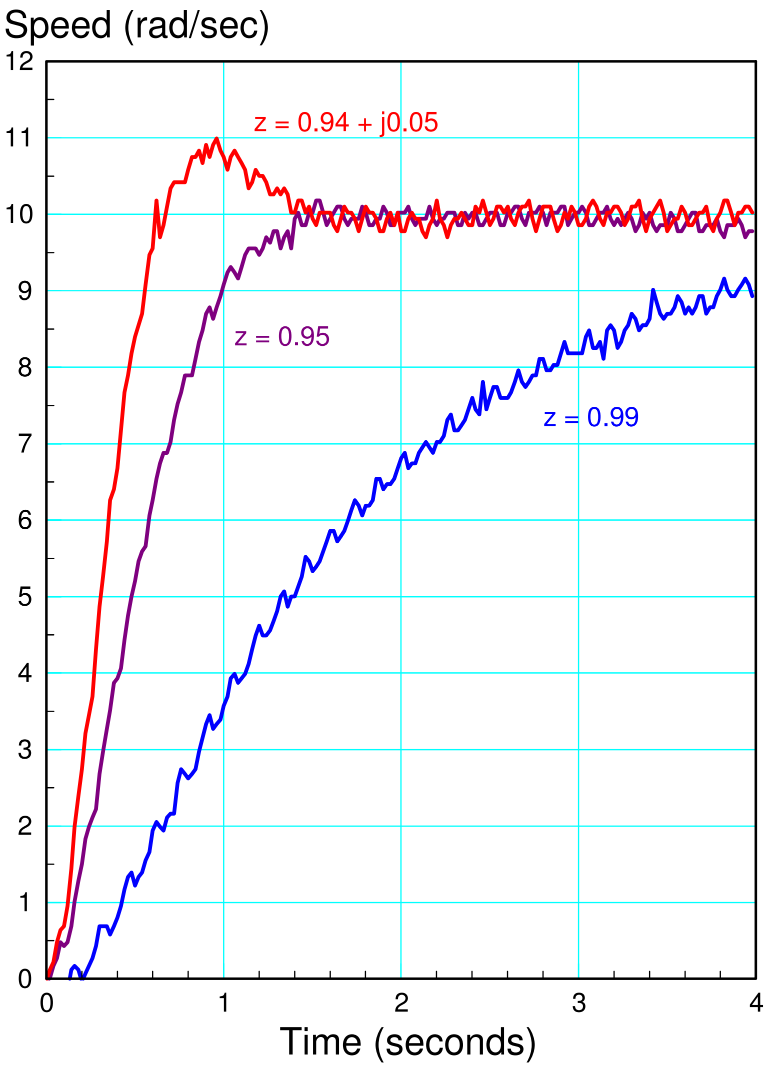
Experimental Results

- I Compensation

Steady-state speed = 10 rad/sec

- No Error for a step input
- Type-1 system

The response is what the root locus plot predicted



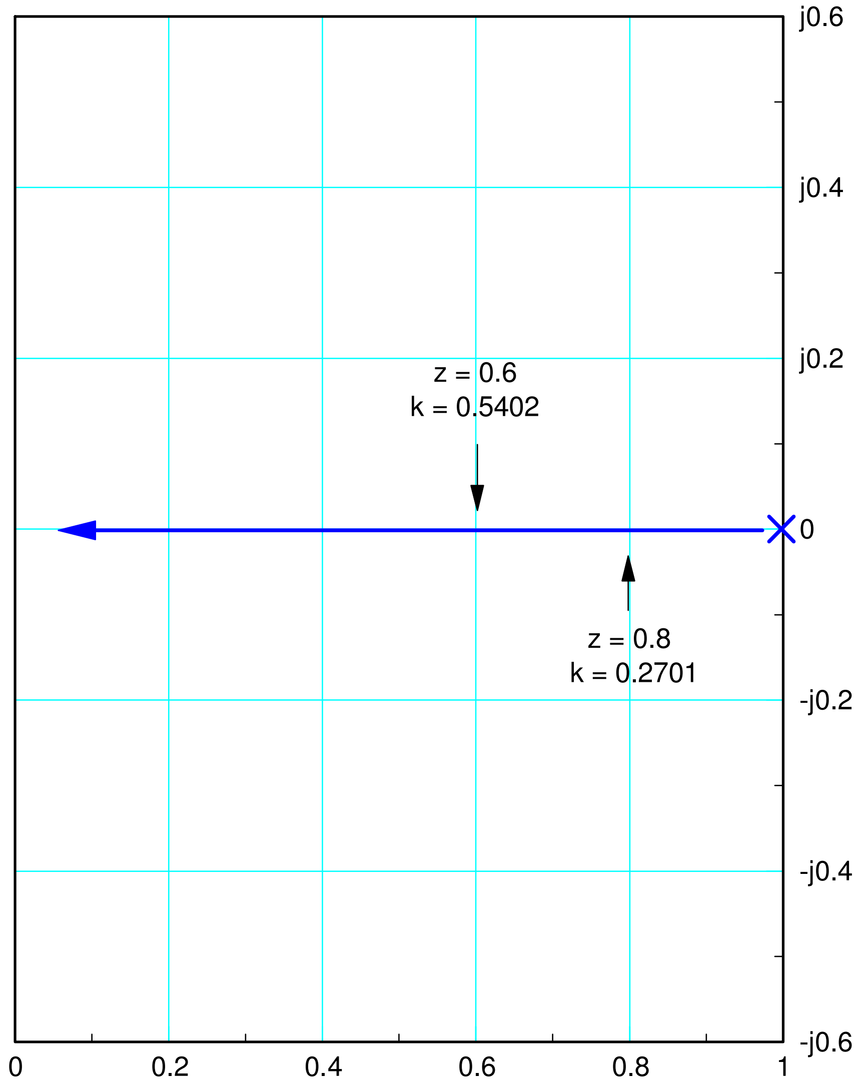
Speed Control: PI Control

$$K(z) = k\left(\frac{z-0.8869}{z-1}\right)$$

$$GK = \left(\frac{0.7404k}{z-1}\right)$$

Code:

```
while(1) {
     :
     E1 = E0;
     E0 = REF - SPEED;
     U = U + k * (E0 - 0.8869*E1);
     D2A(U);
     Wait_20ms();
     }
```



PI Control: Experimental Results

z = 0.99

k = 0.0135

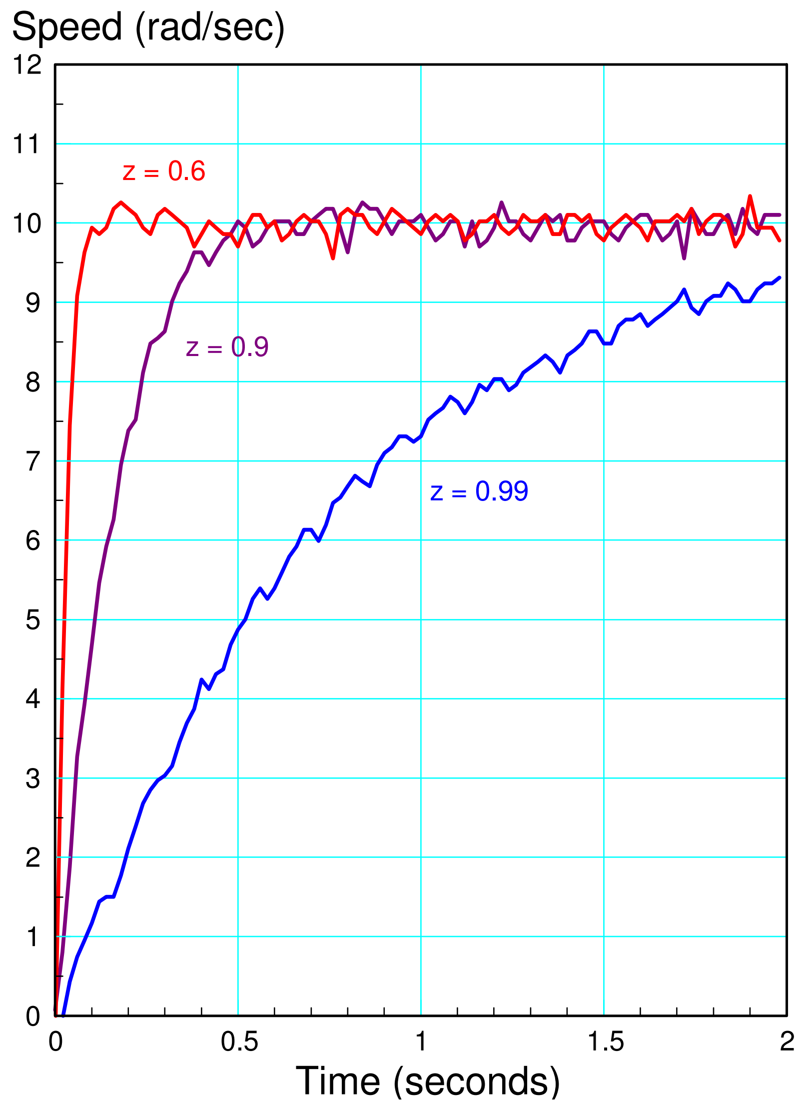
z = 0.9

k = 0.1351

z = 0.6

k = 0.5402

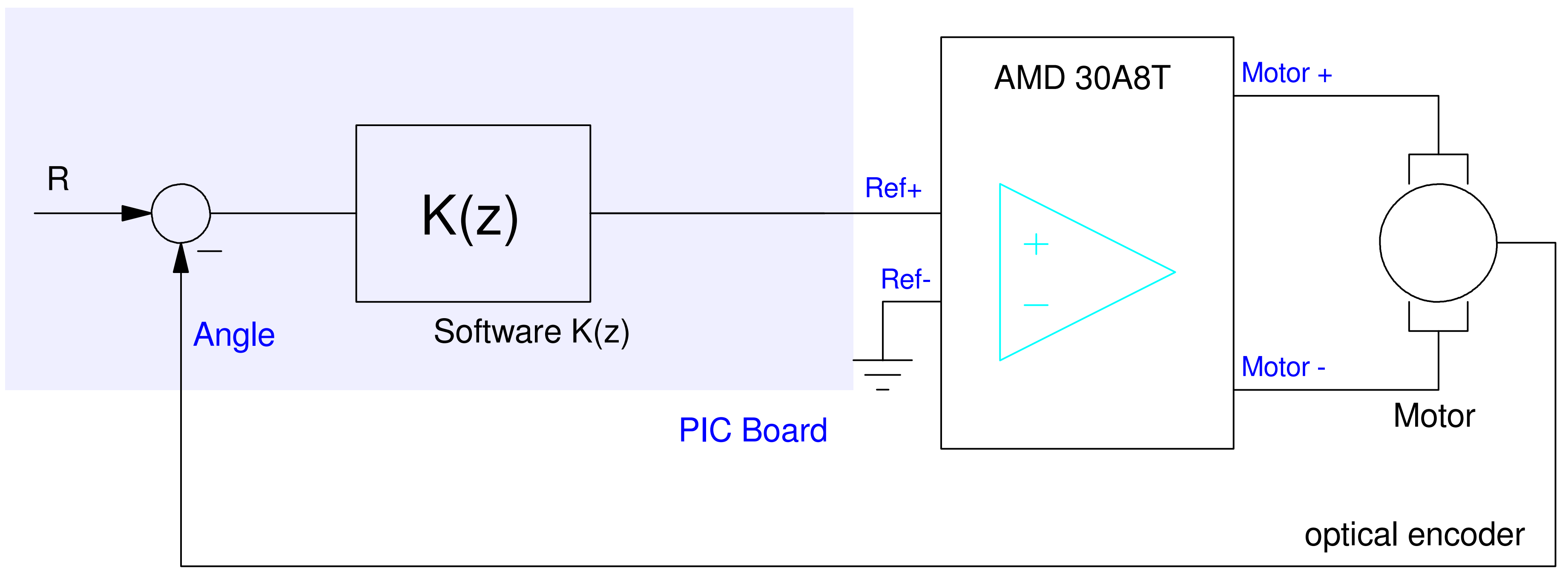
Again, the results are what the root locus plot predicts



Position Control:

- Change the sensor to an angle sensor and you have position control

$$\theta = \left(\frac{39.28}{s(s+6)}\right) V_a = \left(\frac{0.0148z}{(z-1)(z-0.8869)}\right) V_a$$



Position Control: K(z) = k

$$\theta = \left(\frac{0.0148z}{(z-1)(z-0.8869)}\right) V_a$$

z = 0.99

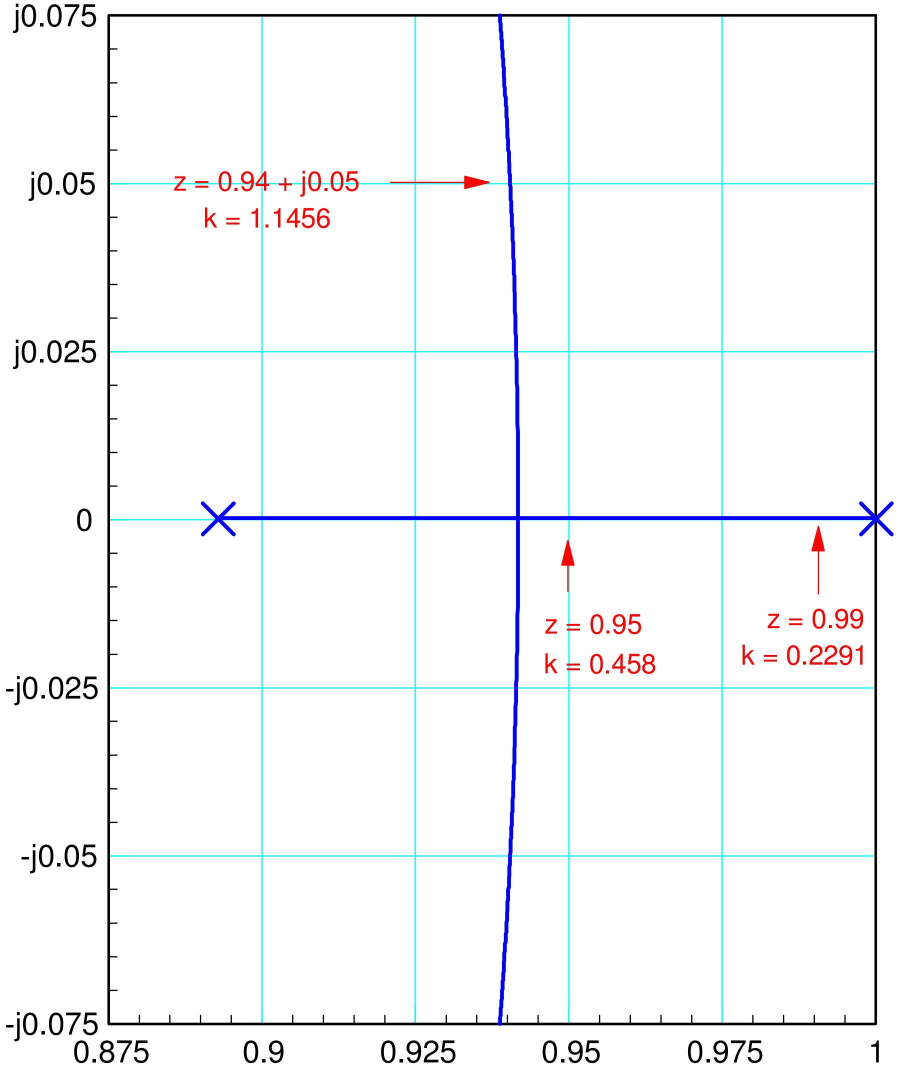
k = 0.2291

z = 0.95

k = 0.458

z = 0.95 + j0.05

k = 1.1456



Position Control: K(s) = k

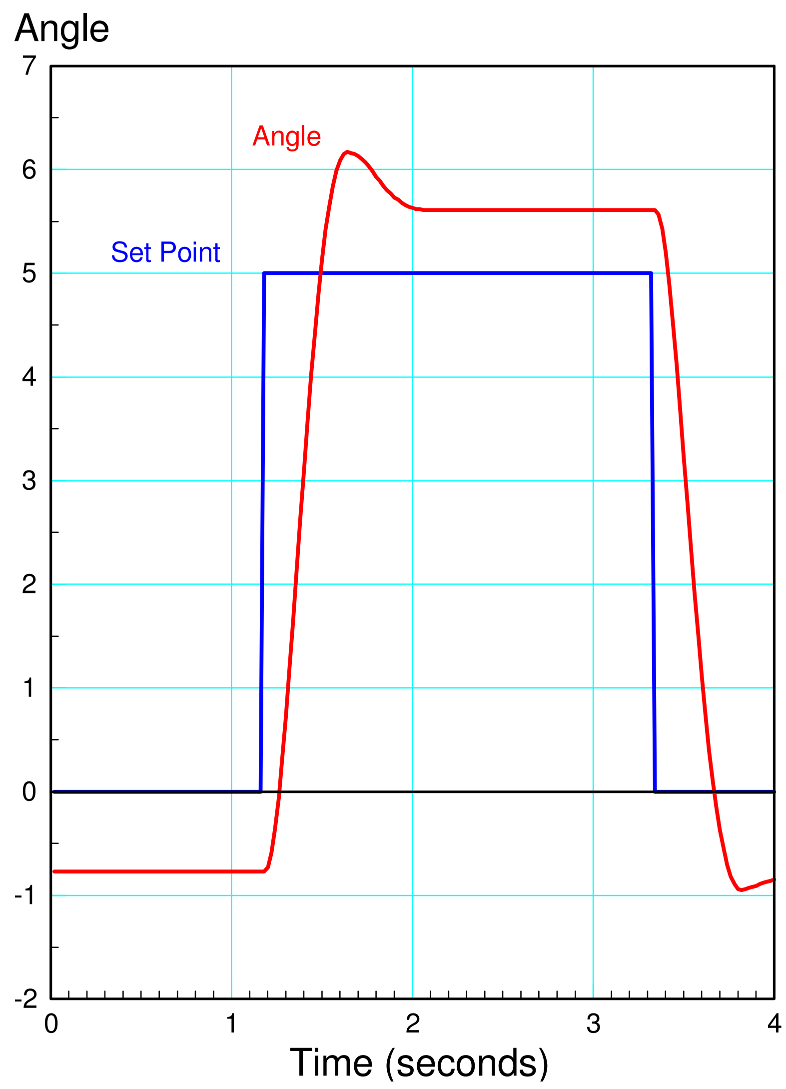
- Experimental Results

In theory, the steady-state error is zero

- Type-1 System

In practice, static friction causes a slight error

Otherwise, the response is what the root locus plot predicts

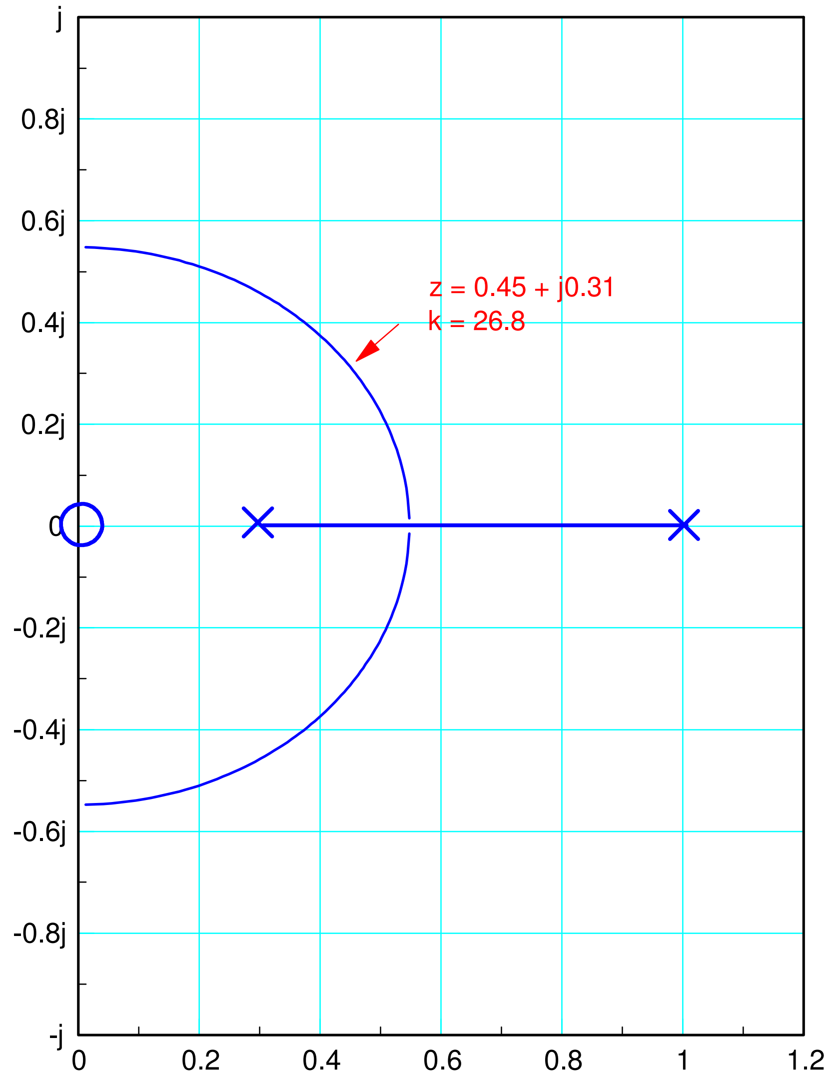


Lead Compensation

- Cancel the pole at s = -6
- Replace it with a pole at s = -19.2

$$K(z) = k\left(\frac{z-0.8869}{z-0.3}\right)$$

$$GK = \left(\frac{0.0148z}{(z-1)(z-0.3)}\right)$$



$$z = 0.4514 + j0.3102$$

$$k = 26.8$$

$$K(z) = 26.8\left(\frac{z-0.8869}{z-0.3}\right)$$

Lead Compensation:

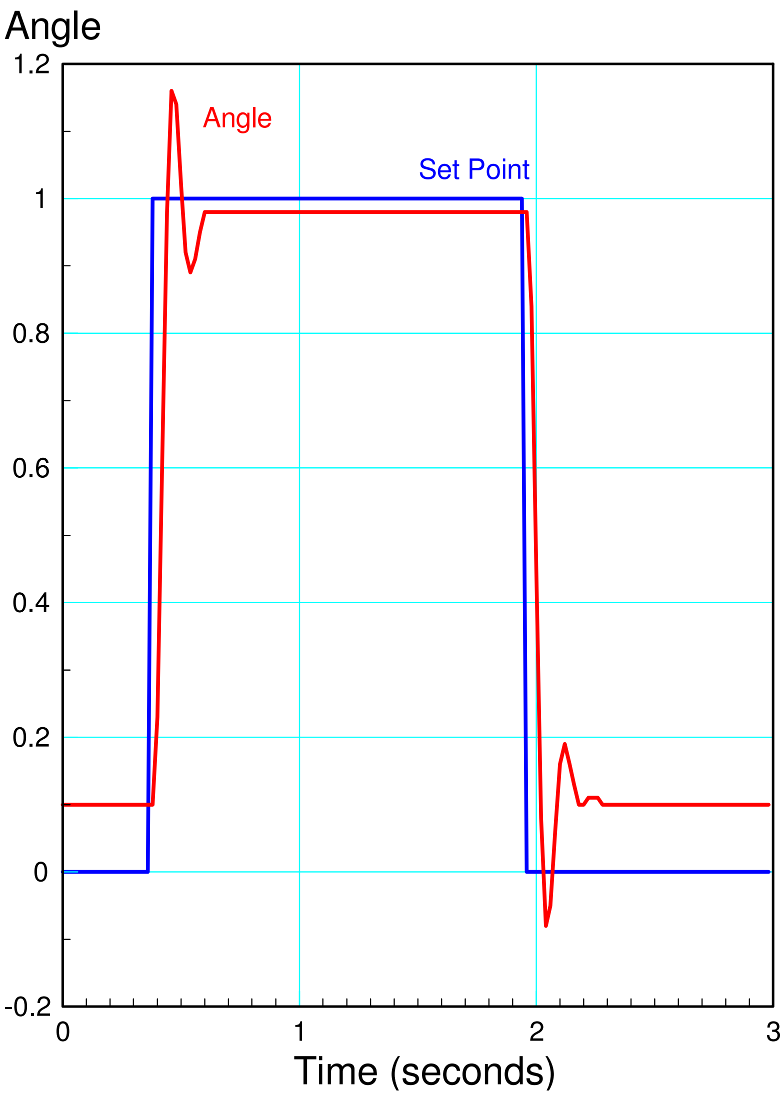
$$K(z) = 26.8\left(\frac{z-0.8869}{z-0.3}\right)$$

Again, the experimental results are what root locus plots predicted

- Slight steady-state error due to static friction
- (nonlinear terms)

Code:

```
while(1) {
     :
     E1 = E0;
     E0 = REF - SPEED;
     U = 0.3*U + k * (E0 - 0.8869*E1);
     D2A(U);
     Wait_20ms();
     }
```



Summary

Root locus really works

- It predicts how the system will behave as the gain changes
- The response is as the root locus plot predicts

Digital Control also really works

- It saves hardware: you don't need to build an op-amp circuit
- It removes the DC offset that op-amps have
- Download a new program and you have a new controller