Servo-Compensators AC Set Points

NDSU ECE 463/663

Lecture #17

Inst: Jake Glower

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

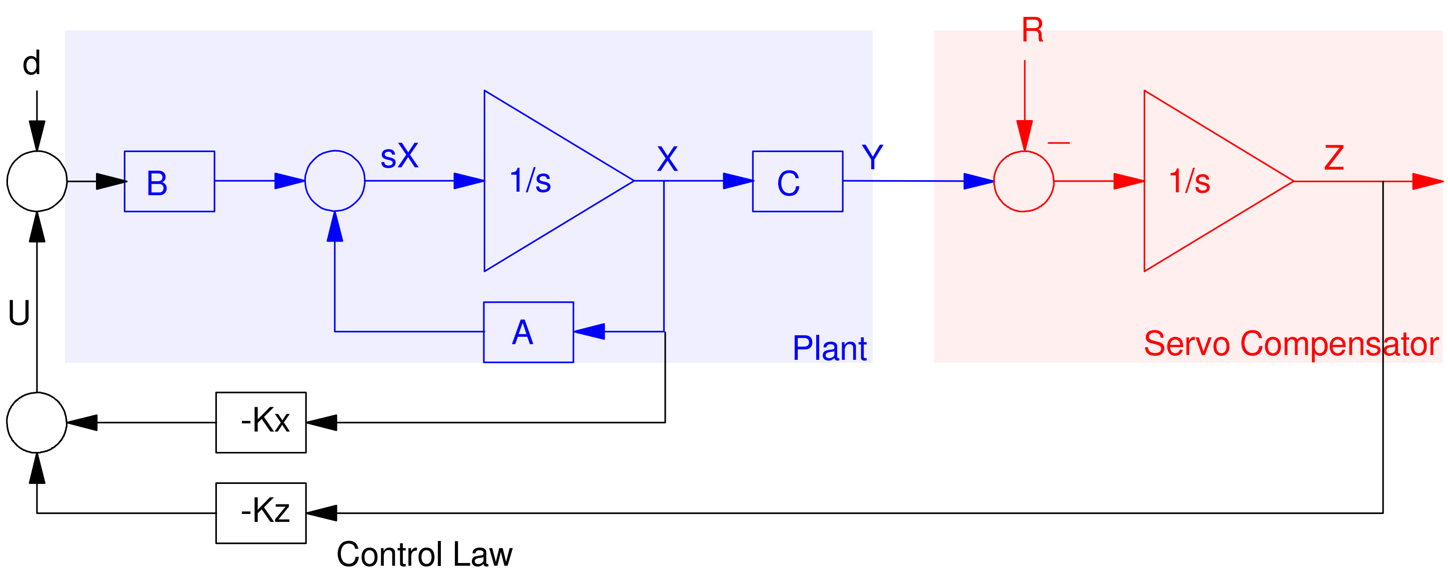
Recap:

If you want to track a constant set point

- Add a servo compensator with a pole at s = 0
- Add full state feedback

The resulting system can

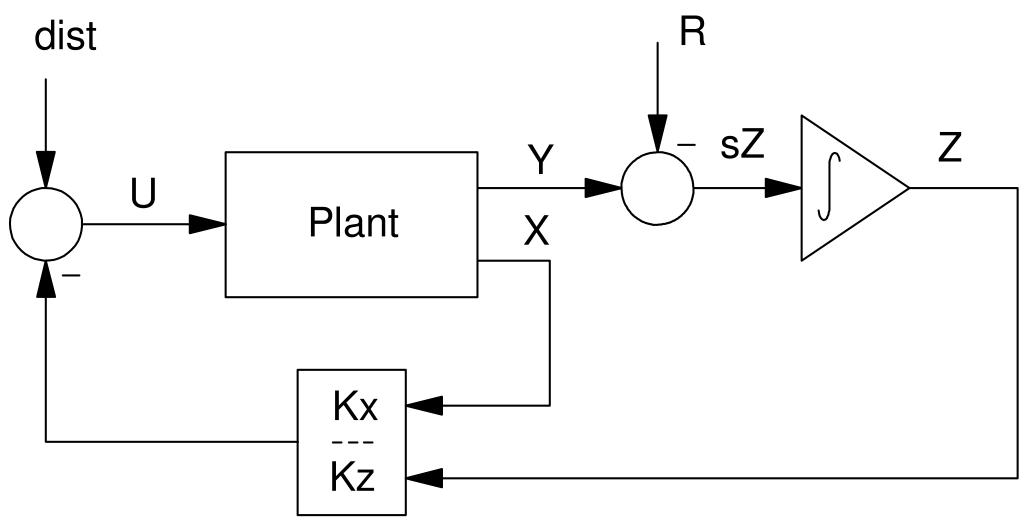
- Track a constant set point, and
- Reject constant disturbances



Problem: Sinusoidal Set Points

What if you want to track a sinusoidal set point?

- The previous design only works at s = 0
- Change the frequency of R, it no longer tracks
- Change the frequency of d, it no longer rejects the disturbance



Solution: Sinusoidal Set Points

At DC, the servo compensator has a gain of infinity

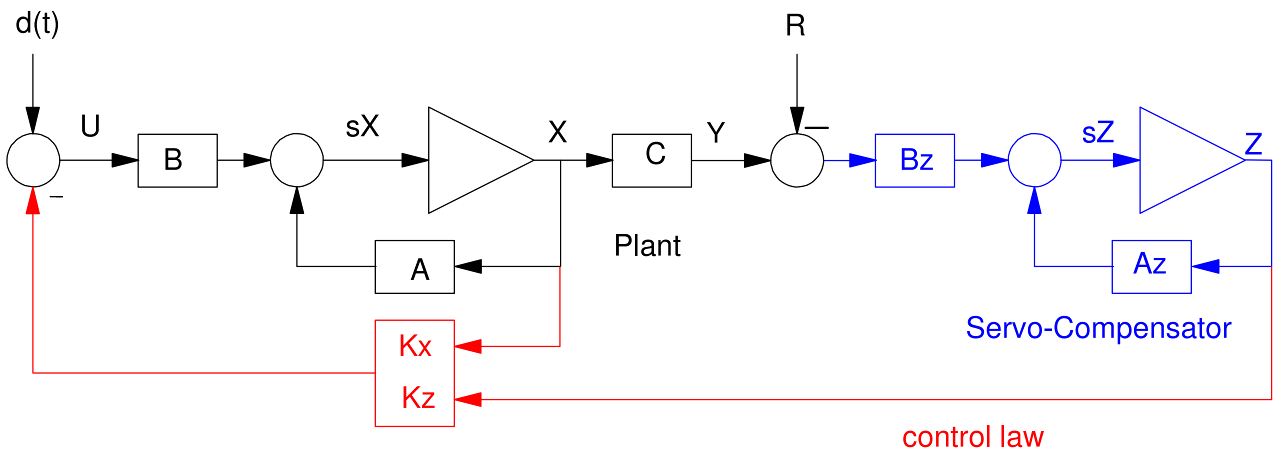
- This forces the error to zero at s = 0

At ω rad/sec, the gain is finite

- This creates finite error at this frequency

Change the servo compensator so that it has infinite gain at ω rad/sec

- Choose Az so that it has poles at $\pm j\omega$



Example: Let the plant be

$$sX = AX + BU$$

$$Y = CX$$

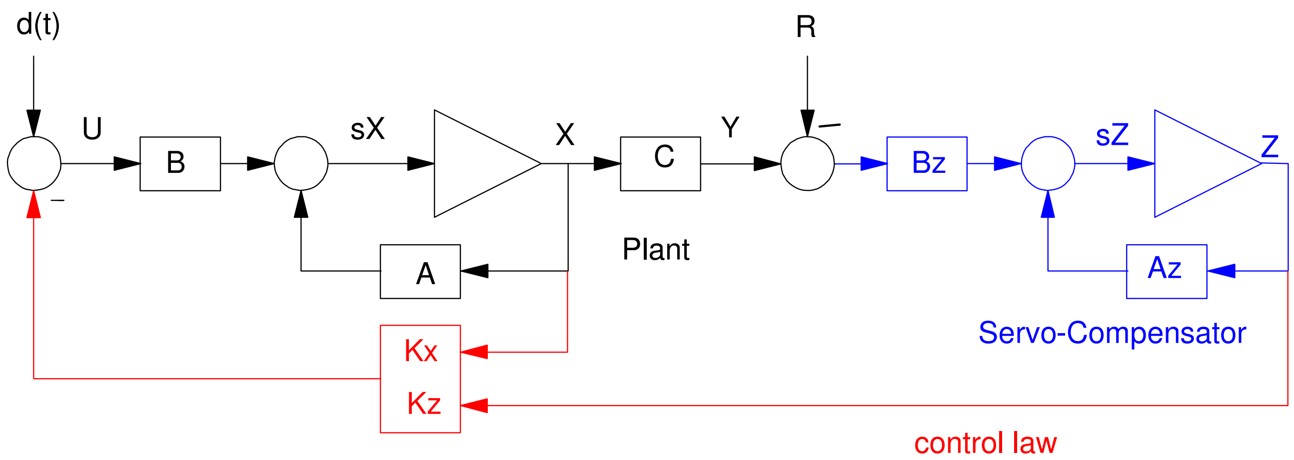
Define a servo-compensator

$$sZ = A_z Z + B_z (Y - R)$$

so that the eigenvalue of Az are

$$eig(A_z) = \pm j\omega$$

Feed the servo-compensator with the difference between Y and the set point R



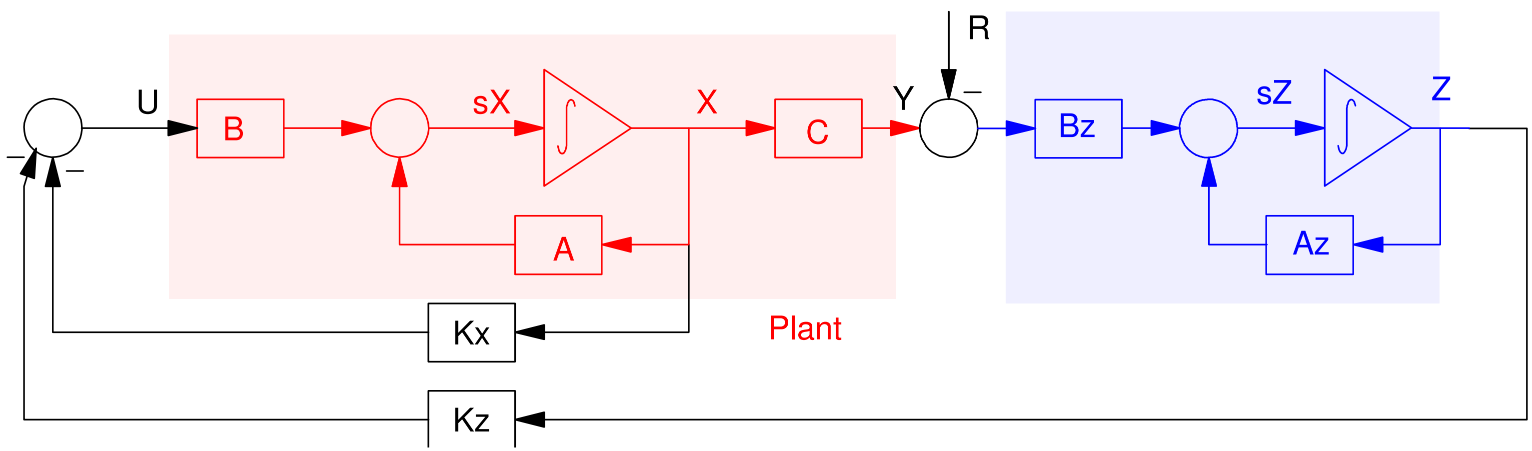
In state-space, the plant plus servo-compensator looks like the following:

$$s\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} A & 0 \\ B_z C & A_z \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} U + \begin{bmatrix} 0 \\ -B_z \end{bmatrix} R$$

$$U = -\begin{bmatrix} K_x & K_z \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix}$$

or

$$s\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} A - BK_x & -BK_z \\ B_z C & A_z \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ -B_z \end{bmatrix} R$$



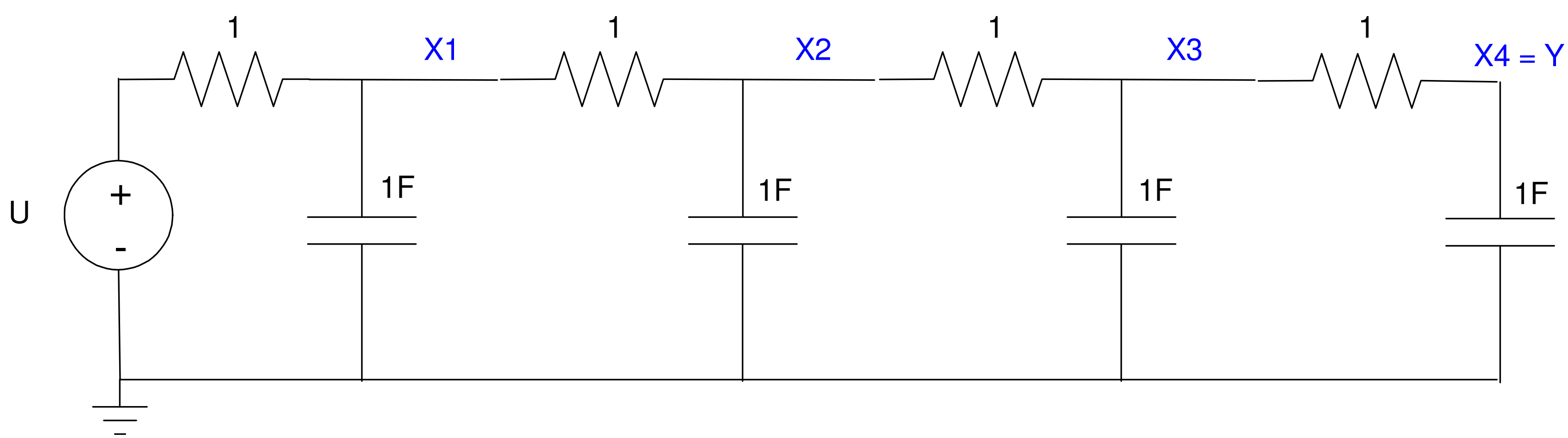
Example: 4th-Order RC Filter

Let

$$R(t) = \sin(2t)$$

Design a feedback control law for the following system so that

- The 2% settling time is 4 seconds,
- $Y \to R$



Solution: First, design a servo compensator with poles at $s = \pm j2$

- There are multiple solutions
- One that works is:

$$sZ = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} Z + \begin{bmatrix} 1 \\ 1 \end{bmatrix} U_z$$

Create an augmented system: plant plus servo

$$s\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} A & 0 \\ B_z C & A_z \end{bmatrix} \begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} U + \begin{bmatrix} 0 \\ -B_z \end{bmatrix} R$$

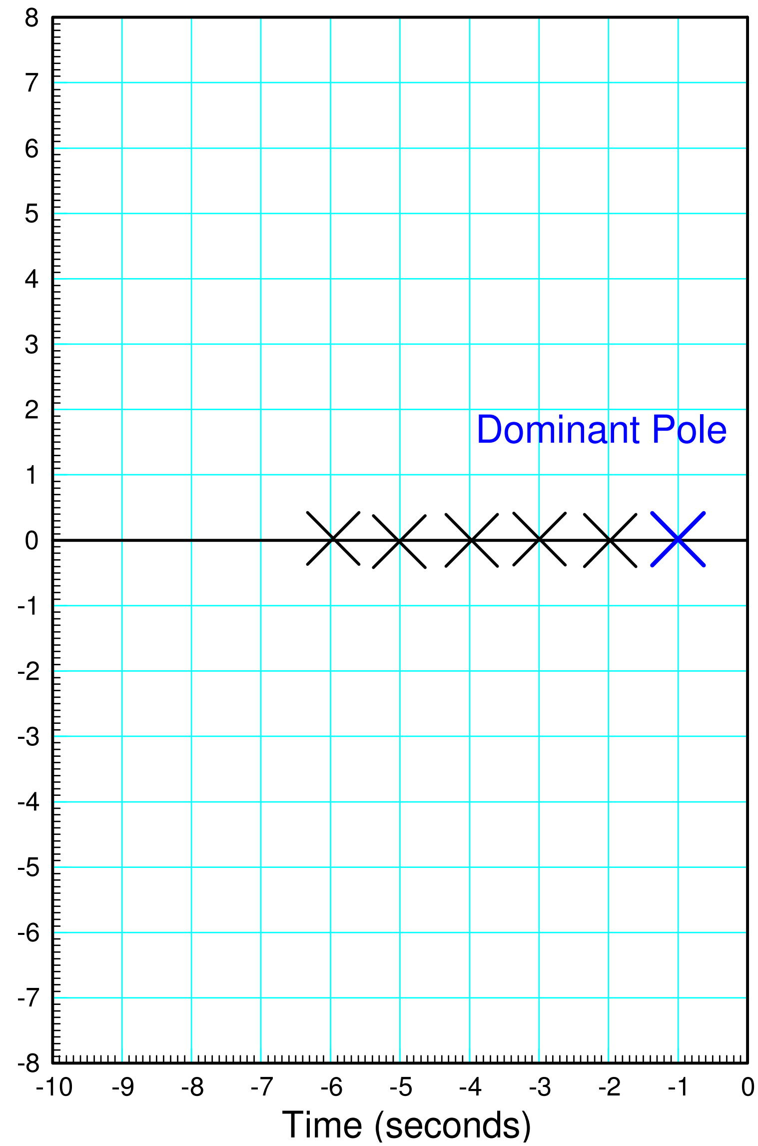
$$s\begin{bmatrix} X \\ \cdots \\ Z \end{bmatrix} = \left[\begin{array}{cccc:cc} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & -2 & 0 \end{array}\right] \begin{bmatrix} X \\ \cdots \\ Z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{bmatrix} U + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ -1 \\ -1 \end{bmatrix} R$$

Use pole placement to meet the requirements

- Dominant pole at s = -1
- Other poles anywhere left of -1

Place the poles at

- s = {-1, -2, -3, -4, -5, -6}
- Somewhat arbitrary
- The pole at -1 isn't *that* dominant...



In Matlab:

```
A = [-2,1,0,0 ; 1,-2,1,0 ; 0,1,-2,1 ; 0,0,1,-1];

  - 2.    1.    0.    0.
    1.  - 2.    1.    0.
    0.    1.  - 2.    1.
    0.    0.    1.  - 1.

B = [1;0;0;0];

    1.
    0.
    0.
    0.

C = [0,0,0,1];

    0.    0.    0.    1.
```

Add in the servo-compensator (any system with poles at $\pm j2$)

```
Az = [0,2;-2,0]

    0.    2.
  - 2.    0.

Bz = [1;1]

    1.
    1.
```

Augment the plant plus servo compensator

```
A6 = [A,zeros(4,2);Bz*C,Az]

  - 2.    1.    0.    0.  :  0.    0.
    1.  - 2.    1.    0.  :  0.    0.
    0.    1.  - 2.    1.  :  0.    0.
    0.    0.    1.  - 1.  :  0.    0.
    - - - - - - - - - - - - - - - - -
    0.    0.    0.    1.  :  0.    2.
    0.    0.    0.    1.  : -2.    0.
```

```
B6 = [B;0;0]

    1.
    0.
    0.
    0.
    0.
    0.
```

Use Bass Gura, find the transformation matrix to take you to controller canonical form:

```
K6 = ppl(A6, B6, [-1,-2,-3,-4,-5,-6])

     14.     86.     299.     540.  - 1180.     340.
    |--------- Kx -------------||--- Kz ---------|
```

Check that the closed-loop poles of (A - BK) are where they should be:

```
>> eig(A6 - B6*K6)

   -6.0000
   -5.0000
   -4.0000
   -3.0000
   -2.0000
   -1.0000
```

Validation: step3.m

This gets a bit tricky. Matlab has the built in function *impulse()* which assumes

$$U = \delta(t)$$

Matlab has the build in function *step()* which assumes

$$U = u(t)$$

Matlab does *not* have a built in function which assumes

$$U = \cos(2t)\, u(t)$$

So, create a function:

```
y = step3(A, B, C, D, t, X0, U);
```

- {A, B, C, D} define the system's dynamics,
- t defines the time points
- X0 is the initial condition (currently not needed but we'll need that later....), and
- U is the input at time points defined in t.

In order to comput y(t), it's easiest to convert to discrete time. In discrete-time (z-domain), the dynamics are:

$$zX = A_z X + B_z U$$

$$Y = C_z X + D_z U$$

This is *much* easier to simulate in Matlab since you avoid numerical integration and all the errors that can produce. X(k) at all times are found from

```
X = X0;
for k=1:length(t)
   X = Az*X + Bz*U
   Y = Cz*X + Dz*U
   end
```

To make this work, you need to convert from continuous time (s-plane) to discrete-time (z-plane).

In the s-plane, the dynamics are:

$$sX = AX + BU$$

$$Y = CX + DU$$

In the z-plane:

$$zX = A_z X + B_z U$$

$$Y = C_z X + D_z U$$

The relationship for each term is:

$A_z = e^{AT}$ *matlab funciton expm(A*T)*

$B_z \approx BT$

$C_z = C$

$D_z = D$

where T is the time step used in the time vector, t.

- *This function requires constant step size in t.*

Finally, to allow for sinusoidal inputs, assume

- t is a column vector defining time at each point
- U is a column vector defining the input at each time point.

For example, if you want to find the step response

```
t = [0:0.01:10]';
U = 0*t + 1;
```

If you want to find the response to a 5 rad/sec sinusoidal input

```
t = [0:0.01:10]';
U = sin(5*t);
```

With that, the function *step3* is:

```
function [ y ] = step3( A, B, C, D, t, X0, U )

T = t(2) - t(1);
[m, n] = size(C);

npt = length(t);

Az = expm(A*T);
Bz = B*T;

X = X0;

y = zeros(npt, m);

y(1,:) = (C*X + D * ( U(1,:)' ) )';

for i=2:npt
    X = Az*X + Bz*( U(i,:)' );
    Y = C*X + D * ( U(i,:)' );

    y(i,:) = Y';

   end

end
```

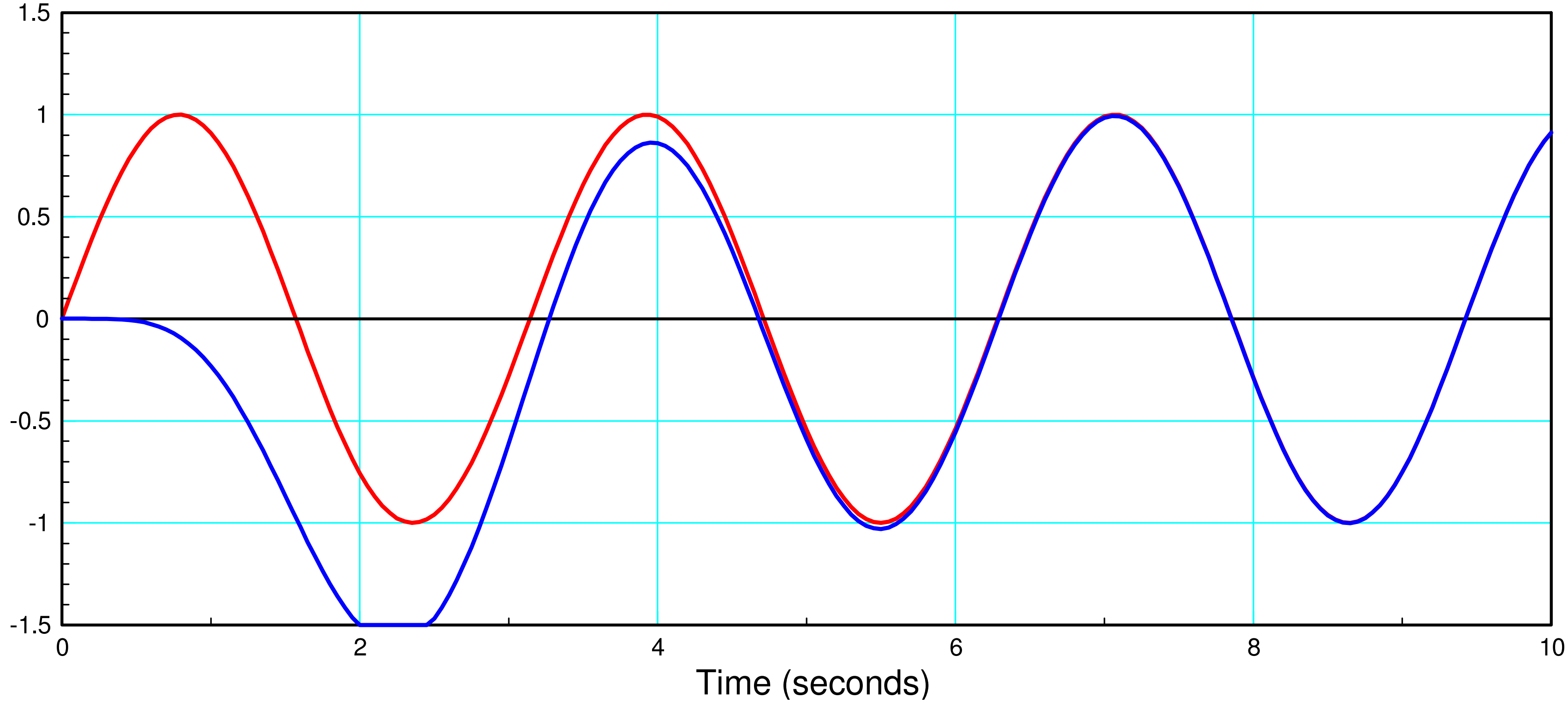
Validation:

Now that we have a fuction that can apply a sinusoidal input to a system, lets validate the previous servo compensator.

```
% Plant
A = [-2,1,0,0 ; 1,-2,1,0 ; 0,1,-2,1 ; 0,0,1,-1];
B = [1;0;0;0];
C = [0,0,0,1];

%Servo Compensator
Az = [0,2;-2,0];
Bz = [1;1];

% Augmented System
A6 = [A,zeros(4,2);Bz*C,Az];
B6u = [B; 0*Bz];
B6r = [0*B, -Bz];

C6 = [C,0,0]
D6 = 0;

K6 = ppl(A6, B6u, [-1,-2,-3,-4,-5,-6]);

t = [0:0.05:10]';
X0 = zeros(6,1);
R = sin(2*t);
```

Case 1: Step Response with Respect to the Set-Point: R

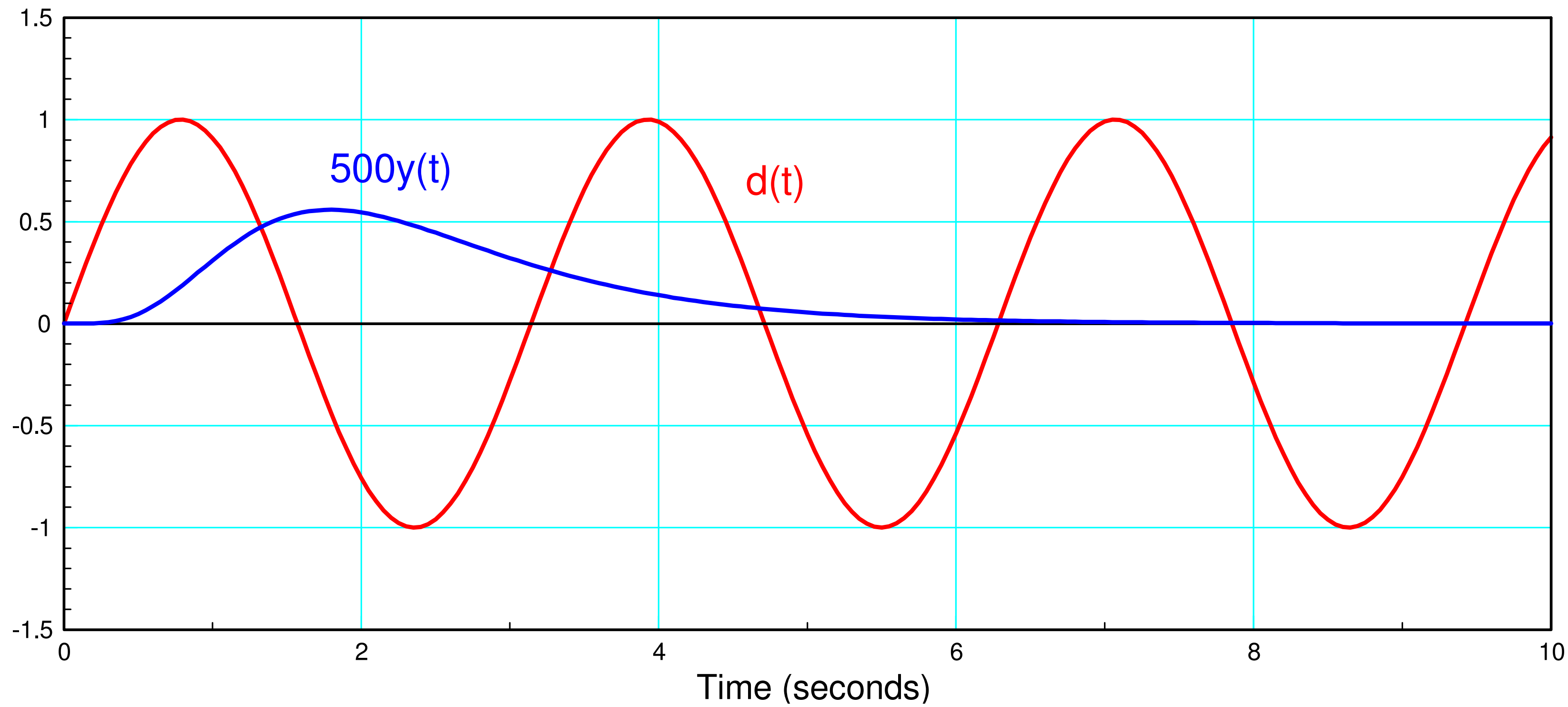
```
t = [0:0.05:10]';
R = sin(2*t);
y = step3(A6-B6u*K6, B6r, C6, D6, t, X0, R);
plot(t,y,'b',t,R,'r');
```



Output (blue) and Set-Point (red)

Case 2: Step Response with Respect to a 2 rad/sec disturbance

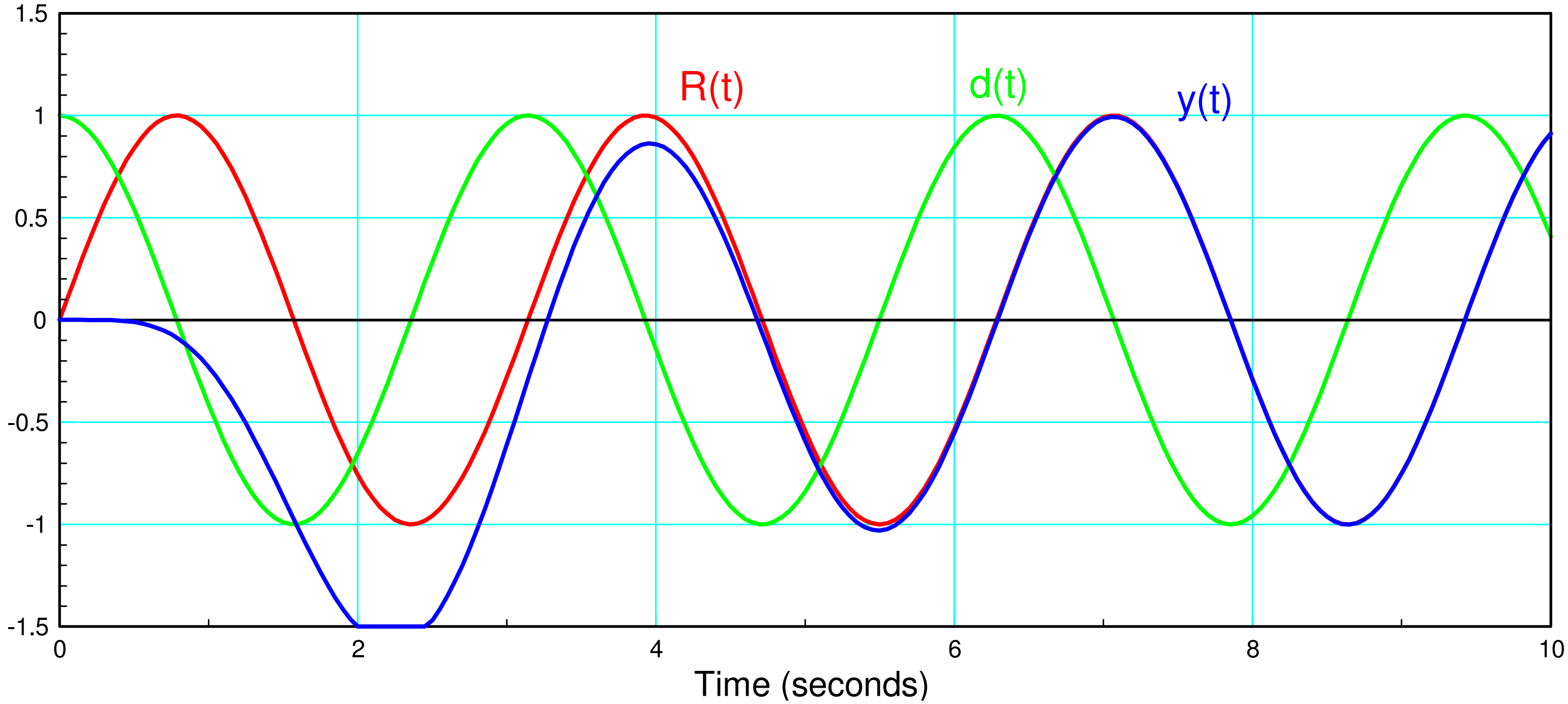
```
d = sin(2*t);
y = step3(A6-B6u*K6, B6u, C6, D6, t, X0, d);
plot(t,y*500,'b',t,R,'r');
```



Case 3: R = sin(2t), d = cos(2t)

- Create two inputs so you can adjust as you like...

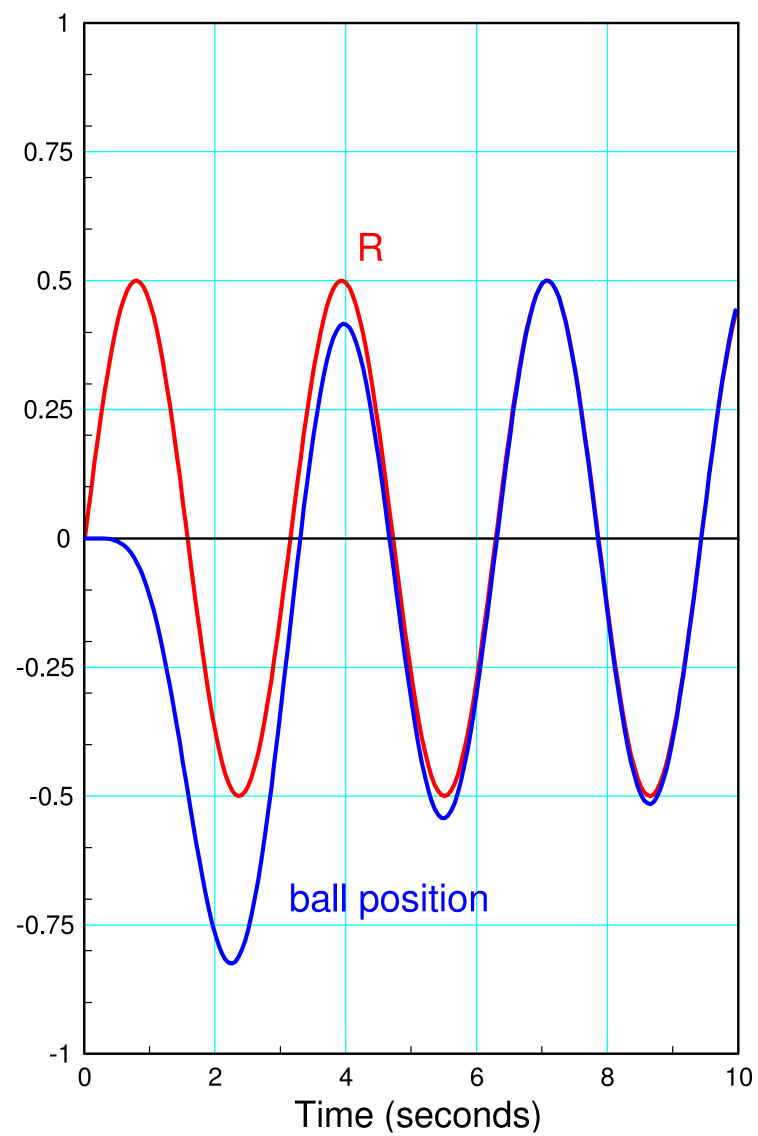
```
R = sin(2*t);
d = cos(2*t);
y = step3(A6-B6u*K6, [B6r, B6u], C6, [0, 0], t, X0, [R, d]);
plot(t,y,'b',t,R,'r',t,d,'g');
```



Ball & Beam Simulation

- Track a 2 rad/sec sinusoid
- Poles placed at {-1, -2, -3, -4, -5, -6}

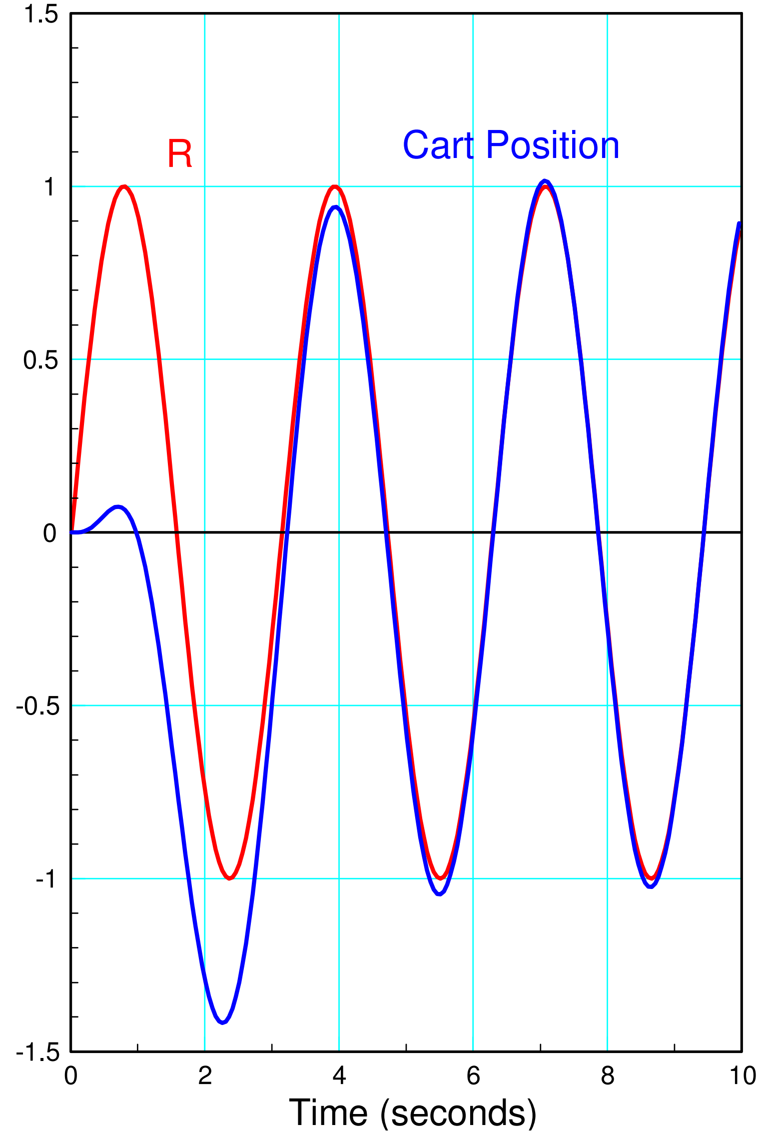
```
X = zeros(4,1);
Z = zeros(2,1);
dt = 0.01;
t = 0;
Kx = [ -170.95  205.21 -111.60  25.20];
Kz = [  202.2938  -58.2880];
y = [];

while(t < 10)
 Ref = 0.5*sin(2*t);
 U = -Kz*Z - Kx*X;
 dX = BeamDynamics(X, U);
 dZ = Az*Z + Bz*(X(1) - Ref);
 X = X + dX * dt;
 Z = Z + dZ*dt;
 y = [y ; Ref, X(1)];
 t = t + dt;
 BeamDisplay(X, Ref);
 end
```



Cart & Pendulum Simulation

- Servo compensator with poles at {j2, -j2}
- Closed-loop poles = {-1, -2, -3, -4, -5, -6}
- Tracks a 2 rad/sec set point

```
X = zeros(4,1);
Z = zeros(2,1);
dX = zeros(4,1);
Ref = 1;
dt = 0.01;
t = 0;
Kx = [ -146.87 -518.27 -120.43 -162.43];
Kz = [  171.0145  -49.2754];
Az = [0,2;-2,0];
Bz = [1;1];

while(t < 10)
  Ref = 1.0*sin(2*t);
  U = - Kx*X - Kz*Z;
  dX = CartDynamics(X, U);
  dZ = Az*Z + Bz*(X(1) - Ref);
  X = X + dX * dt;
  Z = Z  + dZ*dt;
  t = t + dt;
  CartDisplay(X, Ref);
end
```



Summary

Not surprisingly, adding a servo compensator with poles at {+j2, -j2} creates a system which can

- Track 2 rad/sec set points, and
- Rejet 2 rad/sec disturbances

separately or both at the same time.