6.3100 Dynamical System Modeling and Control Design

Spring 2024 – Lecture 6

MATLAB tools

- Line-following example:

System equation (with PD-control)

denominator

$$d[n] - 2d[n-1] + d[n-2]\left(1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T\right) + d[n-3](-K_d V\gamma\Delta T)$$
$$= d_d[n-2]\left(\Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma\right) + d_d[n-3](-K_d\Delta T\, V\gamma)$$

numerator

Coefficients:

MATLAB tools

- Line-following example:

System equation (with PD-control)

denominator

$$d[n] - 2d[n-1] + d[n-2]\left(1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T\right) + d[n-3](-K_d V\gamma\Delta T)$$
$$= d_d[n-2]\left(\Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma\right) + d_d[n-3](-K_d\Delta T\, V\gamma)$$

numerator

Coefficients:

Denominator terms	Numerator terms
d[n]: 1	d_d[n]: 0
d[n-1]: -2	d_d[n-1]: 0
d[n-2]: $1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T$	d_d[n-2]: $\Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma$
d[n-3]:$-K_d V\gamma\Delta T$	d_d[n-3]: $-K_d\Delta T\, V\gamma$

MATLAB tools

- Line-following example:

System equation (with PD-control)

denominator

$$d[n] - 2d[n-1] + d[n-2]\left(1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T\right) + d[n-3](-K_d V\gamma\Delta T)$$
$$= d_d[n-2]\left(\Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma\right) + d_d[n-3](-K_d\Delta T\, V\gamma)$$

numerator

Coefficients:

$$num = [0, 0, \Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma, -K_d\Delta T\, V\gamma]$$

$$den = [1, -2, 1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T, -K_d V\gamma\Delta T\ ]$$

MATLAB tools

- MATLAB code

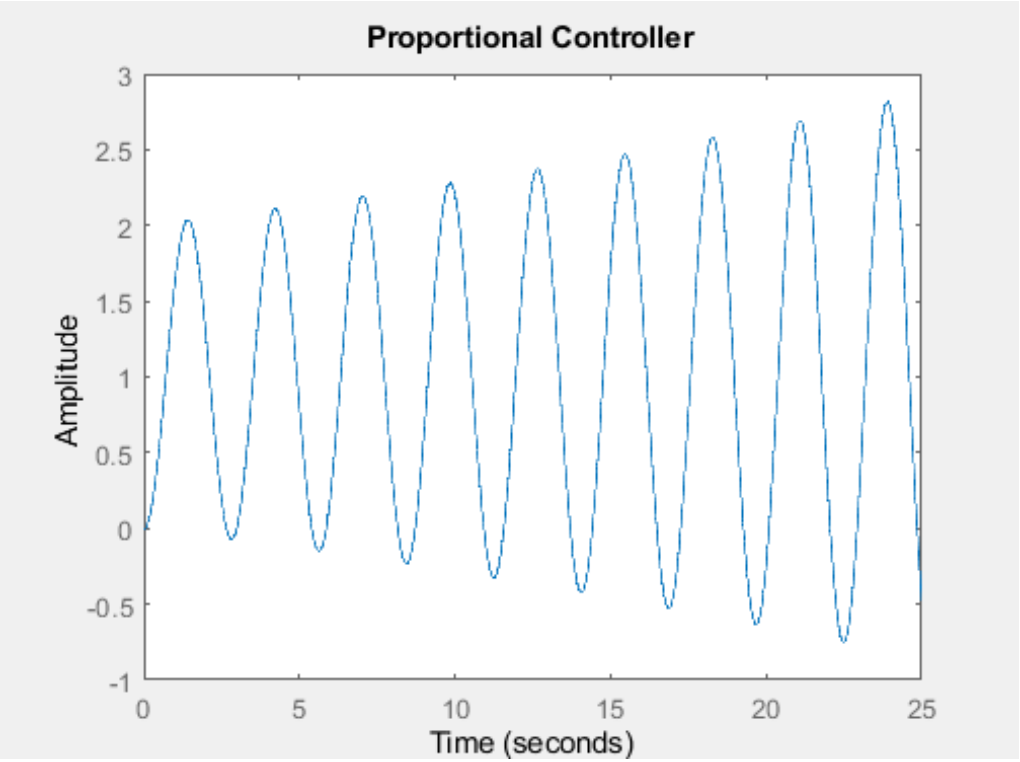
```
%P controller
Kd = 0; Kp = 5; gamma = 1; V = 1; dT=0.01;
num_kp = [0   0 (dT^2*gamma*V*Kp+Kd*dT*V*gamma) -Kd*dT*V*gamma];
den_kp = [1 -2 (1+dT^2*V*gamma*Kp+Kd*V*gamma*dT) -Kd*V*gamma*dT];

%calculate time-domain solution
close all; figure(1); hold on
subplot(1,2,1); hold on
sys_kp = tf(num_kp,den_kp,dT,'variable','z^-1');
step(sys_kp,25)
title('Proportional Controller')
```

Formulate transfer function

Simulate from t = 0 to t = 35

MATLAB tools

- Simulation results



Key takeaway:
Proportional control (of this system) is unstable!

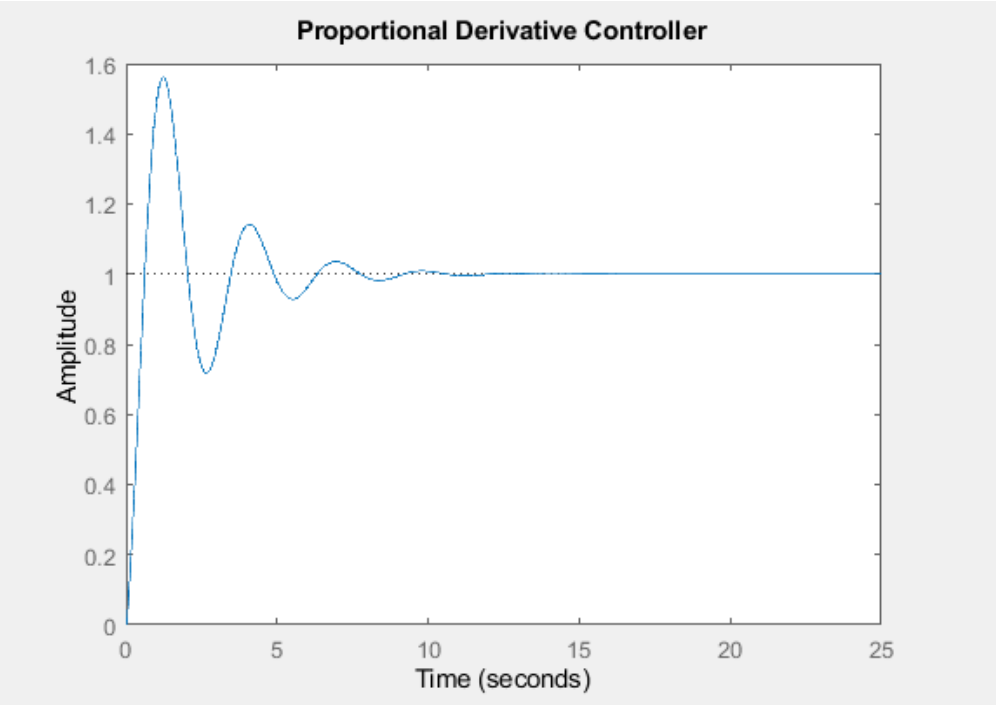
MATLAB tools

- Case 2: Proportional derivative (PD) control

```
%PD controller
Kd = 1;
subplot(1,2,2); hold on
num_kpd = [0  0 (dT^2*gamma*V*Kp+Kd*dT*V*gamma) -Kd*dT*V*gamma];
den_kpd = [1 -2 (1+dT^2*V*gamma*Kp+Kd*V*gamma*dT) -Kd*V*gamma*dT];
sys_kpd = tf(num_kpd,den_kpd,dT,'variable','z^-1');
step(sys_kpd,25)
title('Proportional Derivative Controller')
```

MATLAB tools

- Simulation results



Key takeaway:
Proportional derivation (PD) control (of this system) is stable!

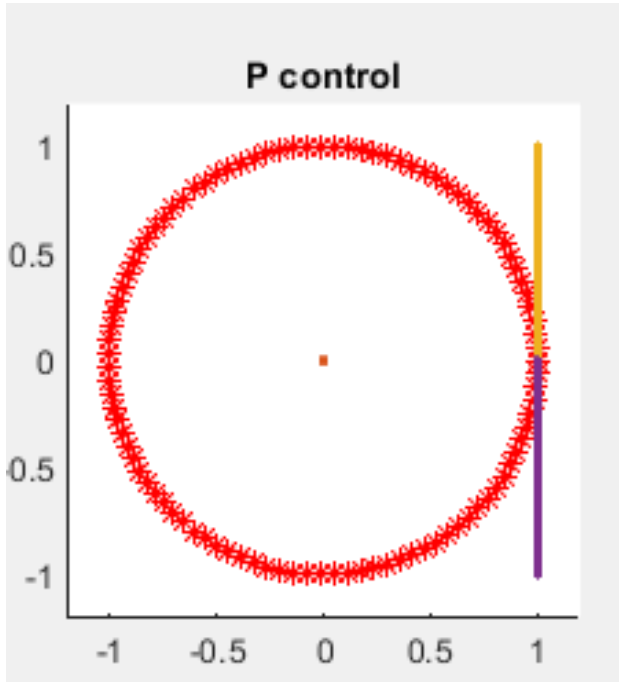
MATLAB tools

- Root locus plots (P-control)

Draw unit circle

```
figure(2)
subplot(1,2,1); hold on; axis equal; axis([-1.2 1.2 -1.2 1.2])
th = linspace(0, 2*pi, 100)';
plot(cos(th)+sqrt(-1)*sin(th), '*r');  % Draw a unit circle.
hold on

N_roots = 1000;
Kp_vec = linspace(0,10000,N_roots);
roots_mat = zeros(N_roots,3);
Kd = 0.0;
for i = 1:length(Kp_vec)
    polyc = [1 -2 (1+dT^2*V*gamma*Kp_vec(i)+Kd*V*gamma*dT) -Kd*V*gamma*dT];
    roots_mat(i,:)= roots(polyc);
end
plot(roots_mat,'.');
title('P control')
```

Draw root locus



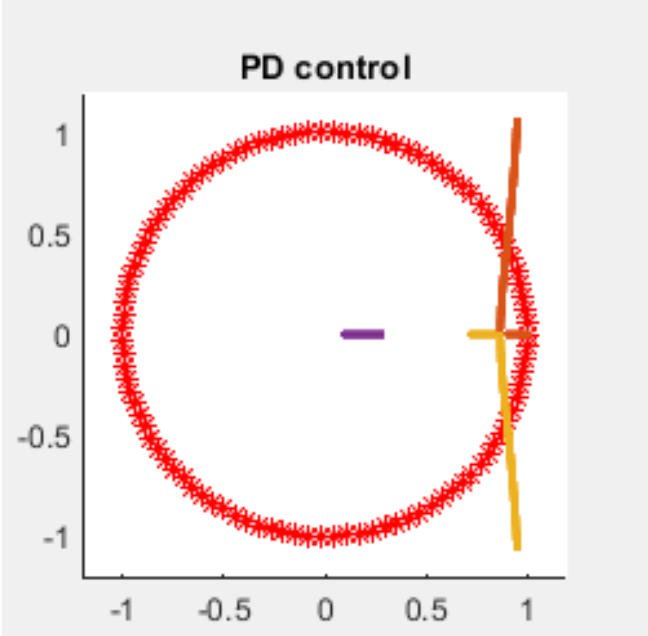
MATLAB tools

- Root locus plots (PD-control)

Draw unit circle

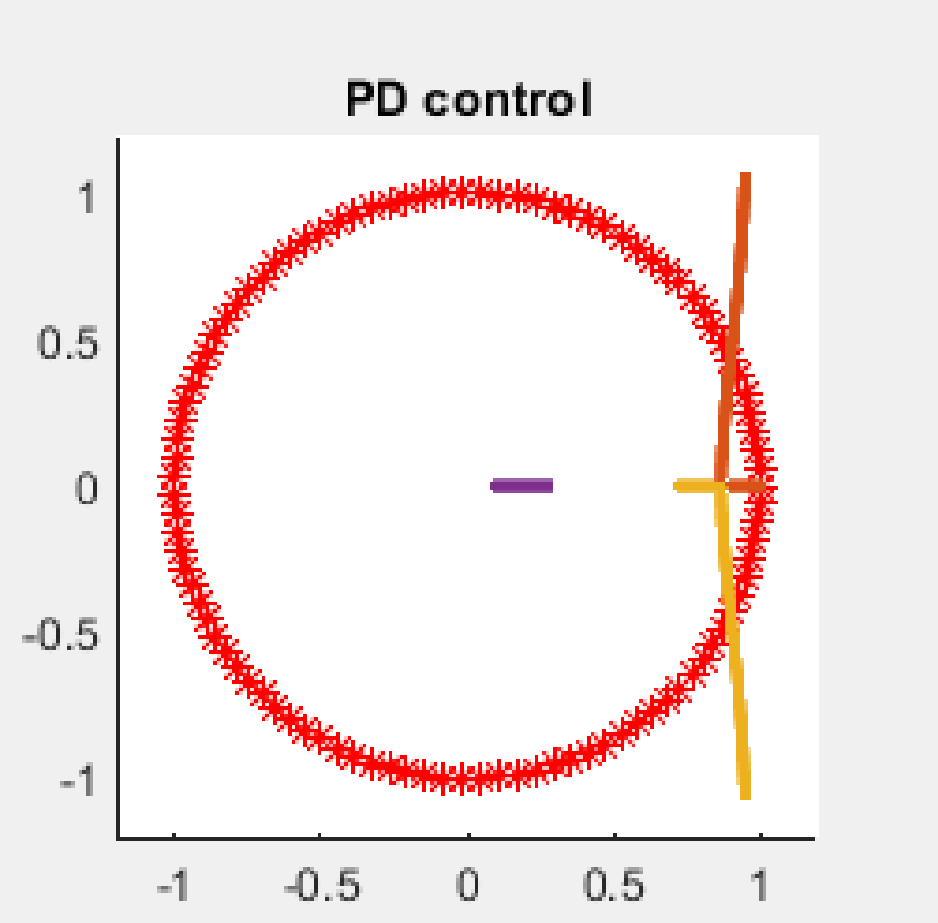
```
subplot(1,2,2); hold on; axis equal; axis([-1.2 1.2 -1.2 1.2])
th = linspace(0, 2*pi, 100)';
plot(cos(th)+sqrt(-1)*sin(th), '*r');  % Draw a unit circle.
hold on

Kp_vec = linspace(0,10000,N_roots);
roots_mat = zeros(N_roots,3);
Kd = 20;
for i = 1:length(Kp_vec)
    polyc = [1 -2 (1+dT^2*V*gamma*Kp_vec(i)+Kd*V*gamma*dT) -Kd*V*gamma*dT];
    roots_mat(i,:)= roots(polyc);
end
plot(roots_mat,'.');
title('PD control')
```

Draw root locus



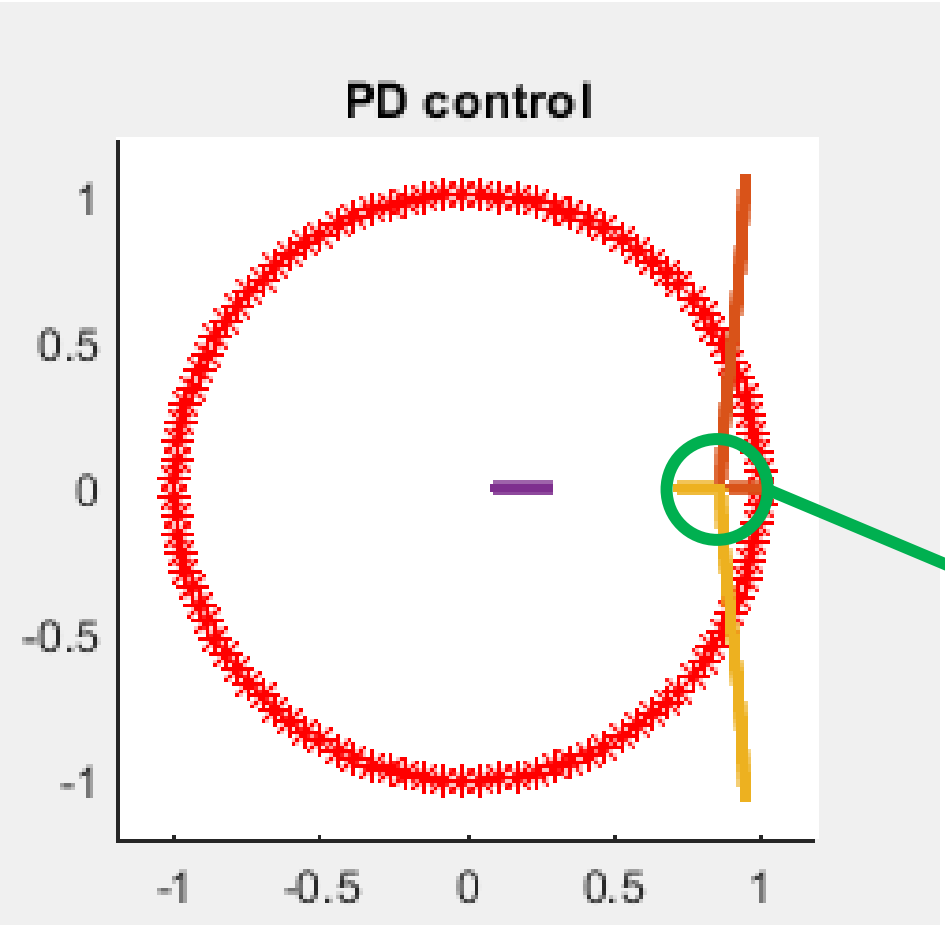
MATLAB tools

- Finding optimal point on the root locus plot (where is the optimal point?)



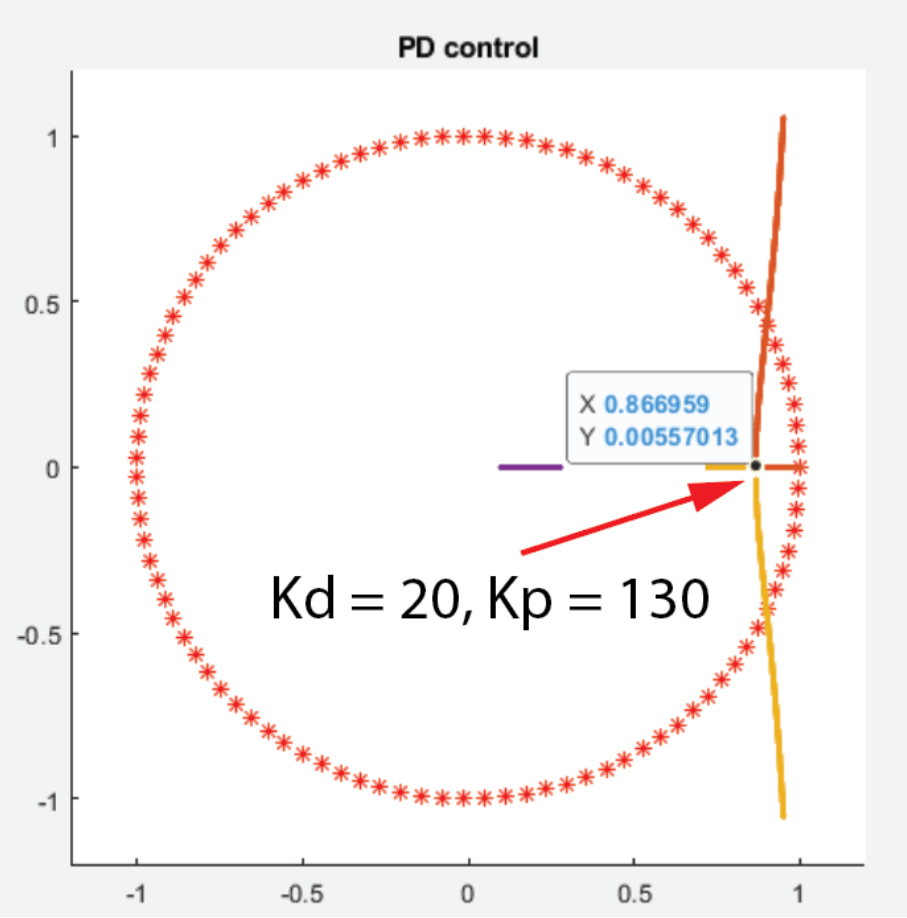
MATLAB tools

- Finding optimal point on the root locus plot



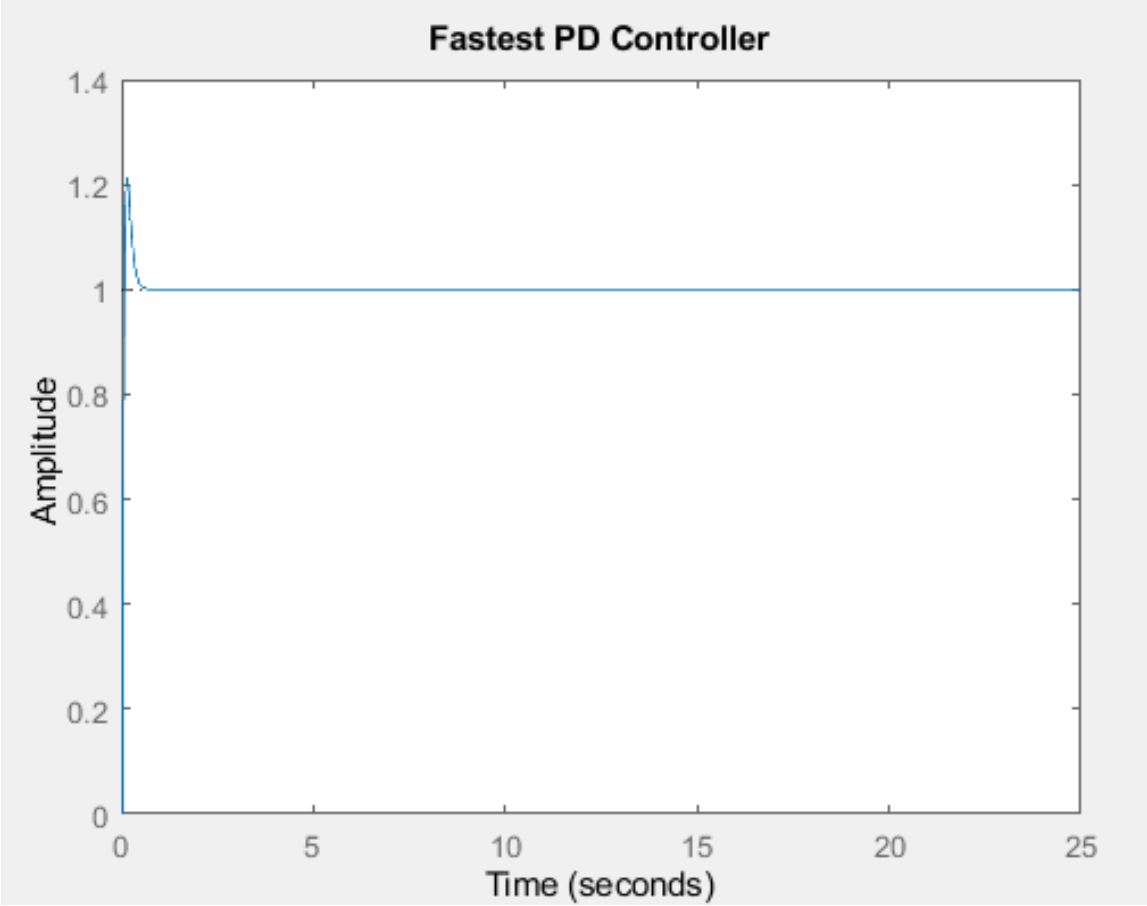
MATLAB tools

```
%find magnitude of 3 natural frequencies for each case
mag_vec_1 = abs(roots_mat(:,1));
mag_vec_2 = abs(roots_mat(:,2));
mag_vec_3 = abs(roots_mat(:,3));

%find maximum for each case
cum_mag_vec = max([mag_vec_1 mag_vec_2 mag_vec_3],[],2);

%find the minimum
[~,ind]=min(cum_mag_vec);
Kp_min = Kp_vec(ind);
```



MATLAB tools

- Fastest convergence



February 21, 2024

6.3100 Lecture 6 Notes – Spring 2024

MATLAB tools for solving 2nd order systems and introduction of PID control

Dennis Freeman, Elfar Adalsteinsson, and Kevin Chen

Outline:

1. MATLAB tools for solving 2nd order systems
2. Introduction of PID control

1. MATLAB tools for solving 2nd order systems

In the previous class, we introduced 2nd and 3rd order control systems. We saw that system stability is determined by the magnitude of natural frequencies, and the natural frequencies are determined by our control parameters Kp and Kd. We also learned that natural frequencies can be complex numbers and calculating them analytically is tedious.

We will solve the same line following problem using MATLAB. From the previous lecture, we calculate the system equation:

$$d[n] - 2d[n-1] + d[n-2]\left(1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T\right) + d[n-3](-K_d V\gamma\Delta T) = d_d[n-2]\left(\Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma\right) + d_d[n-3](-K_d\Delta T\, V\gamma)$$

First, we will extract the "numerator" and "denominator" coefficients. Their naming will be explained next week when we introduce Z-transform techniques. We have:

$$num = [0, 0, \Delta T^2\gamma V k_p + K_d\Delta T\, V\gamma, -K_d\Delta T\, V\gamma]$$

$$den = [1, -2, 1 + \Delta T^2 V\gamma K_p + K_d V\gamma\Delta T, -K_d V\gamma\Delta T\,]$$

Case 1: proportional control

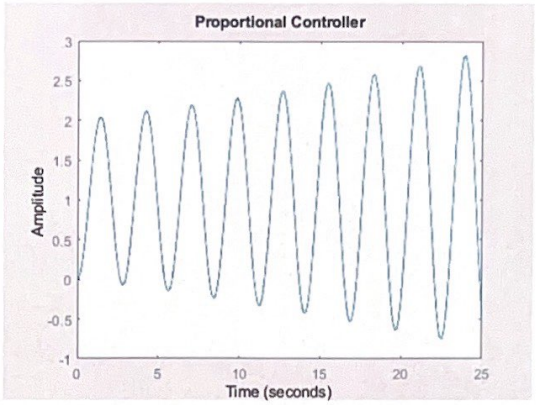
We first define the coefficient values, and then set the numerator and denominator.

```
%P controller
Kd = 0; Kp = 5; gamma = 1; V = 1; dT=0.01;
num_kp = [0  0 (dT^2*gamma*V*Kp+Kd*dT*V*gamma) -Kd*dT*V*gamma];
den_kp = [1 -2 (1+dT^2*V*gamma*Kp+Kd*V*gamma*dT) -Kd*V*gamma*dT];
```

Next, we formulate the control system and then simulate the step response.

```
%calculate time-domain solution
close all; figure(1); hold on
subplot(1,2,1); hold on
sys_kp = tf(num_kp,den_kp,dT,'variable','z^-1');
step(sys_kp,25)
title('Proportional Controller')
```

The time-domain solution is plotted in MATLAB, and it is shown below.



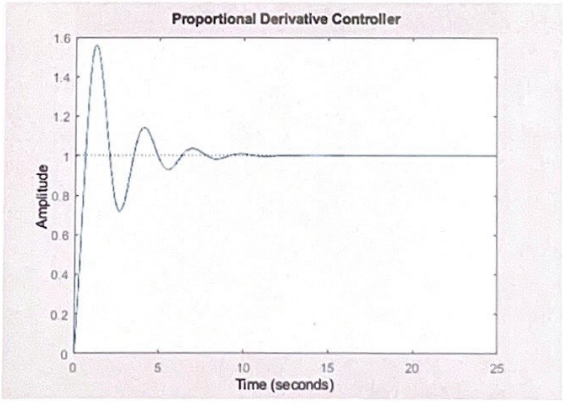
This simulation shows that implementing a proportional controller is unstable.

Case 2: proportional derivative (PD) control

We can solve the same problem again, now setting Kd to 1. The code is given below:

```
%PD controller
Kd = 1;
subplot(1,2,2); hold on
num_kpd = [0  0 (dT^2*gamma*V*Kp+Kd*dT*V*gamma) -Kd*dT*V*gamma];
den_kpd = [1 -2 (1+dT^2*V*gamma*Kp+Kd*V*gamma*dT) -Kd*V*gamma*dT];
sys_kpd = tf(num_kpd,den_kpd,dT,'variable','z^-1');
step(sys_kpd,25)
title('Proportional Derivative Controller')
```

The simulation result is shown below:



Based on this simulation result, we observe that the PD controller is stable. The car can follow a line with no steady-state error. Next, we can use the MATLAB tools to study stability and optimize the values of Kp and Kd.

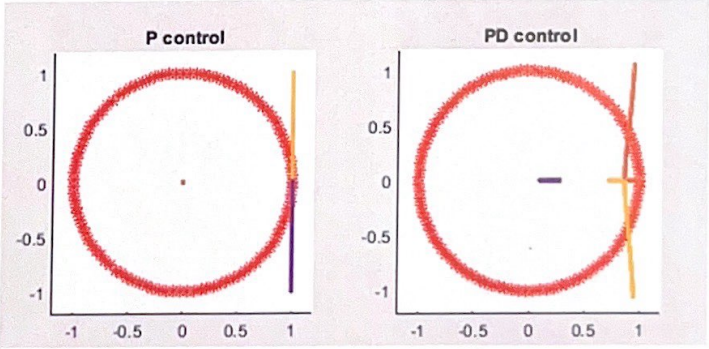
Root locus plots

To study controller stability, we sweep through different combination of Kp and Kd and look at the corresponding natural frequencies. Here the natural frequencies are given by the roots of the "denominator" equation. Equivalently, they are the roots of the characteristic (or homogeneous) equation.

In the first case, we set Kd = 0 and vary Kp in the range of 1 to 10000. In the second case, we set Kd = 20, and vary Kp in the range of 1 to 10000. The code is given below:

```
figure(2)
subplot(1,2,1); hold on; axis equal; axis([-1.2 1.2 -1.2 1.2])
th = linspace(0, 2*pi, 100)';
plot(cos(th)+sqrt(-1)*sin(th), '*r');  % Draw a unit circle.
hold on

N_roots = 1000;
Kp_vec = linspace(0,10000,N_roots);
roots_mat = zeros(N_roots,3);
Kd = 0.0;
for i = 1:length(Kp_vec)
    polyc = [1 -2 (1+dT^2*V*gamma*Kp_vec(i)+Kd*V*gamma*dT) -Kd*V*gamma*dT];
    roots_mat(i,:)= roots(polyc);
end
plot(roots_mat,'.');
title('P control')

subplot(1,2,2); hold on; axis equal; axis([-1.2 1.2 -1.2 1.2])
th = linspace(0, 2*pi, 100)';
plot(cos(th)+sqrt(-1)*sin(th), '*r');  % Draw a unit circle.
hold on

Kp_vec = linspace(0,10000,N_roots);
roots_mat = zeros(N_roots,3);
Kd = 20;
for i = 1:length(Kp_vec)
    polyc = [1 -2 (1+dT^2*V*gamma*Kp_vec(i)+Kd*V*gamma*dT) -Kd*V*gamma*dT];
    roots_mat(i,:)= roots(polyc);
end
plot(roots_mat,'.');
title('PD control')
```

The MATLAB generated plots are shown below.



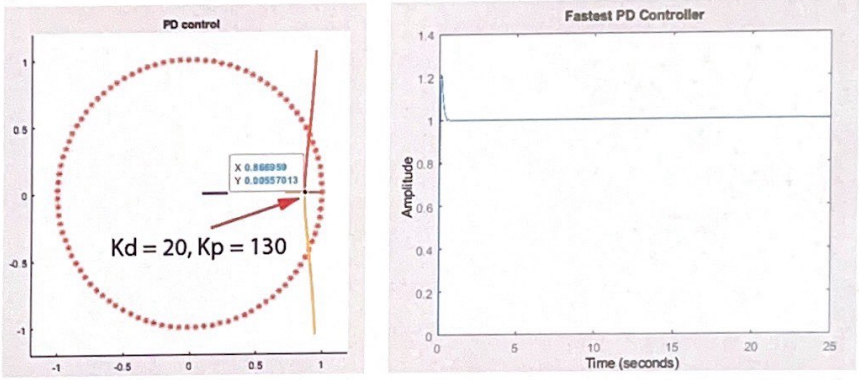
Here the red circle indicates the unit circle in the complex plane. We observe P control is unstable because the natural frequencies are outside of the unit circle regardless of the Kp values we pick. For the case of PD control, there are cases where all 3 natural frequencies are within the unit circle, which implies the system is stable.

Given this range of possible Kp, what is the optimal value that we can pick? We should pick the value of Kp where the maximum of the three natural frequencies is at minimum. This condition implies the solution will decay (converge) as quickly as possible.

The corresponding code is given below:

```
%find magnitude of 3 natural frequencies for each case
mag_vec_1 = abs(roots_mat(:,1));
mag_vec_2 = abs(roots_mat(:,2));
mag_vec_3 = abs(roots_mat(:,3));

%find maximum for each case
cum_mag_vec = max([mag_vec_1 mag_vec_2 mag_vec_3],[],2);

%find the minimum
[~,ind]=min(cum_mag_vec);
Kp_min = Kp_vec(ind);
```

The solution is shown below:



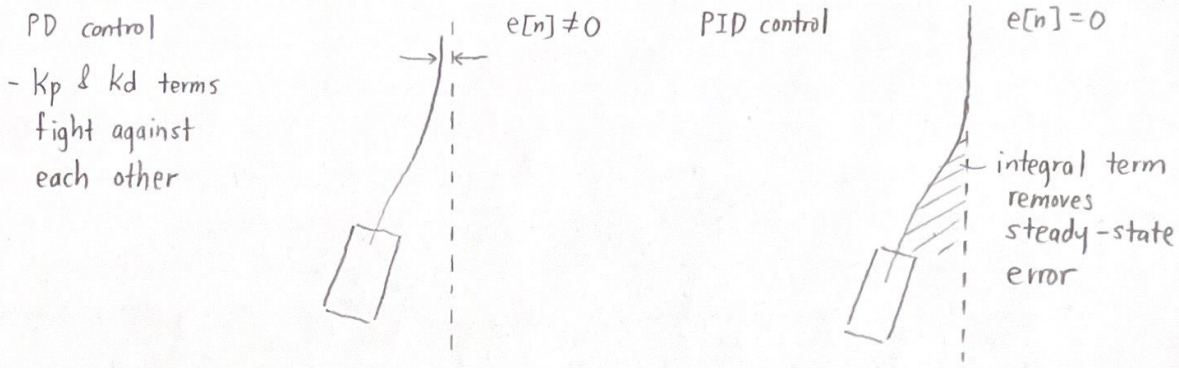
These tools will be helpful when solving Post Lab 1.

2. Introduction of PID control

While a PD controller can stabilize a 2nd or 3rd order system, it has other issues that we need to investigate and solve. For example, what is the steady-state error of our line following example? It turns out that for this simple example, the steady-state error is 0. However, if there exists a small damping term β (which is very common in physical systems), then the steady-state error is non-zero.

How do we remove the steady-state error in a control system? Let's first think about why PD controller can contribute to a steady-state error. Consider the line following example again.

Essentially, the P term and the D term sometimes fight against each other, which can leave a net error. This is drawn below.



How can we remove the steady-state error? We can introduce another term that accumulates all the errors in the past. If there is a steady-state error, then this term will be non-zero, and it will hopefully push the system to 0. This term is called an "integral (or sum) term". A "PID" (or "PSD") controller is defined as:

$$c[n] = k_p e[n] + k_d \frac{e[n] - e[n-1]}{\Delta T} + k_i \Delta T \sum_{m=0}^{m=n} e[m]$$

where the error term is defined as:

$$e[n] = d_d[n] - d[n]$$

Here we need to choose three parameter values: kp, kd, and ki. In addition, note that there is a sum term $k_i \Delta T \sum_{m=0}^{m=n} e[m]$ in our controller. This can be tedious to analyze. Thus far, we have been shifting indices and solving the control system directly. This becomes very tedious when the control system becomes complex. In addition, we have been only studying step response. How does a system follow other trajectories (ramp, impulse, or any arbitrary functions)? We will need a new tool. Starting from next week, we will introduce the Z-transform technique.