Control Systems Engineering

- **Prof. John Wen takes over for the rest of semester**
- **Office Hour: M/Th 1-2pm CII 8105**
- **Contact: wenj@rpi.edu x6156**

http://www.cats.rpi.edu/~wenj/ECSE444F05

Remainder of Semester

Date	Topic
11/4	MATLAB & Simulink, PID control and gain tuning
11/8	Design project description
11/11	Introduction to the state space method
11/15	Full state feedback and pole placement
11/18	State estimator
11/22	Tracking control
11/29	Digital implementation
12/2	Design example
12/6	Design example
12/9	Review

Your Grade

- **Design Project: 25%, group of 2, design report, due on 12/9.**
- **Final exam: 25% (12/14, 6:30-9:30)**
- **Homework: 25% total**

Today

- MATLAB & Simulink
- PID Control and Gain Tuning

MATLAB

A powerful package with built-in math functions, array and matrix manipulation capabilities, plotting and lots of add-on toolboxes (e.g., control, image processing, symbolic manipulation, block diagram programming, i.e., Simulink, etc.)

You can install MATLAB 7.0, Simulink, & Control toolbox on your laptop (you'll need to connect to license server to use it; through VPN off-campus) (http://www.rpi.edu/dept/arc/web/licenses/matlab_license.html)

MATLAB (Cont.)

An interpretive environment, may use scripts

- Vectors: theta=[theta_1;theta_2];
- Matrices: M=[M11 M12; M21 M22];
- Polynomials: p=[a3 a2 a1 a0];
- Transfer functions: G=tf(num,den);
- Linear simulation:
  - step response: step(G);
  - impulse response: impulse(G);
  - general response: y=lsim(G,u,t);

MATLAB (Cont.)

- Plotting: plot(t,x1,t,x2);xlabel('time (sec)');ylabel('theta (deg)'); title('theta(t)'); legend('\theta_1','\theta_2');
- Printing (to printer or file)

  print -f -d<device type> <file name>
- Using m-files in MATLAB

  use any editor (or MATLAB built-in editor, just type in **`edit`**)

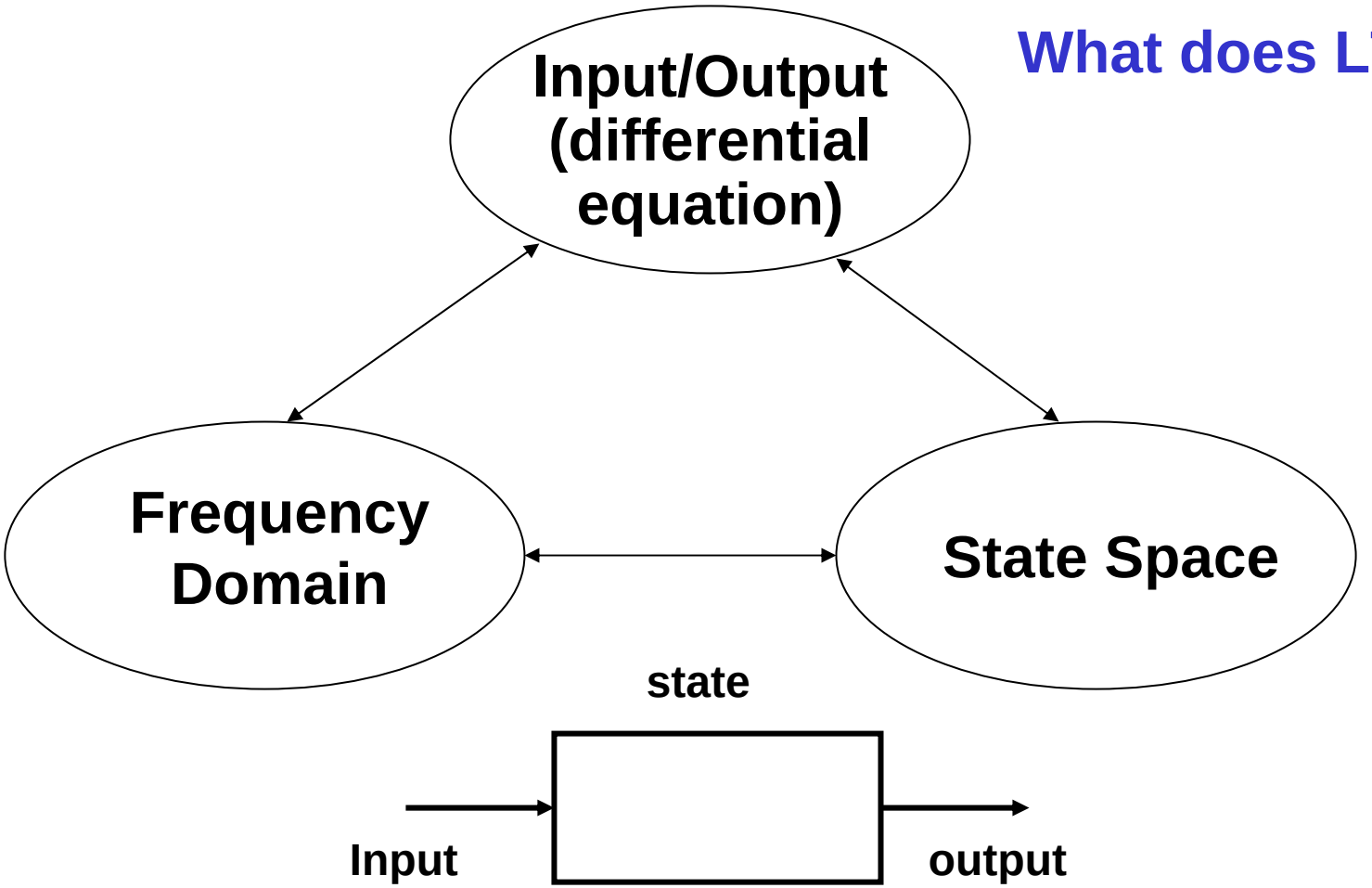
  `function f=func(t,x)` ...
- Getting **help** in MATLAB: help <function name> or just help

Always label axes, include units, and include legends in your plots.

on-line tutorial: **http://www.engin.umich.edu/group/ctm/**

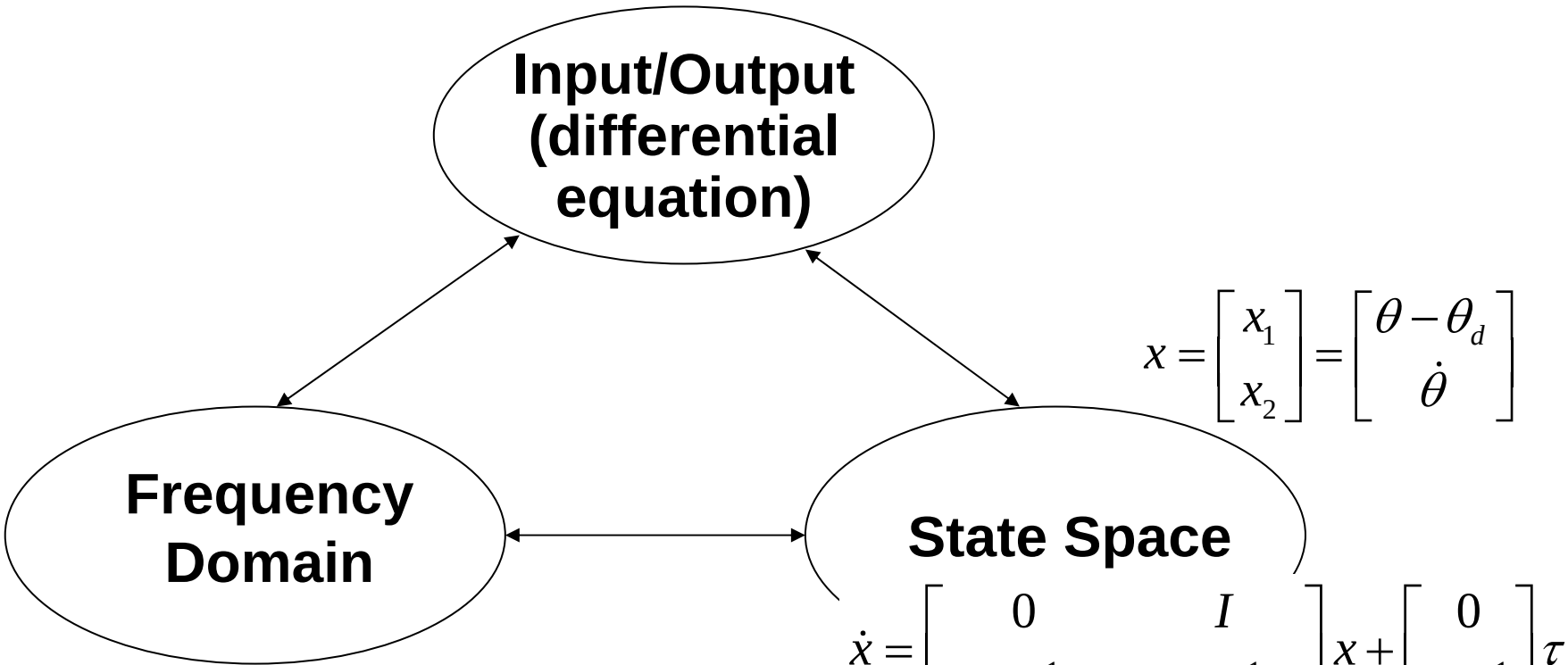
Description of LTI Systems

What does LTI mean?



Description of LTI Systems

$$M\ddot{\theta} + D\dot{\theta} + K(\theta - \theta_d) = \tau$$



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta - \theta_d \\ \dot{\theta} \end{bmatrix}$$

$$\Delta\theta(s) = \underbrace{(s^2 M + Ds + K)^{-1}}_{G(s)} \tau(s)$$

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}}_{A} x + \underbrace{\begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}}_{B} \tau$$

$$y = \underbrace{\begin{bmatrix} I & 0 \end{bmatrix}}_{C} x + \underbrace{0}_{D} \tau$$

MATLAB Description of LTI

Each LTI is treated as an *object* with a variety of possible description:

transfer function: `tf(num,den)` (numerator and denominator polynomials)

pole/zero/gain: `zpk(z,p,k)` (zeros, poles, gain)

state space: `ss(A,B,C,D)` (state space parameters)

Open Loop Linear System Response

Impulse response:

```
y=impulse(G)
```

step response:

```
y=step(G)
```

general response:

```
y=lsim(G,u,t)
```

Bode plot:

```
y=bode(G)
```

```
poles/zeros/dampings
```

```
pole(G),zero(G), damp(G)
```

pole/zero plot:

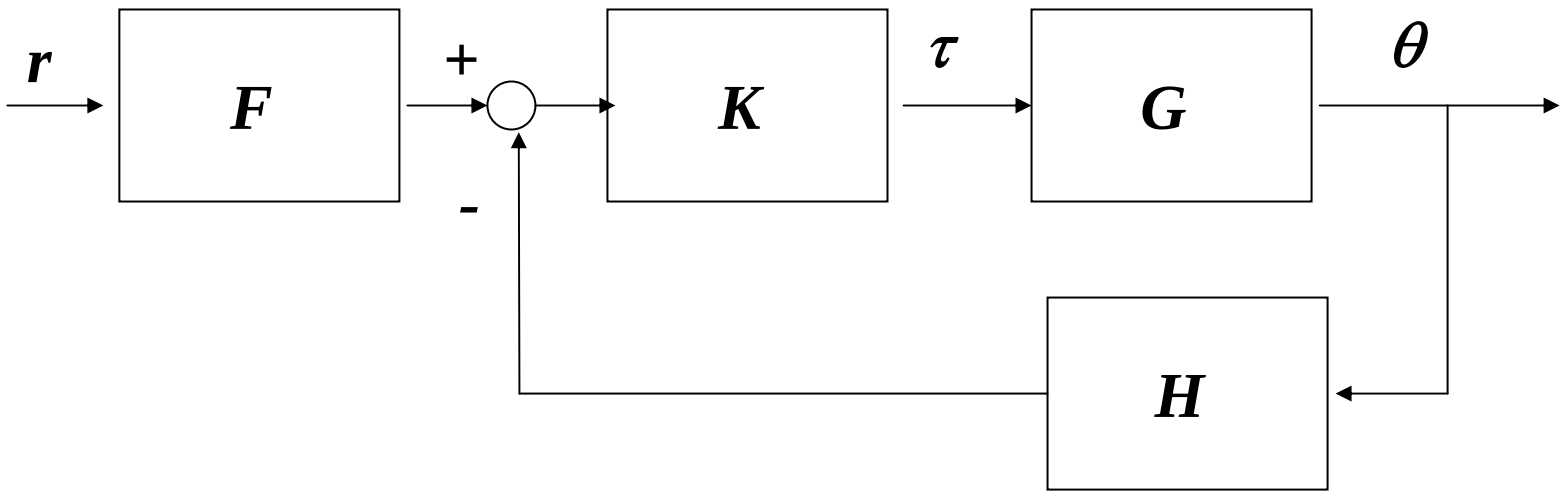
```
pzmap(G)
```

gain/phase margin (robustness):

```
margin(G)
```

Incorporation of Control

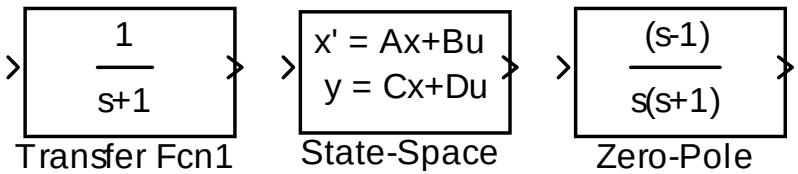
Interconnection of LTI systems:



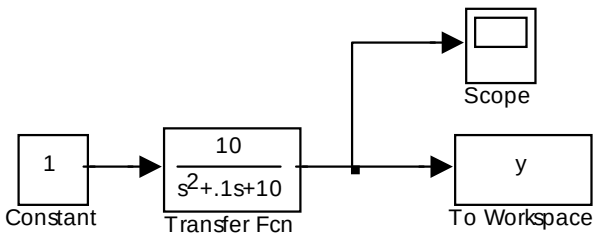

```
Gcl = feedback(G*K,H)*F
```

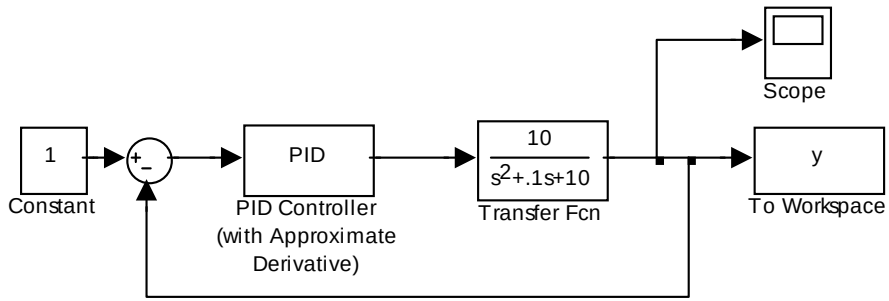
Simulink

Instead of command line entries, it may be easier to use a block diagram programming tool:



LTI block





Effect of Sampling

Most control systems these days are digital in nature so sampling is inherent (through A/D for sensor, which contains a sampler, and D/A for actuators, which contains a zero-order-hold).

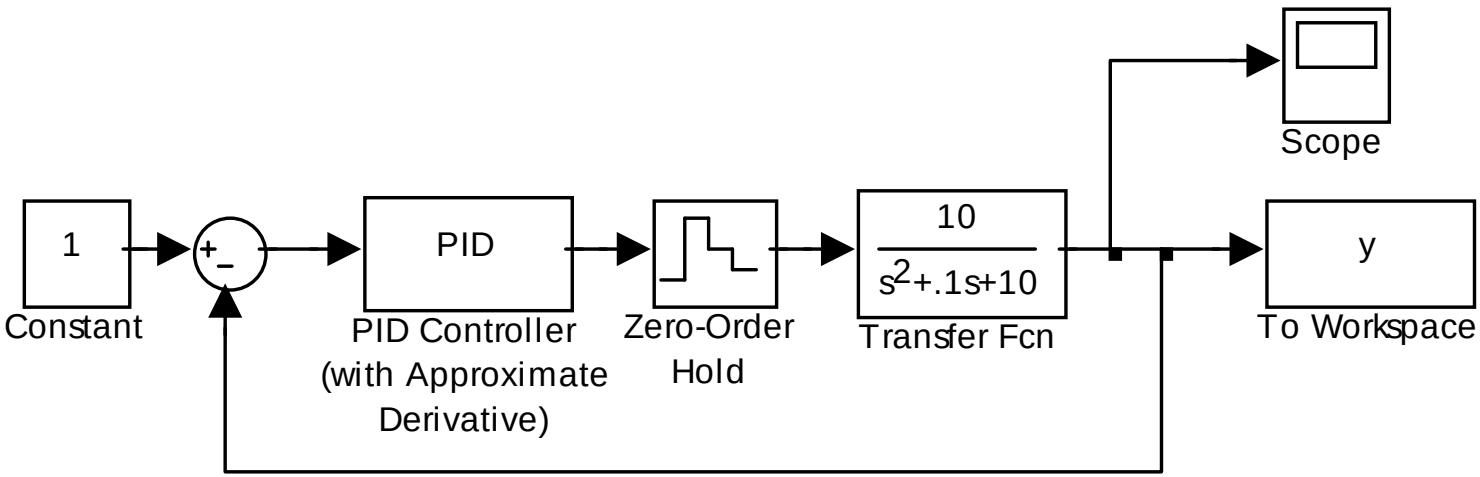
To analyze the effect of sampling, we can find the equivalent discrete time system:

```
Gd = c2d(G,ts);%ts=sampling period (sec)
```

Gd is also an LTI object and the commands for LTI may be applied.

Adding Sampling to Simulink Diagram

To add sampling to your continuous time simulation, just add a zero-order-hold block (in the discrete time system library) to the input, then set the sampling time.



MATLAB and Simulink

MATLAB and Simulink are the de facto industry standard, you should try to master them.

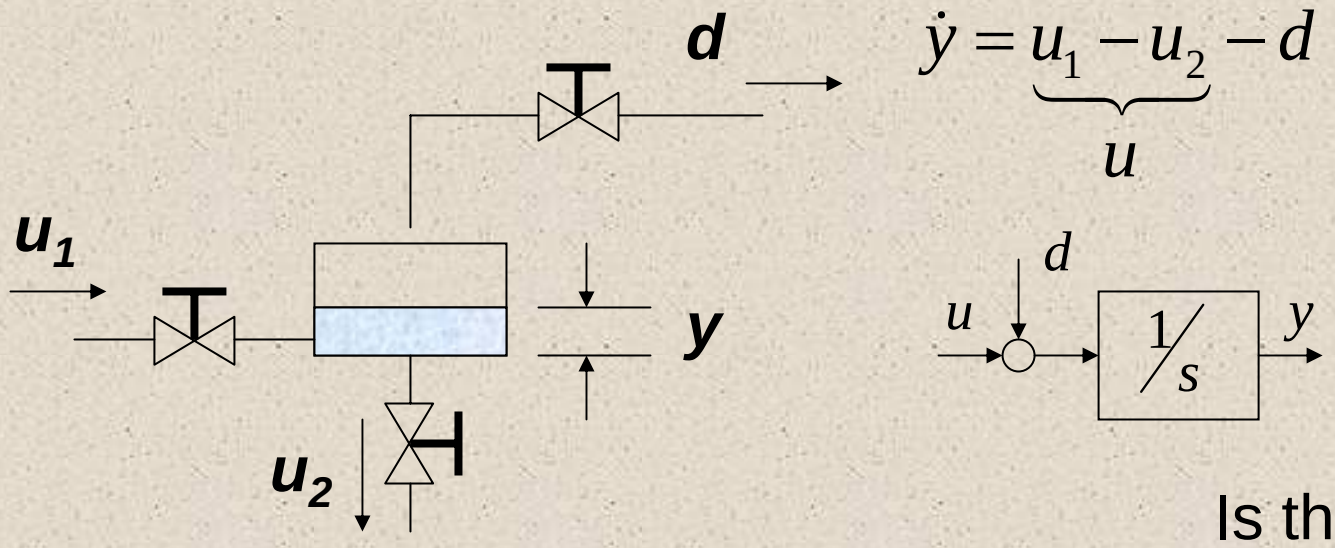
You'll also need to use MATLAB and Simulink in your design project.

Quick Review of Control Design

Why feedback?

- Stability (closed loop poles in LHP)
- Performance (rise time, overshoot, settling time)
- Steady state (steady state error)
- Robustness (gain/phase margin)
- Disturbance rejection (loop shaping)
- Tracking (loop shaping / feedforward)

Example: 1st Order System



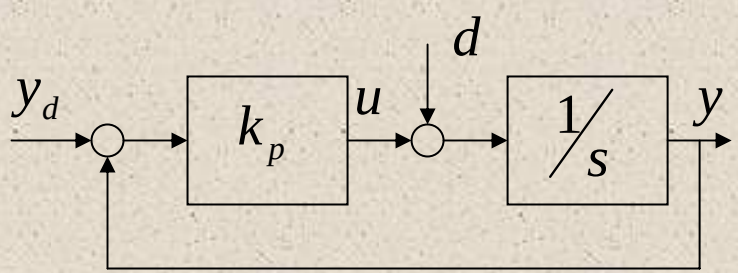
Is this system stable?

- First order system: One pole only
- Target output: y_d
- Proportional control: $u = - k_p (y - y_d)$

Example: 1st Order System

- **Stable if $k_p > 0$**
- **No overshoot, faster rise time, smaller settling time if k_p large**
- **Steady state error:**

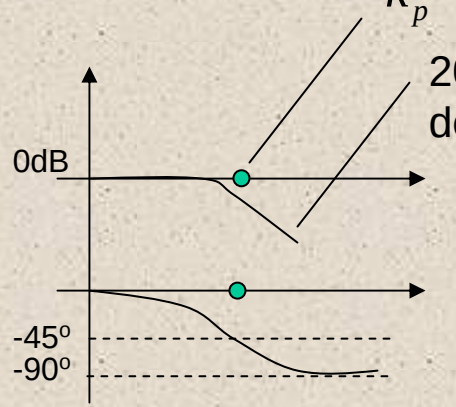
$$\dot{y} = u - d$$

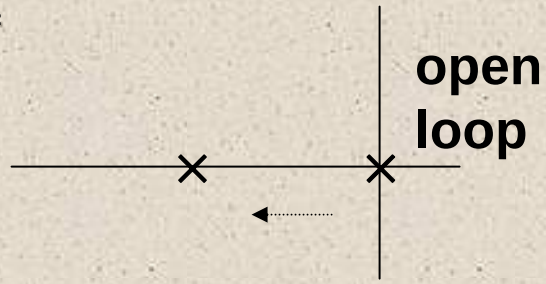


$$e = -\frac{s}{s+k_p} y_d - \frac{d}{s+k_p}$$

$$e_{ss} = -\frac{d}{k_p}$$

$$y = \frac{k_p}{s+k_p} y_d - \frac{d}{s+k_p}$$





Example: 1st Order System

Steady state error:

$$e = -\frac{s}{s+k_p} y_d - \frac{d}{s+k_p}$$

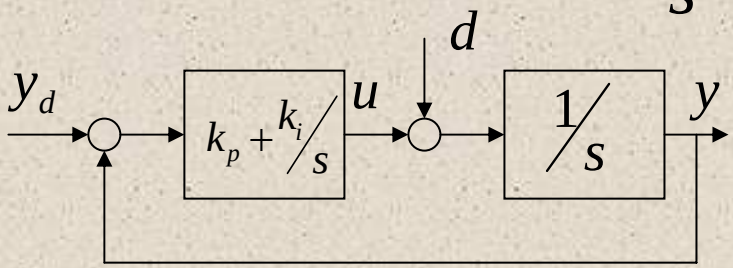
$$e_{ss} = -\frac{d}{k_p}$$

- Large k_p means small steady state error
- Alternatively, use integral control

$$u = -k_p(y - y_d) - k_i \int_0^t (y - y_d) d\tau$$

Create a closed-loop zero

$$y = \frac{sk_p + k_i}{s^2 + sk_p + k_i} y_d - \frac{s}{s^2 + sk_p + k_i} d$$



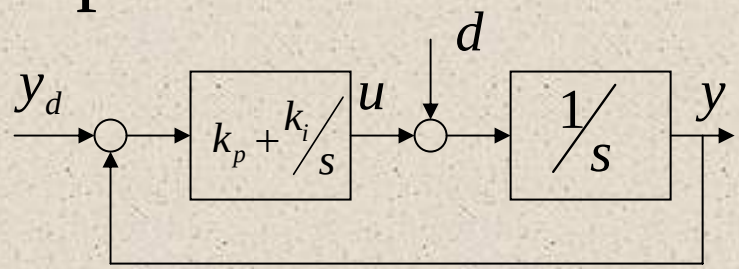
For small k_i, closed loop TF is close to before

Example: 1st Order System

Integral control

$$e = \frac{-s^2}{s^2 + sk_p + k_i} y_d - \frac{s}{s^2 + sk_p + k_i} d$$

- CL system is 2nd order $e_{ss} = 0$
- Choose k_p k_i to achieve desired transient response
- Steady state error due to step disturbance is zero!

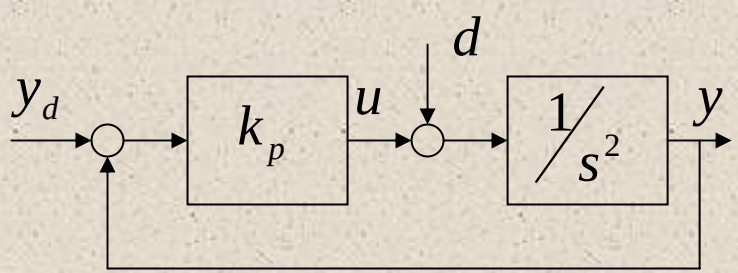


Example: 2nd Order System

Orientation control of Satellite:
Proportional feedback

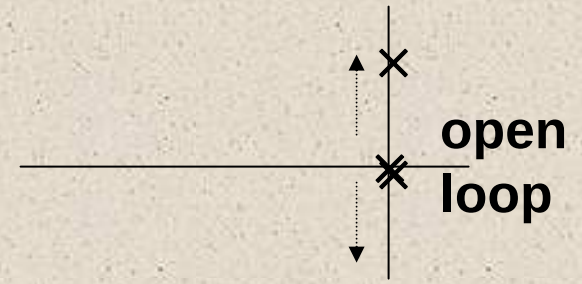
$$\ddot{y} = u + d$$

$$u = -k_p (y - y_d)$$



- **Closed loop poles at +/- $k_p j$**
- **Marginally stable if $k_p > 0$**

$$y = \frac{k_p}{s^2 + k_p} y_d + \frac{d}{s^2 + k_p}$$

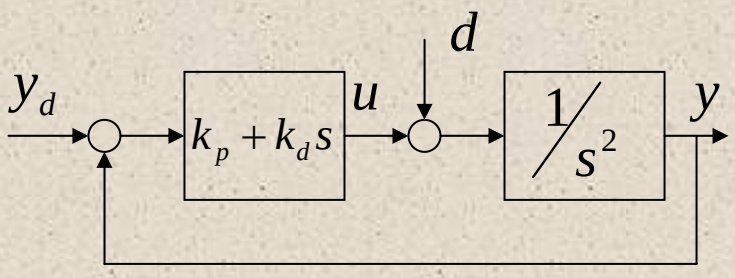


Example: 2nd Order System

Proportional-derivative feedback

$$u = -k_p(y - y_d) - k_d\dot{y}$$

$$\ddot{y} = u + d$$

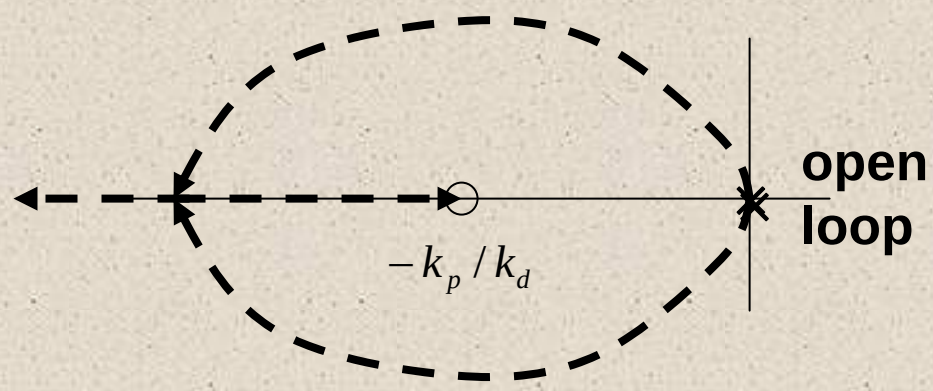


- **Stable if $k_p > 0$ and $k_d > 0$**
- **Choose gains to achieve desired transient response**
- **Steady state error:**

$$y = \frac{k_d s + k_p}{s^2 + k_d s + k_p} y_d + \frac{d}{s^2 + k_d s + k_p}$$

$$e = \frac{-s^2}{s^2 + k_d s + k_p} y_d + \frac{d}{s^2 + k_d s + k_p}$$

$$e_{ss} = \frac{d}{k_p}$$

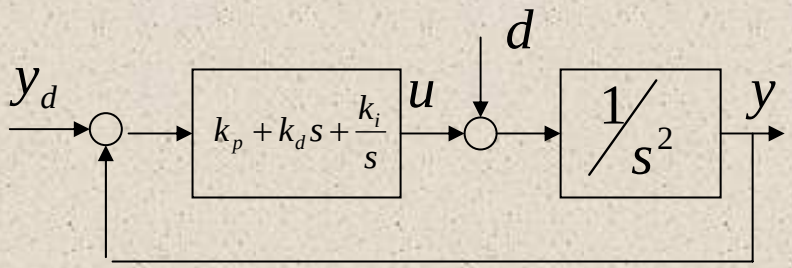


Example: 2nd Order System

Proportional-integral-derivative (PID) control

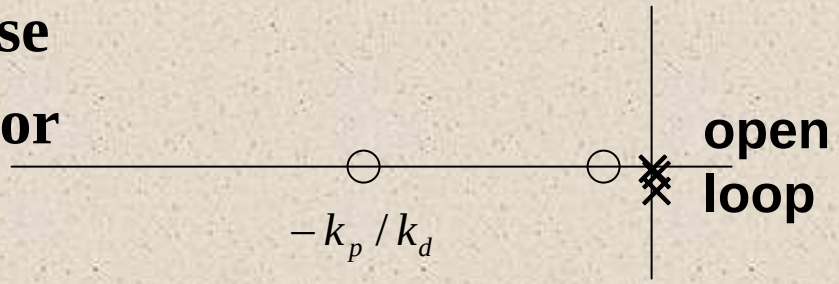
$$\ddot{y} = u + d$$

$$u = -k_p(y - y_d) - k_d\dot{y} - k_i\int_0^t (y - y_d)d\tau$$



- **Stable if $k_p > 0$ and $k_d > 0$ and $k_i > 0$ sufficiently small**
- **Choose gains to achieve desired transient response**
- **Zero steady state error for step disturbance**

$$y = \frac{k_d s^2 + k_p s}{s^3 + k_d s^2 + k_p s + k_i} y_d + \frac{sd}{s^3 + k_d s^2 + k_p s + k_i}$$



Other Issues

- Tracking control: Replace the step y_d by the desired time varying $y_d(t)$ – keep closed loop transfer function around 1 (0dB, 0 phase) in the spectrum of $y_d(t)$.
- What if the system is not second order?
  - Use the dominant second order system (two poles closest to the $j\omega$-axis)
  - Check stability / performance / steady state error on the high order system

Tuning of PID Gains

- Intuition:
  - P gain increases speed of response but also increases overshoot
  - D gain reduces overshoot but decreases speed of response
  - I gain reduces steady state error but can reduce speed of response and lead to instability
- Strategy:
  - Tune PD gain until desired transient response is obtained (use 2^{nd} order formula).
  - Increase I gain until convergence to steady state is satisfactory.
  - Retune PD gains (increase) if necessary.

Ziegler-Nichols Tuning Guide

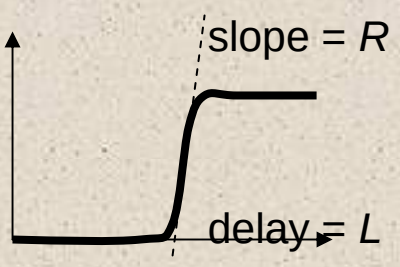
- Approximate the system as a first order system with delay (motivated by process control)

$$D(s) = K(1 + 1/T_I s + T_D s)$$

$$K = 1.2/RL$$

$$T_I = 2L$$

$$T_D = 0.5L$$



- Use proportional feedback to drive the system to the boundary of instability (ultimate gain: K_u ultimate period: P_u)

$$K = 0.6K_u$$

$$T_I = 0.5P_u$$

$$T_D = \frac{1}{8} P_u$$

Computational Approach

- Parameterize PID controller as $G_c(s) = K\frac{(s+a)^2}{s}$ with K and a chosen within a range (based on bandwidth, sampling, and saturation consideration).
- Step through K and a to minimize some combination of performance measures (e.g., overshoot, settling time, rise time, etc.).
- MATLAB is a great tool for this approach.

Digital Control

$$u = -k_p(y - y_d) - k_d\dot{y} - k_i\int_0^t (y - y_d)d\tau$$

- Sample data implementation: we need to approximate the derivative and integral terms
- Derivative term: backward difference
- Integral term: cumulative sum

$$w = \dot{y}$$

$$w_k \cong \frac{y_k - y_{k-1}}{t_s}$$

$$w \cong \left[(1 - z^{-1}) \Big/ t_s \right] y$$

$$w = \int y$$

$$w_k \cong w_{k-1} + y_k t_s$$

$$w \cong \left[t_s \Big/ (1 - z^{-1}) \right] y$$

- Stability: closed loop poles within unit circle

Digital Control

- Stability: closed loop poles within unit circle
- Rule of thumb: 3 to 10 times faster than closed loop bandwidth ω_n
  - With a specified sampling rate, poles cannot be too fast (i.e., gains cannot be too high)
- First order system example: $y_{k+1} = y_k - t_s k_p (y_k - y_d)$

Closed loop characteristic equation $z - (1 - t_s k_p) = 0$

Closed loop poles : $(1 - t_s k_p)$

Condition for stability : $k_p < 1/t_s$

Other Velocity Estimators

- washout filter $G(s) = \dfrac{s}{\left(\dfrac{s}{p} + 1\right)}$
- finite difference + low pass filter
- Kalman predictor (state observer based on an assumed plant, will cover later in course)