01211433 Vision and Control of Industrial Robots

Robot Independent Joint Control

(State Feedback and PID with Feedforward)

Dr.Varodom Toochinda

Outline

- State-space representation
- Conversion between state-space and transfer function
- State feedback control
- State feedback + integrator control
- PID with Feedforward control

State-space representation

DC motor model $J\ddot{\theta}(t)+B\dot{\theta}(t)= u(t)$ (1)

Define system states $x_1 = \theta$

$x_2 = \dot{\theta}$

(1) ➡

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{B}{J}x_2 + \frac{1}{J}u(t)$$

State equation in matrix form

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -\frac{B}{J}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{J}\end{bmatrix}u$$

Output equation with $y = \theta$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

General LTI State-space system

State equation $\dot{x} = Ax + Bu$

Output equation $y = Cx + Du$

Conversion between state-space and transfer function

$$P(s) = C(sI - A)^{-1}B + D$$

Can use Scilab commands ss2tf() and tf2ss()

Ex: $P(s) = 1/(s^2 + s)$

```
s=poly(0,'s');
P = syslin('c',1/(s^2+s));
Pss = tf2ss(P);
Ptf = ss2tf(Pss);
```

State-feedback Control

- Controller $u = -Kx$
- For a specified closed-loop poles, design goal is to compute

$$K = [k_1 \quad k_2 \quad \dots \quad k_n]^T$$

- Closed-loop state equation $\dot{x} = (A - BK)x$
- Closed-loop poles $\det(sI - A + BK) = 0$
- Characteristic polynomial $\alpha(s) = (s - p_1)(s - p_2) \dots (s - p_n)$

Ex 1: basic SFB design

- Let J=1, B=1 in $\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -\frac{B}{J}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{J}\end{bmatrix} uy = [1 \quad 0]\begin{bmatrix}x_1\\ x_2\end{bmatrix}$
- Design specs
  - 1. Overshoot < 5%
  - 2. Rise time < 0.1 sec
- For 2nd order system
  - 1. damping ratio $\zeta \geq 0.7$
  - 2. with $t_r = 1.8/\omega_n$ ➡ $\omega_n \geq 18 rad/s$
- Characteristic polynomial

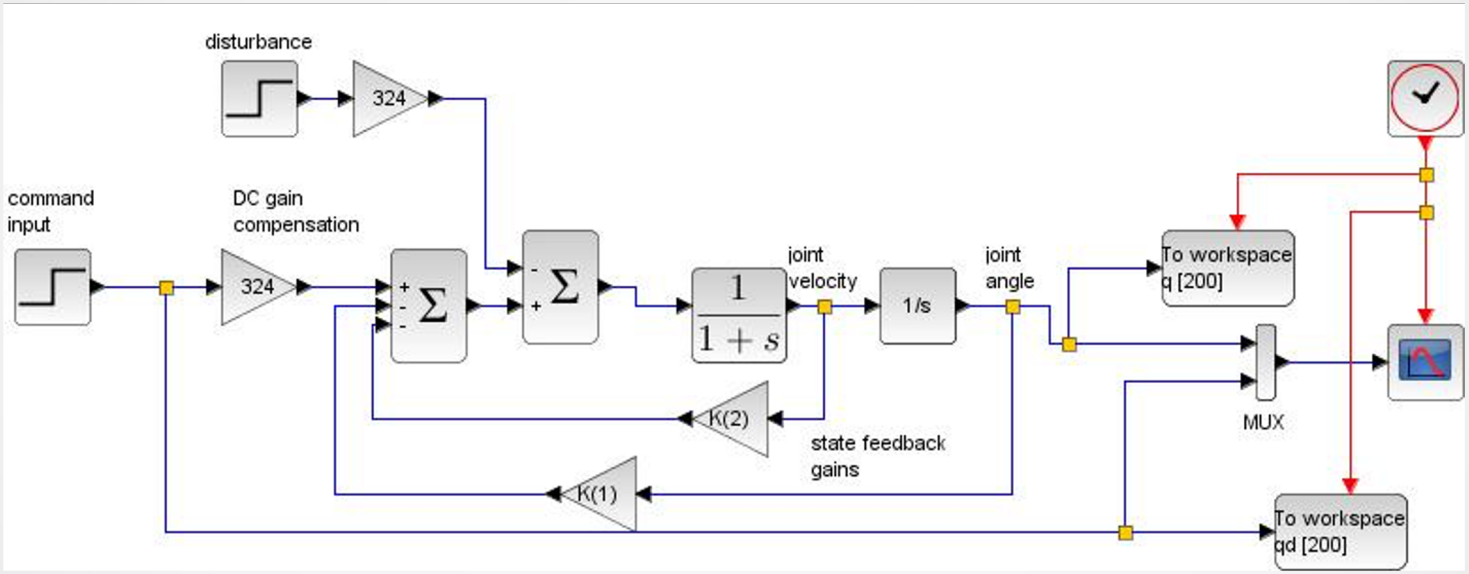
$$\Lambda(s) = s^2 + 2\zeta\omega_n s + \omega_n^2 = s^2 + 25.2s + 324$$

Poles at $-12.6 \pm 12.8546i$

Scilab commands

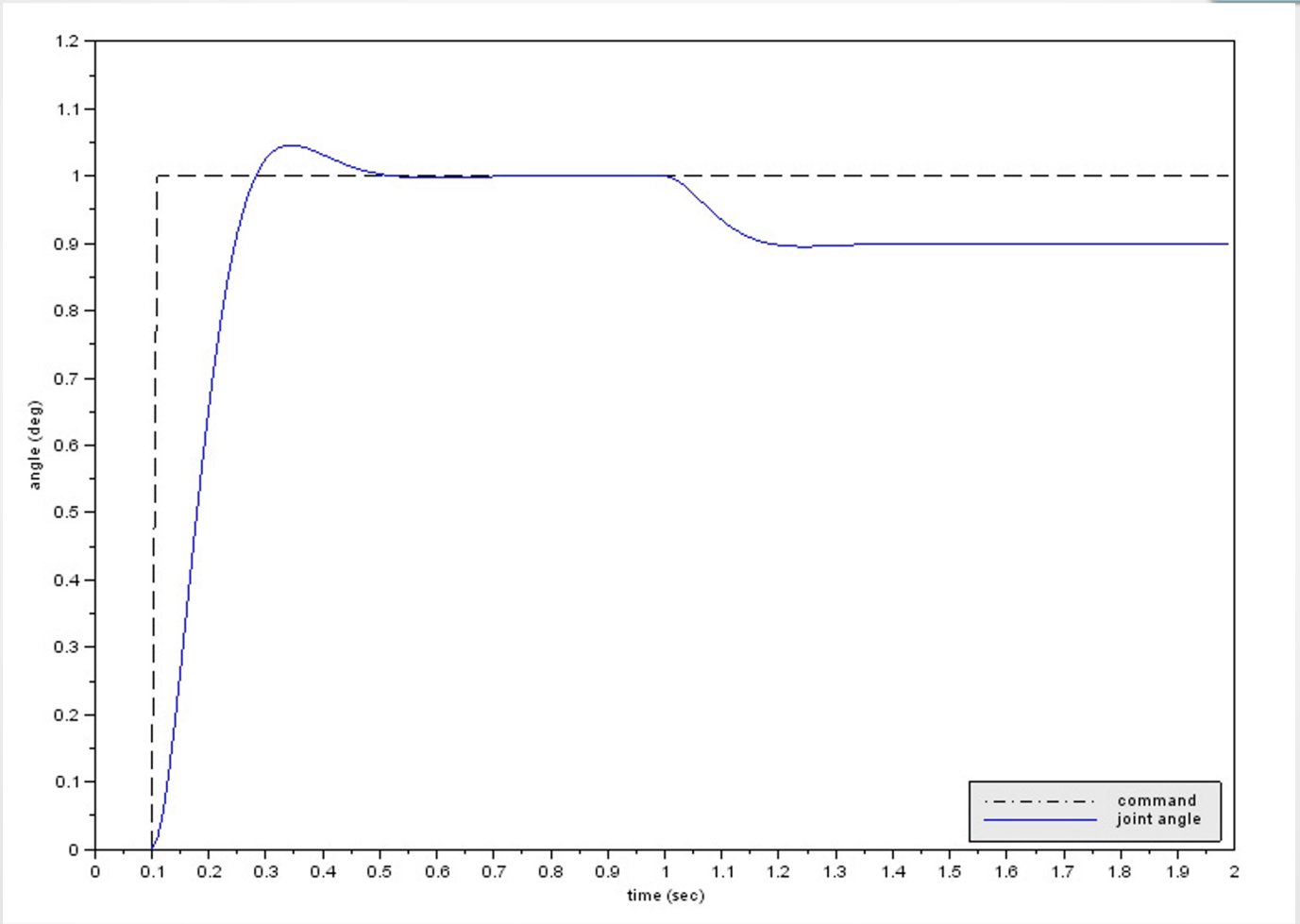
```
-->z=0.7; wn=18;
-->lamda = s^2+2*z*wn*s+wn^2;
-->clpoles = roots(lamda)
 clpoles  =
  - 12.6 + 12.854571i
  - 12.6 - 12.854571i
-->K=ppol(Pss.A,Pss.B,clpoles)
 K  =
    324.    24.2
-->cltf=ss2tf(syslin('c',Pss.A-Pss.B*K,Pss.B,Pss.C))
 cltf  =
            1
   --------------
                  2
   324 + 25.2s + s
```

Xcos simulation



ppol_1joint.zcos

Simulation result



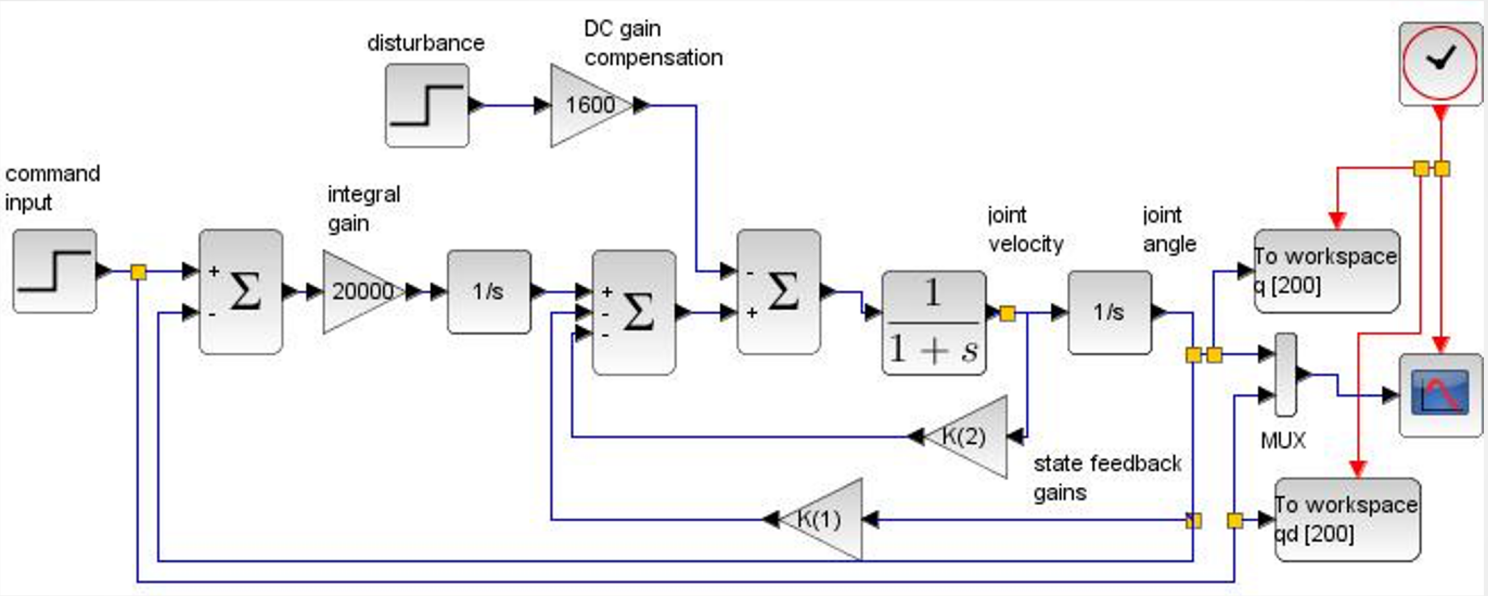
Ex 2: SFB + integrator

- Add an integrator in place of DC gain compensation
- Design SFB gain first, then adjust integral gain
- Increase natural frequency to 40 rad/s

Scilab commands

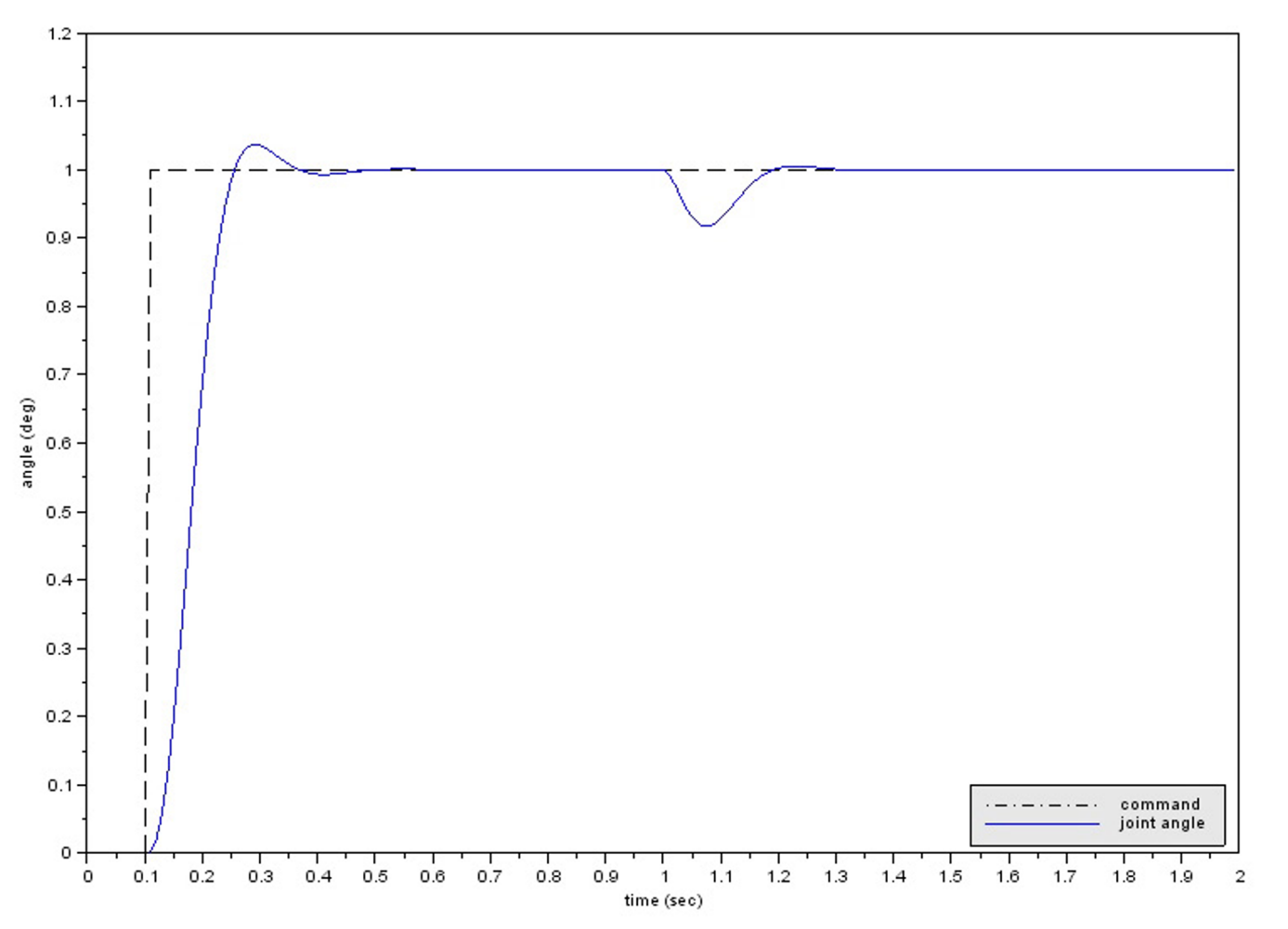
```
-->z=0.7; wn=40;
-->lamda = s^2+2*z*wn*s+wn^2;
-->clpoles = roots(lamda)
 clpoles  =
 - 28. + 28.565714i
 - 28. - 28.565714i
-->K=ppol(Pss.A,Pss.B,clpoles)
 K  =
    1600.    55.
-->cltf=ss2tf(syslin('c',Pss.A-Pss.B*K,Pss.B,Pss.C))
 cltf  =
                1
    -------------
                  2
    1600 + 56s + s
```

Xcos simulation



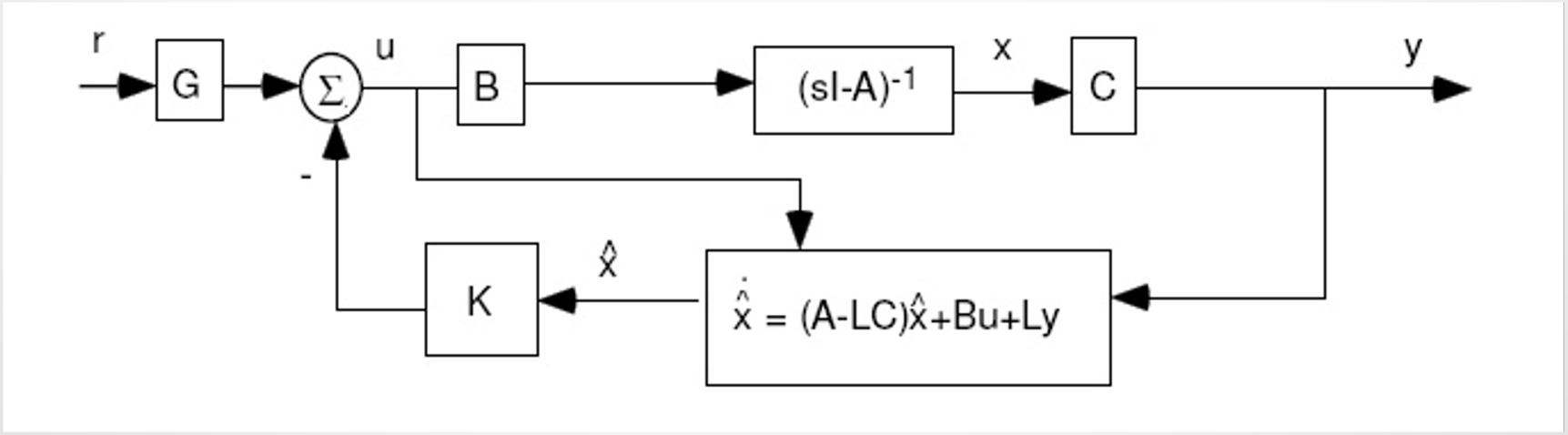
ppol_int_1joint.zcos

Simulation result



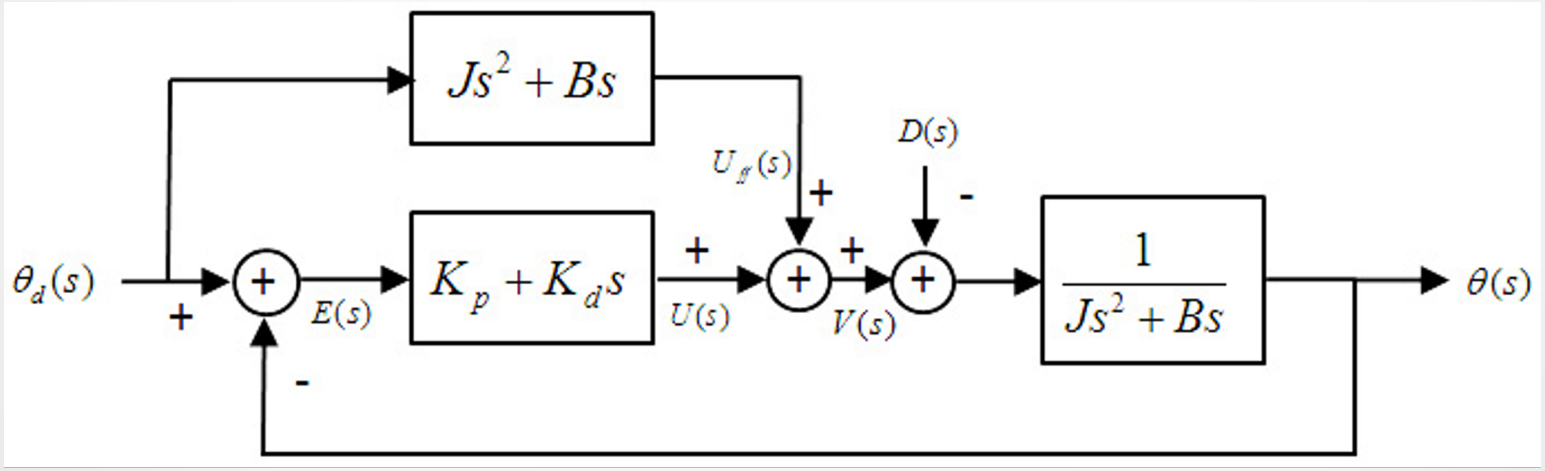
Output Feedback Control

- For a general plant, all states may not be measurable
- Use a state-estimator (often called “observer”) together with state feedback control
- Various design schemes, including “synthesis” methods

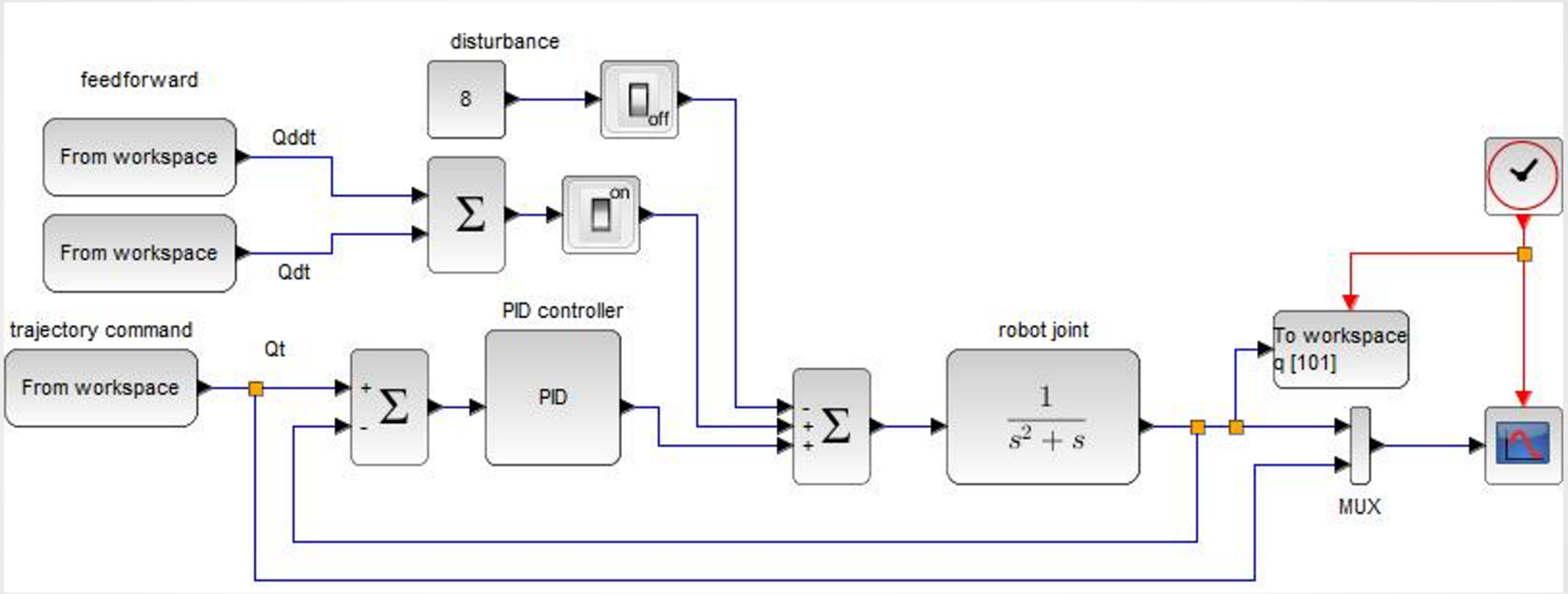


PD with Feedforward Control

Not implementable!



Xcos simulation

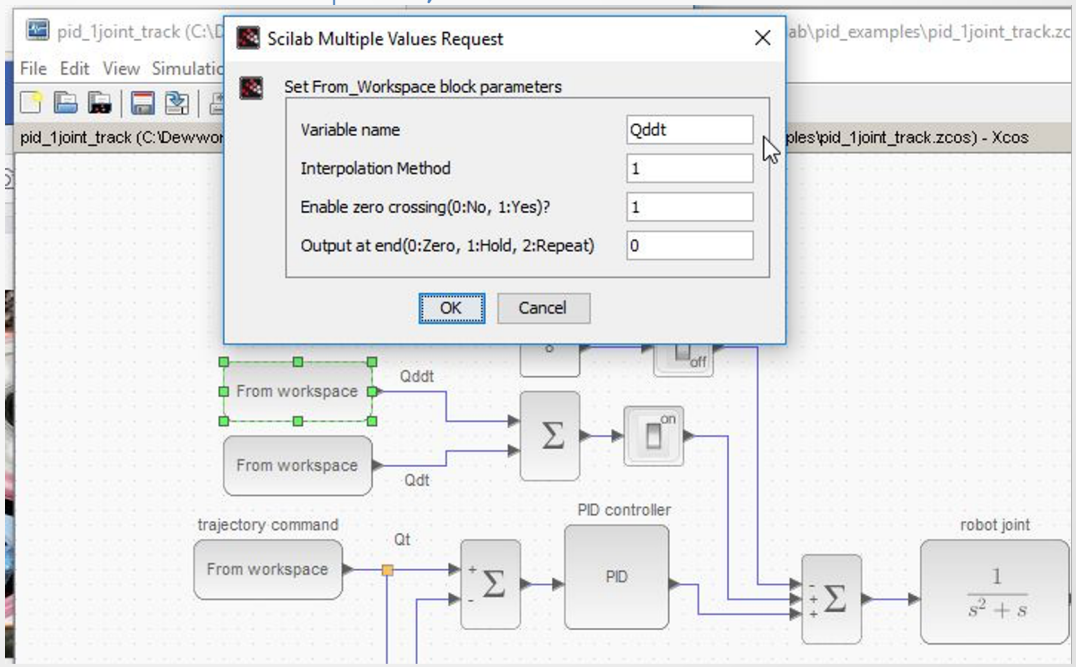


pid_1joint_track.zcos

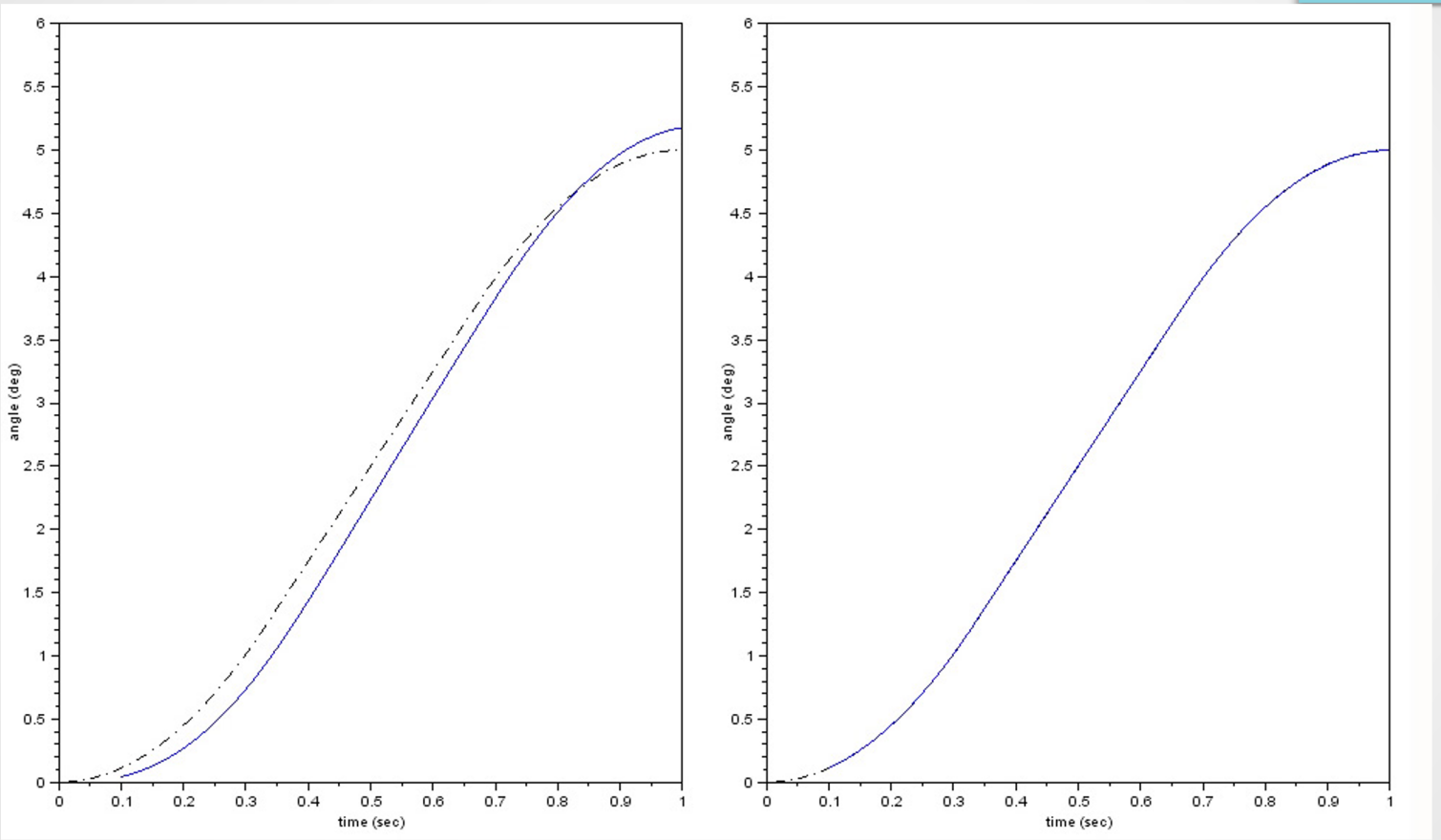
Must run RTSX to generate trajectory

Trajectory generation

```
t=0:0.01:1;
[qt,qdt,qddt]=lspb(0,5,t);
Qt.time = t'; Qt.values = qt;
Qdt.time = t'; Qdt.values = qdt;
Qddt.time = t'; Qddt.values = qddt;
```



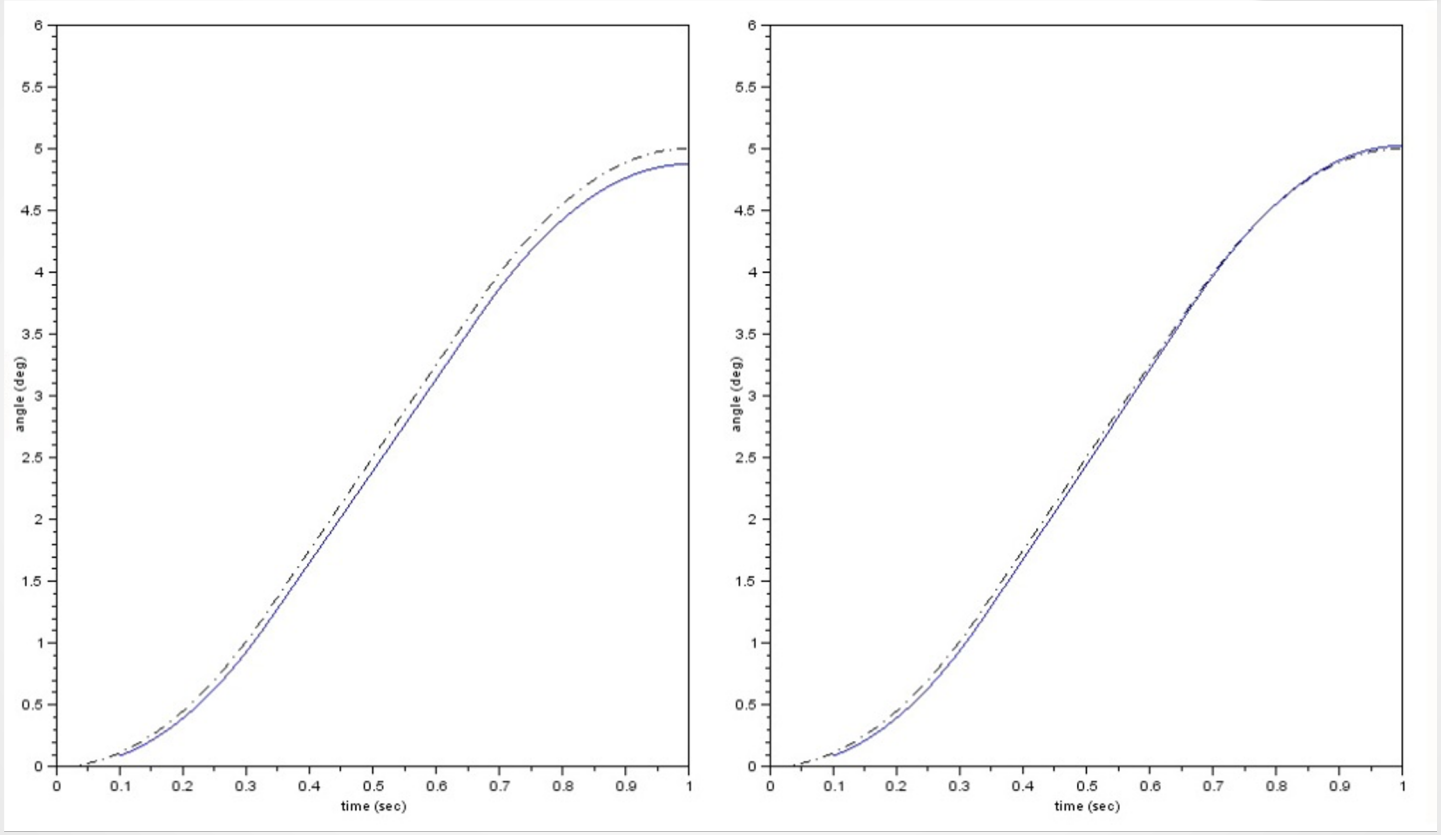
Simulation result



Without feedforward

With feedforward

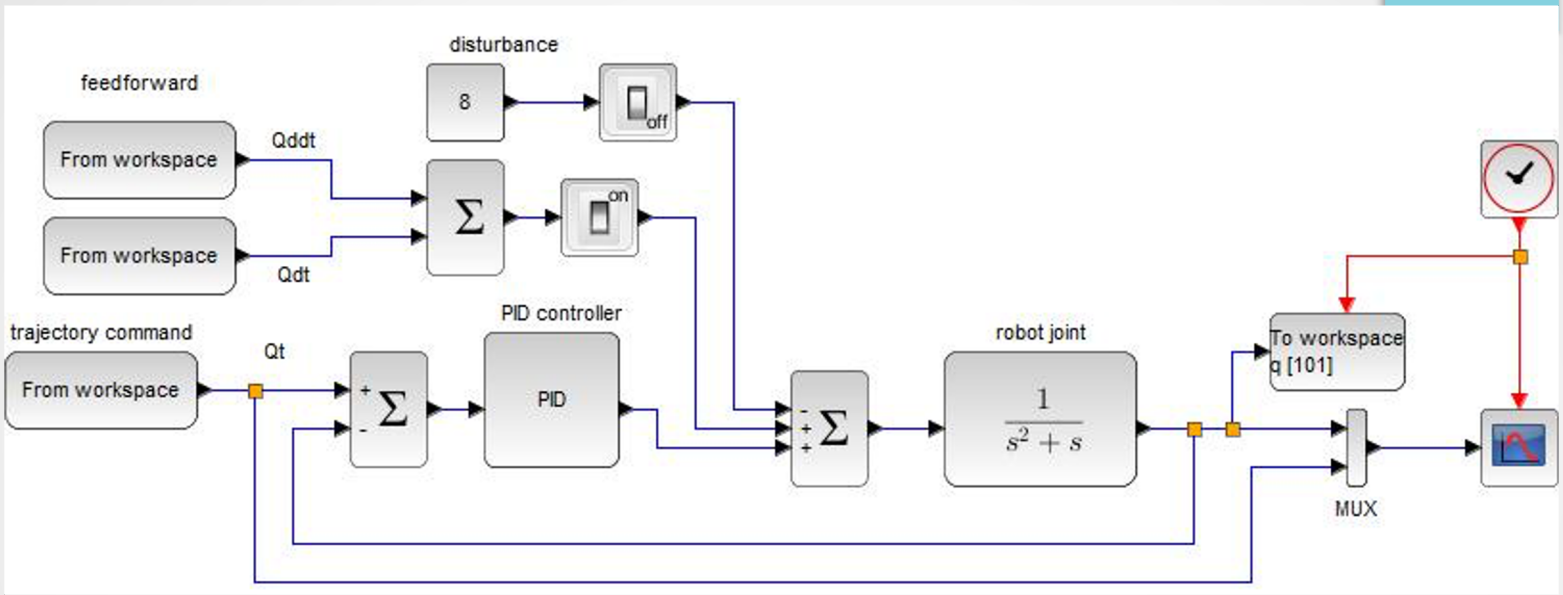
Use integral term to reject disturbance



PD control

PID control with Ki = 300

Exercise : PD with feedforward



Original model : pid_1joint_track.zcos

Instructions

1. Change robot joint model to 1/(10*s^2+0.1*s) ; i.e., J= 10, B = 0.1
2. Optional : generate multi-segment trajectory using RTSX (เหมือนในรูปที่ 8.25)