Robot Control Exp. 3

(Computed Torque Method)

Autumn semester

School of Robotics

BICAR(Biologically-inspired Control and Robot) Lab.

Prof. Woosung Yang

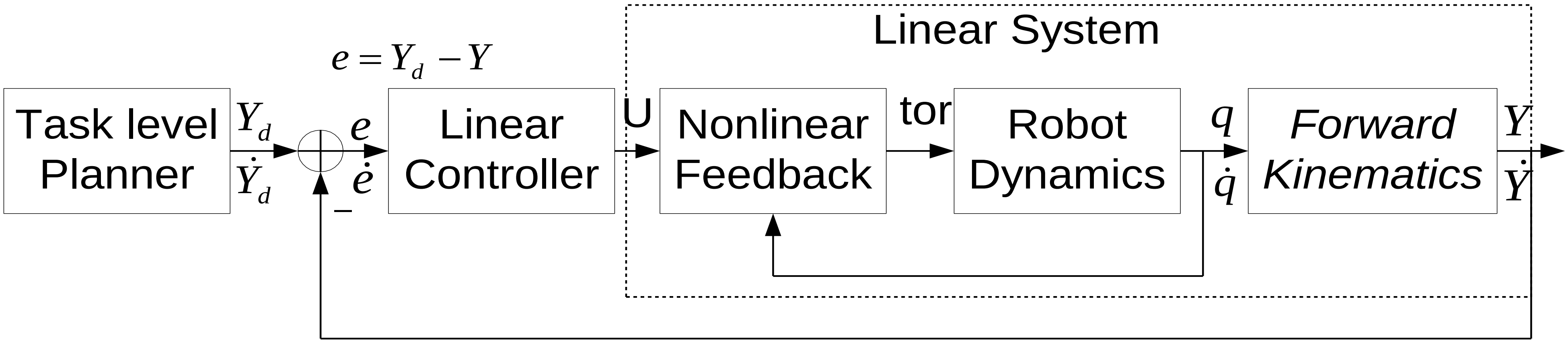
光云大學校
KwangWoon University

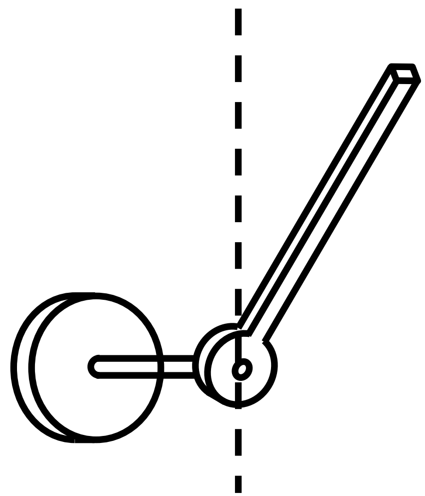
❑ Robot Control EXP. 3

1. Computed Torque Method
2. Joint Space PD CTM Controller
3. Cartesian Space PD CTM Controller
4. Homework
    1. (1-DOF) Joint Space PID CTM Controller
    2. (2-DOF) Cartesian Space PID CTM Controller

❑ Robot Control EXP. 3

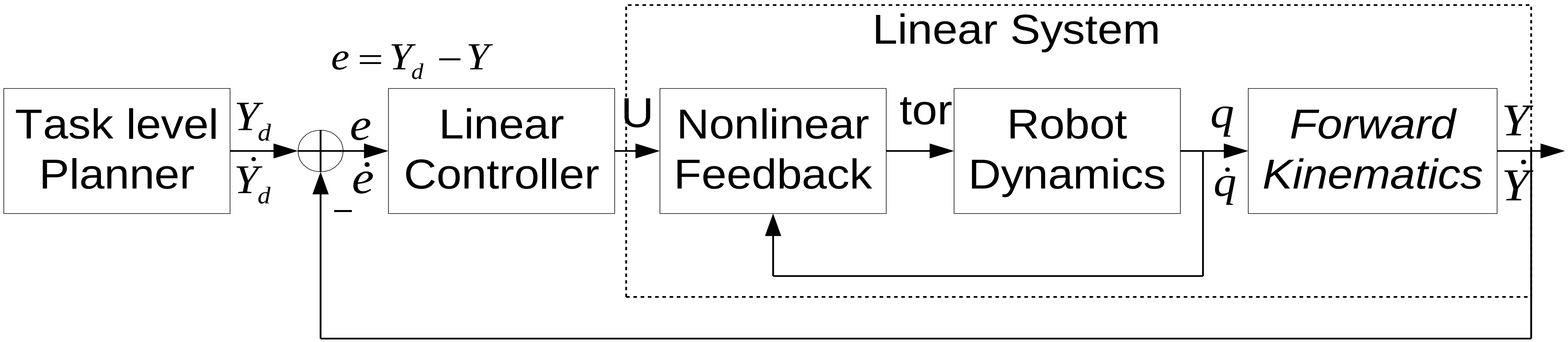
1. Computed Torque Method

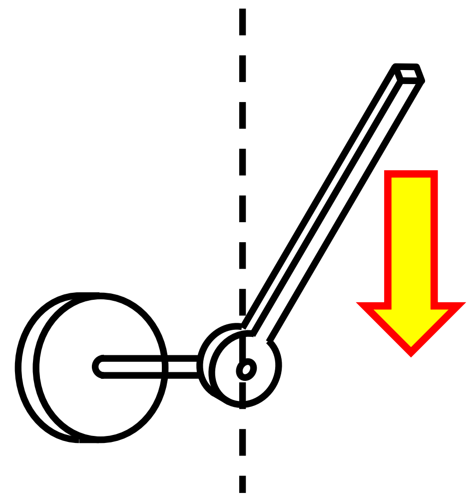




❑ Robot Control EXP. 3

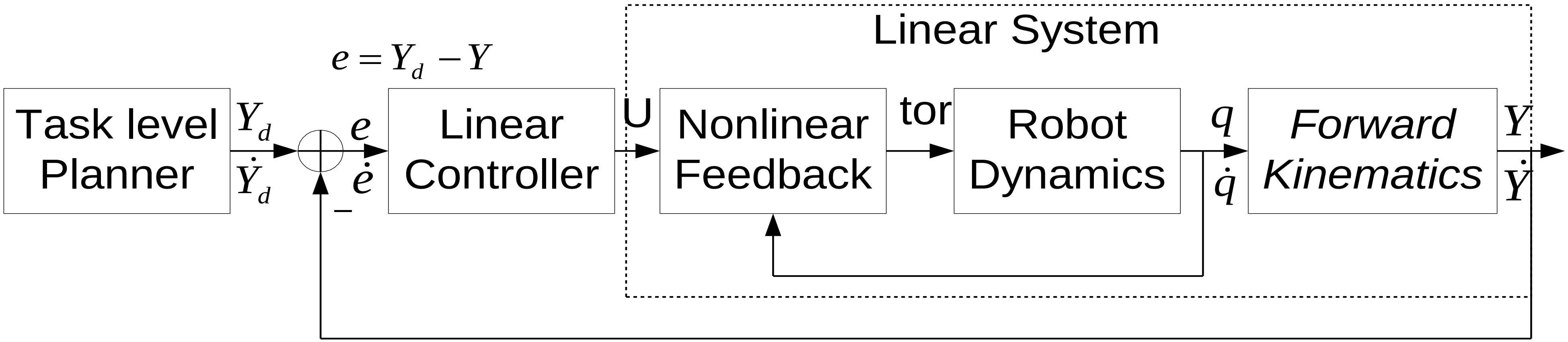
1. Computed Torque Method





❑ Robot Control EXP. 3

1. Computed Torque Method

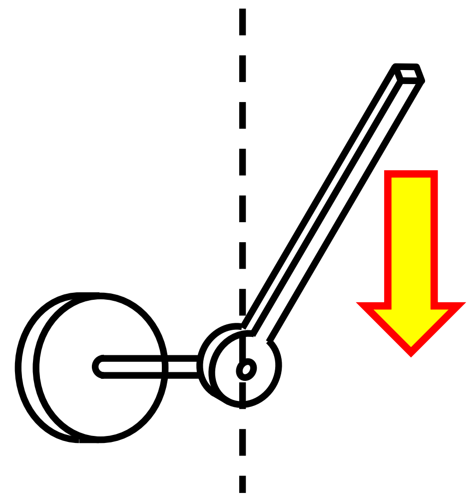


$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q})$$

- **Joint Space**

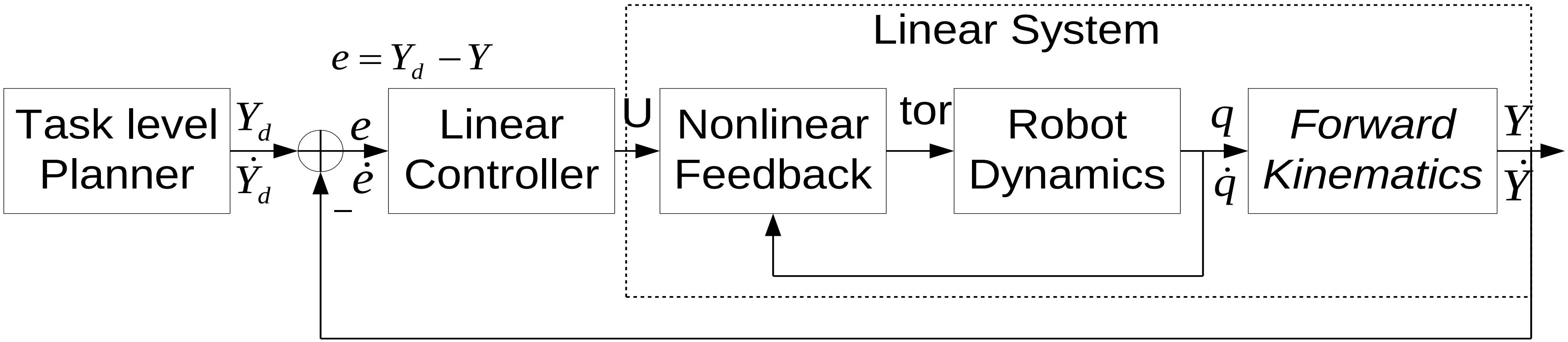
$$\boldsymbol{\tau} = \boldsymbol{J}\ddot{\boldsymbol{q}}$$

$$\boldsymbol{\tau} = \boldsymbol{J}\boldsymbol{u}$$



❑ Robot Control EXP. 3

1. Computed Torque Method



$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q})$$

- **Joint Space**
- **Cartesian Space**

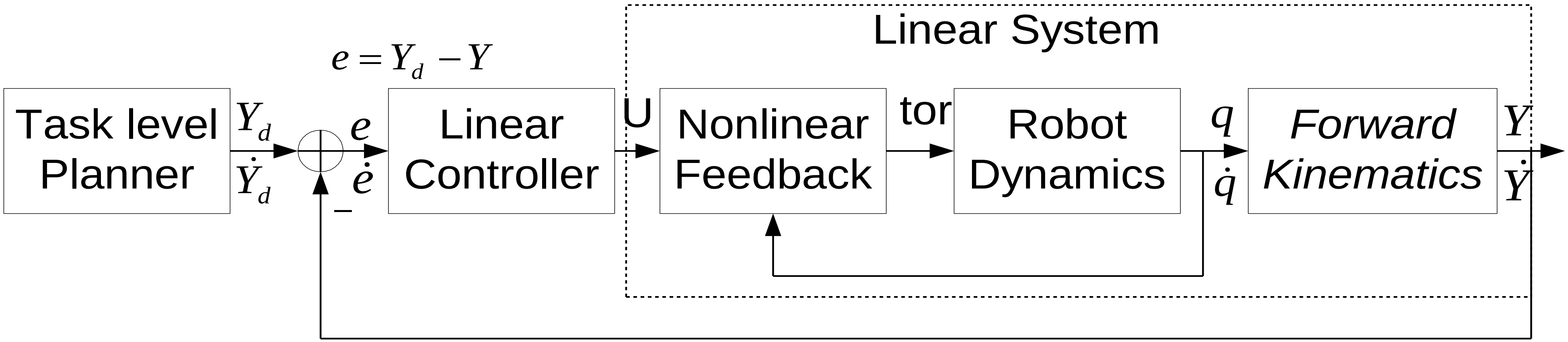
$$\boldsymbol{\tau} = \boldsymbol{J}\ddot{\boldsymbol{q}} \qquad \boldsymbol{f} = \boldsymbol{m}\ddot{\boldsymbol{x}}$$

$$\Downarrow \qquad \Downarrow$$

$$\boldsymbol{\tau} = \boldsymbol{J}\boldsymbol{u} \qquad \boldsymbol{f} = \boldsymbol{m}\boldsymbol{u}$$

❑ Robot Control EXP. 3

1. Computed Torque Method



$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\ddot{\mathbf{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q})$$

- **Joint Space**

$$\boldsymbol{\tau} = \boldsymbol{J}\ddot{\boldsymbol{q}}$$

⇩

$$\boldsymbol{\tau} = \boldsymbol{J}\boldsymbol{u}$$

- **Cartesian Space**

$$\boldsymbol{f} = \boldsymbol{m}\ddot{\boldsymbol{x}}$$

⇩

$$\boldsymbol{f} = \boldsymbol{m}\boldsymbol{u}$$

$$\mathbf{0} = \ddot{\boldsymbol{x}} + 2\zeta_n\omega_n\dot{\boldsymbol{x}} + \omega_n^2$$

⇩

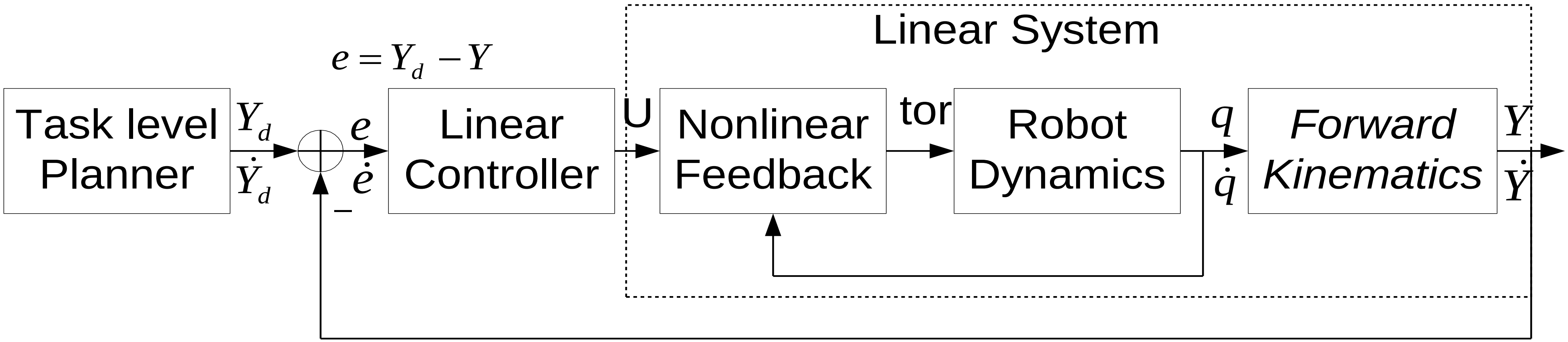
$$\mathbf{0} = (\ddot{\boldsymbol{x}}_0 - \ddot{\boldsymbol{x}}) + 2\zeta_n\omega_n(\dot{\boldsymbol{x}}_0 - \dot{\boldsymbol{x}}) + \omega_n^2(\boldsymbol{x}_0 - \boldsymbol{x})$$

⇩

$$\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}}_0 + 2\zeta_n\omega_n(\dot{\boldsymbol{x}}_0 - \dot{\boldsymbol{x}}) + \omega_n^2(\boldsymbol{x}_0 - \boldsymbol{x})$$

❑ Robot Control EXP. 3

1. Computed Torque Method



$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q})$$

$$\mathbf{0} = \ddot{x} + 2\zeta_n\omega_n\dot{x} + \omega_n^2$$

$$\Downarrow$$

$$\mathbf{0} = (\ddot{x}_0 - \ddot{x}) + 2\zeta_n\omega_n(\dot{x}_0 - \dot{x}) + \omega_n^2(x_0 - x)$$

$$\Downarrow$$

$$\ddot{x} = \ddot{x}_0 + 2\zeta_n\omega_n(\dot{x}_0 - \dot{x}) + \omega_n^2(x_0 - x)$$

- **Joint Space**

$$\boldsymbol{\tau} = \boldsymbol{J}\ddot{\boldsymbol{q}}$$

$$\Downarrow$$

$$\boldsymbol{\tau} = \boldsymbol{J}\boldsymbol{u}$$

- **Cartesian Space**

$$\boldsymbol{f} = \boldsymbol{m}\ddot{\boldsymbol{x}}$$

$$\Downarrow$$

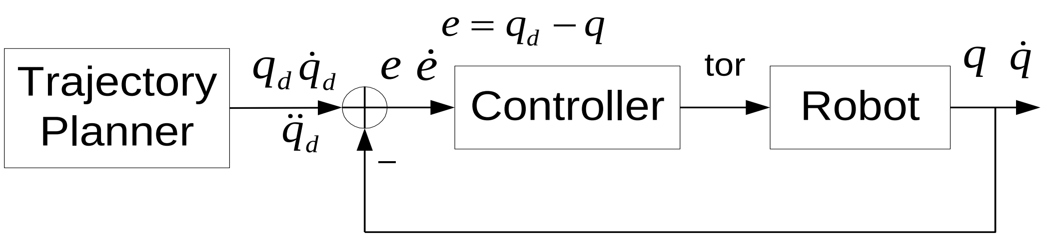
$$\boldsymbol{f} = \boldsymbol{m}\boldsymbol{u} = \boldsymbol{m}\left(\ddot{\boldsymbol{x}}_0 + \boldsymbol{K}_v(\dot{\boldsymbol{x}}_0 - \dot{\boldsymbol{x}}) + \boldsymbol{K}_p(\boldsymbol{x}_0 - \boldsymbol{x})\right)$$

❑ Robot Control EXP. 3

1. Computed Torque Method
2. Joint Space PD CTM Controller
3. Cartesian Space PD CTM Controller
4. Homework
    1. (1-DOF) Joint Space PID CTM Controller
    2. (2-DOF) Cartesian Space PID CTM Controller

❑ Robot Control EXP. 3

2. Joint Space PD CTM Controller



- **Control Input**

$$u = \ddot{q} = \ddot{q}_0 + K_v(\dot{q}_0 - \dot{q}) + K_p(q_0 - q)$$

- **Torque Ouput**

$$\tau = D(q)u + H(q, \dot{q}) + C(q)$$

```
if(flag_Simul == 1)
    % Simulation
        n = 1;
        for (time = start_t:delta_t:finish_t)
            % Set Target Trajectory
                if(time < 1)
                    q_d     = init_q;
                    dq_d    = 0.0;
                    ddq_d   = 0.0;
                else
                    if(q_d < 90*pi/180)
                        q_d = q_d + (45*pi/180/2)*delta_t;
                    else
                        q_d = 90*pi/180;
                    end
                    dq_d    = (q_d  - simul_q_d(n-1))/delta_t;
                    ddq_d   = (dq_d - simul_dq_d(n-1))/delta_t;
                end
            % Get Dynamics
                G        = get_Gravity(q);
            % Controller
                u        = ddq_d + Kv*(dq_d - dq) + Kp*(q_d - q);
                tq_ctrl  = I*u + G * 0.8;
            % Robot Model
                % Inverse Dynamics
                    tq      = tq_ctrl;
                    [t,y]   = ode45('one_link',[0 delta_t],[q; dq]);
                    index   = length(y);
                    q       = y(index,1);
                    dq      = y(index,2);
            % Save Data
                simul_time(n)   = time;     % [sec]
                simul_q(n)      = q;        % [rad]
                simul_dq(n)     = dq;       % [rad/s]
                simul_q_d(n)    = q_d;      % [rad]
                simul_dq_d(n)   = dq_d;     % [rad/s]
                n               = n + 1;
        end
end
```

❑ Robot Control EXP. 3

2. Joint Space PD CTM Controller

How to chose Kp, Kv to make the system stable?

Error dynamics $\ddot{e} + k_v \dot{e} + k_p e = 0$

Characteristic equation: $|\lambda I - A| = \begin{vmatrix} \lambda & -1 \\ k_p & \lambda + k_v \end{vmatrix} = \lambda^2 + k_v \lambda + k_p = 0$

The error system is asymptotically stable as long as the k_v and k_p are all positive

$$p(s) = s^2 + 2\xi\omega_n s + \omega_n^2 \Rightarrow \boxed{k_p = \omega_n^2,\ k_v = 2\xi\omega_n \quad additionally,\ k_i < k_v k_p}$$

It is undesirable for the robot to exhibit overshoot, since this could cause impact. Thus, the PD gains are usually selected for ***critical damping***

The natural frequency ω_n governs the speed of response in each error component

Suppose, the 1st natural frequency ω_r and robots have some flexibility

$\omega_r = \sqrt{k_r / J} \quad \therefore \omega_n < \omega_r / 2 \quad$ J the link inertia, k_r the link stiffness ➡ ***constant?***

❑ Robot Control EXP. 3

2. Joint Space PD CTM Controller

$\omega_r = \sqrt{k_r / J} \quad \therefore \omega_n < \omega_r / 2$ J the link inertia, k_r the link stiffness ➡ ***constant?***

Performance

High Gains: better disturbance rejection

Gains are limited by

structural flexibilities

time delays (actuator-sensing)

sampling rate

$$\omega_n \le \frac{\omega_r}{2}$$

$$\omega_n \le \frac{\omega_{delay}}{3} \quad \leftarrow \text{largest delay } \left(\frac{2\pi}{T_{delay}}\right)$$

$$\boxed{\omega_n \le \frac{\omega_{sampling-rate}}{5}}$$

❑ Robot Control EXP. 3

1. Computed Torque Method
2. Joint Space PD CTM Controller
3. Cartesian Space PD CTM Controller
4. Homework
    1. (1-DOF) Joint Space PID CTM Controller
    2. (2-DOF) Cartesian Space PID CTM Controller

❑ Robot Control EXP. 3

3. Cartesian Space PD CTM Controller

- **Control Input**

$$u = \ddot{Y} = \ddot{Y}_0 + K_v(\dot{Y}_0 - \dot{Y}) + K_p(Y_0 - Y)$$

✓ **Jacobian Matrix**

$$\dot{Y} = J\dot{q}$$

$$\ddot{Y} = \dot{J}\dot{q} + J\ddot{q}$$

$$J\ddot{q} = \ddot{Y} - \dot{J}\dot{q}$$

$$\ddot{q} = J^{-1}(\ddot{Y} - \dot{J}\dot{q})$$

- **Torque Ouput**

$$\tau = D(q)\ddot{q} + H(q,\dot{q}) + C(q)$$

```
if(flag_Simul == 1)
    % Simulation
    n = 1;
    sin_t = 0;
    for (time = start_t:delta_t:finish_t)
        % Set Target Position
            if(time < 1.0)
                X_d      = init_X;
                dX_d     = [0;0];
                ddX_d    = [0;0];
            elseif(time < 2.0)
                X_d(1)   = init_X(1);
                if(X_d(2) < init_X(2) + 0.2)
                   X_d(2) = X_d(2) + (0.2/0.5)*delta_t;
                else
                   X_d(2) = init_X(2) + 0.2;
                end
                dX_d     = (X_d  - [simul_X_d_x(n-1) ;simul_X_d_y(n-1)])./delta_t;
            else
                X_d      = [ 0.2 * sin((2*pi*sin_t)/2) + init_X(1);
                             0.2 * cos((2*pi*sin_t)/2) + init_X(2);];
                sin_t = sin_t + delta_t;
                dX_d     = (X_d  - [simul_X_d_x(n-1) ;simul_X_d_y(n-1)])./delta_t;
                ddX_d    = (dX_d - [simul_dX_d_x(n-1);simul_dX_d_y(n-1)])./delta_t;
            end
        % Get Dynamics
            J   = get_Jacobian(  q(1),q(2));
            dJ  = get_dJacobian( q(1),q(2));
            X   = get_Kinematics(q(1),q(2));
            dX  = J*dq;
            D   = get_Inertia(        q(2));
            H   = get_Coriollis(      q(2),dq(1),dq(2));
            C   = get_Gravity(   q(1),q(2));
        % Control
            u       = ddX_d + Kv*(dX_d - dX) + Kp*(X_d - X);
            ddq_ref = inv(J)*(u - dJ*dq);

            tq_ctrl = D*ddq_ref + H + C*0.8;
        % Robot Model
            % Inverse Dynamics
                tq  = tq_ctrl;
                tq1 = tq(1);    tq2 = tq(2);
                [t,y] = ode45('two_link',[0 delta_t],[q(1); dq(1); q(2); dq(2)]);
                index = length(y);
                q   = [ y(index,1); y(index,3)];
                dq  = [ y(index,2); y(index,4)];
        % Save Data
            simul_time(n)  = time;          % [sec]
            simul_q1(n)    = q(1);          % [rad]
            simul_q2(n)    = q(2);          % [rad]
            simul_dq1(n)   = dq(1);         % [rad/s]
            simul_dq2(n)   = dq(2);         % [rad/s]
            simul_X_x(n)   = X(1);          % [m]
            simul_X_y(n)   = X(2);          % [m]
            simul_dX_x(n)  = dX(1);         % [m/s]
            simul_dX_y(n)  = dX(2);         % [m/s]
            simul_X_d_x(n) = X_d(1);        % [m]
            simul_X_d_y(n) = X_d(2);        % [m]
            simul_dX_d_x(n)= dX_d(1);       % [m/s]
            simul_dX_d_y(n)= dX_d(2);       % [m/s]
            n              = n + 1;
    end
end
```

❑ Robot Control EXP. 3

1. Computed Torque Method
2. Joint Space PD CTM Controller
3. Cartesian Space PD CTM Controller
4. Homework
    1. (1-DOF) Joint Space PID CTM Controller
    2. (2-DOF) Cartesian Space PID CTM Controller

❑ Robot Control EXP. 3

1. Computed Torque Method
2. Joint Space PD CTM Controller
3. Cartesian Space PD CTM Controller
4. Homework
   1. (1-DOF) Joint Space PID CTM Controller
   2. (2-DOF) Cartesian Space PID CTM Controller

$$u = \ddot{Y} = \ddot{Y}_0 + K_v(\dot{Y}_0 - \dot{Y}) + K_p(Y_0 - Y) + K_i \int (Y_0 - Y)\, dt$$

$$\Downarrow$$

$$\tau = D(q)\ddot{q} + H(q, \dot{q}) + C(q)$$

❑ Robot Control EXP. 3

4. Homework

- 이론 내용(수식) 부실 시 감점
- 조교 자료 복사 제출 시 감점
- 그래프 정보(제목, 라벨, 레전드 등) 부족 시 감점
- 주석 부실 시 감점

1. (1-DOF) Joint Space PID CTM Controller

- 목표 위치 : 0 deg → 90 deg
- 목표 속도 : 30 deg/s
- PID 게인 설정 및 게인 별 특성 보일 것
- 중력 보상 오차 적용 할 것

2. (2-DOF) Cartesian Space PID CTM Controller

- 목표 궤적 : 반경 0.1m / 주기 1초 원 그리기
- PID 게인 설정 및 게인 별 특성 보일 것
- 중력 보상 오차 적용 할 것

❑ Robot Control EXP. 3

4. Homework

- 이론 내용(수식) 부실 시 감점
- 조교 자료 복사 제출 시 감점
- 그래프 정보(제목, 라벨, 레전드 등) 부족 시 감점
- 주석 부실 시 감점

1. (1-DOF) Joint Space PID CTM Controller

- 목표 위치 : 0 deg → 90 deg
- 목표 속도 : 30 deg/s
- PID 게인 설정 및 게인 별 특성 보일 것
- 중력 보상 오차 적용 할 것

2. (2-DOF) Cartesian Space PID CTM Controller

- 목표 궤적 : 반경 0.1m / 주기 1초 원 그리기
- PID 게인 설정 및 게인 별 특성 보일 것
- 중력 보상 오차 적용 할 것

➔ **텀프 : 3-DOF**