Tutorial on Robotic Manipulator Control

(Part I: Position Control in Joint & Task Spaces)

Yan-Bin Jia

Department of Computer Science

Iowa State University

Ames, IA 50011

Outline

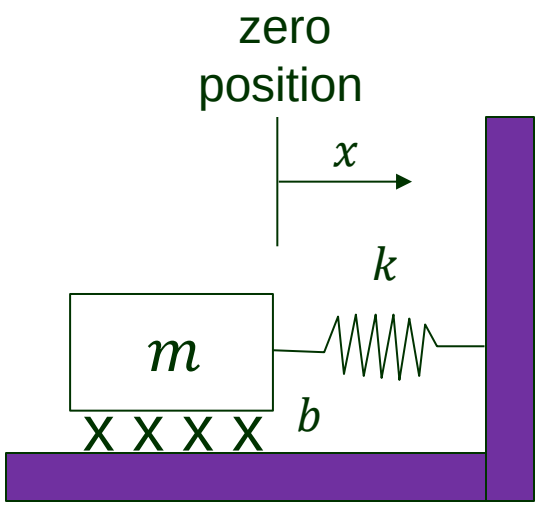
I. Mass-spring-damper system

II. Control of the 2nd order system

III. Joint space control

IV. Task space control

I. Mass-Spring-Damper System



m: mass of the block

k: stiffness of the spring ($k > 0$)

b: damping coefficient ($b > 0$)

x: displacement of the block

Newton's 2nd law:

$$m\ddot{x} = -b\dot{x} - kx$$

$$\Downarrow$$

$$m\ddot{x} + b\dot{x} + kx = 0$$ (2nd order ODE)

Solution of the ODE

$$m\ddot{x} + b\dot{x} + kx = 0$$

$$\Downarrow \quad x = e^{st}$$

$$ms^2 + bs + k = 0 \quad \text{(characteristic equation)}$$

$$\Downarrow$$

roots

$$s_1 = \frac{-b + \sqrt{b^2 - 4mk}}{2m}$$

$$s_2 = \frac{-b - \sqrt{b^2 - 4mk}}{2m}$$

} two poles of the ODE

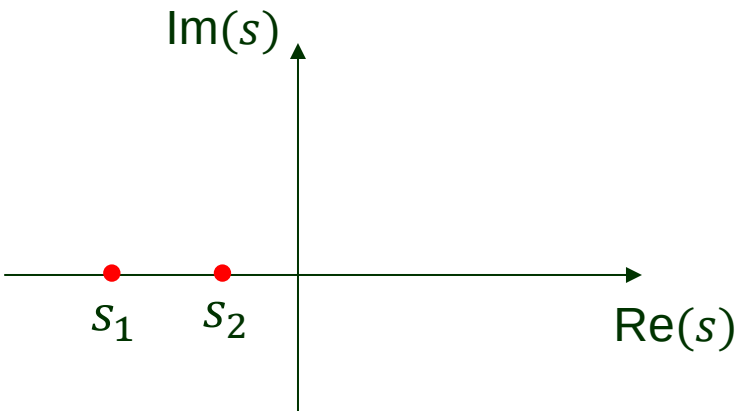
$$\Downarrow$$

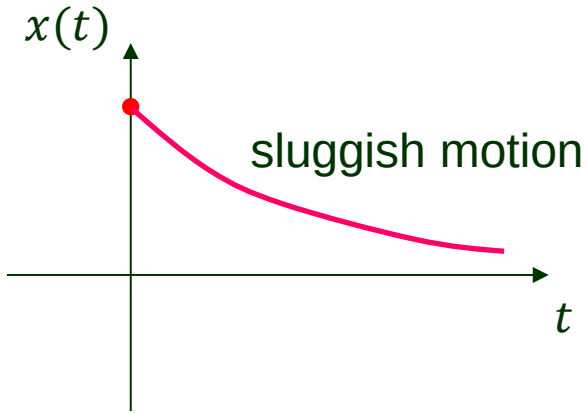
solution

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

Overdamped System

- $b^2 > 4mk$

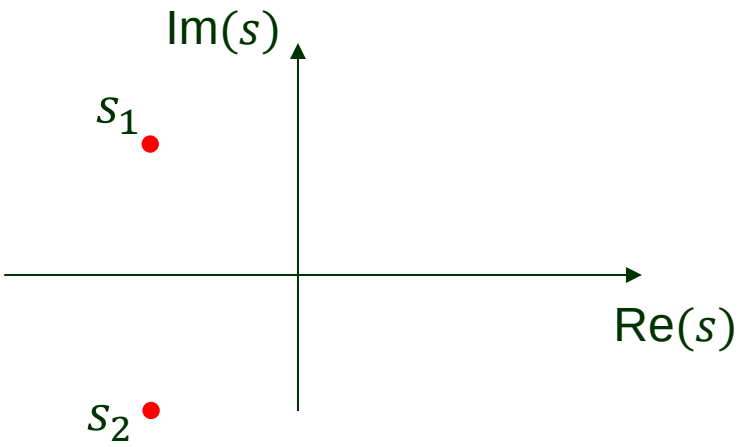


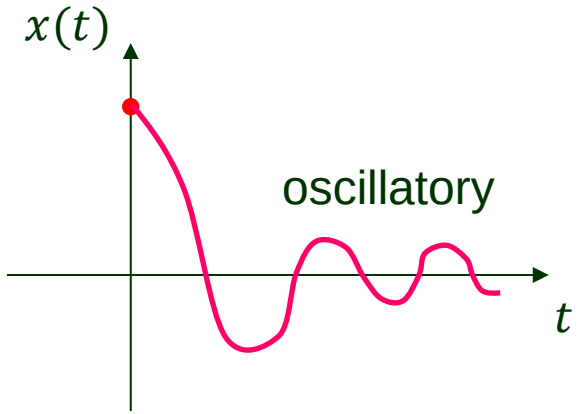


- ♣ Real and unequal roots.
- ♣ Friction (responsible for damping) dominates stiffness.

Underdamped System

- $b^2 < 4mk$

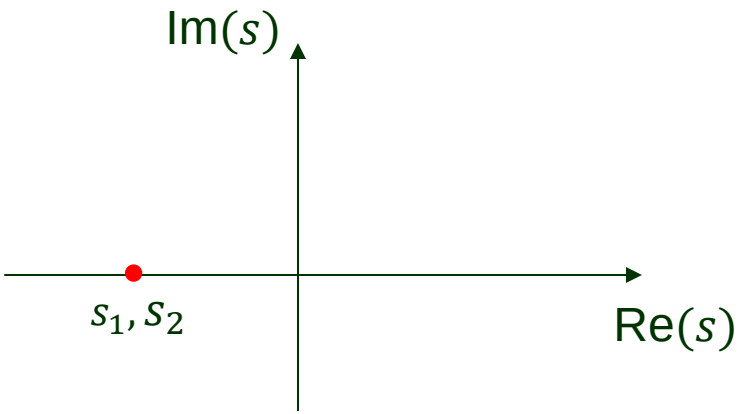


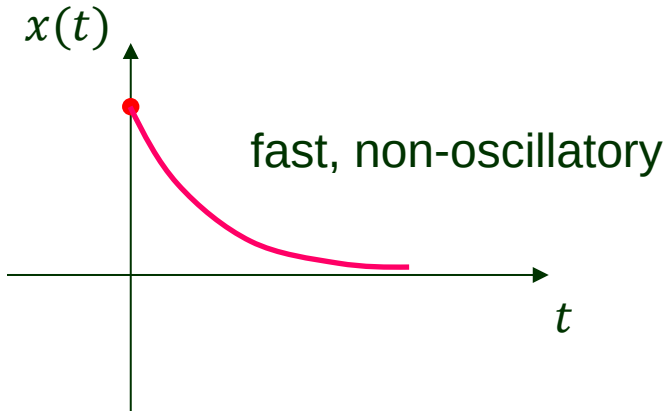


- ♣ Two complex roots.
- ♣ Stiffness dominates damping.

Critically Damped System

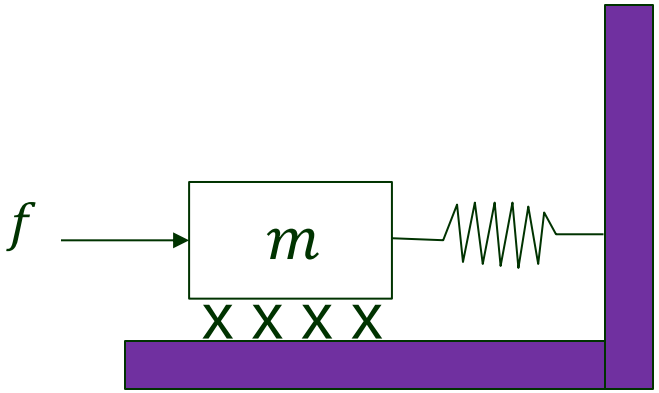
- $b^2 = 4mk$





- ♣ Equal real roots.
- ♣ Friction and stiffness balance.

II. Control of the 2nd-Order System



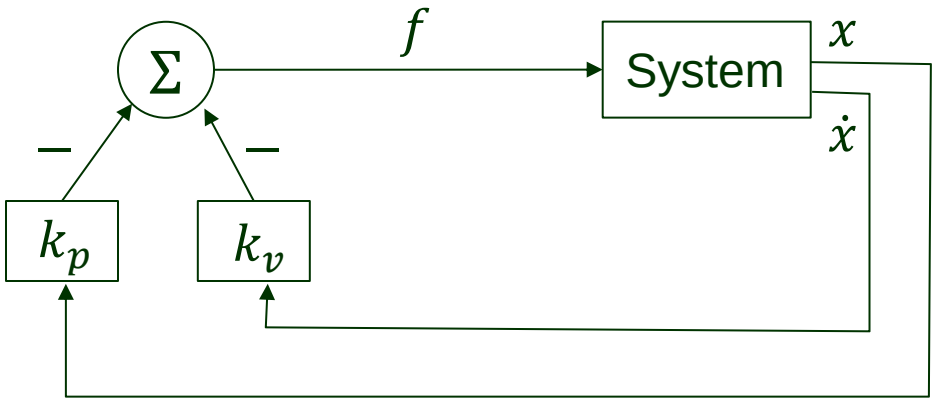
System dynamics:

$$f = m\ddot{x} + b\dot{x} + kx$$

Control law:

$$f = -k_v\dot{x} - k_p x$$

Control gains



Closed loop

Closed Loop System

$$f = m\ddot{x} + b\dot{x} + kx \qquad f = -k_v\dot{x} - k_p x$$

$$\Downarrow$$

$$m\ddot{x} + \underbrace{(b + k_v)}_{b'}\dot{x} + \underbrace{(k + k_p)}_{k'}x = 0$$

How to choose the gains k_p and k_v?

- Set a desired closed loop stiffness $k' \Longrightarrow k_p = k' - k$
- Obtain critical damping. $\Longrightarrow b' = 2\sqrt{mk'} \Longrightarrow k_v = 2\sqrt{mk'} - b$

Control Law Partitioning

Open loop equation (dynamics):

$$f = m\ddot{x} + b\dot{x} + kx$$

The control law consists of

- a model-based portion $f = \alpha f' + \beta$

  where $\alpha = m$

  $\beta = b\dot{x} + kx$

  Idea: If f' is taken as the new input, the system appears to be a unit mass.

- a servo portion $\ddot{x} = f'$

  Idea: Proceed as if the above were the open loop of a system to be controlled.

  $f' = -k_v\dot{x} - k_p x$

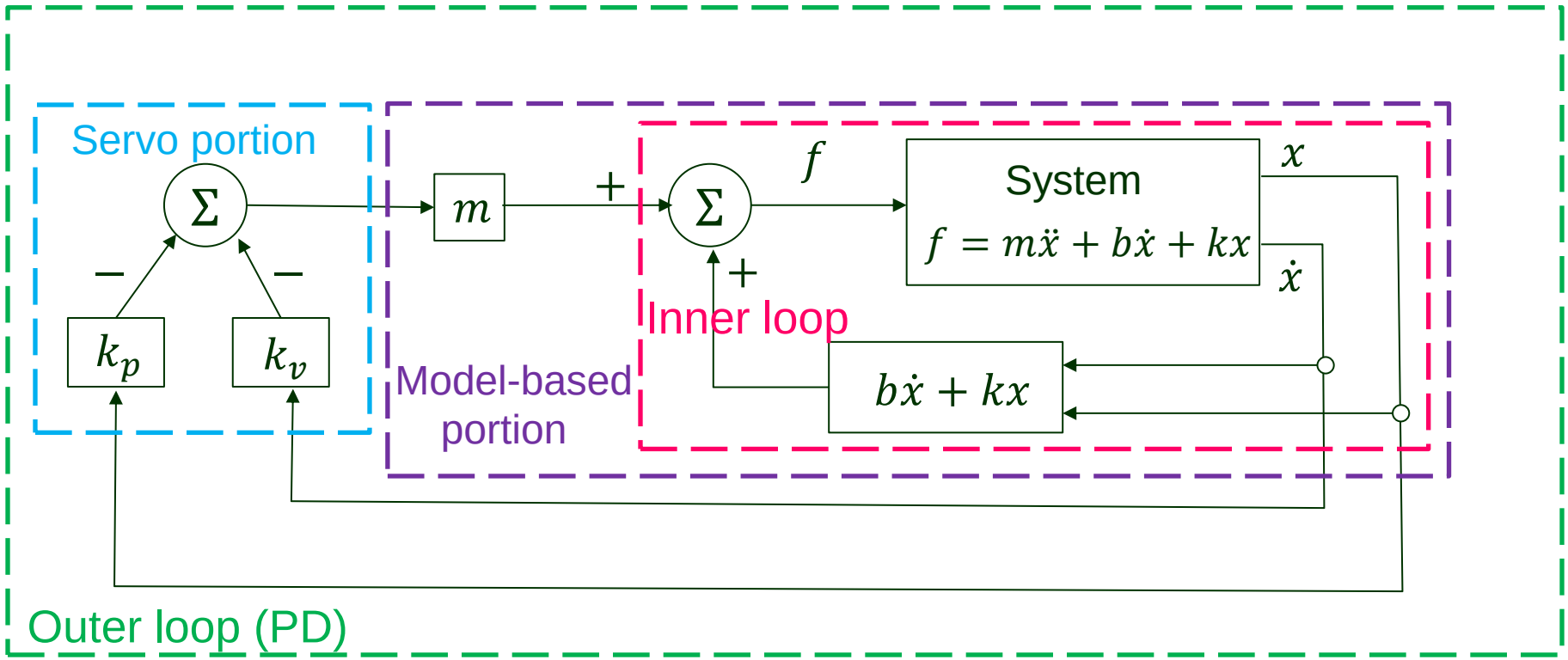
  $\Rightarrow \ddot{x} + k_v\dot{x} + k_p x = 0$

  (set $k_v = 2\sqrt{k_p}$ for critical damping)

Block Diagram

Proportional-derivative (PD) controller:

$$f = m\left(-k_p x - k_v \dot{x}\right) + b\dot{x} + kx$$



Trajectory Following

The current controller maintains the block at $x = 0$.

How to enhance it to make the block track a prescribed trajectory?

$x_d(t)$: desired trajectory

$e(t) \equiv x_d(t) - x(t)$: tracking error

Control law: $f' = \ddot{x}_d + k_v \dot{e} + k_p e$

$$\Downarrow \quad \ddot{x} = f'$$

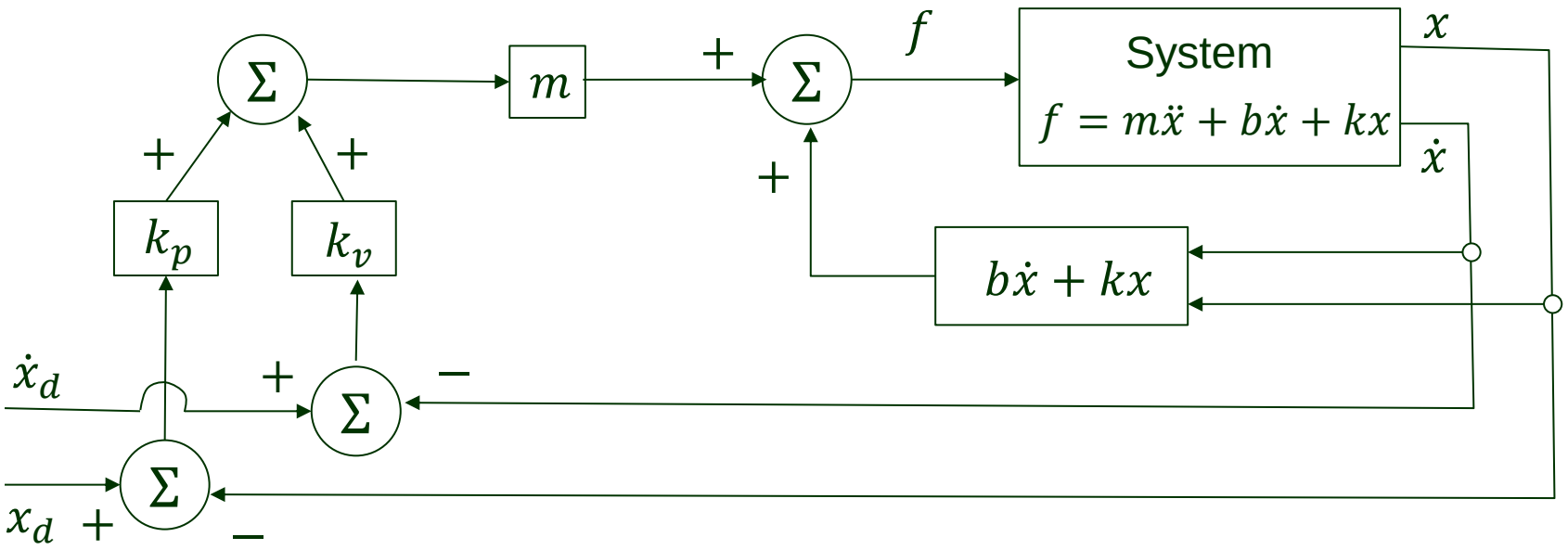
$$\ddot{e} + k_v \dot{e} + k_p e = 0 \qquad \text{(error dynamics)}$$

- We can design any response we wish, with critical damping often the choice.
- An initial error e_0 will be suppressed according to the error dynamics, after that the system will be following the desired trajectory.

Block Diagram for Trajectory Following

Controller:

$$f = m\left(\ddot{x}_d + k_v\dot{e} + k_p e\right) + b\dot{x} + kx$$



Disturbance Rejection

In the presence of a disturbance force ma_{dist}, the error equation changes to:

$$\ddot{e} + k_v\dot{e} + k_p e = a_{dist}$$

Consider the simplest case of f_{dist} being a constant.

Set all derivatives in the error equation to zero (i.e., consider when the state stops changing):

$$e = \frac{a_{dist}}{k_p}$$ (steady-state error)

To eliminate the error, add an integral term to the control law.

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e + k_i \int e\, dt$$

PID Control

$$f = m\left(\ddot{x}_d + k_p e + k_i \int e\,dt + k_v \dot{e}\right) + b\dot{x} + kx$$

proportional integral derivative

$$f' = \ddot{x}_d + k_v \dot{e} + k_p e + k_i \int e\,dt$$

$$\Downarrow \quad f' = \ddot{x}$$

$$\ddot{e} + k_v \dot{e} + k_p e + k_i \int e\,dt = 0$$

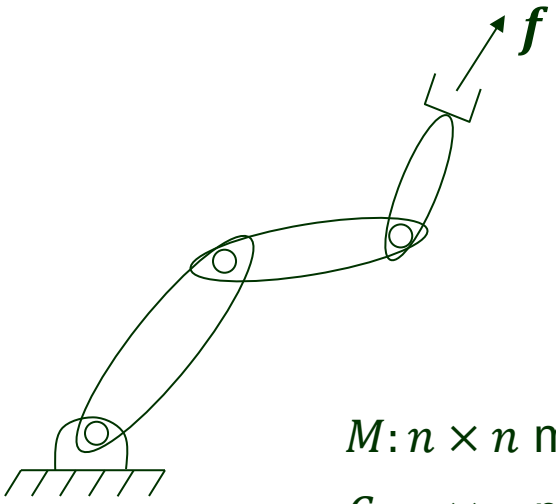
$$\Downarrow$$

$$\dddot{e} + k_v \ddot{e} + k_p \dot{e} + k_i e = 0$$

$$\Downarrow$$

$e = 0$ in a steady state ($\dddot{e} = \ddot{e} = \dot{e} = 0$)

III. Dynamics of an n-DOF Manipulator



$$M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f} = \boldsymbol{\tau}$$

$\boldsymbol{\theta} = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$: joint angles $\quad$ $\boldsymbol{\tau}$: joint torques

M: $n \times n$ mass matrix

C: $n \times n$ matrix of Coriolis and centrifugal terms

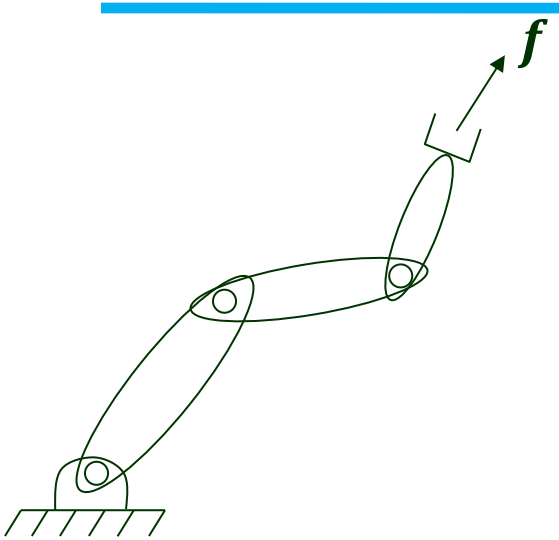
N: gravity term

J: end-effector Jacobian matrix

$\boldsymbol{f}$: force acted by the end-effector

- $M(\boldsymbol{\theta})$ is symmetric and positive definite.
- $\dot{M} - 2C$ is antisymmetric, i.e., $\left(\dot{M} - 2C\right)^T = -\dot{M} + 2C^T$.

Joint Space Control



Servo portion

Feedforward terms

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\boldsymbol{a} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T\boldsymbol{f}$$

Model-based portion

(containing system parameters)

Combine dynamics with control, since M is nonsingular:

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{a} \quad \text{(feedback)}$$

It then comes down to designing the servo $\boldsymbol{a}$ …

Servo Portion for Trajectory Tracking

Feedback-based:

$\boldsymbol{\theta}_d$: desired joint trajectory

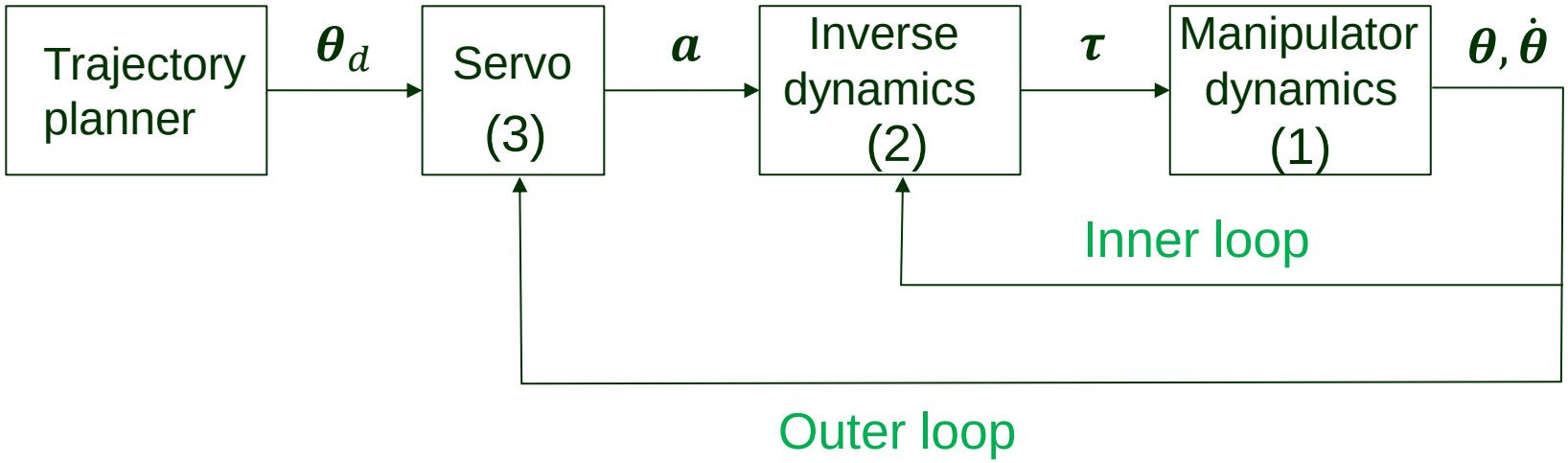
$\boldsymbol{\theta}_e = \boldsymbol{\theta}_d - \boldsymbol{\theta}$: joint error vector

Servo (PID):

$$\boldsymbol{a} = \ddot{\boldsymbol{\theta}}_d + K_v \dot{\boldsymbol{\theta}}_e + K_p \boldsymbol{\theta}_e + K_i \int \boldsymbol{\theta}_e \, dt$$

K_v, K_p, K_i: diagonal gain matrices (for decoupling of error dynamics) with positive elements on their diagonals.

Control Diagram



$$M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f} = \boldsymbol{\tau} \quad (1)$$

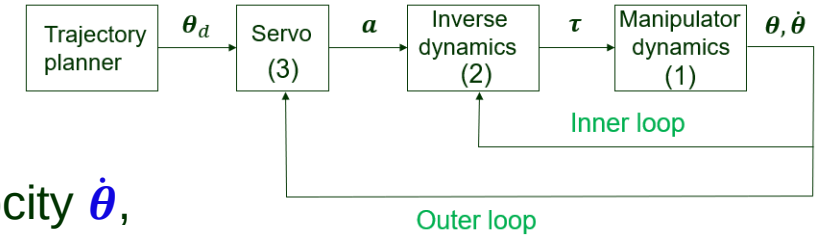
$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\boldsymbol{a} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f} \quad (2)$$

$$\boldsymbol{a} = \ddot{\boldsymbol{\theta}}_d + K_v \dot{\boldsymbol{\theta}}_e + K_p \boldsymbol{\theta}_e + K_i \int \boldsymbol{\theta}_e \, dt \quad (3)$$

Complete Expression

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\left(\ddot{\boldsymbol{\theta}}_d + K_v\left(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}\right) + K_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) + K_i\int(\boldsymbol{\theta}_d - \boldsymbol{\theta})dt\right)$$

$$+C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T\boldsymbol{f}$$



- The robot can read joint angle $\boldsymbol{\theta}$, joint velocity $\dot{\boldsymbol{\theta}}$, and external force $\boldsymbol{f}$ (via a force sensor).
- The controller can differentiate $\boldsymbol{\theta}_d$ to obtain its first two derivatives.
- But we cannot differentiate $\boldsymbol{\theta}$ to obtain $\ddot{\boldsymbol{\theta}}$, which is updated by forward dynamics as below. (Otherwise, we would be creating a loop for ourselves.)

$$\ddot{\boldsymbol{\theta}} = M(\boldsymbol{\theta})^{-1}\left(\boldsymbol{\tau} - C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} - V(\boldsymbol{\theta}) - J^T\boldsymbol{f}\right)$$

- The idea of control is to set $\boldsymbol{\tau}$ properly to alter $\ddot{\boldsymbol{\theta}}$ (and thus $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$) in order to realize some objective.
- Due to time discretization, the value of $\boldsymbol{\tau}$ computed over current readings of $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\boldsymbol{f}$ will affect $\ddot{\boldsymbol{\theta}}$ at the next (not the current) time instant.

Error Dynamics

$$M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f} = \boldsymbol{\tau} \qquad \boldsymbol{\tau} = M(\boldsymbol{\theta})\boldsymbol{a} + C\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f}$$

$$M(\boldsymbol{\theta})\,\ddot{\boldsymbol{\theta}} = M(\boldsymbol{\theta})\,\boldsymbol{a}$$

$M(\boldsymbol{\theta})$ square and invertible

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{a}$$

(This is the starting point as long as the controller is under the same scheme.)

$$\boldsymbol{a} = \ddot{\boldsymbol{\theta}}_d + K_v \dot{\boldsymbol{\theta}}_e + K_p \boldsymbol{\theta}_e + K_i \int \boldsymbol{\theta}_e \, dt$$

$$\ddot{\boldsymbol{\theta}}_e + K_v \dot{\boldsymbol{\theta}}_e + K_p \boldsymbol{\theta}_e + K_i \int \boldsymbol{\theta}_e \, dt = \mathbf{0}$$

Matrices with positive elements on the diagonals and zeros elsewhere

$$\dddot{\boldsymbol{\theta}}_e + K_v \ddot{\boldsymbol{\theta}}_e + K_p \dot{\boldsymbol{\theta}}_e + K_i \boldsymbol{\theta}_e = \mathbf{0}$$

Remarks on PID Control

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\left(\ddot{\boldsymbol{\theta}}_d + K_v \dot{\boldsymbol{\theta}}_{\boldsymbol{e}} + K_p \boldsymbol{\theta}_e + K_i \int \boldsymbol{\theta}_e \, dt \right) + C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + V(\boldsymbol{\theta}) + J^T \boldsymbol{f}$$

$$\dddot{\boldsymbol{\theta}}_e + K_v \ddot{\boldsymbol{\theta}}_e + K_p \dot{\boldsymbol{\theta}}_e + K_i \boldsymbol{\theta}_e = \mathbf{0} \qquad \text{(error dynamics)}$$

- The proportional gain K_p acts like a virtual spring to reduce the position error $\boldsymbol{\theta}_e$.
- The derivative gain K_v acts like a virtual damper to reduce the velocity error $\dot{\boldsymbol{\theta}}_{\boldsymbol{e}}$.
- The integral gain K_i acts to eliminate the steady-state error.
- The controller can track a desired joint trajectory $\boldsymbol{\theta}_d$, with zero steady-state error, under the condition that the largest eigenvalue of K_i is less than the product of the two smallest eigenvalues respectively of K_v and K_p.

IV. Task Space Control

- From the robot's perspective, joint space control is convenient because trajectories are described by joint angles whose limits are easily expressed.
- From the task's perspective, it is often more convenient to describe the trajectory of the end-effector in the task space, in which constraints due to the environment and involved objects are naturally expressed.

$\boldsymbol{x}(\boldsymbol{\theta}) \in \mathbb{R}^m$: end-effector position and orientation ($m = 3$ or 6)

$J(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}$: Jacobian matrix

$$\dot{\boldsymbol{x}} = J\dot{\boldsymbol{\theta}} \quad \Longrightarrow \quad \ddot{\boldsymbol{x}} = \dot{J}\dot{\boldsymbol{\theta}} + J\ddot{\boldsymbol{\theta}}$$

$$\Longrightarrow \quad \ddot{\boldsymbol{\theta}} = J^{\dagger}\left(\ddot{\boldsymbol{x}} - \dot{J}\dot{\boldsymbol{\theta}}\right) + \left(I - J^{\dagger}J\right)\boldsymbol{\beta}$$

Pseudoinverse of J

Vector from the null space of J is zeroed as we are considering the joint acceleration with the smallest magnitude for designing control next.

- Often the robot has at least m degrees of freedom ($n \geq m$). When this holds and J has linearly independent rows, $J^{\dagger} = J^T(JJ^T)^{-1}$ such that $JJ^{\dagger} = I_m$.

Trajectory Tracking in the Task Space

$$\ddot{\boldsymbol{\theta}} = J^{\dagger}\left(\ddot{\boldsymbol{x}} - \dot{J}\dot{\boldsymbol{\theta}}\right)$$

$\boldsymbol{a}$ feedforward terms

Idea: Replace the acceleration $\ddot{\boldsymbol{x}}$ with the servo $\boldsymbol{a}$ to be constructed over the position error feedback.

$\boldsymbol{x_d(\theta)} \in \mathbb{R}^m$: desired end-effector position and orientation

$\boldsymbol{x_e} = \boldsymbol{x_d} - \boldsymbol{x}$: position error vector

$$\boldsymbol{a} = J^{\dagger}\left(\ddot{\boldsymbol{x}}_d + K_v\dot{\boldsymbol{x}}_e + K_p\boldsymbol{x}_e + K_i\int \boldsymbol{x}_e\, dt - \dot{J}\dot{\boldsymbol{\theta}}\right)$$

Error Dynamics for Task Space Control

$$\ddot{\boldsymbol{\theta}} = J^{\dagger}\left(\ddot{\boldsymbol{x}} - \dot{J}\dot{\boldsymbol{\theta}}\right) \qquad \boldsymbol{a} = J^{\dagger}\left(\ddot{\boldsymbol{x}}_d + K_v\dot{\boldsymbol{x}}_e + K_p\boldsymbol{x}_e + K_i\int \boldsymbol{x}_e\, dt - \dot{J}\dot{\boldsymbol{\theta}}\right)$$

$$\Downarrow \quad \ddot{\boldsymbol{\theta}} = \boldsymbol{a}, \quad JJ^{\dagger} = I_m$$

$$\ddot{\boldsymbol{x}}_e + K_v\dot{\boldsymbol{x}}_e + K_p\boldsymbol{x}_e + K_i\int \boldsymbol{x}_e\, dt = \mathbf{0}$$

- When the robotic arm has redundant DOFs ($n > m$), the Jacobian J has a non-empty null space, which can be used to realize an additional control objective such as realizing a “smoothest” or “minimum energy” trajectory.
- The above is the subject of redundant manipulator control.

References

1. John J. Craig. *Introduction to Robotics: Mechanics and Control*, 2nd edition. Addision-Wesley, 1989.

2. Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

3. Frank L. Lewis, Darren M. Dawson, and Chaouki T. Abdallah. *Robot Manipulator Control: Theory and Practice,* 2nd edition. Marcel Dekker, Inc., 2004.

4. Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control.* Cambridge University Press, 2017.

5. Mark W. Spong, Seth Hutchinson, M. Vidyasagar. *Robot Modeling and Control*. John Wiley and Sons, Inc., 2005.