Kinematics, Jacobian, Dynamics and Control - Planar RR Robot Example

Yu Chung (Paul) Lee

Department of Electrical and Computer Engineering

The University of British Columbia

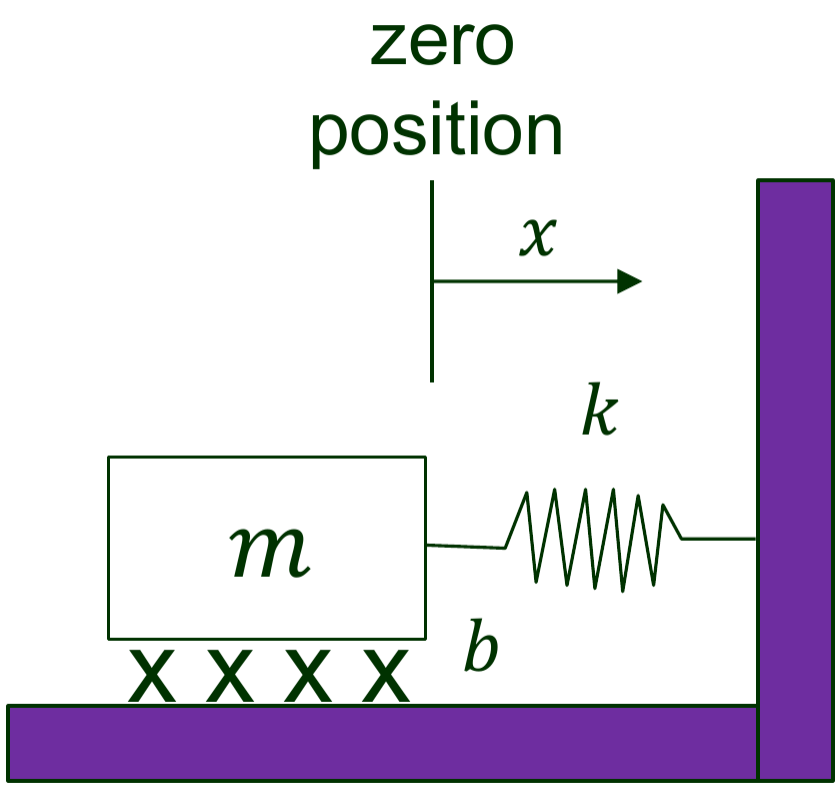
The University of Hong Kong

Version 1.6, 2026

Disclaimer: These notes referenced the contents of UBC robotics course by Prof. Tim Salcudean, ISU Tutorial on Robotic Manipulator Control by Prof. Yan-Bin Jia, and ChatGPT.

Table of contents

Mass-Spring-Damper System



- m: mass of the block
- k: spring stiffness ($k > 0$)
- b: damping coefficient ($b > 0$)
- x: displacement of the block

Newton's 2nd law:

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

$$\Downarrow$$

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

Reference: Tutorial on Robotic Manipulator Control by Prof. Yan-Bin Jia

Source of Spring Force

Spring force comes from anything that stores mechanical energy when displaced:

- **Actual coil spring**: The most obvious case.
- **Elastic deformation of materials**: Beams, shafts, robot links, joints, seals, tires, belts, tendons, and even frames flex slightly and act like springs.
- **Torsional stiffness in rotating systems**: For rotation, the equivalent is $\tau_{\text{spring}} = -k_\theta \theta$ This happens in flexible couplings, gear trains, harmonic drives, cable drives, etc.
- **Gravity linearized around equilibrium**: Near an equilibrium point, gravity can behave like an effective spring. Example: small-angle pendulum $\tau \approx -mgl\,\theta$ which looks like a rotational spring.
- **Compressed fluids or air**: Pneumatic systems, soft actuators, and trapped fluid volumes can create restoring forces.

Source of Damping Force

Damping comes from anything that dissipates energy, usually into heat:

- **Viscous friction**: Oil dashpots, hydraulic dampers, lubricated sliding surfaces, fluid drag with $F_{\text{damping}} = -b\dot{x}$.
- **Material internal losses**: Rubber, polymers, soft tissues, and metal structures under vibration lose energy internally. This is often called structural damping.
- **Dry friction approximated as damping**: Real Coulomb friction is not exactly proportional to velocity, but near some operating point engineers often approximate it as viscous damping.
- **Air resistance / fluid drag**: At low speeds this can be approximated as proportional to velocity.
- **Electrical damping in motors**: In electromechanical systems, back-EMF and electrical resistance can create damping-like behavior.
- **Control damping**: A feedback controller can intentionally create damping. Example: a PD controller $u = -K_p x - K_d \dot{x}$ adds an artificial spring term and damping term.

Solution of the ODE

Characteristic equation:

$$m\ddot{x} + b\dot{x} + kx = 0 \xrightarrow{x=e^{st}} ms^2 + bs + k = 0$$

roots:

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

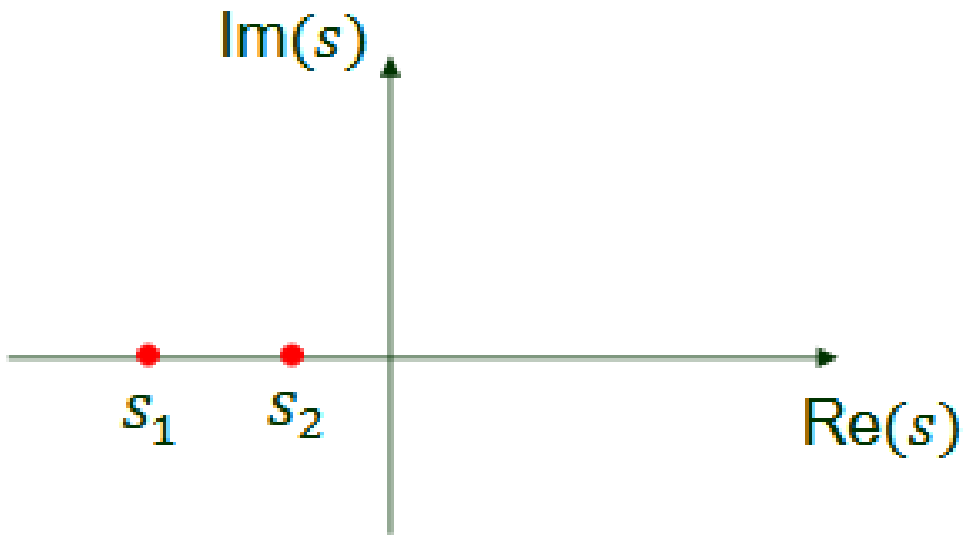
$$s_2 = \frac{-b - \sqrt{b^2 - 4mk}}{2m}$$

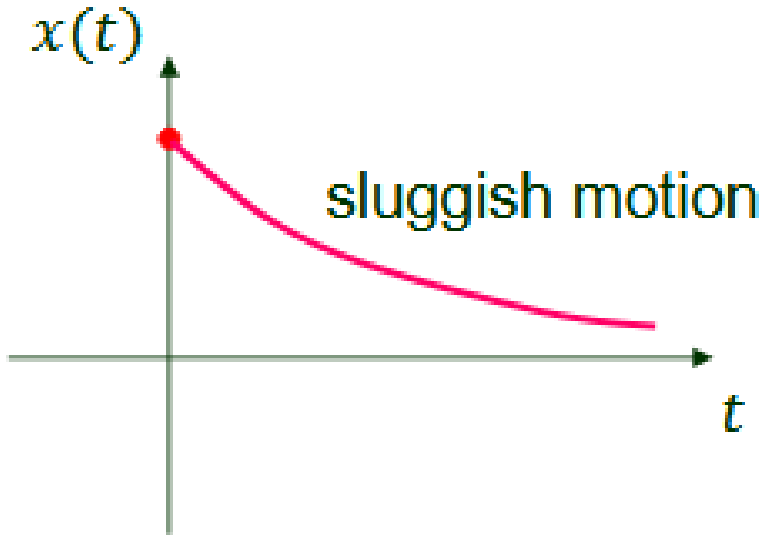
two poles of the ODE

$$\xrightarrow{\text{solution}} \quad x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

Overdamped System

$b^2 > 4mk$

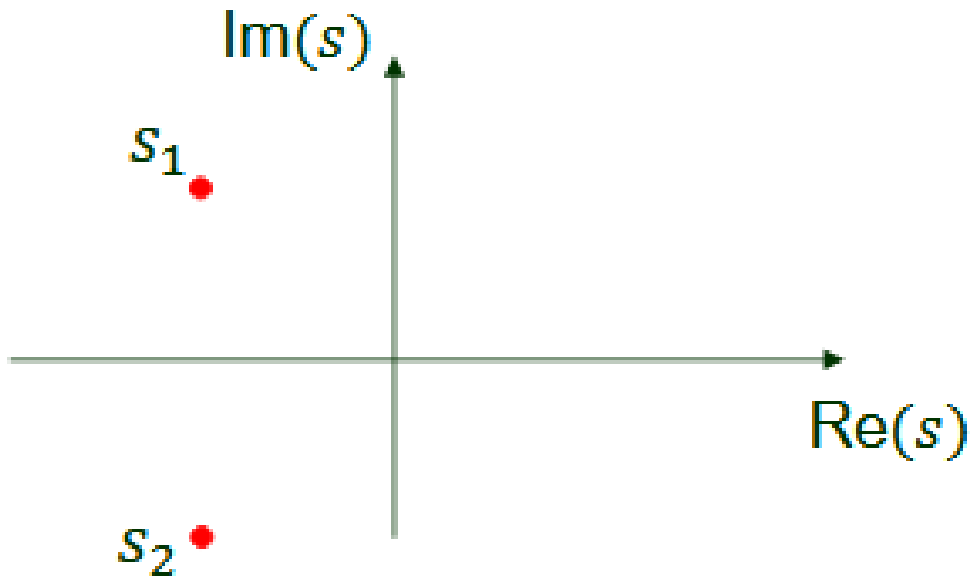


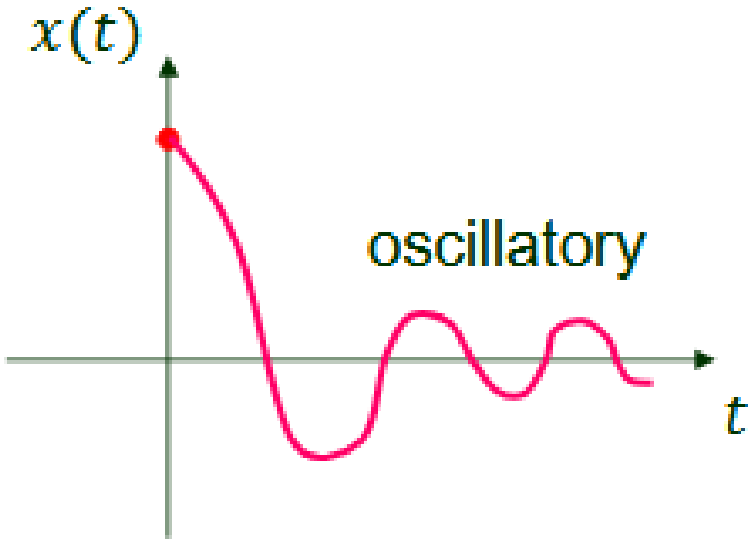


Results in two distinct real roots: $s_1, s_2 \in \mathbb{R}, \quad s_1 \neq s_2$. Solution becomes $x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$. The system has no oscillation and return to equilibrium state slowly. Friction (responsible for damping) dominates stiffness.

Uncerdamped System

$b^2 < 4mk$

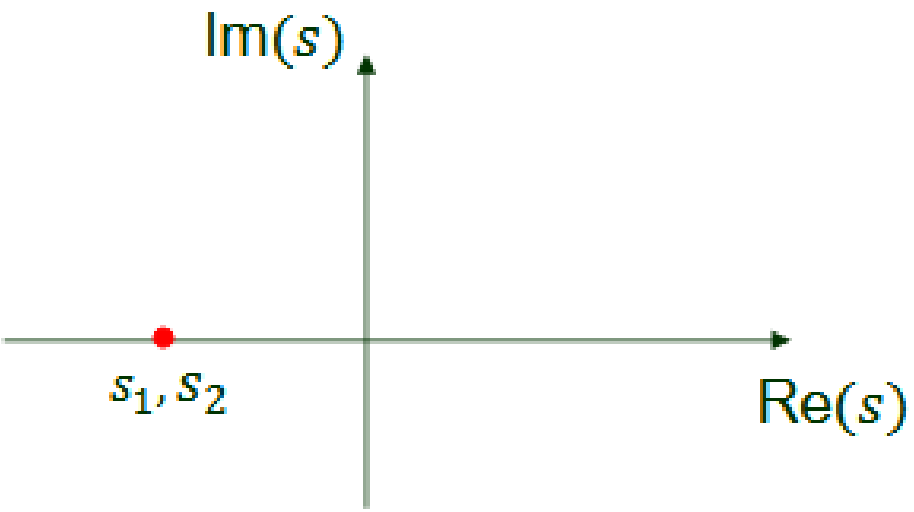


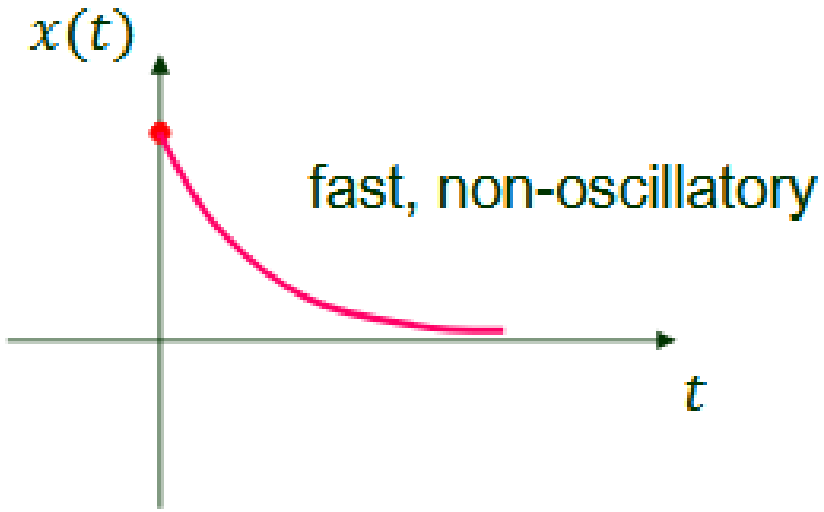


Results in two complex conjugate roots: $s_{1,2} = -\frac{b}{2m} \pm j\frac{\sqrt{4mk-b^2}}{2m}$. Solution becomes $x(t) = e^{-\frac{b}{2m}t}(c_1 \cos(\omega_d t) + c_2 \sin(\omega_d t))$, where $\omega_d = \sqrt{\frac{k}{m} - (\frac{b}{2m})^2}$. The system has oscillation with shrinking amplitude. Stiffness dominates damping in this case.

Critically Damped System

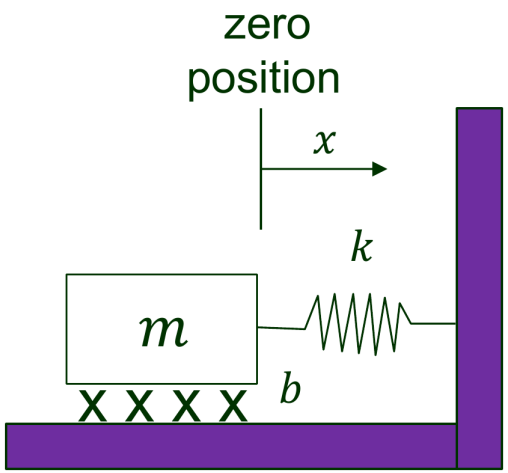
$$b^2 = 4mk$$

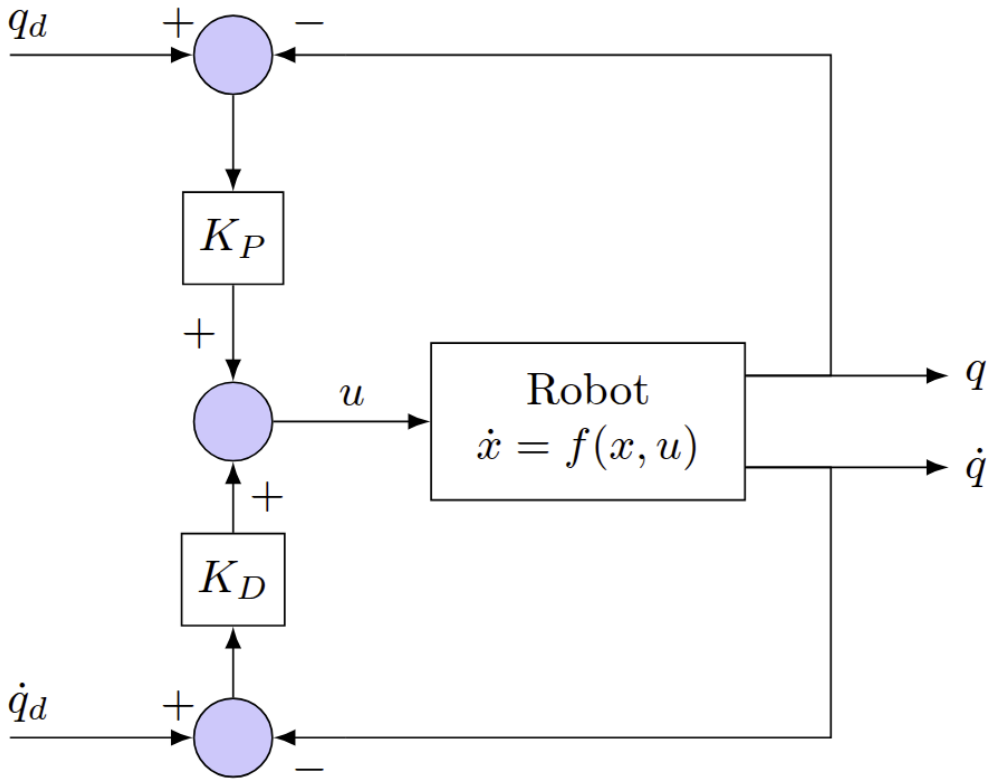




Results in equal real roots with $s = -\frac{b}{2m}$. Solution becomes $x(t) = (c_1 + c_2 t)e^{st}$ provides the fastest return to equilibrium without oscillation, balancing friction and stifness. This is the most desirable case in most control system.

Control of the 2^{nd}-Order System





System dynamics:

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

Control law of the closed loop feedback system:

$$f_{control} = -k_v\dot{x} - k_p x$$

Closed Loop Control System

Combining the two, with $f_{control} = f_{system}$

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

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

How to choose the gains k_p and k_v?

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

Control Law Partitioning

Open loop equation (dynamics)

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

The control law consists of

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

  where $\alpha = m, \qquad \beta = b\dot{x} + kx$

  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.

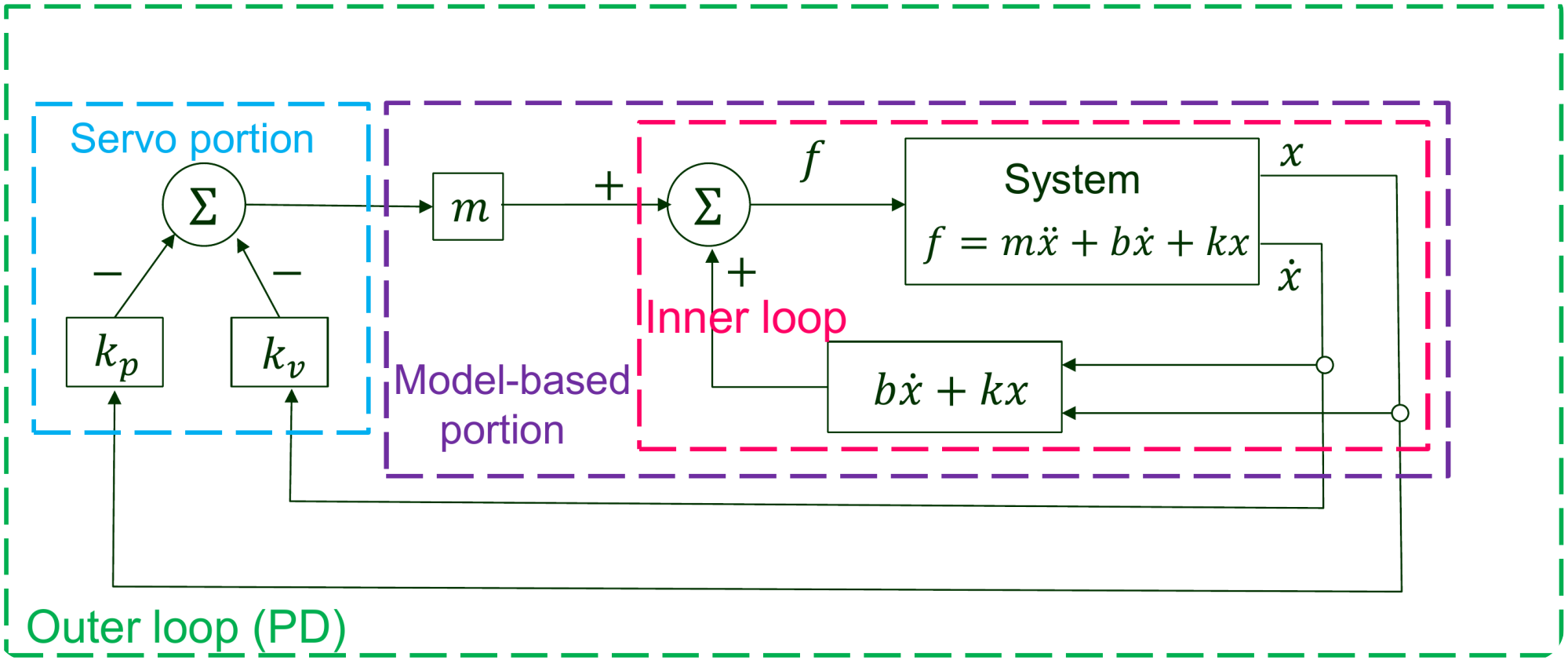
$$f' = -k_v\dot{x} - k_p x \implies \ddot{x} + k_v\dot{x} + k_p x = 0$$

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

Model based PD Control Block Diagram

Proportional-derivative (PD) controller:

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



Trajectory Following

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

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

desired trajectory: $x_d(t)$

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

Control law:

$$f' = \ddot{x}_d + k_v \dot{e} + k_p e$$
$$\Downarrow \ddot{x} = f'$$
$$\ddot{e} + k_v \dot{e} + k_p e = 0$$

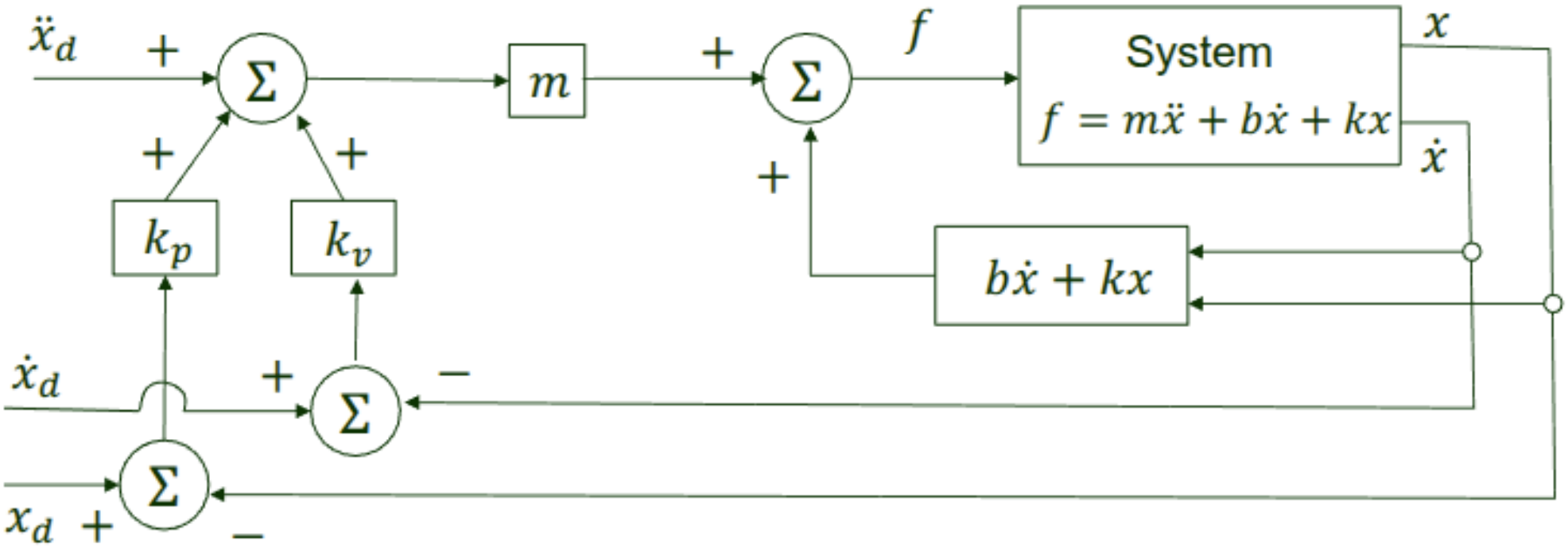
where $\ddot{e}$ is the *error dynamics*.

Block Diagram for Trajectory Following

Controller:

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

With consideration of desired x_d and $\dot{x}_d$



Distrubance 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):

Steady-State Error:

$$e = \frac{a_{dist}}{k_p}$$

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

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

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

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

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

$$\Downarrow$$

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

$$\Downarrow$$

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

PID Tuning Heuristic Method

Ziegler–Nichols method: K_p is increased from zero to ultimate gain K_u with stable output and consistent oscillations. K_u and the oscillation period T_u are then used to set the PID gains.

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.83\bar{T}_u$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.10K_uT_u$
classic PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$
Pessen Integral Rule	$0.7K_u$	$0.4T_u$	$0.15T_u$	$1.75K_u/T_u$	$0.105K_uT_u$
some overshoot	$0.33K_u$	$0.50T_u$	$0.33\bar{T}_u$	$0.66K_u/T_u$	$0.11K_uT_u$
no overshoot	$0.20K_u$	$0.50T_u$	$0.33\bar{T}_u$	$0.40K_u/T_u$	$0.066K_uT_u$

General State-Space Model

For a general Linear Time Invariant (LTI) model (possibly multi-input, multi-output):

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

where:

$\mathbf{x}(t) \in \mathbb{R}^n$: the n state variables

$\mathbf{u}(t) \in \mathbb{R}^m$: the m state inputs

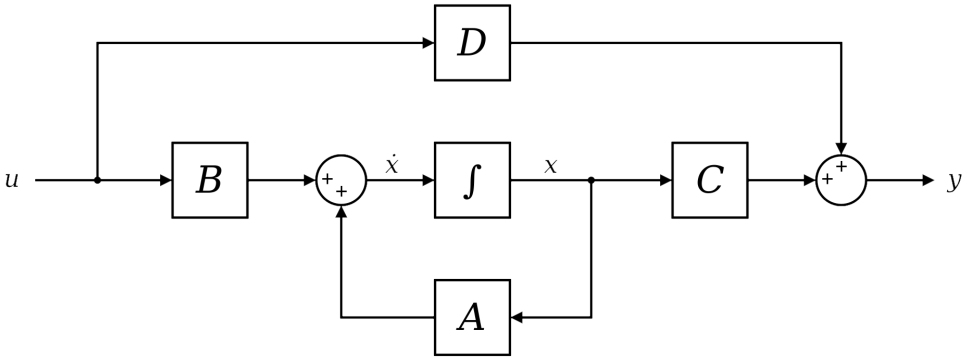
$\mathbf{y}(t) \in \mathbb{R}^p$: the p outputs

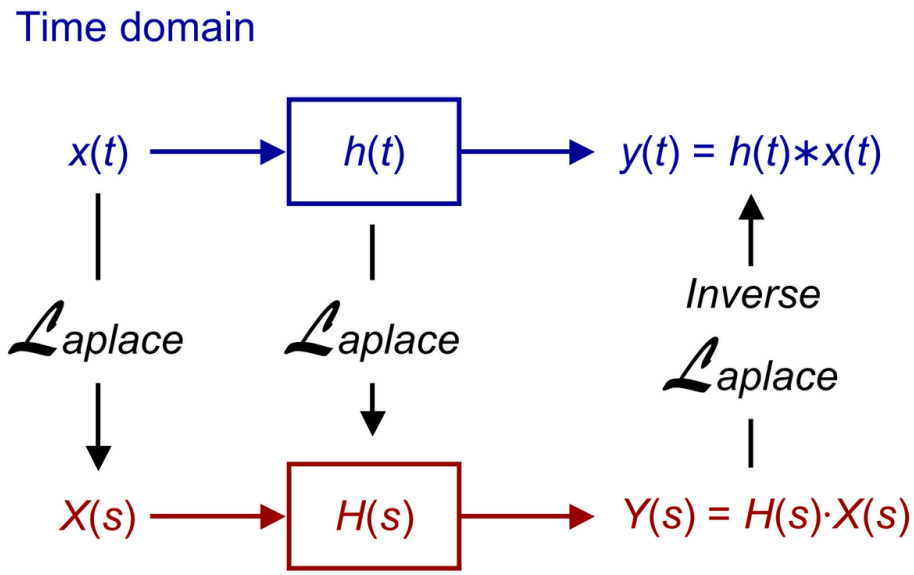
$A \in \mathbb{R}^{n \times n}$: state matrix (controls the latent state)

$B \in \mathbb{R}^{n \times m}$: control matrix

$C \in \mathbb{R}^{p \times n}$: output matrix

$D \in \mathbb{R}^{p \times m}$: command matrix





State-Space Model of Mass-Damper-Spring System

The second-order differential equation:

$$\ddot{x} = -\frac{k}{m}x - \frac{b}{m}\dot{x} + \frac{1}{m}u$$

can be written in the standard state-space form:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

$$\text{where:} \qquad \mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

Typical output choices are:

$$\text{Position output:} \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = 0$$
$$\text{Velocity output:} \qquad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0$$

Stability Analysis: Root Locus Method

Root Locus Key Concepts

The root locus plot indicates how the closed loop poles of a system vary with a system parameter (typically a gain, K).

We can choose a value of 's' on this locus that will give us good results. The shape of the locus can also give us information on design of a more complex (lead/lag, PID controller).

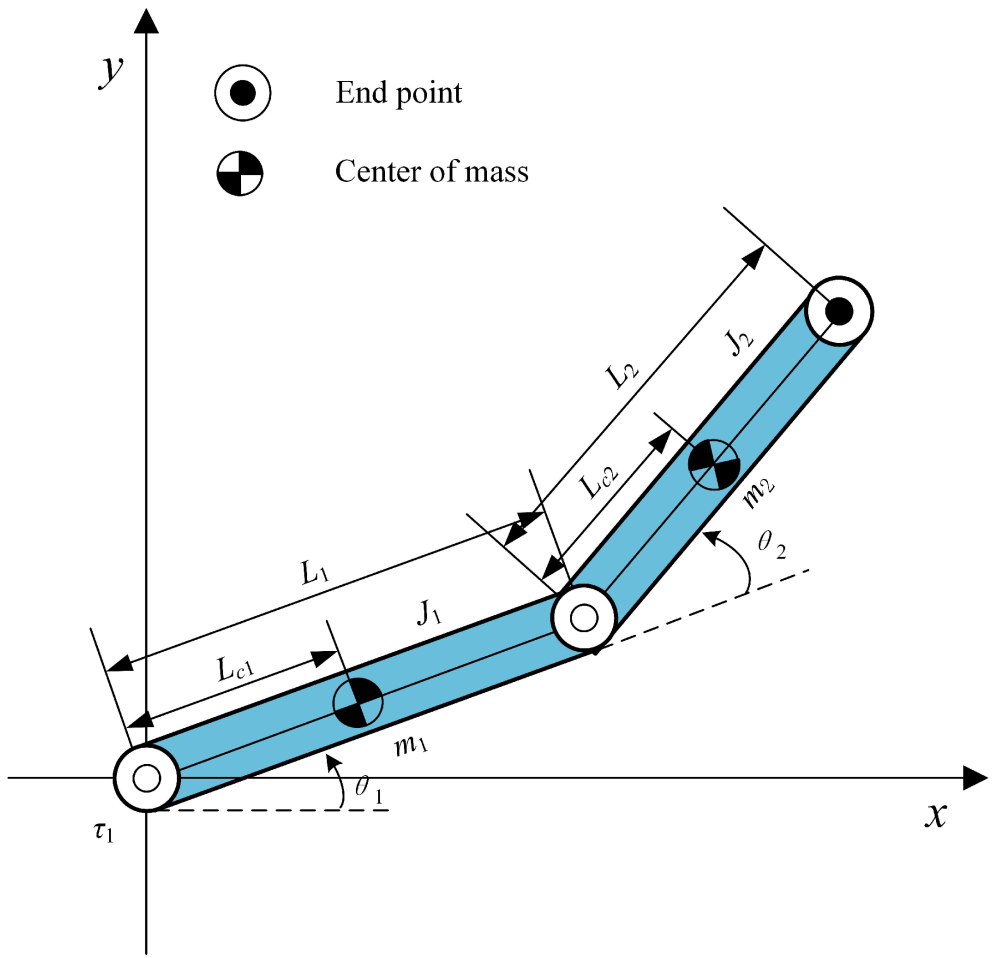
Planar RR Robot Overview

The standard derivation for a **planar double pendulum**, treated as a **2-link revolute manipulator (2R robot)**. Assume:

- link lengths: l_1, l_2
- joint angles: q_1, q_2
- q_1 is measured from the inertial x-axis
- q_2 is the **relative** angle between link 1 and link 2

So the absolute orientation of link 2 is

$$\theta_2 = q_1 + q_2$$



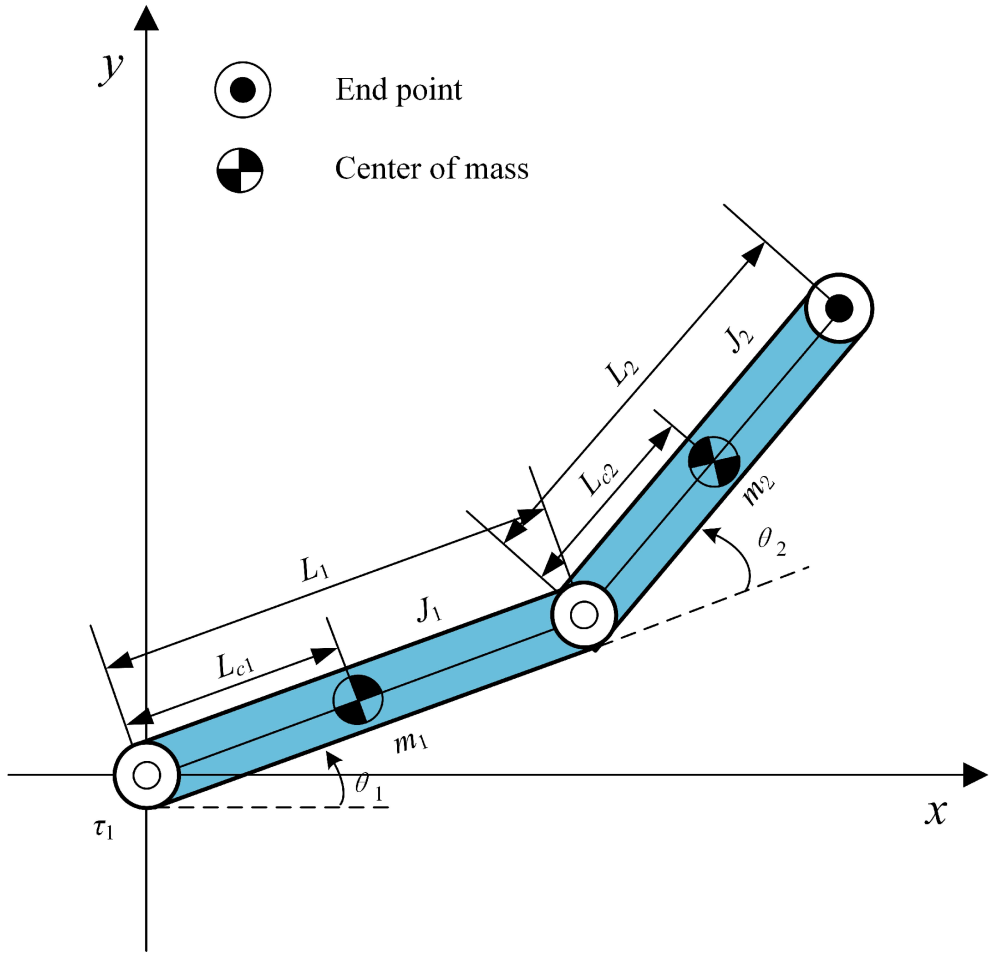
(Zixin Huang, 2025)

Geometry and Coordinate Frames

Let the base be at the origin.

- Joint 1 is at $(0, 0)$
- Joint 2 is at the end of link 1
- The end-effector (tip of link 2) is the tip of the double pendulum

$$c_1 = \cos q_1, \quad s_1 = \sin q_1,$$
$$c_{12} = \cos(q_1 + q_2), \quad s_{12} = \sin(q_1 + q_2).$$



Forward kinematics by geometry

Position of joint 2

The position of joint 2 (the end of the first link) is:

$$p_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_1 \cos q_1 \\ l_1 \sin q_1 \end{bmatrix}$$

Position of the end-effector

The second link contributes a vector of length l_2 oriented at angle $q_1 + q_2$. Therefore,

$$p_e = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \end{bmatrix}$$

So the forward kinematics are and the end-effector orientation is

$$x = l_1 c_1 + l_2 c_{12}, \qquad y = l_1 s_1 + l_2 s_{12}, \qquad \phi = q_1 + q_2.$$

Homogeneous transformations

For a planar 2R manipulator, the homogeneous transform of each link can be written as

$$
{}^0T_1 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & l_1 \cos q_1 \\ \sin q_1 & \cos q_1 & 0 & l_1 \sin q_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1T_2 = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & l_2 \cos q_2 \\ \sin q_2 & \cos q_2 & 0 & l_2 \sin q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_2 = {}^0T_1 {}^1T_2 = \begin{bmatrix} \cos(q_1+q_2) & -\sin(q_1+q_2) & 0 & l_1 \cos q_1 + l_2 \cos(q_1+q_2) \\ \sin(q_1+q_2) & \cos(q_1+q_2) & 0 & l_1 \sin q_1 + l_2 \sin(q_1+q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

From the last column, the end-effector position is again

$$
p_e = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1+q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1+q_2) \end{bmatrix}
$$

Velocity kinematics

The end-effector velocity is obtained by differentiating x and y.

Differentiate x

$$x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$\dot{x} = -l_1 \sin q_1 \dot{q}_1 - l_2 \sin(q_1 + q_2)(\dot{q}_1 + \dot{q}_2)$$

$$\dot{x} = [-l_1 \sin q_1 - l_2 \sin(q_1 + q_2)]\dot{q}_1 - l_2 \sin(q_1 + q_2)\dot{q}_2$$

Differentiate y

$$y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{y} = l_1 \cos q_1, \dot{q}_1 + l_2 \cos(q_1 + q_2)(\dot{q}_1 + \dot{q}_2).$$

$$\dot{y} = [l_1 \cos q_1 + l_2 \cos(q_1 + q_2)]\dot{q}_1 + l_2 \cos(q_1 + q_2)\dot{q}_2$$

Jacobian derivation

By definition,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

where the Jacobian is

$$J(q) = \frac{\partial(x, y)}{\partial(q_1, q_2)} = \frac{\partial \mathbf{x}}{\partial \mathbf{q}}$$

Compute the partial derivatives:

$$\frac{\partial x}{\partial q_1} = -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) \qquad \frac{\partial x}{\partial q_2} = -l_2 \sin(q_1 + q_2)$$

$$\frac{\partial y}{\partial q_1} = l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \qquad \frac{\partial y}{\partial q_2} = l_2 \cos(q_1 + q_2)$$

Jacobian derivation

Jacobian matrix:

$$J(q) = \frac{\partial(x, y)}{\partial(q_1, q_2)} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} \end{bmatrix}$$

$$J(q) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2)[4pt] \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix}$$

Using the shorthand notation,

$$J(q) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}$$

Full planar task Jacobian (x, y, ϕ)

If you also want the end-effector orientation ($\phi = q_1 + q_2$), then

$$\dot{\phi} = \dot{q}_1 + \dot{q}_2$$

So the planar geometric Jacobian becomes

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

Thus,

$$J_{xy\phi}(q) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 1 & 1 \end{bmatrix}$$

Jacobian Columns Interpretation

Each Jacobian column tells you how the end-effector moves if only one joint moves.

First column - Effect of $\dot{q}_1$

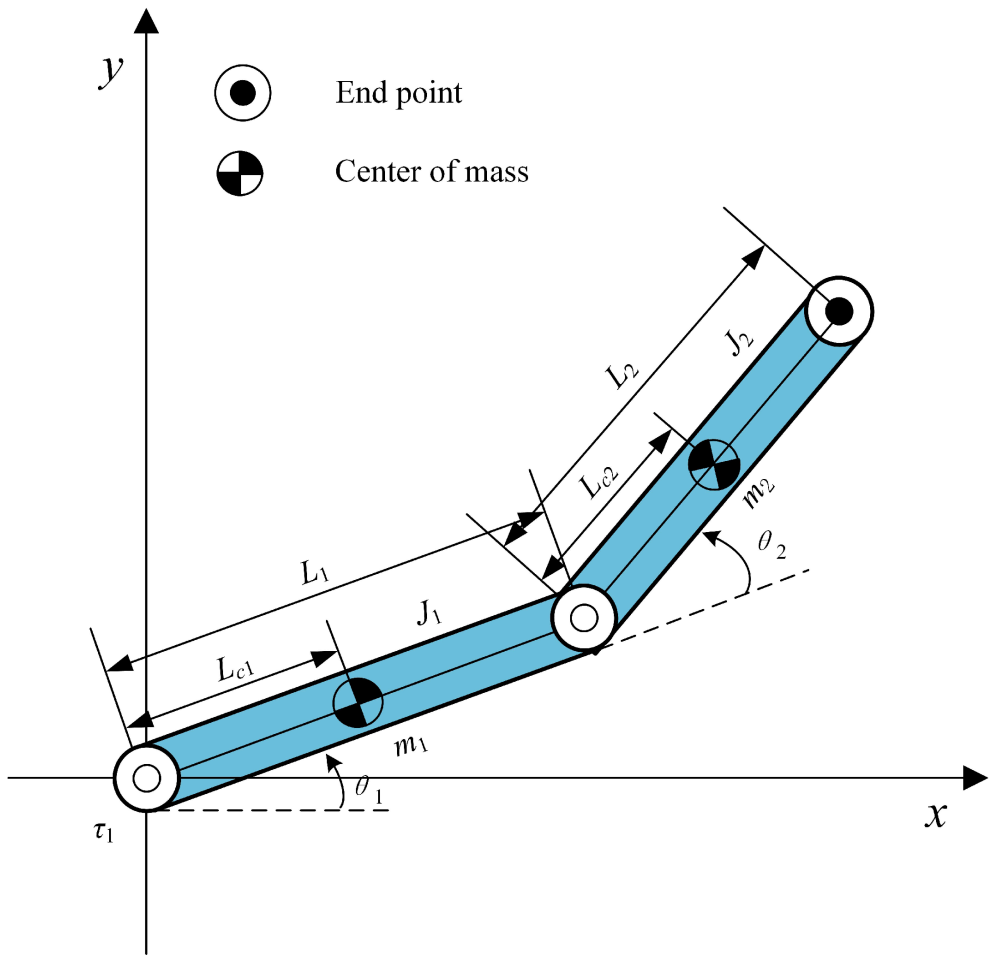
$$\begin{bmatrix} -l_1 s_1 - l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} \end{bmatrix}$$

This makes sense because rotating joint 1 moves **both** links.

Second column - Effect of $\dot{q}_2$

$$\begin{bmatrix} -l_2 s_{12} \\ l_2 c_{12} \end{bmatrix}$$

Only moves the second link relative to the first.



Singularities

The determinant of the 2×2 position Jacobian is

$$\det J = l_1 l_2 \sin q_2$$

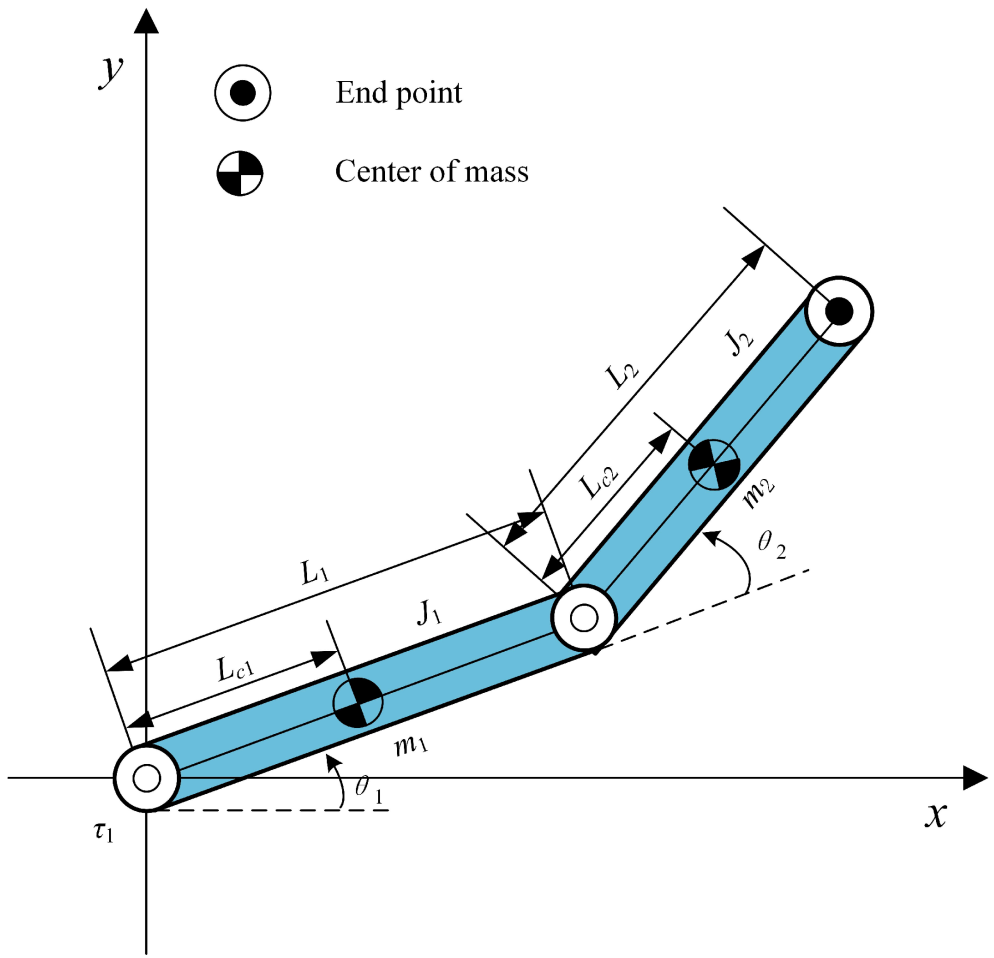
So the manipulator is singular when

$$\sin q_2 = 0 \Longrightarrow q_2 = 0,\ \pi$$

These are the configurations where the two links are collinear:

- fully stretched
- fully folded back

At those configurations, the end-effector loses one instantaneous direction of motion.



Jacobian from the cross-product formula

For a revolute joint in planar motion, the linear velocity contribution is $v_i = z \times (p_e - p_i)\dot{q}_i$

where $z = [0, 0, 1]^T$, and p_i is the joint position.

- Joint 1 position: $p_0 = [0, 0, 0]^T$
- Joint 2 position: $p_1 = [l_1 c_1, l_1 s_1, 0]^T$
- End-effector: $p_e = [l_1 c_1 + l_2 c_{12}, l_1 s_1 + l_2 s_{12}, 0]^T$

$$J_{v1} = z \times p_e = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_1 c_1 + l_2 c_{12} \\ l_1 s_1 + l_2 s_{12} \\ 0 \end{bmatrix} = \begin{bmatrix} -(l_1 s_1 + l_2 s_{12}) \\ l_1 c_1 + l_2 c_{12} \\ 0 \end{bmatrix}$$

$$J_{v2} = z \times (p_e - p_1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_2 c_{12} \\ l_2 s_{12} \\ 0 \end{bmatrix} = \begin{bmatrix} -l_2 s_{12} \\ l_2 c_{12} \\ 0 \end{bmatrix}$$

So the translational Jacobian is exactly the same as above.

Jacobians for link centers of mass

These are useful for dynamics.

Let the COM distances from each joint be l_{c1} and l_{c2}.

Link 1 COM

$$p_{c1} = \begin{bmatrix} l_{c1}\cos q_1 \\ l_{c1}\sin q_1 \end{bmatrix} \qquad J_{c1} = \begin{bmatrix} -l_{c1}\sin q_1 & 0 \\ l_{c1}\cos q_1 & 0 \end{bmatrix}$$

Link 2 COM

$$p_{c2} = \begin{bmatrix} l_1\cos q_1 + l_{c2}\cos(q_1+q_2) \\ l_1\sin q_1 + l_{c2}\sin(q_1+q_2) \end{bmatrix} \qquad J_{c2} = \begin{bmatrix} -l_1\sin q_1 - l_{c2}\sin(q_1+q_2) & -l_{c2}\sin(q_1+q_2) \\ l_1\cos q_1 + l_{c2}\cos(q_1+q_2) & l_{c2}\cos(q_1+q_2) \end{bmatrix}$$

Summary: RR Jacobian

For the standard planar double pendulum / RR robot arm,

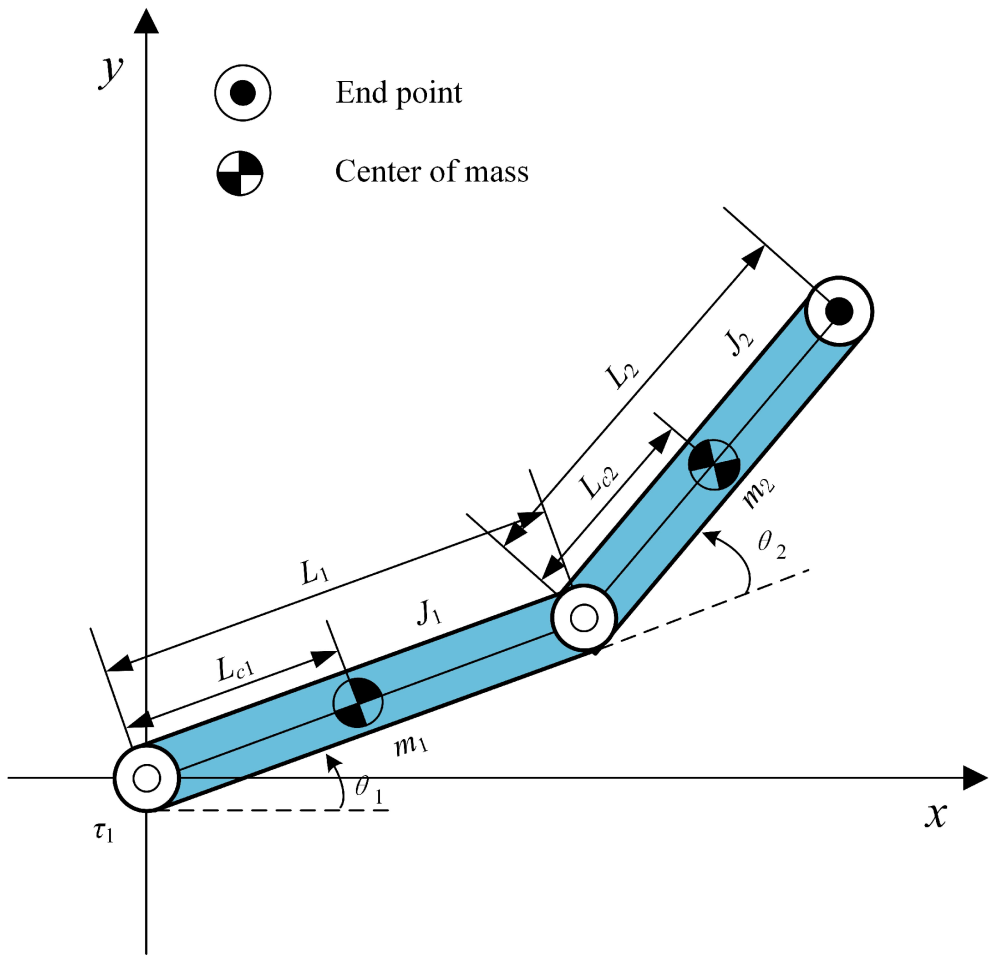
End-Effector Position p_e

$$p_e(q) = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \end{bmatrix}$$

and End-Effector Position

$$J(q) = \frac{\partial(x, y)}{\partial(q_1, q_2)} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} \end{bmatrix}$$

$$J(q) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix}$$



DH-parameter derivation in robotics notation

Using the standard DH convention

$$^{i-1}T_i = R_z(\theta_i)\, T_z(d_i)\, T_x(a_i)\, R_x(\alpha_i)$$

the homogeneous transform is

$$^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

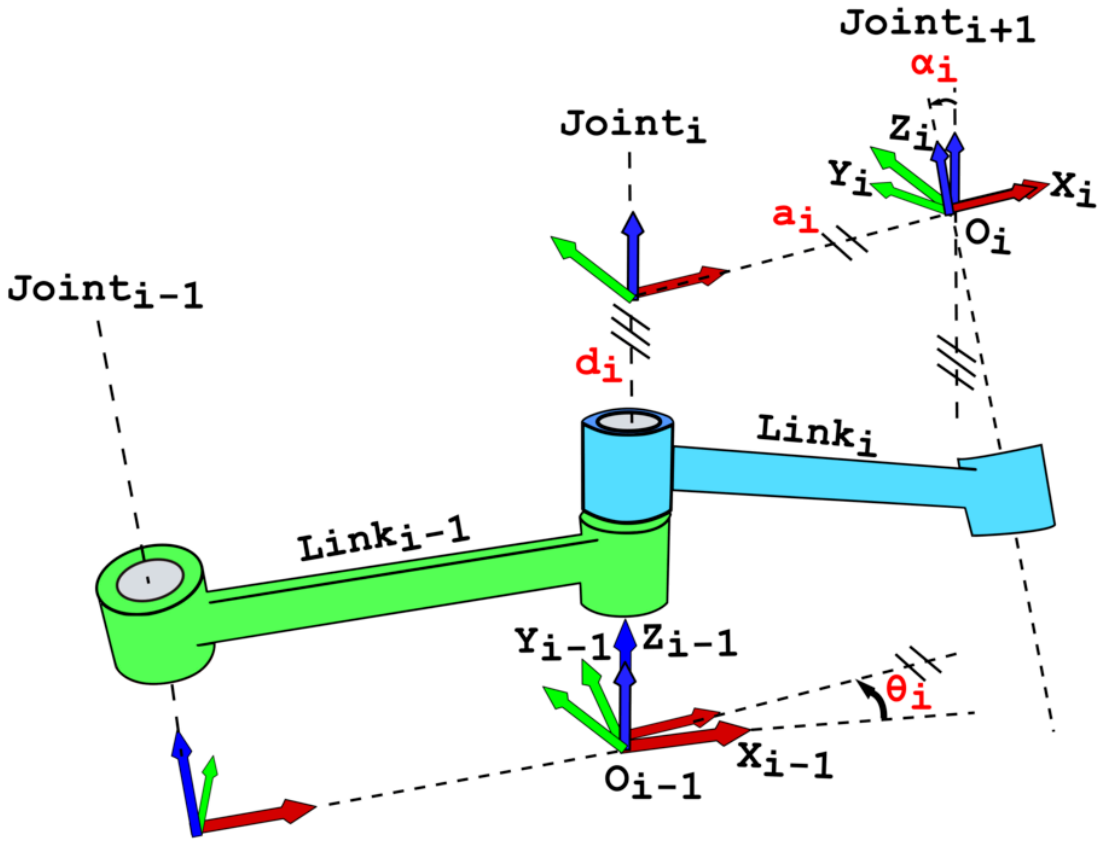
DH-parameter derivation in robotics notation

For a planar 2R manipulator:

$$\alpha_1 = \alpha_2 = 0, \qquad d_1 = d_2 = 0$$
$$a_1 = l_1, \qquad a_2 = l_2$$
$$\theta_1 = q_1, \qquad \theta_2 = q_2$$

So the DH table is

i	a_i	α_i	d_i	θ_i
1	l_1	0	0	q_1
2	l_2	0	0	q_2



DH-parameter Link transforms

Because $\alpha_i = 0$ and $d_i = 0$,

$$ {}^0T_1 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & l_1 \cos q_1 \\ \sin q_1 & \cos q_1 & 0 & l_1 \sin q_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1T_2 = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & l_2 \cos q_2 \\ \sin q_2 & \cos q_2 & 0 & l_2 \sin q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0T_2 = {}^0T_1 \, {}^1T_2. $$

$$ {}^0T_2 = \begin{bmatrix} \cos(q_1+q_2) & -\sin(q_1+q_2) & 0 & l_1 \cos q_1 + l_2 \cos(q_1+q_2) \\ \sin(q_1+q_2) & \cos(q_1+q_2) & 0 & l_1 \sin q_1 + l_2 \sin(q_1+q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

So the end-effector pose is

$$ x = l_1 \cos q_1 + l_2 \cos(q_1+q_2) \qquad y = l_1 \sin q_1 + l_2 \sin(q_1+q_2) \qquad \phi = q_1 + q_2 $$

Jacobian in strict geometric robotics notation

For revolute joints, the geometric Jacobian columns are

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1}) \qquad J_{\omega_i} = z_{i-1}$$

where o_i is the origin of frame i and z_{i-1} is the joint axis.

For this planar arm:

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad o_1 = \begin{bmatrix} l_1 \cos q_1 \\ l_1 \sin q_1 \\ 0 \end{bmatrix}, \qquad o_2 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ 0 \end{bmatrix}, \qquad z_0 = z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{v_1} = z_0 \times (o_2 - o_0) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ 0 \end{bmatrix}$$

$$J_{v_2} = z_1 \times (o_2 - o_1) = \begin{bmatrix} -l_2 \sin(q_1 + q_2) \\ l_2 \cos(q_1 + q_2) \\ 0 \end{bmatrix}$$

Jacobian in strict geometric robotics notation

Thus the full 6D geometric Jacobian is

$$J(q) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

The reduced planar Jacobian for (x, y, ϕ) is

$$J_p(q) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = J_p(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

Numerical Inverse Kinematics

Inverse Kinematics problems: given a desired end-effector pose x_d (in the operational space), we want to find the joint values q such that $\mathcal{FK}(\cdot) = x_d$. Here, $\mathcal{FK}(\cdot)$ is the forward kinematics. source

Defining operational space error between the desired x_d and the current end-effector pose x_e:

$$\dot{x}_e = J(q)\dot{q} \qquad\qquad e = x_d - x_e = x_d - \mathcal{FK}(q)$$

time derivative of the above operational space error

$$\dot{e} = \dot{x_d} - \dot{x_e} = \dot{x_d} - J(q)\dot{q}$$

This is called error dynamics, as it shows ODE of end-effector pose error $e(t)$ over time t. Here, $\dot{q}$ can be viewed as the control input. From a control perspective, we want to design a control law

$$\dot{q} = \text{controller}(x_d, \dot{x}_d, q, e)$$

which depends on all current information $(x_d, \dot{x}_d, q, e)$, such that, when plug the controller to the error dynamics, we have $e(t) \to 0$ as $t \to \inf$. At convergence, the resulting q is the IK solution!

Jacobian Inverse (Newton-Raphson) Method

We set

$$\dot{\boldsymbol{q}} = \text{controller}(\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \boldsymbol{q}, \boldsymbol{e})$$

as

$$\dot{\boldsymbol{q}} = \begin{cases} \boldsymbol{J}_A^{-1}(\boldsymbol{q})\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}\right) & \text{for non-redundant robot arms} \\ \boldsymbol{J}_A^{\dagger}\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}\right) + \left(\boldsymbol{I}_n - \boldsymbol{J}_A^{\dagger}\boldsymbol{J}_A\right)\dot{\boldsymbol{q}}_{\text{ref}} & \text{for redundant robot arms} \end{cases}$$

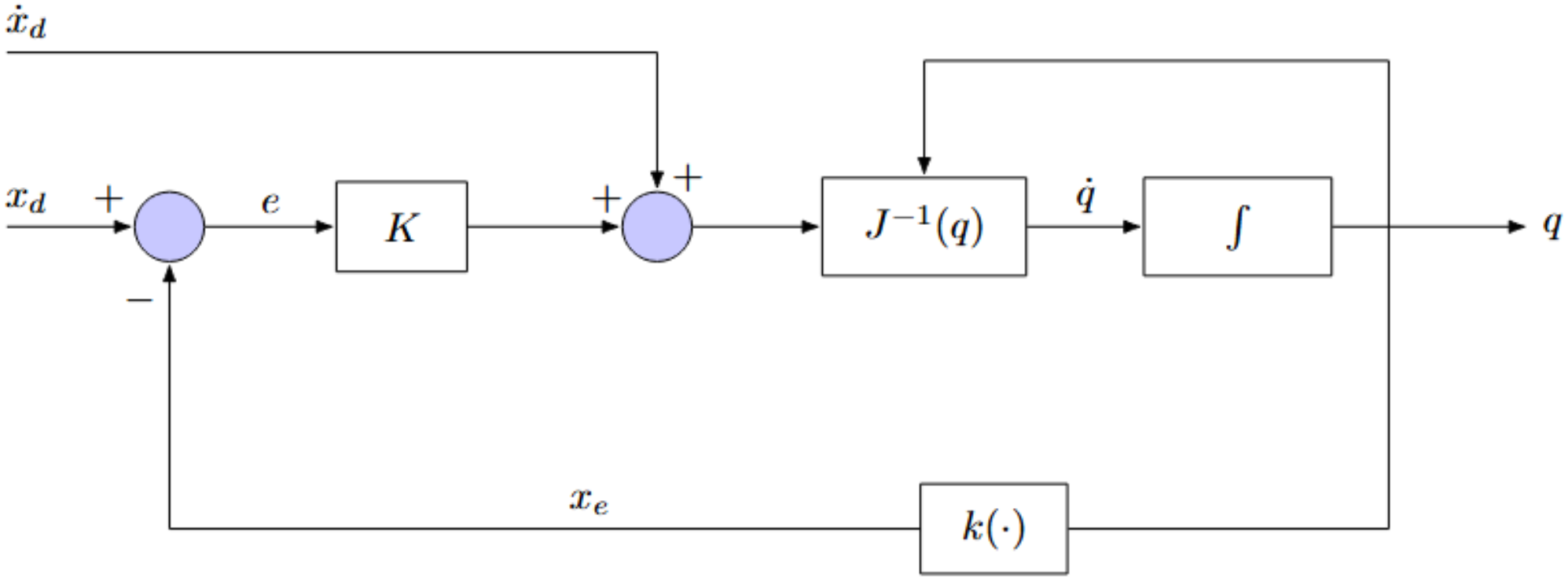
Here $\boldsymbol{K}$ is a positive definite (usually diagonal) matrix, and $\boldsymbol{q}_{\text{ref}}$ is any reference joint velocity that makes the robot move aways from the singularity. $\boldsymbol{J}_A^{\dagger}$ is the pseudo-inverse of Jacobian. For non-singular robot arm configurations, $\boldsymbol{J}_A^{\dagger} = \boldsymbol{J}_A^T\left(\boldsymbol{J}_A\boldsymbol{J}_A^T\right)^{-1}$.

To verify if the above controller can solve IK, we next examine if the error dynamics converges to 0. Submitting the above controllers into the error dynamics, we have

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = 0.$$

Jacobian Inverse (Newton-Raphson) Method

The block scheme for the above IK algorithm is shown below (non-redundant robot arm), where $\boldsymbol{k}(\cdot)$ means the forward kinematics.



Euler-Lagrangian dynamics

Below is the full derivation for a **planar RR robot**.

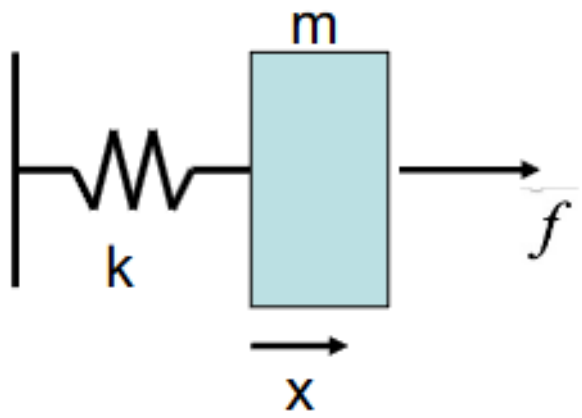
- q_1 : angle of link 1 measured from the inertial x -axis
- q_2 : relative angle of link 2 with respect to link 1
- absolute angle of link 2 is $q_1 + q_2$
- link lengths: l_1, l_2
- COM distances from each joint: r_1, r_2
- masses: m_1, m_2
- planar link inertias about each COM: I_1, I_2
- gravity acts in the negative y -direction, so potential is $V = mgy$

Euler-Lagrange of mass-spring system

To obtain the Euler-Lagrange Approach to Manipulator Dynamics for Serial Mechanisms.

First, consider the analogy of some familiar linear 2^{nd} order systems:

Mass-spring system:



$$KE = T = \frac{1}{2}m\dot{x}^2$$

$$PE = V = \frac{1}{2}kx^2$$

$$L = T - V = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2$$

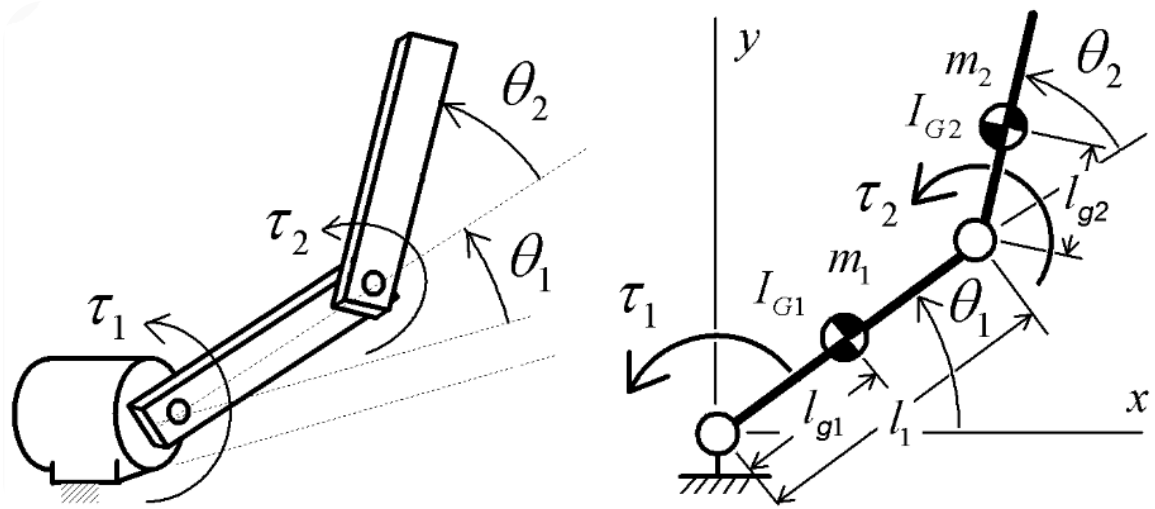
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = \sum f_{ext} = f$$

$$\frac{d}{dt}(m\dot{x}) - (-kx) = f$$

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

Lagrangian dynamics

Euler-Lagrange Approach to Manipulator Dynamics for Serial Mechanisms



(Asaji Sato, 2007)

Now we derive the dynamics using

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \qquad \text{with} \quad i = 1, 2,$$

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q)$$

COM positions

For link 1 COM:

$$p_{c1} = \begin{bmatrix} r_1 \cos q_1 \\ r_1 \sin q_1 \end{bmatrix}$$

For link 2 COM:

$$p_{c2} = \begin{bmatrix} l_1 \cos q_1 + r_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + r_2 \sin(q_1 + q_2) \end{bmatrix}$$

Lagrangian dynamics - COM velocities

$$\dot{p}_{c1} = \begin{bmatrix} -r_1 \sin q_1 \, \dot{q}_1 \\ r_1 \cos q_1 \, \dot{q}_1 \end{bmatrix} \qquad \|\dot{p}_{c1}\|^2 = r_1^2 \dot{q}_1^2$$

For link 2:

$$\dot{p}_{c2} = \begin{bmatrix} -l_1 \sin q_1 \, \dot{q}_1 - r_2 \sin(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \\ l_1 \cos q_1 \, \dot{q}_1 + r_2 \cos(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \end{bmatrix}$$

and after simplification,

$$\|\dot{p}_{c2}\|^2 = l_1^2 \dot{q}_1^2 + r_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2 l_1 r_2 \cos q_2 \, \dot{q}_1 (\dot{q}_1 + \dot{q}_2)$$

Expanding,

$$\|\dot{p}_{c2}\|^2 = (l_1^2 + r_2^2 + 2 l_1 r_2 \cos q_2) \dot{q}_1^2 + r_2^2 \dot{q}_2^2 + 2(r_2^2 + l_1 r_2 \cos q_2) \dot{q}_1 \dot{q}_2$$

Angular velocities:

$$\omega_1 = \dot{q}_1, \qquad \omega_2 = \dot{q}_1 + \dot{q}_2$$

Kinetic Energy

$$T = \frac{1}{2} m_1 \|\dot{p}_{c1}\|^2 + \frac{1}{2} I_1 \dot{q}_1^2 + \frac{1}{2} m_2 \|\dot{p}_{c2}\|^2 + \frac{1}{2} I_2 (\dot{q}_1 + \dot{q}_2)^2$$

Substitute the velocity expressions:

$$T = \frac{1}{2}(I_1 + m_1 r_1^2)\dot{q}_1^2 + \frac{1}{2} m_2 \Big[(l_1^2 + r_2^2 + 2 l_1 r_2 \cos q_2)\dot{q}_1^2 + r_2^2 \dot{q}_2^2 + 2(r_2^2 + l_1 r_2 \cos q_2)\dot{q}_1 \dot{q}_2 \Big] + \frac{1}{2} I_2 (\dot{q}_1 + \dot{q}_2)^2$$

It is convenient to define

$$a = I_1 + I_2 + m_1 r_1^2 + m_2 (l_1^2 + r_2^2)$$

$$b = m_2 l_1 r_2$$

$$d = I_2 + m_2 r_2^2$$

Then

$$T = \frac{1}{2}(a + 2b \cos q_2)\dot{q}_1^2 + (d + b \cos q_2)\dot{q}_1 \dot{q}_2 + \frac{1}{2} d\, \dot{q}_2^2$$

Potential Energy

With y upward,

$$V = m_1 g y_{c1} + m_2 g y_{c2}$$

So

$$V = m_1 g r_1 \sin q_1 + m_2 g \big(l_1 \sin q_1 + r_2 \sin(q_1 + q_2) \big)$$

Hence

$$V = (m_1 r_1 + m_2 l_1) g \sin q_1 + m_2 r_2 g \sin(q_1 + q_2)$$

Therefore

$$L = T - V$$

First equation

$$\frac{\partial L}{\partial \dot{q}_1} = (a + 2b\cos q_2)\dot{q}_1 + (d + b\cos q_2)\dot{q}_2$$

Differentiate:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right) = (a + 2b\cos q_2)\ddot{q}_1 + (d + b\cos q_2)\ddot{q}_2 - 2b\sin q_2\,\dot{q}_1\dot{q}_2 - b\sin q_2\,\dot{q}_2^2$$

Also,

$$\frac{\partial L}{\partial q_1} = -\frac{\partial V}{\partial q_1} = -(m_1 r_1 + m_2 l_1)g\cos q_1 - m_2 r_2 g\cos(q_1 + q_2)$$

So the first joint equation is

$$(a + 2b\cos q_2)\ddot{q}_1 + (d + b\cos q_2)\ddot{q}_2 - 2b\sin q_2\,\dot{q}_1\dot{q}_2 - b\sin q_2\,\dot{q}_2^2 + (m_1 r_1 + m_2 l_1)g\cos q_1 + m_2 r_2 g\cos(q_1 + q_2) = \tau_1$$

Second equation

$$\frac{\partial L}{\partial \dot{q}_2} = (d + b\cos q_2)\dot{q}_1 + d\dot{q}_2$$

Differentiate:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right) = (d + b\cos q_2)\ddot{q}_1 + d\ddot{q}_2 - b\sin q_2\, \dot{q}_1\dot{q}_2$$

Also,

$$\frac{\partial L}{\partial q_2} = -b\sin q_2\, \dot{q}_1^2 - b\sin q_2\, \dot{q}_1\dot{q}_2 - m_2 r_2 g\cos(q_1 + q_2)$$

So

$$(d + b\cos q_2)\ddot{q}_1 + d\ddot{q}_2 + b\sin q_2\, \dot{q}_1^2 + m_2 r_2 g\cos(q_1 + q_2) = \tau_2$$

Standard dynamic model of a robotic manipulator

We can represent in the manuipulator form

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

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

In compact a, b, d notation

$$D(q) = \begin{bmatrix} a + 2b\cos q_2 & d + b\cos q_2 \\ d + b\cos q_2 & d \end{bmatrix}$$

A valid Coriolis matrix is

$$C(q,\dot{q}) = \begin{bmatrix} -b\sin q_2\, \dot{q}_2 & -b\sin q_2 (\dot{q}_1 + \dot{q}_2) \\ b\sin q_2\, \dot{q}_1 & 0 \end{bmatrix}$$

Standard dynamic model of a robotic manipulator

with

$$C(q,\dot{q})\dot{q} = \begin{bmatrix} -2b\sin q_2\,\dot{q}_1\dot{q}_2 - b\sin q_2\,\dot{q}_2^2 \\ b\sin q_2\,\dot{q}_1^2 \end{bmatrix}$$

The gravity matrix is

$$G(q) = \begin{bmatrix} (m_1 r_1 + m_2 l_1) g\cos q_1 + m_2 r_2 g\cos(q_1+q_2) \\ m_2 r_2 g\cos(q_1+q_2) \end{bmatrix}$$

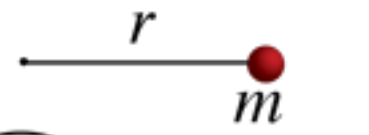
$D(q)$ directly in physical parameters, the Fully expanded mass matrix:

$$D(q) = \begin{bmatrix} I_1 + I_2 + m_1 r_1^2 + m_2(l_1^2 + r_2^2 + 2 l_1 r_2 \cos q_2) & I_2 + m_2(r_2^2 + l_1 r_2 \cos q_2) \\ I_2 + m_2(r_2^2 + l_1 r_2 \cos q_2) & I_2 + m_2 r_2^2 \end{bmatrix}$$

where I_i means the moment of inertia of link i about its own COM.

Moment of Inertia Examples

Moment of inertia is defined with respect to a specific rotation axis. The moment of inertia of a point mass with respect to an axis is defined as the product of the mass times the distance from the axis squared. source

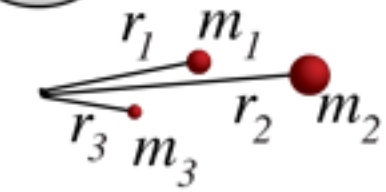


$$I = mr^2$$

For a point mass the moment of inertia is just the mass times the radius from the axis squared. For a collection of point masses (below) the moment of inertia is just the sum for the masses.

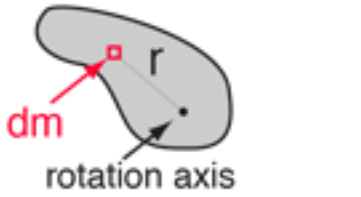
$$I = kmr^2$$

For an object with an axis of symmetry, the moment of inertia is some fraction of that which it would have if all the mass were at the radius r.



$$I = \sum_i m_i r_i^2 = m_1 r_1^2 + m_2 r_2^2 + m_3 r_3^2 + \ldots$$

Sum of the point mass moments of inertia.

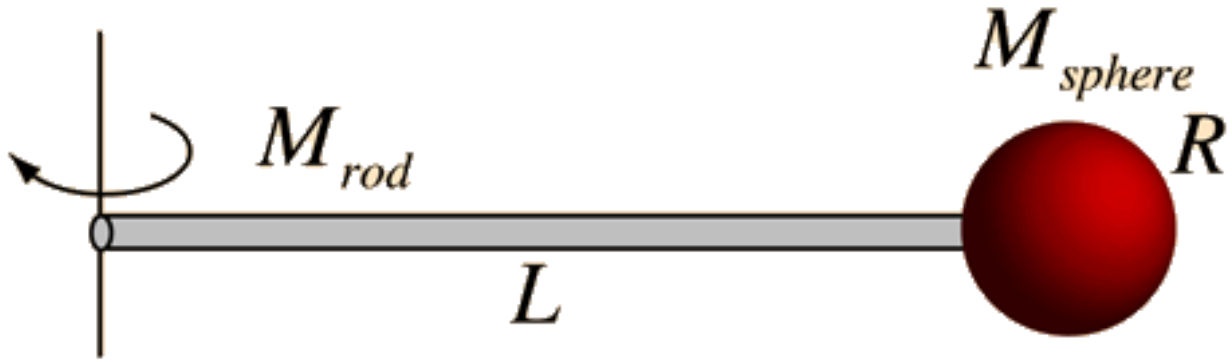


$$I = \int_0^M r^2 dm$$

Continuous mass distributions require an infinite sum of all the point mass moments which make up the whole. This is accomplished by an integration over all the mass.

Superposition of Moments of Inertia

The moment of inertia of a composite object can be obtained by superpositon of the moments of its constituent parts. The Parallel axis theorem is an important part of this process. For example, a spherical ball on the end of a rod: source:

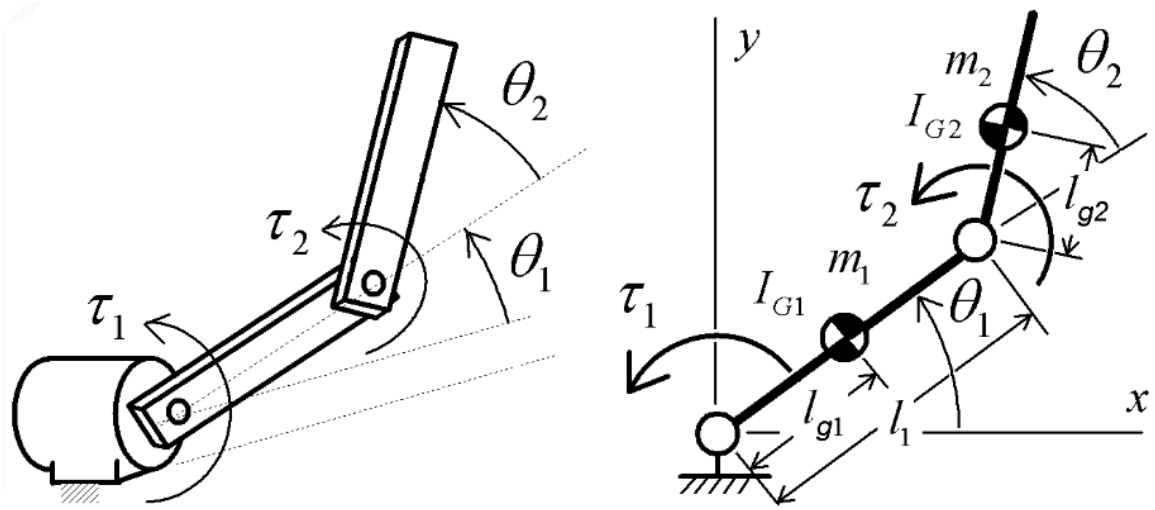


$$I = \frac{1}{3} M_{rod} L^2 + \frac{2}{5} M_{sphere} R^2 + M_{sphere} (L + R)^2$$

$$I = I_{\text{rod about end}} + I_{\text{sphere about center}} + \text{Parallel axis contribution}$$

E-L Derivation for Planar RR (point mass case)

Euler-Lagrange Approach to Manipulator Dynamics for Serial Mechanisms



To simplify our calculation, for geared serial link robotic arm, we can assume mass is concentrated at the joint (joint motor). We then have a massless rods and endpoint point mass model:

- link 1 mass m_1 is concentrated at the end of link 1; link 2 mass m_2 is concentrated at the end of link 2, so
  $$r_1 = l_1, \qquad r_2 = l_2$$
- and since each link mass is treated as a point mass, all the mass is concentrated at one point, its inertia about an axis through itself is
  $$I_1 = 0, \qquad I_2 = 0.$$

E-L Derivation for Planar RR (point mass case)

Mass m_1 is at the end of link 1; Mass m_2 is at the end of link 2:

$$p_1 = \begin{bmatrix} l_1 \cos q_1 \\ l_1 \sin q_1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \end{bmatrix}$$

Differentiate p_1 and p_2:

$$\dot{p}_1 = \begin{bmatrix} -l_1 \sin q_1, \dot{q}_1 \\ l_1 \cos q_1, \dot{q}_1 \end{bmatrix}, \qquad \dot{p}_2 = \begin{bmatrix} -l_1 \sin q_1, \dot{q}_1 - l_2 \sin(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \\ l_1 \cos q_1, \dot{q}_1 + l_2 \cos(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \end{bmatrix}$$

We have velocity of each joint:

$$v_1^2 = l_1^2 \dot{q}_1^2, \qquad v_2^2 = l_1^2 \dot{q}_1^2 + l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2 l_1 l_2 \cos q_2, \dot{q}_1 (\dot{q}_1 + \dot{q}_2)$$

Expanding:

$$v_2^2 = (l_1^2 + l_2^2 + 2 l_1 l_2 \cos q_2) \dot{q}_1^2 + 2(l_2^2 + l_1 l_2 \cos q_2) \dot{q}_1 \dot{q}_2 + l_2^2 \dot{q}_2^2$$

E-L Derivation for Planar RR (point mass case)

Because the masses are point masses, there is only translational kinetic energy:

$$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2$$

Substitute the expressions above:

$$T = \frac{1}{2} m_1 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 \left[(l_1^2 + l_2^2 + 2 l_1 l_2 \cos q_2) \dot{q}_1^2 + 2(l_2^2 + l_1 l_2 \cos q_2) \dot{q}_1 \dot{q}_2 + l_2^2 \dot{q}_2^2 \right]$$

$$T = \frac{1}{2} \left[m_1 l_1^2 + m_2 (l_1^2 + l_2^2 + 2 l_1 l_2 \cos q_2) \right] \dot{q}_1^2 + \left[m_2 (l_2^2 + l_1 l_2 \cos q_2) \right] \dot{q}_1 \dot{q}_2 + \frac{1}{2} (m_2 l_2^2) \dot{q}_2^2$$

$$T = \frac{1}{2} \begin{bmatrix} \dot{q}_1 & \dot{q}_2 \end{bmatrix} D(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$D(q_1, q_2) = \begin{bmatrix} m_1 l_1^2 + m_2 (l_1^2 + l_2^2 + 2 l_1 l_2 \cos q_2) & m_2 (l_2^2 + l_1 l_2 \cos q_2) \\ m_2 (l_2^2 + l_1 l_2 \cos q_2) & m_2 l_2^2 \end{bmatrix}$$

Assume the usual dynamic robot model (we will ignore environment forces unless otherwise noted):

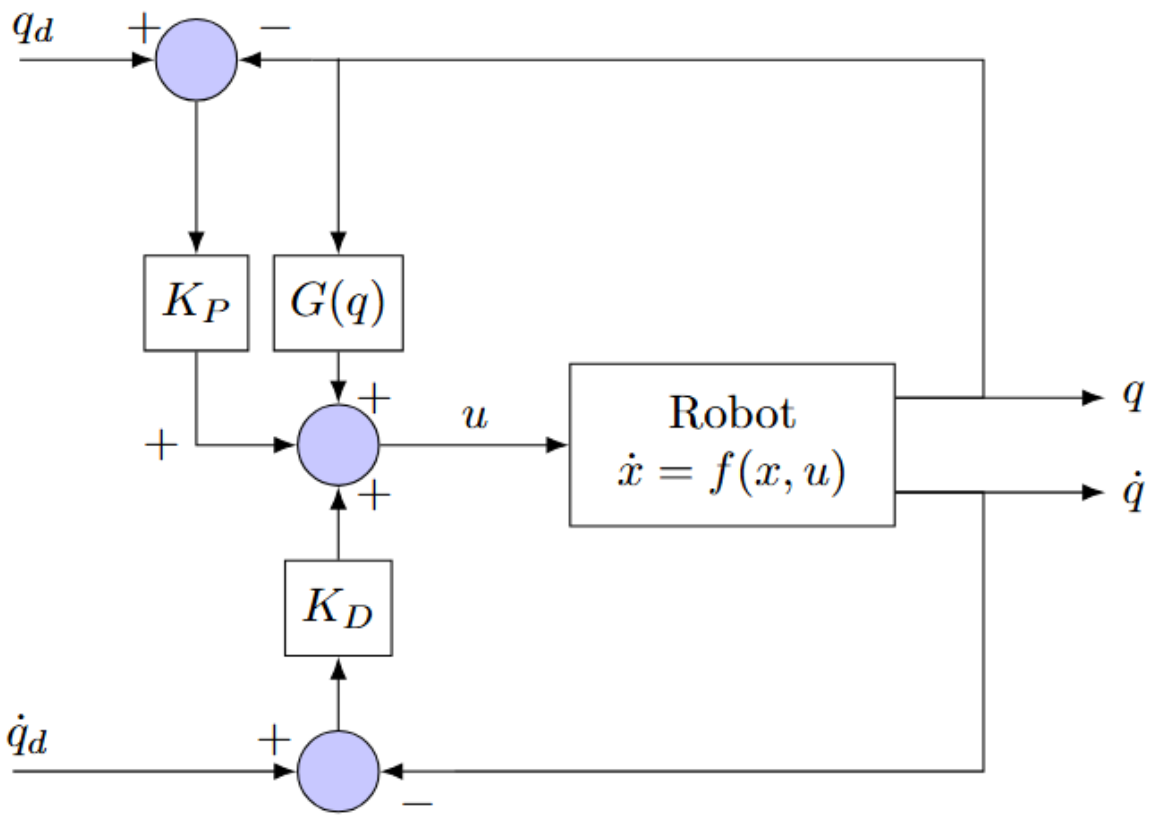
$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u$$

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ -D(q)^{-1}\left[C(q,\dot{q})\dot{q} + G(q)\right] \end{bmatrix} + \begin{bmatrix} 0 \\ D(q)^{-1}u \end{bmatrix}$$

We have our state x representing the joint status q and $\dot{q}$,
and it is essentially a function of the state x and control u.

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \qquad \dot{x} \triangleq f(x,u)$$

PD + Gravity Set-Point Control



Simple controller, instead of canceling all non-linearities, it only compensates gravity and adds a PD term. Controller for PD control with gravity set point follows:

$$\boldsymbol{u} = \underbrace{\boldsymbol{G}(\boldsymbol{q})}_{\text{gravitational compensation}} + \underbrace{\boldsymbol{K}_P(\boldsymbol{q}_d - \boldsymbol{q})}_{\text{proportional control}} \underbrace{+\boldsymbol{K}_D(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}})}_{\text{derivative control}}$$

with $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ are positive definite matrices.

PD + Gravity Set-Point Control

To obtain the closed-loop dynamics under PD + gravity setpoint, substitute control u into dynamics

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

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + G(q).$$

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} = K_p e + K_d \dot{e}.$$

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + K_d(\dot{q} - \dot{q}_d) + K_p(q - q_d) = 0$$

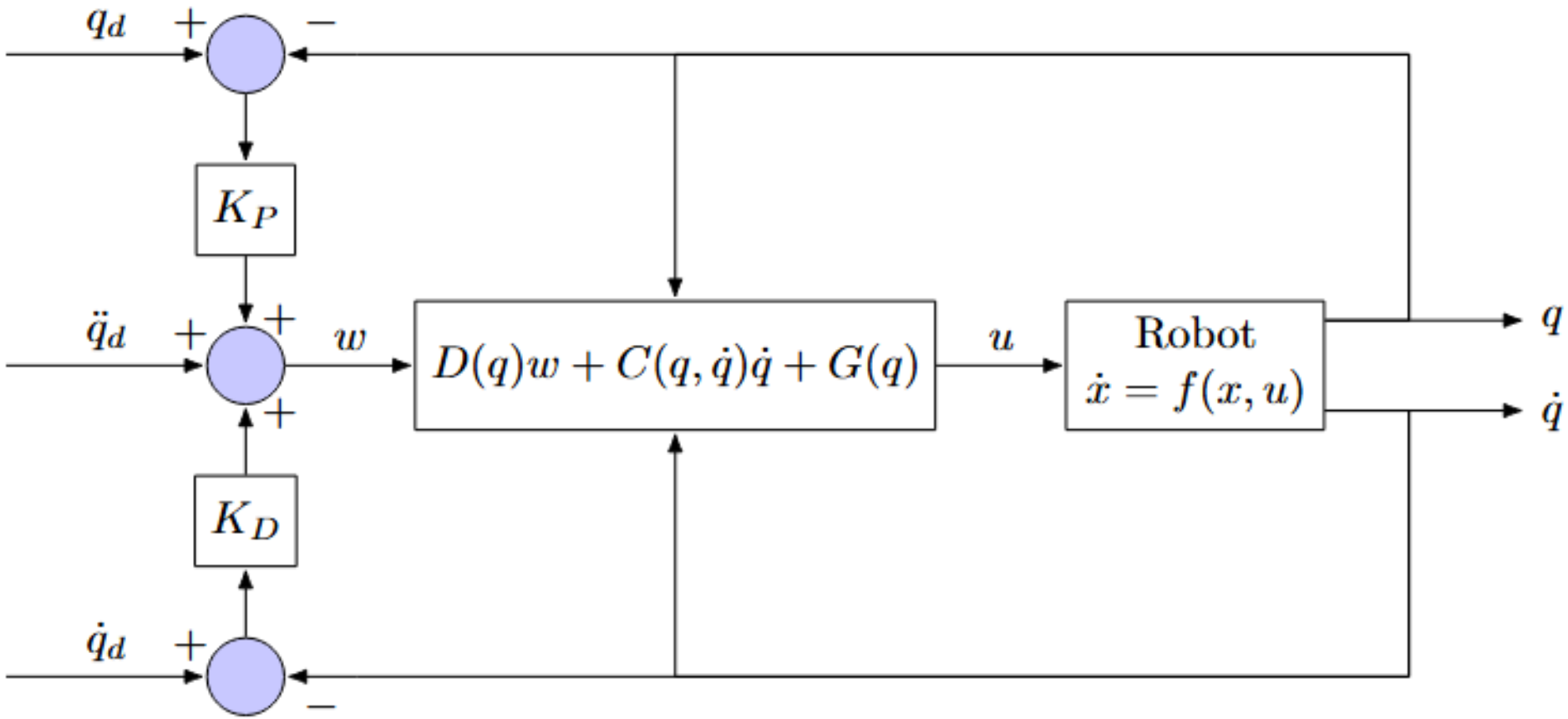
or in terms of error for regulation ($\dot{q}_d = 0$, constant q_d), known as gravity compensation

$$D(q)\ddot{e} + C(q,\dot{q})\dot{e} + K_d\dot{e} + K_p e = 0$$

up to sign convention depending on whether you define $e = q_d - q$ or $e = q - q_d$. The key point is (a) gravity is removed, and (b) inertia and Coriolis coupling remain. So the system is not exactly linearized.

Computed-Torque Joint Position Control

To determine the control inputs to track this desired trajectory based on measurements of the joint positions and velocities, we might transform the problem into one of determining another input w to the control input u. It is also known as inverse dynamics control.



Computed-Torque Joint Position Control

One limitation of the PD Control with Gravity Compensation is that the controller can only follow a stationary desired $\boldsymbol{q}_d$, not good at tracking a fast-changing desired joint. source

$$\dot{\boldsymbol{q}}_d(t) \neq \boldsymbol{0} \quad \text{and} \quad \ddot{\boldsymbol{q}}_d(t) \neq \boldsymbol{0}$$

Inverse dynamics control is to address the above limitation. The idea of inverse dynamics control is to find a controller that can make the robot arm behave like a mass-spring-damper system. Controller for inverse dynamics control follows,

$$\boldsymbol{u} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{w} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q})$$

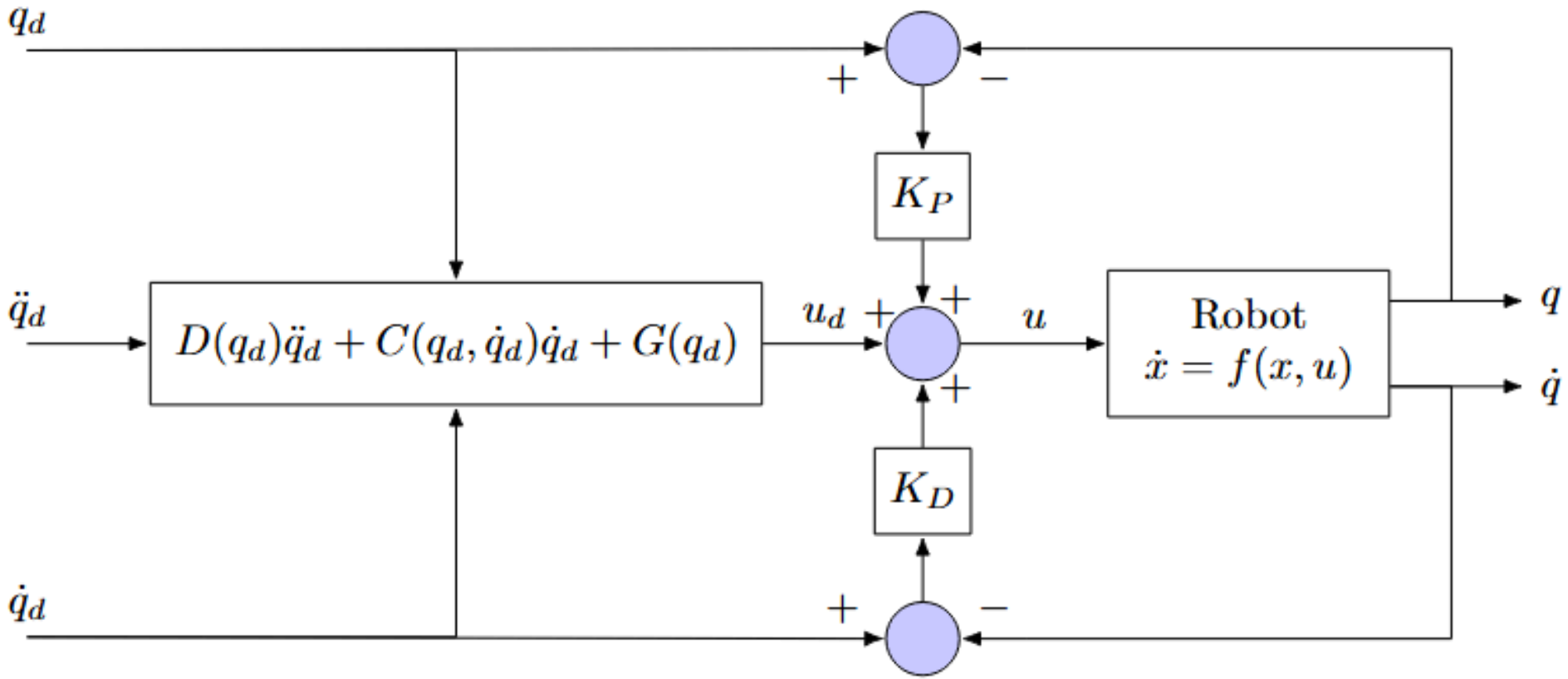
$$\boldsymbol{w} = \ddot{\boldsymbol{q}}_d + \boldsymbol{K}_P(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_D(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}})$$

$$\boldsymbol{K}_P = \text{diag}\left\{\omega_1^2, \ldots, \omega_n^2\right\} \quad \boldsymbol{K}_D = \text{diag}\left\{2\zeta_1\omega_1, \ldots, 2\zeta_n\omega_n\right\}$$

with ζ_i the damping ratio and ω_i the natural frequency for joint i.

Feedforward Position Control

If the nominal desired trajectory is known ahead of time, a path planner can be used to determine the nominal control inputs. State feedback control corrections only need to be made for deviations from the nominal trajectory.



Feedforward Position Control

The control of such feedforward + PD position feedback controller:

$$\tau = D(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + G(q_d) + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$$

Substitute into the true dynamics:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = D(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + G(q_d) + K_p e + K_d \dot{e}$$

Rearranging,

$$D(q)\ddot{q} = D(q_d)\ddot{q}_d + \big(C(q_d, \dot{q}_d)\dot{q}_d - C(q, \dot{q})\dot{q}\big) + \big(G(q_d) - G(q)\big) + K_p e + K_d \dot{e}$$

This is not perfectly linear in the error, because the true and desired dynamics are evaluated at different states. But if $q \approx q_d$ and $\dot{q} \approx \dot{q}_d$, the mismatch terms are small, and the PD terms pull the trajectory back.

Feedforward Position Control

What if I use only the feedforward part? It should match the exact torque required?

$$\tau = \tau_{ff}$$

That is,

$$\tau = D(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + G(q_d)$$

This is usually not robust in practice because:

- model mismatch
- unknown friction
- disturbances
- initial condition error
- numerical tracking error

Even tiny mismatch can cause drift. So pure feedforward alone is rarely enough.

Feedforward + PD vs Inverse Dynamics Control

Feedforward + PD

$$\tau = D(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + G(q_d) + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$$

Computed torque

$$\tau = D(q)\left[\ddot{q}_d + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q)\right] + C(q, \dot{q})\dot{q} + G(q)$$

Difference:

- feedforward + PD uses **desired-state model terms**
- computed torque uses **actual-state model terms**

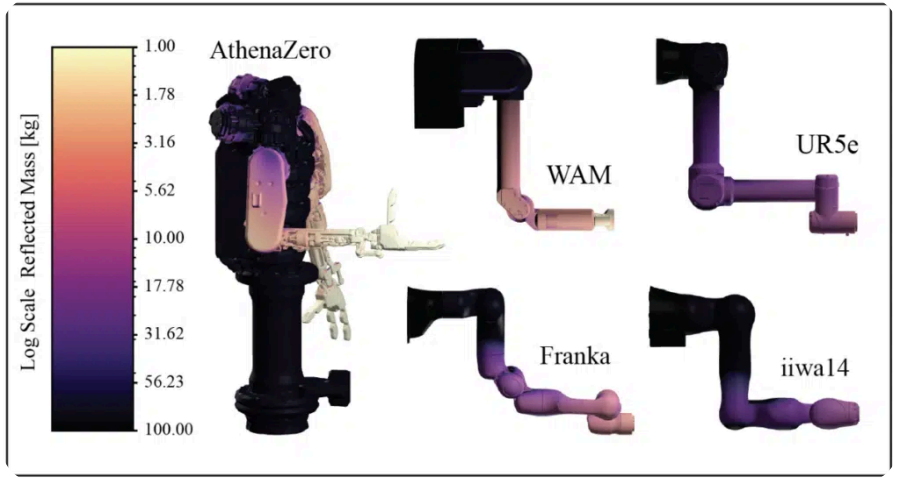
Computed torque gives exact cancellation in the ideal model case.
Feedforward + PD feedback apply the torque that should be needed for the desired motion, and correct the error by feedback loops to yield a faster tracking.

Why Dynamics Matters?

Latest research from Robotics and AI Institute (Boston Dynamics spin off)

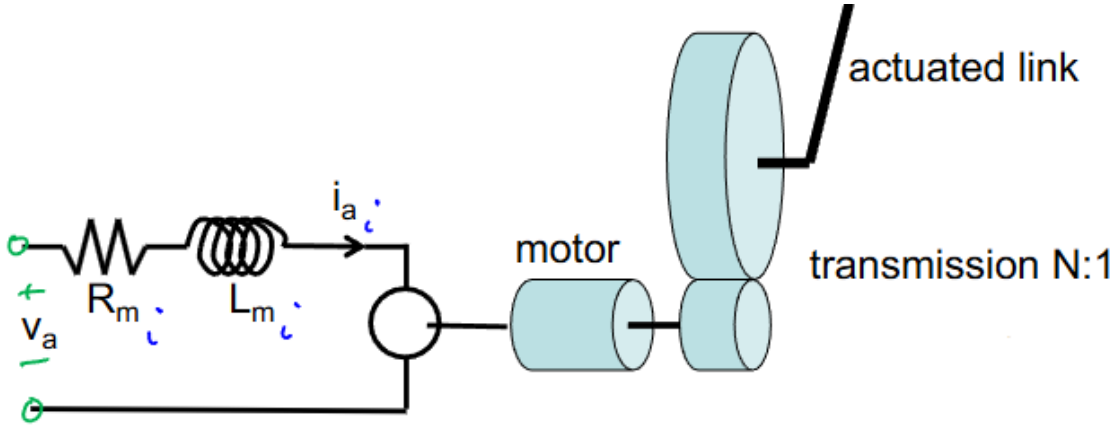
AthenaZero : A Bimanual Robot for Dynamic Manipulation Video

"AthenaZero's *low-inertia design* requires control approaches that differ from those used with traditional manipulators. The system's dynamical behavior requires *careful attention to torque control* and *trajectory acceleration components for feedforward dynamics* that go well beyond kinematic planning or feedforward position control." ... "As the actuator's reflected inertia, a key ingredient of the *effective mass, scales by the square of the gear ratio*, we kept these values low, using *5:1 in most joints*. We also made use of efficient planetary gearsets (97% or better)."



Why Dynamics (Sometimes) Doesn't Matters?

Considering this geared robot dynamic model



The relationship between motor input angle θ_{mi}, motor torque u_{mi} and joint output angle θ_i, joint output torque u_i

$$\theta_{mi} = N\theta_i \qquad \dot{\theta}_{mi} = N\dot{\theta}_i$$

$$u_{mi} = \frac{1}{N}u_i$$

Considering back emf $K_{mi}\dot{\theta}_{mi}$, friction in the bearing $B_i\dot{\theta}_{mi}$, reflected motor dynamics $1/Nu$

$$L_{mi}\frac{di_{ai}}{dt} + R_{mi}i_{ai} = v_{ai} - K_{mi}\dot{\theta}_{mi}$$

$$J_{mi}\ddot{\theta}_{mi} = u_{mi} - \frac{1}{N_i}u_i$$

$$u_{mi} = -B_i\dot{\theta}_{mi} + K_{Ti}i_{ai}$$

$$i_{ai} = \frac{1}{R_{mi}}v_{ai} - \frac{K_{mi}}{R_{mi}}\dot{\theta}_{mi}$$

Applied motor torque

$$u_{mi} = -\left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)\dot{\theta}_{mi} + \frac{K_{Ti}}{R_{mi}}v_{ai}$$

Geared robot dynamic model

$$J_{mi}\ddot{\theta}_{mi} = u_{mi} - \frac{1}{N_i}u_i$$

$$u_{mi} = -\left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)\dot{\theta}_{mi} + \frac{K_{Ti}}{R_{mi}}v_{ai}$$

$$\implies J_{mi}\ddot{\theta}_{mi} + \left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)\dot{\theta}_{mi} + \frac{1}{N_i}u_i = \frac{K_{Ti}}{R_{mi}}v_{ai}$$

where $J_{mi}\ddot{\theta}_{mi}$ is the motor rotor inertia and acceleration,
B_i is the mechanical damping, $\frac{K_{Ti}K_{mi}}{R_{mi}}$ is the electrical damping.
Substituting back $\theta_{mi} = N_i\theta_i$,

$$J_{mi}N_i\ddot{\theta}_i + \left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)N_i\dot{\theta}_i + \frac{1}{N_i}u_i = \frac{K_{Ti}}{R_{mi}}v_{ai}$$

Geared robot dynamic model

$$J_{mi}N_i\ddot{\theta}_i + \left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)N_i\dot{\theta}_i + \frac{1}{N_i}u_i = \frac{K_{Ti}}{R_{mi}}v_{ai}$$

$$J_m = \begin{bmatrix} J_{m1} & 0 & \cdots & 0 \\ 0 & J_{m2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & J_{mn} \end{bmatrix} \quad N = \begin{bmatrix} N_1 & 0 & \cdots & 0 \\ 0 & N_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & N_n \end{bmatrix} \quad u_m = \begin{bmatrix} \frac{K_{T1}}{R_{m1}}v_{a1} \\ \vdots \\ \vdots \\ \frac{K_{Tn}}{R_{mn}}v_{an} \end{bmatrix}$$

$$B_m = \begin{bmatrix} \left(B_1 + \frac{K_{T1}K_{m1}}{R_{m1}}\right) & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & \left(B_n + \frac{K_{Tn}K_{mn}}{R_{mn}}\right) \end{bmatrix}$$

Geared robot dynamic model

$$J_{mi}N_i\ddot{\theta}_i + \left(B_i + \frac{K_{Ti}K_{mi}}{R_{mi}}\right)N_i\dot{\theta}_i + \frac{1}{N_i}u_i = \frac{K_{Ti}}{R_{mi}}v_{ai} \Longrightarrow NJ_m\ddot{q} + NB_m\dot{q} + N^{-1}u = u_m$$

Subsituting joint torque $u = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)$

$$\left(N^{-1}D(q) + NJ_m\right)\ddot{q} + \left(N^{-1}C(q,\dot{q}) + NB_m\right)\dot{q} + N^{-1}G(q) = u_m$$

In highly geared case, N is large and $1/N$ is relatively small, it reduces to

$$(NJ_m)\ddot{q} + (NB_m)\dot{q} = u_m$$

This behaves similar to a mass-damper system $m\ddot{x} + b\dot{x} = u$

PD controller is often sufficient to control.

Reference

[1] UBC EECE589 - Introduction to Robotics by Prof. Tim Salcudean. course website

[2] Tutorial on Robotic Manipulator Control by Prof. Tan-Bin Jia, Iowa State University notes

[3] ASU MAE 547 - Modeling and Control of Robotics (MAE 547) from Irislab

[4] hyperphysics - moment of Intrtia link

[5] MIT865.24 Tutorial on Model free control link