Lecture 1, Sep 3, 2025

- The 5 basic problems of robotics:
  1. Forward kinematics: find the pose of the end effector given joint states
  2. Inverse kinematics: given a desired end effector pose, find the joint states
     - Solution is often not unique (underdetermined problem)
  3. Dynamics: Derive a mathematical model for simulation and control design
     - We will arrive at a canonical model
  4. Motion planning: given a target pose, plan a path for the end effector (joint states as a function of time) so that it reaches the goal while avoiding collisions
  5. Control: design feedback controllers to execute the planned motions (reference signals) with accuracy, speed and robustness

Lecture 2, Sep 5, 2025

Robot Manipulators Basics

- Manipulators are represented by a collection of rigid links connected by joints, which can be revolute (R, rotation around an axis) or prismatic (P, translation along an axis)
  - A series of links and joints forms a *kinematic chain*
- A spherical wrist is composed of 3 revolute joints (this is the case on the lab KUKA robots)
  - Typically all 3 degrees of freedom are consolidated into a single mechanical part, but expanded out in mathematical diagrams

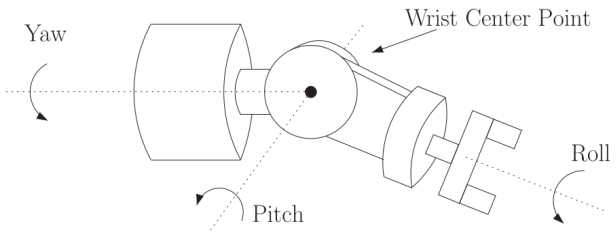

Figure 1: Spherical wrist diagram.

- The *topology* of a manipulator describes how links are connected; examples include serial chain (no loops), closed chain, or tree structure
  - Serial chains are much easier to control since we have a single fixed base, so we do not need to account for complex coupling like in closed chains
  - This course focuses on serial chains, the simplest topology
- Some popular serial chain robots:
  - Articulated manipulator: RRR, typically with wrist for 6 degrees of freedom
  - Spherical manipulator: RRP (named so because it maps out spherical coordinates)
  - SCARA: RRP, but unlike the spherical manipulator all 3 axes are parallel; typically used for pick-and-place tasks
- Our objective is to represent the pose of all the links (which leads to the pose of the end effector) as a function of the joint variables (rotation and translation)

Notation

- We fix the world coordinate frame 0, with origin (*base point*) O_0 and right-hand coordinate axes (orthonormal basis) x_0, y_0, y_0
  - Frame 0 is typically attached at base of the robot
- A superscript is used to denote the reference frame: p^0 denotes a point p in frame 0 and v^0 denotes a vector

  - Note while points are defined as an offset with respect to a reference origin, vectors are just a direction and magnitude so they do not need a coordinate origin
  - $p^0 = O_0^0 + \begin{bmatrix} a \\ b \\ c \end{bmatrix} = O_0 + ax_0^0 + by_0^0 + cz_0^0$
  - $v^0 = \begin{bmatrix} d \\ e \\ f \end{bmatrix} = dx_0^0 + ey_0^0 + fz_0^0$
  - e.g. x_0^1 is the x unit vector for frame 0, expressed in frame 1
- Note by definition $O_i^i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, x_i^i = e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, y_i^i = e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, z_i^i = e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ for any frame i
- To express p^0 in frame 1, we can write $p^1 = O_0^1 + ax_0^1 + by_0^1 + cz_0^1$ and once we find the basis vector representations, the point can be computed

Lecture 3, Sep 8, 2025

Rotations

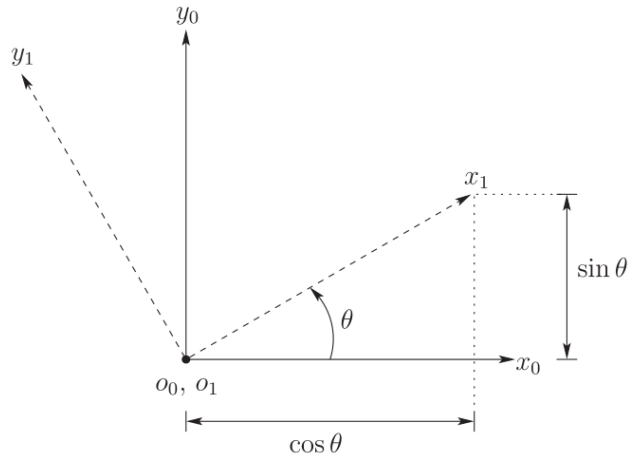

Figure 2: Coordinate frames 1 and 2.

- Consider 2 frames O_0, x_0, y_0 and O_1, x_1, y_1 in $\mathbb{R}^2$ where $O_0 = O_1$; the two frames are separated by an angle θ
- There are 2 ways to describe the relationship between the frames:
  1. Geometric: Specify the angle θ from x_0 to x_1
  2. Algebraic: Specify x_1, y_1 as expressed in frame 0, i.e. x_1^0, y_1^0
     - This is the preferred method that we will explore
- Let $R_1^0 = \begin{bmatrix} x_1^0 & y_1^0 \end{bmatrix}$ be the *rotation matrix* of **frame 1 with respect to frame 0**
  - Since the dot product represents a projection (when normalized), we can obtain the components of x_1^0 by taking the dot product $x_1 \cdot x_0$ and $x_1 \cdot y_0$ to project it onto the axes of frame 0
    * Recall $v_1 \cdot v_2 = \|v_1\|\|v_2\| \cos\theta$, so to actually compute the dot products we can use geometry
  - $x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix}, y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix}$
  - Therefore $R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix}$
- Generalizing to $\mathbb{R}^3$, $R_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix} = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$

– Notice $R_0^1 = \begin{bmatrix} x_0 \cdot x_1 & y_0 \cdot x_1 & z_0 \cdot x_1 \\ x_0 \cdot y_1 & y_0 \cdot y_1 & z_0 \cdot y_1 \\ x_0 \cdot z_1 & y_0 \cdot z_1 & z_0 \cdot z_1 \end{bmatrix} = (R_1^0)^T$

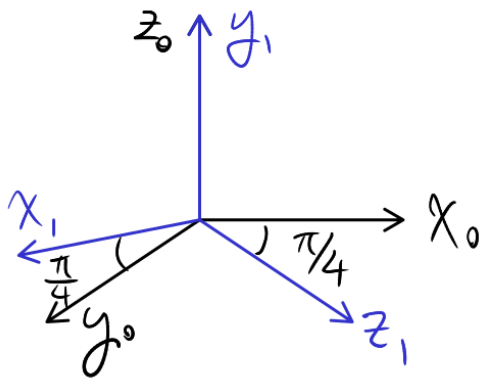

Figure 3: Coordinate frames for the example.

- Example: Consider the frames as arranged in the figure; find R_1^0
  - By geometry: $x_1^0 = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \\ 0 \end{bmatrix}, y_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, z_1^0 = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \\ 0 \end{bmatrix}$
- Rotation matrices are good for changing vector representations between frames
  - Let $v^1 = \begin{bmatrix} a & b & c \end{bmatrix}^T = ax_1 + by_1 + cz_1$
  - To transform the vector, $v_0 = ax_1^0 + by_1^0 + cz_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = R_1^0 v^1$
  - Note we used the rotation matrix of frame 1 with respect to frame 0, to transform v^1 to get v^0 – the frame that the rotation matrix is with respect to is the frame we end up getting
    - i.e. The **superscript** on v must match the **subscript** on R
- By the same logic, $v^1 = R_0^1 v^0$, so $R_1^0 = (R_0^1)^{-1} \implies (R_1^0)^T = (R_1^0)^{-1}$
  - To invert a rotation matrix, we can simply take its transpose

Definition

A matrix $R \in \mathbb{R}^{m \times m}$ such that $R^T = R^{-1}$ is an *orthogonal* matrix.

For all orthogonal matrices, all its columns must be unit vectors and are mutually orthogonal (dot product of 0).

- Note that since $1 = \det(I) = \det(RR^T) = \det(R)\det(R^T) = (\det(R))^2 \implies \det(R) = \pm 1$
  - If we choose +1 we get a right-handed coordinate system; similarly -1 gives a left-handed coordinate system
  - We are only interested in right-handed coordinate systems

Definition

The *special orthogonal group* is defined as

$$SO(3) = \{ R \in \mathbb{R}^{3\times 3} \mid R^T = R^{-1}, \det(R) = 1 \}$$

- The *elementary rotation matrices* correspond to rotations by θ about one of the coordinate axes

  - Frame 1 is obtained from frame 0 by rotating by θ about one of the axes; the following matrices give R_1^0
  - $R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$
  - $R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
  - $R_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Lecture 4, Sep 10, 2025

Rotational Transformations

- Fix a frame O_0, x_0, y_0, z_0 and let $R \in SO(3)$ be a rotation matrix; we will use R to rotate vectors in frame 0
- Consider $w^0 = Rv^0$ – now R is a rotation matrix in frame 0, representing the action of rotating v^0 by some angle
  - This is now a *rotational transformation* in frame 0
    - All rotational transformations (matrices) must have an associated coordinate frame; the matrix operates only on coordinates in that frame
- We can use rotational transformations to generate new frames by rotating the coordinate vectors x, y, z of an existing frame
  - We can define a new frame 1, resulting from rotating each of x_0, y_0, z_0 by R
  - Then $x_1^0 = Rx_0^0$, $y_1^0 = Ry_0^0$ and $z_1^0 = Rz_0^0$ so $R_1^0 = R\begin{bmatrix} x_0^0 & y_0^0 & z_0^0 \end{bmatrix}$
  - But note $x_0^0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, y_0^0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, z_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, therefore $R_1^0 = R$
- The rotation matrix of frame 1 with respect to frame 0 (R_1^0) is simply the rotation we apply to the coordinate vectors of frame 0 to get the coordinate vectors of frame 1
- What if we want to represent R in frame 1 instead? If we have $w^0 = Rv^0$, what is R' such that $w^1 = R'v^1$?
  - $w_1 = R_0^1 w^0 = (R_1^0)^T w^0 = (R_1^0)^T R v^0 = (R_1^0)^T R R_1^0 v^1$
  - Therefore $R' = (R_1^0)^T R (R_1^0)$
  - Recall from linear algebra that this is a similarity transformation used in a change of basis

Composition of Rotational Transformations

- Consider a vector v and 3 frames
- $v^0 = R_2^0 v^2 = R_1^0 v_1 = R_1^0 R_2^1 v^2 \implies R_2^0 = R_1^0 R_2^1$
  - This allows us to compose relative transformations between intermediate frames
  - As a shortcut, notice that the superscript matches the superscript of the left, and the subscript matches the subscript of the right
- Let $R \in SO(3)$ be a rotational transformation in frame 1; we generate frame 2 by rotating frame 1 by R; what is R_2^0?
  - We know $R_2^0 = R_1^0 R_2^1$
  - Since frame 2 is generated by rotating frame 1 by R, $R_2^1 = R$, so $R_2^0 = R_1^0 R$
    - Note the important point part is that R is a rotation in frame 1 – this has to match the superscript on R_2^1
- Now consider the same problem but R is in frame 0 instead (still used to rotate frame 1 to get frame 2); what is R_2^0?
  - $R_2^0 = R_1^0 R_2^1 = R_1^0 (R_1^0)^T R R_1^0 = R R_1^0$

    * R is in frame 0, but we need it in frame 1, so we apply $R_0^1 = (R_1^0)^T$ on the left and R_1^0 on the right
  - Notice that this time the order is reversed!
- Suppose we rotate frame 0 by θ around x_0, then ϕ around z_1 (of the new frame!) to get frame 2; find R_2^0
  - $R_2^0 = R_1^0 R_2^1 = R_{x,\theta} R_{z,\phi}$
  - Note we are able to do this only because the rotations are represented in the respective frame that they are happening to
- Now suppose we rotate frame 0 by θ around x_0, then ϕ around z_0 (old frame), then ψ around z_2; find R_3^0
  - $R_3^0 = R_1^0 R_2^1 R_3^2$
  - $R_1^0 = R_{x,\theta}$ and $R_3^2 = R_{z,\psi}$
  - R_2^1 is $R_{z,\phi}$ in frame 0, so $R_2^1 = (R_1^0)^T R_{z,\phi} R_1^0$
  - Therefore $R_3^0 = R_1^0 R_2^1 R_3^2 = R_1^0 (R_1^0)^T R_{z,\phi} R_1^0 R_{z,\psi} = R_{z,\phi} R_{x,\theta} R_{z,\psi}$
- Let $M_1, M_2, M_3 \in SO(3)$; define frame 1 by rotating frame 0 by M_1; define frame 2 by rotating frame 2 by rotating frame 1 by M_2, expressed in frame 0; define frame 3 by rotating frame 2 by M_3, expressed in frame 1; find R_3^0
  - $R_1^0 = M_1$
  - $R_2^1 = (R_1^0)^T M_2 R_1^0$
  - $R_3^2 = (R_2^1)^T M_3 R_2^1$
  - $\begin{aligned} R_3^0 &= R_1^0 R_2^1 R_3^2 \\ &= R_1^0 R_2^1 (R_2^1)^T M_3 R_2^1 \\ &= R_1^0 M_3 R_2^1 \\ &= M_1 M_3 M_1^T M_2 M_1 \end{aligned}$

Lecture 5, Sep 12, 2025

Euler Angle Parameterization of Rotations

- We will parametrize rotations using zyz Euler angles, i.e. $R_1^0 = R_{z,\phi} R_{y,\theta} R_{z,\psi}$
  - Expanded: $R_1^0 = \begin{bmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix}$
  - This is the representation we choose to use since it directly coincides with the joint angles of a spherical wrist, e.g. on the KUKA robot
- Given R, we can do the inverse and find (ϕ, θ, ψ)
  - Assuming $\sin\theta \neq 0$:
    * If $\sin\theta > 0$:
      - $\theta = \cos^{-1}(r_{33})$
      - $\psi = \text{atan2}(r_{32}, -r_{31})$
      - $\phi = \text{atan2}(r_{23}, r_{13})$
    * If $\sin\theta < 0$:
      - $\theta = -\cos^{-1}(r_{33})$
      - $\psi = \text{atan2}(-r_{32}, r_{31})$
      - $\phi = \text{atan2}(-r_{23}, -r_{13})$
  - Explanation:
    * We can recover θ from r_{33}, but since $\cos^{-1}$ is not unique, this gives rise to 2 different solutions
    * The other angles can be recovered by atan2, but note multiplying the two arguments of atan2 by a negative factor changes the result
    * If $\sin\theta > 0$ the factor is essentially cancelled out, but for $\sin\theta < 0$ we need to explicitly negate both arguments to undo the sign flip
  - If $\sin\theta = 0$, we have a *singularity* (e.g. when the robot folds in on itself); in this case there are an infinite number of solutions

* These are undesirable because it is very hard to move the robot in this pose

Rigid Motions in 3D

Definition

A *rigid motion* in frame 0 is a function

$$T(p^0) = Rp^0 + d^0$$

where $R \in SO(3)$ is a rotational transformation, and d^0 is a coordinate vector in frame 0; i.e. a rotation followed by a translation.

- Note p is a point, so now we add the ability to translate
- Note if we reverse the order (translate first and then rotate), the translation will be different

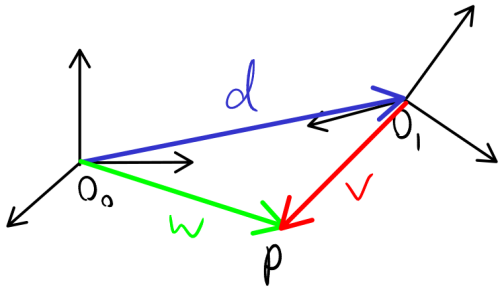

Figure 4: Diagram of the setup used to derive the rigid transformation rule.

- Consider two frames, O_0, x_0, y_0, z_0 and O_1, x_1, y_1, z_1, and consider a point p; we have $O_0 + d = O_1$, $O_0 + w = p$, and $O_1 + v = p$
  - Geometrically (without considering coordinates), we know $w = d + v$
  - If we think about coordinates, we have, $w^0 = p^0$, $v^1 = p^1$ and $d^0 = O_1^0$
  - Then $w^0 = d^0 + v^0 \implies w^0 = d^0 + R_1^0 v^1$
  - Substituting, we get the result $p^0 = O_1^0 + R_1^0 p^1$
  - Notice that this is similar to when we worked with vectors, except now we need to know O_1^0

Lecture 6, Sep 15, 2025

Homogeneous Transformations

- Motivation: Suppose we have p^2 and we want to get p^0; we know $p^0 = R_1^0 p^1 + O_1^0$ and $p^1 = R_2^1 p^2 + O_2^1$
  - $p^0 = R_1^0(R_2^1 p^2 + O_2^1) + O_1^0 = R_1^0 R_2^1 p^2 + R_1^0 O_2^1 + O_1^0$
  - The more frames we have, the messier it gets – the algebra is not scalable
  - Can we turn this into a simple matrix multiplication?
- Consider $p^0 = \begin{bmatrix} a & b & c \end{bmatrix}^T$; apply a rigid motion $q^0 = Rp^0 + d^0$ where $R \in SO(3), d^0 \in \mathbb{R}^3$
  - Let the *homogeneous coordinates* be $Q^0 = \begin{bmatrix} q^0 \\ 1 \end{bmatrix}, P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix} \in \mathbb{R}^4$ (capital letters denote homogeneous coordinates)
  - Let the *homogeneous transformation* $H = \begin{bmatrix} R & d^0 \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$
  - Notice $HP^0 = \begin{bmatrix} R & d^0 \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} p^0 \\ 1 \end{bmatrix} = \begin{bmatrix} Rp^0 + d^0 \\ 0 + 1 \end{bmatrix} = Q^0$
  - We have turned the entire rigid motion transformation into a single matrix multiplication by H, which is much easier to compose

Definition

$Q^0 = HP^0$ is a *homogeneous transformation*, where

$$P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix}, Q^0 = \begin{bmatrix} q^0 \\ 1 \end{bmatrix}$$

are the *homogeneous coordinates*, formed by adding a 1 after the normal coordinates. H is a member of the *special Euclidean group*

$$SE(3) = \left\{ H = \begin{bmatrix} R & d \\ 0_{1\times 3} & 1 \end{bmatrix} \middle| R \in SO(3), d \in \mathbb{R}^3 \right\}$$

Each member of $SE(3)$ represents a rigid motion in 3D corresponding to a rotation by R followed by a translation by d.

- For our original problem, let $P^i = \begin{bmatrix} p^i \\ 1 \end{bmatrix}$ be the homogeneous coordinates of p in the 3 frames
  - Let $H_1^0 = \begin{bmatrix} R_1^0 & O_1^0 \\ 0 & 1 \end{bmatrix}$ be the homogeneous transformation from frame 1 to frame 0, and so on for H_2^1 and H_2^0
  - With these definitions, $P^0 = H_1^0 P^1$, $P^1 = H_2^1 P^2$, so $P^0 = H_1^0 P^1 = H_1^0 H_2^1 P^2$
  - If we compute this matrix product, we get the same expression from before, but now we have a much more compact way of representing it
- Similar to the rotations, we define the *elementary homogeneous transformations*:
  - Pure rotations: $\text{Rot}_{x,\alpha} = \begin{bmatrix} R_{x,\alpha} & 0 \\ 0 & 1 \end{bmatrix}$ denotes a rotation by α around x with no translation
    - Similar definitions for $\text{Rot}_{y,\beta}$ and $\text{Rot}_{z,\gamma}$
  - Pure translations: $\text{Trans}_{x,a} = \begin{bmatrix} I & \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} \\ 0 & 1 \end{bmatrix}$ denotes a translation of a in x with no rotation
    - Similar definitions for $\text{Trans}_{y,b}$ and $\text{Trans}_{z,c}$

Forward Kinematics

- The forward kinematic problem: Given the joint variables $(q_1, \ldots, q_n)$ for an n-joint robot (where q_i denotes the angle for a revolute joint or a length for the prismatic joint), compute the end-effector position and orientation with respect to frame 0
- Consider a manipulator with n links and a base, denoted $0, 1, \ldots, n$ (where 0 is the base), and joints $1, \ldots, n$ (notice the base has no joint); joint i connects link $i-1$ to link i, and is rigidly attached to link $i-1$; when joint i is actuated, link i moves
  - For each link $i \in [1, n]$, we attach a corresponding frame O_i, x_i, y_i, z_i (so this frame moves when joint i moves)
  - Note frame 0 is the inertial frame, which is attached to the fixed base link (link 0)
- Suppose we have all the R_i^{i-1}, O_i^{i-1} relating each frame to the previous frame; let $H_i^{i-1} = \begin{bmatrix} R_i^{i-1} & O_i^{i-1} \\ 0 & 1 \end{bmatrix}$, then the solution of the forward kinematics problem (i.e. the end-effector pose), is $H_n^0 = H_1^0 H_2^1 \cdots H_n^{n-1}$
  - These rotations and translations are functions of the joint variables

Lecture 7, Sep 17, 2025

Denavit-Hartenberg (DH) Parameters

- Recall that we can compute the pose of the end-effector by multiplying together all the H_i^{i-1} corresponding to the homogeneous transformations of each of the joints; however, the assignment of frames is still arbitrary, and we still need a way to efficiently compute R_i^{i-1}, O_i^{i-1}
- This method of systematically assigning coordinate frames and describing the links/joints is called the *Denavit-Hartenberg* (DH) convention
- First, we assign coordinate frames according to the following rules, starting from the inertial frame 0 all the way to the end effector n:
  - Attach frames as follows:
    * Attach frame 0 (the inertial frame) to the base link, which never moves
    * Attach frame i to link i where $i \in [1, n-1]$ such that when joint i is actuated, frame i and link i move
    * Attach frame n to the end effector
  - All frames follow the two fundamental rules:
    * (DH1) x_i is orthogonal to z_{i-1}
    * (DH2) x_i intersects z_{i-1} (i.e. if we extend the axes out infinitely in both directions, there is a point where they meet)
  - Note frame n is not as constrained, so we generally put it on the end-effector

Note

Due to the Denavit-Hartenberg frame rules, the origins of the frames are not necessarily physically on the joints or links, but they are always rigid with respect to the link. For most conventional geometries however, frame i is usually at the center of joint i.

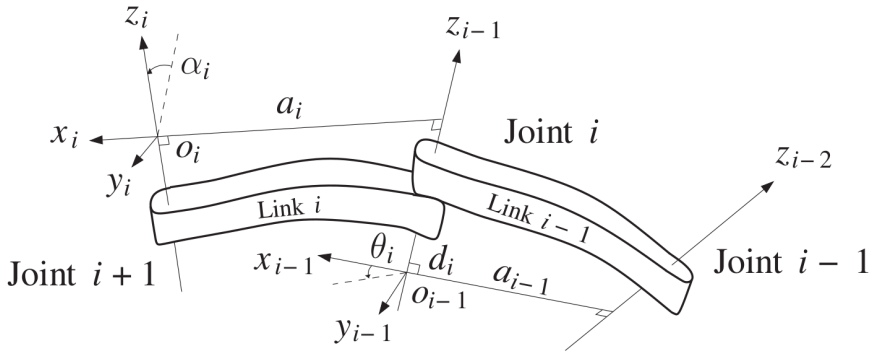

Figure 5: Assignment of DH parameters in the case where z_i and z_{i-1} are not coplanar.

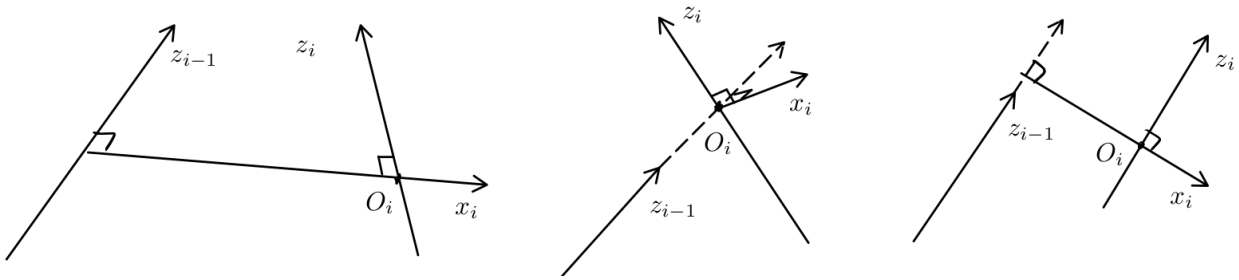

Figure 6: Illustration of the 3 cases of assigning the x axes.

- The procedure for assigning frames in detail is:

1. Assign $z_0, \ldots, z_{n-1}$ such that z_i is the axis of actuation of joint $i+1$ (i.e. the axis of rotation for revolute joints, or the axis of translation for prismatic joints)
2. Choose the base frame such that we have a right-handed frame
3. Assign the x axes for each of the frames in sequence:
   - If z_{i-1} and z_i are not coplanar, then find the unique shortest line segment s between z_{i-1} and z_i to define x_i, and define O_i as the intersection of s and z_i
     * Due to the geometry, this is guaranteed to be orthogonal to both z_{i-1} and z_i, and by construction it intersects z_{i-1} and z_i
     * The point of intersection of x_i and z_i is defined as O_i
     * Choose x_i so that it is parallel to s, and in the direction towards the next link
   - If z_{i-1} and z_i intersect transversally (i.e. intersect but not parallel), declare O_i as the intersection of the vectors, and define x_i so that it is normal to the plane formed by z_{i-1} and z_i
     * Note there are two possible directions we can choose x_i; the choice does not matter
   - If z_{i-1} and z_i are parallel (or they are the same), choose O_i anywhere along z_i; x_i can be chosen as any vector orthogonal to both z_{i-1} and z_i as long as it intersects z_{i-1}
4. Assign the frame for the end-effector so that O_n is on the end-effector and select x_n to satisfy the two DH rules
   - The assignment of z_n doesn't really matter in this case (as long as it is orthogonal to x_n) so do x_n first
   - Typically the math works out easier if we make z_n parallel to z_{n-1}, if possible, so the transformation between the last 2 frames is simpler

Lecture 8, Sep 19, 2025

[No notes for this class since it was all examples for DH frame assignments and I can't be bothered to draw diagrams.]

Lecture 9, Sep 22, 2025

DH Tables

- Now that we have systematic frame assignments, we need a way to also systematically compute the homogeneous transformation for each joint/link
- After the frame assignment, we can describe every joint/link with just 4 parameters:
  1. Link twist α_i: the signed angle between z_{i-1} and z_i, about x_i
  2. Link length a_i: the signed distance between z_{i-1} and z_i, along x_i
  3. Link offset d_i: the signed distance between O_{i-1} and O_i, along z_{i-1}
  4. Joint angle θ_i: the signed angle between x_{i-1} and x_i, about z_{i-1}
- The angles can be better illustrated if we bring frame $i-1$ and frame i together
- This allows us to form a *DH table*, which lists out $a_i, \alpha_i, d_i, \theta_i$ for each $i \in [1, n]$
- Notice that each of the parameters corresponds to a single operation about a single axis, so we can get the overall homogeneous transformation for each stage of the manipulator by combining the 4 operations
  - $H_i^{i-1} = \text{Rot}_{z,\theta_i}\text{Trans}_{z,d_i}\text{Trans}_{x,a_i}\text{Rot}_{x,\alpha_i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  - The order of transformations here can be deduced by noticing the axis that each operation operates on
- Consider the example in the image; the DH table for this example is the following

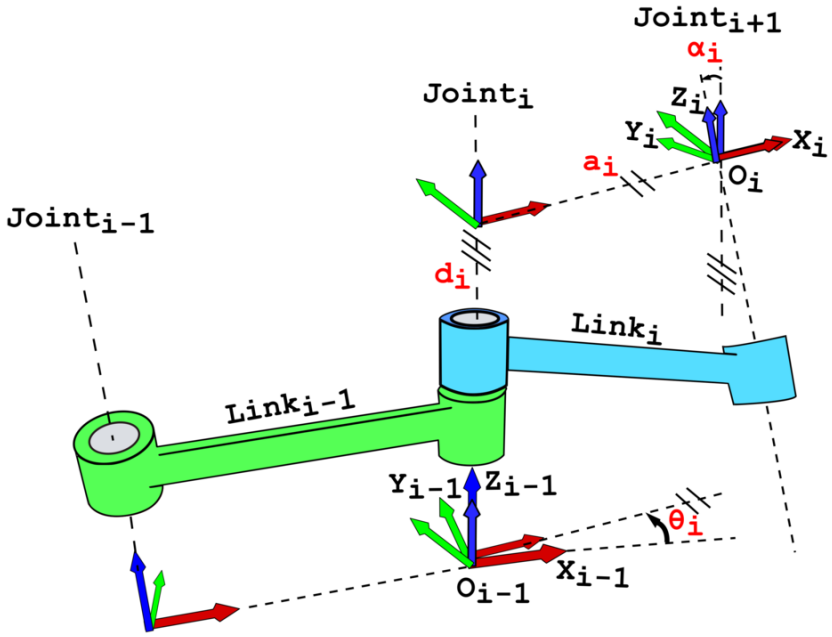

Figure 7: Illustration of the 4 DH parameters.

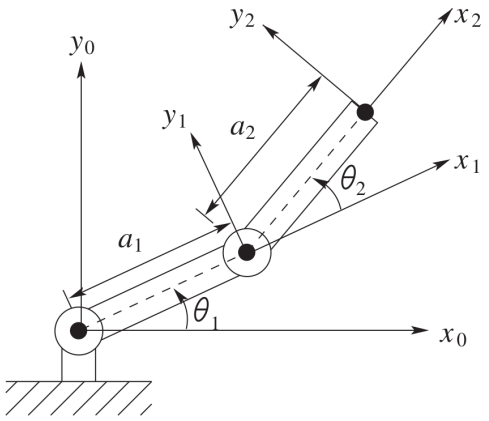

Figure 8: Example 2-stage manipulator annotated with DH parameters. For all frames, z points out of the page.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1^*
2	a_2	0	0	θ_2^*

- Often we mark the variables that will be changed by joint movement with $*$ (these variables later become the joint variables q); the rest of the variables are rigid
  - For prismatic joints, this is always d_i, while for revolute joints this is θ_i

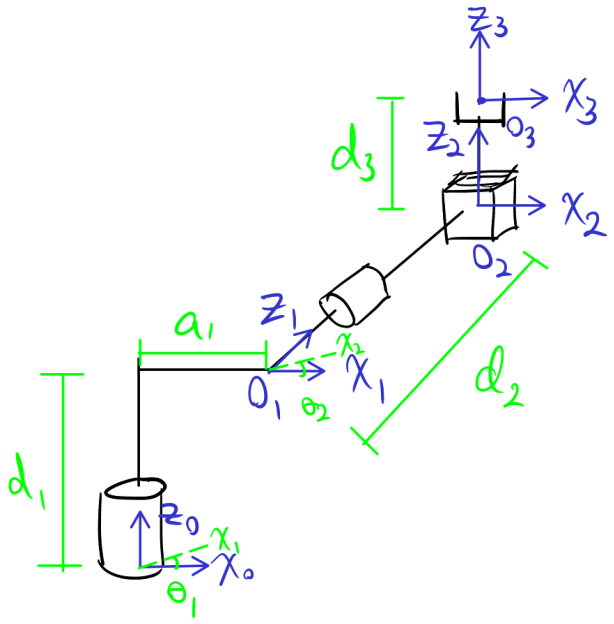

Figure 9: Another example of assigning DH parameters.

- Consider another example with an elbow in the first stage as in the above image; the DH table is the following

Link	a_i	α_i	d_i	θ_i
1	a_1	$-\pi/2$	d_1	θ_1^*
2	0	$\pi/2$	d_2	θ_2^*
3	0	0	d_3^*	0

Lecture 10, Sep 24, 2025

Inverse Kinematics

- Given a desired position $O_d^0 \in \mathbb{R}^3$ and orientation $R_d^0 \in \mathbb{R}^{3\times 3}$ for the end-effector in the world frame, we want to find $q_1, \ldots, q_n$ such that $H_n^0(q_1, \ldots, q_n)$ (the end-effector pose as a function of the joint variables) gives the pose that we want
  - The end-effector pose has 6 degrees of freedom, so whether we can get a solution depends on the number of joint variables n
  - If $n < 6$, then there are no general solutions (many poses will be impossible to reach)
  - If $n > 6$, then there are an infinite number of solutions (kinematically redundant manipulator)
  - If $n = 6$, then there are a finite number of solutions
    - This is why most practical robot manipulators (e.g. KUKA) has 6 joints

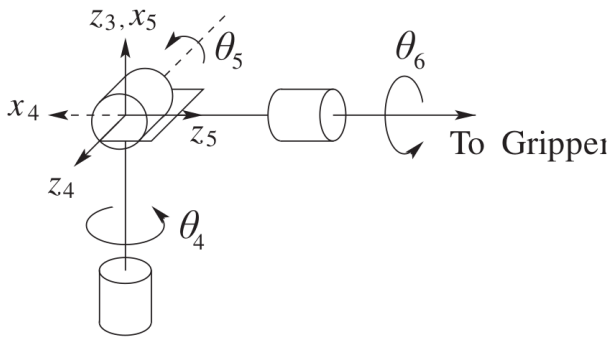

Figure 10: A spherical wrist arrangement. For kinematic decoupling, the last 3 joints must be in this configuration; however the first 3 joints/links are generic.

- We can do this numerically or analytically, but for analytic solutions we need to make some more assumptions about the robot:
  - Assume the robot has 6 joints exactly
  - Assume that the last 3 joints are in a spherical wrist arrangement
- *Kinematic decoupling* is an analytical approach to inverse kinematics under these assumptions, which allows us to split the overall inverse kinematics problem into 2 parts
  - Let O_c be the centre of the wrist; then between O_c and O_d, there is a constant offset of d_6 along axis z_6 by construction of the spherical wrist; furthermore, O_c is not affected by q_4, q_5, q_6
    * Note O_c is the same as O_4 and O_5
    * Therefore we can write $O_d^0 = O_c^0 + R_6^0 \begin{bmatrix} 0 \\ 0 \\ d_0 \end{bmatrix} \implies O_c^0 = O_d^0 - R_d^0 \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$
    * Now the problem becomes solving for q_1, q_2, q_3 so $O_c^0(q_1, q_2, q_3) = O_d^0 - R_d^0 \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$
      - We've decomposed the problem into a part with only 3 unknowns, since everything on the right hand side is known
  - Inverse position kinematics problem: given O_d^0, R_d^0, solve for q_1, q_2, q_3 such that we have the correct O_c^0
  - Inverse orientation kinematics problem: given R_d^0 and solutions for q_1, q_2, q_3, find q_4, q_5, q_6 to get the desired orientation
    * For this part we decompose as $R_d^0 = R_3^0(q_1, q_2, q_3) R_6^3(q_4, q_5, q_6)$; since q_1, q_2, q_3 are known, we can compute the first matrix and invert it
    * $R_6^3(q_4, q_5, q_6) = (R_3^0)^T R_d^0$

Lecture 11, Sep 26, 2025

Inverse Kinematics by Kinematic Decoupling

- Recall that we decomposed the problem so that $O_c(q_1, q_2, q_3) = O_d^0 - R_d^0 \begin{bmatrix} 0 \\ 0 \\ d_6 \end{bmatrix}$
- For inverse position kinematics, we usually do this by analyzing the geometry
- For inverse orientation kinematics, notice that the 3 joints in a spherical wrist form zyz Euler angles, so we can directly compute the joint angles (ϕ, θ, ψ) using the formula introduced previously
- Example: Consider the RRR manipulator with spherical wrist as in the figure above
  - The center of the spherical wrist is $O_c^0 = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$

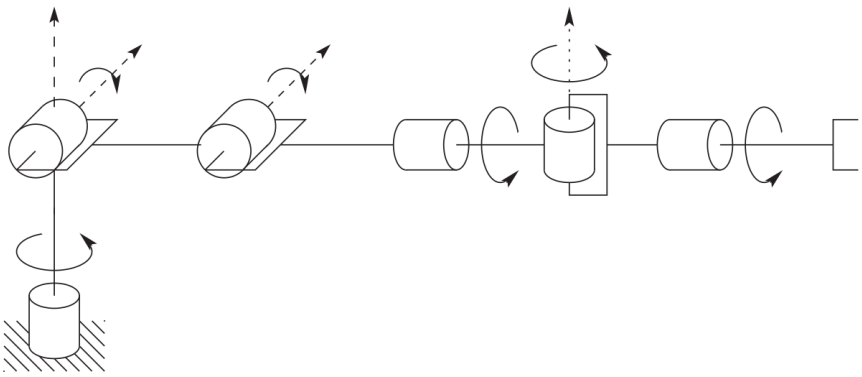

Figure 11: Elbow manipulator with spherical wrist.

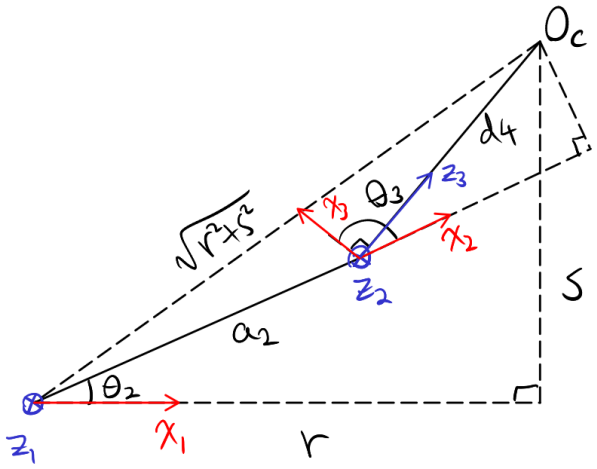

Figure 12: Diagram of link 2 and link 3 for the geometric determination of θ_2, θ_3.

- Let $r = \sqrt{x_c^2 + y_c^2}$ be the horizontal distance from the spherical wrist center to O_0
- Let $s = |z_c - d_1|$ be the vertical distance between the spherical wrist center and the top of link 1 (position of joint 2)
- By geometry, $\theta_1 = \text{atan2}(y_c, x_c)$
- Using the cosine law: $r^2 + s^2 = a_2^2 + d_4^2 - 2a_2 d_4 \cos\left(\frac{3\pi}{2} - \theta_3\right) = a_2^2 + d_4^2 + 2a_2 d_4 \sin\theta_3$
  * $\theta_3 = \sin^{-1}\left(\frac{r^2 + s^2 - a_2^2 - d_4^2}{2a_2 d_4}\right)$
  * Another solution is $\theta_3 = \pi - \sin^{-1}\left(\frac{r^2 + s^2 - a_2^2 - d_4^2}{2a_2 d_4}\right)$
    - There are 2 configurations possible, the "elbow-up" and "elbow-down" configurations
  * Note we can also write $\theta_3 = \text{atan2}(D, \pm\sqrt{1 - D^2})$ where $D = \frac{r^2 + s^2 - a_2^2 - d_4^2}{2a_2 d_4}$
    - This gives a solution in the range $[-\pi, +\pi]$ while the former is in $[0, 2\pi]$
- Using θ_3, now $\theta_2 = \text{atan2}(s, r) - \text{atan2}\left(d_4 \sin\left(\theta_3 - \frac{\pi}{2}\right), a_2 + d_4 \cos\left(\theta_3 - \frac{\pi}{2}\right)\right)$
- For inverse orientation kinematics, if we carry out the computation for H_6^3 we get something very similar to the zyz Euler rotation matrix, but some of the signs may be different
  * The differences in sign are due to the assignment of the x axes, since they can be flipped and still follow the DH rules
  * Watch out for this in labs

Lecture 12, Sep 29, 2025

Forward Velocity Kinematics

- Given the joint variables $q_i(t)$ as functions of time, we want to find the relationship between the joint velocities $\dot{q}_i(t)$ and the linear and angular velocities of the end-effector, with applications in motion planning
  - To do this, we need to know how velocities are transformed between frames
- Define the operation $S(w) = S\left(\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}\right) = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$
  - This allows us to compute the cross product easily as $w \times v = S(w)v$
  - Note this produces a skew-symmetric matrix, i.e. $M = -M^T$; furthermore, every skew-symmetric matrix can be expressed as $S(w)$ for some unique $w \in \mathbb{R}^3$
- Note the following properties of $S(w)$:
  1. Linearity: $S(w_1 + \lambda w_2) = S(w_1) + \lambda S(w_2)$
  2. Similarity transform: $RS(w)R^T = S(Rw)$ for all $R \in SO(3)$
- Lemma: there exists a unique $w_1^0 \in \mathbb{R}^3$ such that $\dot{R}_1^0 = S(w_1^0)R_1^0$
  - This means if we can find w_1^0 we can compute the derivative of a rotation matrix easily without taking the derivative for each term in the matrix
  - Proof: differentiate $R_1^0(R_1^0)^T = I$
    * $\dot{R}_1^0(R_1^0)^T + R_1^0(\dot{R}_1^0)^T = 0$
    * $\dot{R}_1^0(R_1^0)^T = -R_1^0(\dot{R}_1^0)^T = -\left(\dot{R}_1^0(R_1^0)^T\right)^T$
    * This means that $\dot{R}_1^0(R_1^0)^T$ is skew-symmetric, so there exists w_1^0 such that $\dot{R}_1^0(R_1^0)^T = S(w_1^0)$
    * Rearrange to yield $\dot{R}_1^0 = S(w_1^0)R_1^0$
- Consider a point moving in frame 0, $p^0(t)$; its linear velocity is defined as $v^0(t) = \frac{\mathrm{d}}{\mathrm{d}t}p^0(t) = \dot{p}^0(t)$
- Now suppose frame 1 is moving relative to frame 0, i.e. O_1^0, R_1^0 are functions of t; consider a point p fixed in frame 1
  - $p^0 = O_1^0 + R_1^0 p^1$

  - Differentiate: $\dot{p}^0 = \dot{O}_1^0 + \dot{R}_1^0 p^1 + R_1^0 \dot{p}^1$
    $= \dot{O}_1^0 + S(w_1^0) R_1^0 p^1 + R_1^0 \dot{p}^1$
    $= \dot{O}_1^0 + w_1^0 \times (R_1^0 p^1) + R_1^0 \dot{p}^1$
  - $\dot{O}_1^0$ is the *linear velocity* of frame 1 with respect to frame 0
  - w_1^0 is the *angular velocity* of frame 1 with respect to frame 0
- Consider frames 0, 1, 2, where 0 is the inertial frame and frames 1 and 2 are moving; how can we express w_2^0 in terms of w_1^0 and w_2^1?
  - We know $\dot{R}_2^0 = S(w_2^0) R_2^0$ and $R_2^0 = R_1^0 R_2^1$
  - Take derivative: $\dot{R}_2^0 = \dot{R}_1^0 R_2^1 + R_1^0 \dot{R}_2^1$
    $= S(w_1^0) R_1^0 R_2^1 + R_1^0 S(w_2^1) R_2^1$
    $= S(w_1^0) R_2^0 + R_1^0 S(w_2^1) (R_1^0)^T R_1^0 R_2^1$
    $= S(w_1^0) R_2^0 + S(R_1^0 w_2^1) R_1^0 R_2^1$
    $= S(w_1^0) R_2^0 + S(R_1^0 w_2^1) R_2^0$
    $= (S(w_1^0) + S(R_1^0 w_2^1)) R_2^0$
    $= S(w_1^0 + R_1^0 w_2^1) R_2^0$
  - Compare this with the above, we get $S(w_2^0) = S(w_1^0 + R_1^0 w_2^1) \implies w_2^0 = w_1^0 + R_1^0 w_2^1$
- Note the similarity between $w_2^0 = w_1^0 + R_1^0 w_2^1$ and the transformation for points, $O_2^0 = O_1^0 + R_1^0 O_2^1$
  - Angular velocities work similarly to points – we can "add" them
- Therefore, for an n-link manipulator:
  - $O_i^0 = O_{i-1}^0 + R_{i-1}^0 O_i^{i-1}$
  - $\dot{O}_i^0 = \dot{O}_{i-1}^0 + \dot{R}_{i-1}^0 O_i^{i-1} + R_{i-1}^0 \dot{O}_i^{i-1}$
  - $w_i^0 = w_{i-1}^0 + R_{i-1}^0 w_i^{i-1}$
  - This is a set of recursive formulas for computing the end-effector linear and angular velocities
- Next, we can write explicit formulas for w_i^{i-1} and $\dot{O}_i^{i-1}$, which will be easy to do since all joints operate in the z axis

Lecture 13, Oct 1, 2025

Forward Velocity Kinematics (Continued)

- Recall that we want to find a formula for the end-effector linear and angular velocity, so we want to find $\dot{O}_i^{i-1}, w_i^{i-1}$ in terms of the DH parameters
- Denote $w_{i,j}^k = R_i^k w_j^i$ as a shorthand (notation used by textbook)
  - This means "the angular velocity of frame j with respect to frame i, expressed in frame k"
  - Using this notation, $w_n^0 = w_{0,1}^0 + \cdots + w_{n-1,n}^0 = \sum_{i=1}^{n} w_{i-1,i}^0$
- Recall that rotating a body around a fixed axis z yields an angular velocity $w = \dot{\theta} z$ where $\dot{\theta}$ is the rate of rotation, and the linear velocity of any point on the object is given by $v = w \times p$, where p is a vector from the axis of rotation
- Therefore, we can derive $\dot{O}_i^{i-1}, w_i^{i-1}$ for each type of joint as follows:
  - Revolute joint: $w_i^{i-1} = \dot{q}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \dot{O}_i^{i-1} = w_i^{i-1} \times O_i^{i-1}$
    * $w_i = \dot{\theta}_i z_{i-1}$ since the joint rotates around axis z_{i-1} and the angle of rotation is θ
    * Then $w_i^{i-1} = \dot{\theta}_i z_{i-1}^{i-1} = \dot{\theta}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
    * From physics, $\dot{O}_i^{i-1} = w_i^{i-1} \times O_i^{i-1}$
      - In the common case where O_i and O_{i-1} are aligned on z_{i-1}, notice that this evaluates to zero (since z_{i-1} is parallel to w_i), which is consistent with our intuition that in this case,

frame i is not translating with respect to frame $i-1$

- Prismatic joint: $w_i^{i-1} = 0, \dot{O}_i^{i-1} = \dot{q}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
  * This can be easily seen by noting that there is no rotation happening
  * The only component of O_i^{i-1} that is changing is the z component, which has a value of d_i

Angular Velocity Jacobian

- We will show the following:
  - $\dot{O}_n^0 = J_v(q)\dot{q}$, where $J_v(q) = \dfrac{\partial O_n(q)}{\partial q}$ is the *linear velocity Jacobian*
  - $w_n^0 = J_w(q)\dot{q}$, where $J_w(q) = \dfrac{\partial w_n}{\partial q}$ is the *angular velocity Jacobian*
- For the angular velocity, recall $w_n^0 = w_{0,1}^0 + \cdots + w_{n-1,n}^0 = \sum_{i=1}^{n} w_{i-1,i}^0$
  - If joint i is revolute, then $w_{i-1,i}^0 = R_{i-1}^0 w_i^{i-1} = R_{i-1}^0 \dot{q}_i z_{i-1}^{i-1} = \dot{q}_i R_{i-1}^0 z_{i-1}^{i-1} = \dot{q}_i z_{i-1}^0$
  - If joint i is prismatic, then $w_{i-1,i}^0 = 0$
- As a shorthand, define the *indicator function* $\rho_i = \begin{cases} 0 & \text{joint } i \text{ is prismatic} \\ 1 & \text{joint } i \text{ is revolute} \end{cases}$
- Using the indicator function,
$$\begin{aligned} w_n^0 &= \sum_{i=1}^{n} w_{i-1,i}^0 \\ &= \rho_i \dot{q}_i z_{i-1}^0 \\ &= \begin{bmatrix} \rho_1 z_0^0 & \rho_1 z_1^0 & \cdots & \rho_n z_{n-1}^0 \end{bmatrix}_{3\times n} \dot{q} \\ &= J_w(q)\dot{q} \end{aligned}$$

Lecture 14, Oct 3, 2025

Forward Velocity Kinematics (Continued)

Linear Velocity Jacobian

- Now we want to find $J_v(q)$ such that $\dot{O}_n^0 = J_v(q)\dot{q}$, where $J_v(q) = \dfrac{\partial O_n(q)}{\partial q}$
  - Note for simple cases, if we have an explicit expression for the end-effector position as a function of q (from forward kinematics), we can simply differentiate it as a consequence of the chain rule
  - However we want a systematic approach from only the DH parameters so we can do this for any general manipulator
- Recall that $O_i^0 = O_{i-1}^0 + R_{i-1}^0 O_i^{i-1} \implies O_i^0 - O_{i-1}^0 = R_{i-1}^0 O_i^{i-1}$
  - Therefore $O_n^0 = \sum_{i=1}^{n} O_n^0 - O_{n-1}^0 = \sum_{i=1}^{n} R_{i-1}^0 O_i^{i-1} \implies \dot{O}_n^0 = \sum_{i-1}^{n} (\dot{O}_i^0 - \dot{O}_{i-1}^0)$
- Differentiate each term in the sum:
$$\begin{aligned} \dot{O}_i^0 - \dot{O}_{i-1}^0 &= \dot{R}_{i-1}^0 O_i^{i-1} + R_{i-1}^0 \dot{O}_i^{i-1} \\ &= S(w_{i-1}^0) R_{i-1}^0 O_i^{i-1} + R_{i-1}^0 \dot{O}_i^{i-1} \\ &= w_{i-1}^0 \times (R_{i-1}^0 O_i^{i-1}) + R_{i-1}^0 \dot{O}_i^{i-1} \end{aligned}$$
  - For prismatic joints, $\dot{O}_i^{i-1} = \dot{q}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \dot{q}_i z_{i-1}^{i-1} \implies R_{i-1}^0 \dot{O}_i^{i-1} = \dot{q}_i R_{i-1}^0 z_{i-1}^{i-1} = \dot{q}_i z_{i-1}^0$
  - For revolute joints, $\dot{O}_i^{i-1} = \dot{q}_i z_{i-1}^{i-1} \times O_i^{i-1} \implies$
$$\begin{aligned} R_{i-1}^0 \dot{O}_i^{i-1} &= \dot{q}_i R_{i-1}^0 z_{i-1}^{i-1} \times R_{i-1}^0 O_i^{i-1} \\ &= (\dot{q}_i z_{i-1}^0) \times (R_{i-1}^0 O_i^{i-1}) \\ &= \dot{q}_i (z_{i-1}^0 \times (R_{i-1}^0 O_i^{i-1})) \end{aligned}$$

- Then $\dot{O}_i^0 - \dot{O}_{i-1}^0 = w_{i-1}^0 \times (R_{i-1}^0 O_i^{i-1}) + \dot{q}_i \begin{cases} z_{i-1}^0 & \text{joint } i \text{ is prismatic} \\ z_{i-1}^0 \times (R_{i-1}^0 O_i^{i-1}) & \text{joint } i \text{ is revolute} \end{cases}$
- Using a similar derivation as the angular velocity Jacobian, $w_{i-1}^0 = \sum_{j=1}^{i-1} \dot{q}_j \rho_j z_{j-1}^0$
- Finally, $\dot{O}_i^0 - \dot{O}_{i-1}^0 = \left(\sum_{j=1}^{i-1} \dot{q}_j \rho_j z_{j-1}^0 \right) \times (R_{i-1}^0 O_i^{i-1}) + \dot{q}_i \begin{cases} z_{i-1}^0 & \text{joint } i \text{ is prismatic} \\ z_{i-1}^0 \times (R_{i-1}^0 O_i^{i-1}) & \text{joint } i \text{ is revolute} \end{cases}$

- To collect everything, note the following:
  - If joint k is prismatic, then $\rho_k = 0$, and so $\dot{q}_k$ can only show up in the second term, and will have a coefficient of z_{i-1}^0
  - If joint k is revolute, then $\rho_k = 1$ and $\dot{q}_k$ shows up in both terms
    * $\dot{q}_k$ appears in each term $\dot{O}_i^0 - \dot{O}_{i-1}^0$ where $i \geq k$ due to the double sum
    * Intuitively, the revolute joint's angular velocity affects all links down the chain
    * Using this, we can show that the coefficient of $\dot{q}_k$ in $\dot{O}_n^0$ is $z_{k-1}^0 \times \left(\sum_{i=k}^{n} R_{i-1}^0 O_i^{i-1} \right)$
      - We also know $O_n^0 = O_{k-1}^0 + R_{k-1}^0 O_k^{k-1} + \cdots + R_{n-1}^0 O_n^{n-1}$ so $\sum_{i=k}^{n} R_{i-1}^0 O_i^{i-1} = O_n^0 - O_{k-1}^0$
- Finally, $J_v(q) = \begin{bmatrix} J_{v,1}(q) & \cdots & J_{v,n}(q) \end{bmatrix} \in \mathbb{R}^{3 \times n}$, where each column can be written as:
  - $J_{v,i}(q) = \begin{cases} z_{i-1}^0 & \text{joint } i \text{ is prismatic} \\ z_{i-1}^0 \times (O_n^0 - O_{i-1}^0) & \text{joint } i \text{ is revolute} \end{cases}$

Lecture 15, Oct 6, 2025

Basic Motion Planning With Velocity Jacobian

- We can use velocity Jacobians to create a very basic motion planning algorithm for the manipulator through gradient descent
  - Given a desired $O_d^0 \in \mathbb{R}^3$, we want to find $(q_1(t), \ldots, q_n(t))$, the joint variables as a function of time, that takes us to the desired position
- Recall that the gradient for $f : \mathbb{R}^n \mapsto \mathbb{R}$ is $\nabla_x f(x) = \left(\frac{\partial f(x)}{\partial x} \right)^T$, a column vector
- Also, $\frac{\partial \|x\|}{\partial x} = \frac{1}{\|x\|} x^T$
  - This can be shown by checking componentwise, with $\|x\| = \sqrt{x^T x}$
- The algorithm: Define a quadratic cost function $\mathcal{U}(q) = \frac{1}{2} \|O_d^0 - O_n^0(q)\|^2$ (penalizing the difference between desired and current end-effector position), we move in the direction of the negative gradient
  - $$\begin{aligned} \frac{\partial \mathcal{U}}{\partial q} &= \|O_d^0 - O_n^0(q)\| \left. \frac{\partial \|O_d^0 - O_n^0(q)\|}{\partial (O_d^0 - O_n^0(q))} \right|_{O_n^0(q)} \left(-\frac{\partial O_n^0(q)}{\partial q} \right) \\ &= \|O_d^0 - O_n^0(q)\| \frac{1}{\|O_d^0 - O_n^0(q)\|} (O_d^0 - O_n^0(q))^T \left(-\frac{\partial O_n^0(q)}{\partial q} \right) \\ &= -(O_d^0 - O_n^0(q))^T \frac{\partial O_n^0(q)}{\partial q} \end{aligned}$$
  - $\nabla_q \mathcal{U}(q) = \left(\frac{\partial \mathcal{U}}{\partial q} \right)^T = -\left(\frac{\partial O_n^0(q)}{\partial q} \right)^T (O_d^0 - O_n^0(q))$
  - Notice that the second term is simply the direction we want to move in, in the workspace of the robot (Euclidean space), and then we transformed it using the Jacobian to get the direction in q space
- In continuous time, $\dot{q} = -\gamma \nabla_q \mathcal{U}(q) = \gamma \left(\frac{\partial O_n^0(q)}{\partial q} \right)^T (O_d^0 - O_n^0(q))$ where $\gamma > 0$ is the learning rate

  - We can forward simulate this with any numerical ODE solver to get a reference signal to track
- In discrete time, $q(t_{k+1}) = q(t_k) + \gamma \left(\frac{\partial O_n^0}{\partial q}(q_{t_k}) \right)^T (O_d^0 - O_n^0(q_{t_k}))$
- Note this is an asymptotic algorithm, i.e. it will never reach the goal exactly, instead closing in on the goal as the error decays exponentially
- This can also be seen as a method to do inverse kinematics without relying on kinematic decoupling or any geometric constraints on the manipulator
- In the future we will add collision avoidance and other features to this algorithm

Velocity Kinematics – Examples

- Example: 3-link articulated manipulator, RRR, find $J_v(q), J_w(q)$
  - $\rho = 1$ for all joints
  - $J_w(q) = \begin{bmatrix} z_0^0 & z_1^0 & z_2^0 \end{bmatrix}$
  - $J_v(q) = \begin{bmatrix} J_{v,1} & J_{v,2} & J_{v,3} \end{bmatrix}$
    * $J_{v,1} = z_0^0 \times (O_3^0 - O_0^0)$
    * $J_{v,2} = z_1^0 \times (O_3^0 - O_1^0)$
    * $J_{v,3} = z_2^0 \times (O_3^0 - O_2^0)$
  - Now find z_0^0, z_1^0, z_2^0 and O_1^0, O_2^0, O_3^0
    * $z_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
    * z_1^0 can be found by extracting the last column of R_1^0, which is contained in H_1^0
    * Find O_1^0 in H_1^0
    * Repeat for all the other variables
- Example: A helicopter flies horizontally along a circle of radius 30 at a height of 100, speed of 10, counterclockwise about z_0; frame 1 is attached to the helicopter, with x_1 forward and z_1 up; find the linear and angular velocities of the helicopter with respect to frame 0
  - We want $\dot{O}_1^0$ and w_1^0
  - Can think of there being a revolute joint between the two frames, so q_1 is the angle between x_0, x_1 about z_0
  - Because we can easily write a closed form for O_1^0, the simplest method to get the linear velocity is to simply differentiate it
    * $O_1^0 = \begin{bmatrix} 30\cos(\omega t) + c_1 \\ 30\sin(\omega t) + c_2 \\ 100 \end{bmatrix}$
    * c_1, c_2 account for the initial conditions
    * $\dot{O}_1^0 = \begin{bmatrix} -30\omega\sin(\omega t) \\ 30\omega\cos(\omega t) \\ 0 \end{bmatrix}$
    * Now $\|\dot{O}_1^0\| = \sqrt{(-30\omega\sin(\omega t))^2 + (30\omega\cos(\omega t))^2} = 30\omega = 10 \implies \omega = \frac{1}{3}$
    * $\dot{O}_1^0 = \begin{bmatrix} -10\sin(t/3) \\ 10\cos(t/3) \\ 0 \end{bmatrix}$
  - For the angular velocity, we make the assumption that the helicopter always flying forward, i.e. the direction of x_1 is the same as the direction of the linear velocity $\dot{O}_1^0$
  - Since the angular velocity Jacobian is trivial, we can simply compute
$$\begin{aligned} w_1^0 &= J_w(q)\dot{q} \\ &= J_w(q_1)\dot{q}_1 \\ &= \begin{bmatrix} \rho_1 z_0^0 \end{bmatrix} \dot{q}_1 \\ &= \begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix} \end{aligned}$$

– The brute-force approach is to find R_1^0, compute its derivative $\dot{R}_1^0$, and extract w_1^0 from $S(w_1^0) = \dot{R}_1^0 (R_1^0)^T$
  * Find x_1^0 by normalizing $\dot{O}_1^0$, so $x_1^0 = \begin{bmatrix} -\sin(t/3) \\ \cos(t/3) \\ 0 \end{bmatrix}$
  * We also have z_1 parallel to z_0 so $z_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
  * Finally to get y_1^0 we need to take a cross product, so its normal to the other 2 and the right hand rule holds
  * $R_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix} = \begin{bmatrix} -\sin(t/3) & -\cos(t/3) & 0 \\ \cos(t/3) & -\sin(t/3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  * Now we can take the derivative of each term with respect to t, multiply by $(R_1^0)^T$ and simplify
  * In the end we're left with $S(w_1^0) = \begin{bmatrix} 0 & -1/3 & 0 \\ 1/3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies w_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix}$

Lecture 16, Oct 8, 2025

Applications of Jacobians

- The immediate applications of velocity Jacobians are
  1. Forward and inverse velocity kinematics
  2. Motion planning (as seen in the previous lecture)
  3. Inverse kinematics (numerically, but without having to do kinematic decoupling)
  4. End-effector force and torque (i.e. calculating the joint torques required to produce a force/torque at the end-effector)

Inverse Velocity Kinematics

- Given the desired linear and angular velocities at the end effector $\xi = \begin{bmatrix} \dot{O}_n^0 \\ w_n^0 \end{bmatrix}$, compute $\dot{q}$ that gets us these velocities
- Recall $\begin{bmatrix} \dot{O}_n^0 \\ w_n^0 \end{bmatrix} = J(q)\dot{q} = \begin{bmatrix} J_v(q) \\ J_w(q) \end{bmatrix} \dot{q}$ – the problem basically comes down to whether the Jacobian can be inverted
- $J(q)$ will be a $6 \times n$ matrix, so 3 cases exist depending on the dimensions of q
  – If $n = 6$, a numerical solution can be computed if and only if $J(q)$ is invertible, and the solution will be unique
  – If $n < 6$, in general no solution exists
  – If $n > 6$, solutions exist if and only if $\text{rank}(J(q)) = 6$ (solutions will not be unique)
    * One possible solution is to use a psuedoinverse, $\dot{q} = J^\dagger(q)\xi$ where $J^\dagger(q) = J^T(q)(J(q)J(q)^T)^{-1}$
      - Due to the rank restriction, we know the psuedoinverse can be calculated
      - Verify: $J(q)\dot{q} = J(q)J^\dagger(q)\xi = J(q)J^T(q)(J(q)J^T(q))^{-1}\xi = \xi$
      - The psuedoinverse has the important property that $J(q)J^\dagger(q) = I$
    * We can extend this to an infinite number of solutions as $\dot{q} = J^\dagger(q)\xi + (I - J^\dagger(q)J(q))b$ where $b \in R^n$ is any arbitrary vector
      - Verify: $\begin{aligned} J(q)\dot{q} &= J(q)J^\dagger(q)\xi + J(q)(I - J^\dagger(q)J(q))b \\ &= \xi + (J(q) - J(q))b \\ &= \xi \end{aligned}$

Inverse Kinematics Without Kinematic Decoupling

- Given $R_n^0(q) = R_d, O_n^0(q) = O_d^0$ and suppose $n > 6$, how can we solve for q?
- Suppose R_d is written in terms of zyz Euler angles
  - $R_d = \begin{bmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix}$
- Let the state $x = \begin{bmatrix} O_n^0 \\ \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^6$ which describes the pose of the end-effector
- The *analytic Jacobian* $J_a(q)$ relates the derivatives of x with respect to q, and we can show $\dot{x} = J_a(q)\dot{q}$ and $\dot{q} = J_a^\dagger(q)\dot{x}$
  - This is the same psuedoinverse defined earlier, so $J_a(q)J_a^\dagger(q) = I_{6\times 6}$
  - Note however that this does not mean $J_a(q)^\dagger J_a(q) = I$, because the solution for $\dot{q}$ is not unique!
    - Even though we can write $\dot{q} = J^\dagger(q)J(q)\dot{q}$, we can't say $J_a(q)^\dagger J_a(q) = I$ because the $\dot{q}$ on both sides may be different
    - Another way to think about this is to note $J_a(q)^\dagger J_a(q)$ has a nontrivial kernel (since it is $n \times n$ and $n > 6$)
- From R_d, O_d^0 we can get x_d, and from this we define the tracking error $e = x_d - x(q)$
  - $\dot{e} = -\dot{x}(q) = -J_a(q)\dot{q}$
- Choose $\dot{q} = J_a^\dagger(q)Ke$, analogous to a P controller, then $\dot{e} = -J_a(q)J_a^\dagger(q)Ke = -Ke$, and so $e(t)$ will converge to 0 exponentially
  - Therefore, if we have q satisfy the ODE $\dot{q} = J_a^\dagger(q)K(x_d - x(q))$, then the tracking error will decrease to 0, and we will converge on a value of q which solves the inverse kinematics problem
  - Practically we can form this ODE and forward simulate with any ODE solver, and as the "time" goes to infinity, the steady state value of q gives us the solution

Lecture 17, Oct 10, 2025

End-Effector Force and Torque Problem

- Given a desired force f^0 and torque n^0 at the end-effector, what are the corresponding forces and torques we need to apply at the joints to produce this force and torque?
  - In physics, $F^0 = \begin{bmatrix} f^0 \\ n^0 \end{bmatrix} \in \mathbb{R}^6$ is known as a *wrench*
- Consider a body with linear velocity $v^0(t)$ and angular velocity $w^0(t)$ subject to a wrench F^0, then from physics the work performed by the wrench over $[t_1, t_2]$ is $W = \int_{t_1}^{t_2} (v^0(t)^T f^0(t) + w^0(t)^t n^0(t))\,\mathrm{d}t$
  - More compactly, $W = \int_{t_1}^{t_2} \xi^0(t)^T F^0(t)\,\mathrm{d}t$ where $\xi = \begin{bmatrix} v^0(t) \\ w^0(t) \end{bmatrix}$
  - Substitute $\xi = \begin{bmatrix} \dot{O}_n^0 \\ w_n^0 \end{bmatrix} = J(q)\dot{q}$ we get $W = \int_{t_1}^{t_2} \dot{q}(t)^T J(q(t))^T F^0(t)\,\mathrm{d}t$
- Now consider the total work done at each joint (assuming all revolute joints for now for simplicity): $W_r = \int_{t_1}^{t_2} \sum_{i=1}^{n} \dot{q}_i(t)\tau_i(t)\,\mathrm{d}t = \int_{t_1}^{t_2} \dot{q}(t)^T\tau(t)\,\mathrm{d}t$ where $\tau(t) = \begin{bmatrix} \tau_1(t) & \cdots & \tau_n(t) \end{bmatrix}^T \in \mathbb{R}^n$
- We can equate them due to energy conservation: $\int_{t_1}^{t_2} \dot{q}(t)^T\tau(t)\,\mathrm{d}t = \int_{t_1}^{t_2} \dot{q}(t)^T J(q(t))^T F^0(t)\,\mathrm{d}t$
  - Using the argument that $\dot{q}(t)$ is entirely arbitrary, we conclude $\tau(t) = J(q(t))^T F^0(t)$
  - Now we can use this formula to get the joint torques we need
- A typical application: given $F^{\text{load}} = \begin{bmatrix} f^{\text{load}} \\ n^{\text{load}} \end{bmatrix}$, ignoring the robot's own weight, we apply $F^0 = -F^{\text{load}}$

to counteract the load, so the torque we need to apply at each joint is $\tau = J(q)^T F^0$

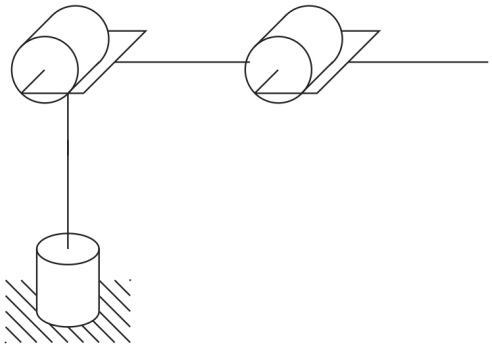

Figure 13: Robot for the torque example.

- Example: consider a 3-link RRR articulated robot; the end-effector must hold a mass M which applies a force $Mg = 10\,\text{N}$; ignoring the weight of the robot, what torques do we need to apply?
  - $F^{\text{load}} = \begin{bmatrix} 0 \\ 0 \\ -10 \\ 0 \\ 0 \\ 0 \end{bmatrix} \implies F^0 = \begin{bmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 0 \\ 0 \end{bmatrix}$
  - $\tau = J(q)^T F^0$

    $$= \begin{bmatrix} z_0^0 \times O_3^0 & z_1^0 \times (O_3^0 - O_1^0) & z_2^0 \times (O_3^0 - O_2^0) \\ z_0^0 & z_1^0 & z_2^0 \end{bmatrix}^T F^0$$

    $$= \begin{bmatrix} (z_0^0 \times O_3^0)^T \\ (z_1^0 \times (O_3^0 - O_1^0))^T \\ (z_2^0 \times (O_3^0 - O_2^0))^T \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}$$

    $$= \begin{bmatrix} 0 \\ 10a_3 \cos(\theta_2 + \theta_3) + 10a_2 \cos(\theta_2) \\ 10a_3 \cos(\theta_2 + \theta_3) \end{bmatrix}$$
  - Notice how the first component is 0, which makes sense since the first joint does not contribute to holding up a load at all
  - Note how the torque we need to apply depends on what the joint angles are
    - τ_2, τ_3 are maximum when $\cos(\theta_2 + \theta_3) = \cos(\theta_2) = 1$; intuitively this corresponds to the situation where the joints are fully extended and lined up
    - Conversely we get $\tau_2 = \tau_3 = 0$ if $\cos(\theta_2 + \theta_3) = \cos(\theta_2) = 0$, which intuitively corresponds to the situation where both joints are pointing straight up, so we don't need to apply any torque to hold up the load

Lecture 18, Oct 15, 2025

Obstacle Avoidance via Potential Field

- Given a start joint position q^s and end position q^f, the motion planning problem involves determining the path of the manipulator in q-space, $q(t)$, such that the robot doesn't hit any obstacles
  - Denote by $\bar{O}_i^0$ the final position of the base point O_i^0 at q_f
- We will use a potential field approach, with $\mathcal{U}(q) = \mathcal{U}_{att}(q) + \mathcal{U}_{rep}(q)$
  - $\mathcal{U}_{att}(q)$: the attractive potential function, which pulls each O_i^0 to $\bar{O}_i^0$
    - This potential alone would simply pull each base point in a straight line to their final positions

  - $\mathcal{U}_{rep}(q)$: the repulsive potential function, which pushes each O_i^0 away from obstacles
    * The strength of this potential is based on the distance between the base points and the nearest point on an obstacle, i.e. the orthogonal projection
- Note this assumes the obstacles are convex, otherwise looking at only the base points is not enough to avoid the obstacle (with some other assumptions, e.g. obstacles are not too small, etc)
  - More complex versions of this algorithm can use various heuristics or intermediate waypoints, etc
  - In general this problem is computationally intractable (NP-hard)
    * Potential fields are not guaranteed to always solve the problem but they are easy to compute
- For now we work in Euclidean space instead of q-space, but in the end we need to define potentials in q
- Define the attractive potential $\mathcal{U}_{i,att}(O_i^0) = \frac{1}{2} c_i \|O_i^0 - \bar{O}_i^0\|^2$ where $c_i > 0$ is a constant weight
  - The weights allow us to place different importances for the different base points, effectively having some of them converge faster than the others
- Attractive gradient: $F_{i,att}(O_i^0) = -\nabla \mathcal{U}_{i,att}(O_i^0)$

$$\begin{aligned} &= -\left(\frac{\partial \mathcal{U}_{i,att}(O_i^0)}{\partial O_i^0} \right)^T \\ &= -\left(c_i \|O_i^0 - \bar{O}_i^0\| \cdot \frac{(O_i^0 - \bar{O}_i^0)^T}{\|O_i^0 - \bar{O}_i^0\|} \right)^T \\ &= -c_i (O_i^0 - \bar{O}_i^0) \end{aligned}$$

  - Notice how this directly pulls towards $\bar{O}_i^0$ in a straight line, with a strength proportional to the distance times a weight
  - Practically, we want to avoid very large gradients, so we often cap the magnitude of the gradient at some value, or normalize the gradient past a certain point
  - $F_{i,att}(O_i^0) = \begin{cases} -c_i (O_i^0 - \bar{O}_i^0) & \|O_i^0 - \bar{O}_i^0\| \le d \\ -c_i \dfrac{(O_i^0 - \bar{O}_i^0)}{\|O_i^0 - \bar{O}_i^0\|} d & \|O_i^0 - \bar{O}_i^0\| > d \end{cases}$
- Define the repulsive potential $\mathcal{U}_{i,rep} = \begin{cases} \dfrac{\eta_i}{2} \left(\dfrac{1}{\|O_i^0 - \pi(O_i^0)\|} - \dfrac{1}{\rho_0} \right)^2 & \|O_i^0 - \pi(O_i^0)\| \le \rho_0 \\ 0 & \|O_i^0 - \pi(O_i^0)\| > \rho_0 \end{cases}$
  - $\mathcal{O}$ is a convex set representing an obstacle ($\forall p, q \in \mathcal{O}, \overline{pq} \in \mathcal{O}$) in frame 0, and $\pi(p)$ is the orthogonal projection of p onto $\mathcal{O}$
    * Note for all convex $\mathcal{O} \in \mathbb{R}^3$, for each $p \in \mathbb{R}^3$ where $p \notin \mathcal{O}$, there exists a unique $\pi(p) \in \mathcal{O}$ such that $\|p - \pi(p)\|$ is minimum, i.e. there is a unique closest point in $\mathcal{O}$ to p
    * The line joining p and $\pi(p)$ is orthogonal to the boundary of $\mathcal{O}$
  - η_i is another weight parameter similar to in the attractive potential
  - ρ_0 is some distance threshold; if the base point is outside this threshold, the obstacle has no effect
    * $\{ O_i^0 \mid \|O_i^0 - \pi(O_i^0)\| \le \rho_0 \}$ is the *region of influence* of $\mathcal{O}$ for collision avoidance
  - We've construed the repulsive potential so that it is continuous, and $\mathcal{U}_{i,rep} \to \infty$ as $O_i^0 \to \pi(O_i^0)$, i.e. the repulsion is stronger the closer we are to the obstacle
- Note the columns of $\frac{\partial \pi}{\partial O_i^0}$ are orthogonal to $O_i^0 - \pi(O_i^0)$, i.e. $(O_i^0 - \pi(O_i^0))^T \frac{\partial \pi}{\partial O_i^0} = 0$, due to the geometry of the orthogonal projection
  - Intuitively $\frac{\partial \pi}{\partial O_i^0}$ is a line tangent to the boundary of $\mathcal{O}$, so it should be orthogonal to the line $O_i^0 - \pi(O_i^0)$
- $$\begin{aligned} \frac{\partial \mathcal{U}_{i,rep}(O_i^0)}{\partial O_i^0} &= -\eta_i \left(\frac{1}{\|O_i^0 - \pi(O_i^0)\|} - \frac{1}{\rho_0} \right) \frac{1}{\|O_i^0 - \pi(O_i^0)\|^2} \frac{(O_i^0 - \pi(O_i^0))^T}{\|O_i^0 - \pi(O_i^0)\|} \frac{\partial (O_i^0 - \pi(O_i^0))}{\partial O_i^0} \\ &= -\eta_i \left(\frac{1}{\|O_i^0 - \pi(O_i^0)\|} - \frac{1}{\rho_0} \right) \frac{(O_i^0 - \pi(O_i^0))^T}{\|O_i^0 - \pi(O_i^0)\|^3} \left(I - \frac{\partial \pi(O_i^0)}{\partial O_i^0} \right) \\ &= -\eta_i \left(\frac{1}{\|O_i^0 - \pi(O_i^0)\|} - \frac{1}{\rho_0} \right) \frac{(O_i^0 - \pi(O_i^0))^T}{\|O_i^0 - \pi(O_i^0)\|^3} \end{aligned}$$

- $F_{i,rep}(O_i^0) = \begin{cases} \eta_i \left(\dfrac{1}{\|O_i^0 - \pi(O_i^0)\|} - \dfrac{1}{\rho_0} \right) \dfrac{(O_i^0 - \pi(O_i^0))}{\|O_i^0 - \pi(O_i^0)\|^3} & \|O_i^0 - \pi(O_i^0)\| \leq \rho_0 \\ 0 & \|O_i^0 - \pi(O_i^0)\| > \rho_0 \end{cases}$

Lecture 19, Oct 17, 2025

Obstacle Avoidance via Potential Field (Continued)

- Given the initial joint variables $q^0 = q^s \in \mathbb{R}^n$ and final joint variables $q^f \in \mathbb{R}^n$, with the gradients, we can plan a path iteratively as $q^{k+1} = q^k - \alpha_k \nabla_q \mathcal{U}(q^k)$ where $\alpha_k > 0$ is the learning rate
- $\mathcal{U}(q)$ is the total potential function, $\mathcal{U}(q) = \sum_{i=1}^{n} \left(\mathcal{U}_{i,att}(O_i^0(q)) + \mathcal{U}_{i,rep}(O_i^0(q)) \right)$
  - A simple sum might not always work; sometimes we cannot find the global minimum of the potential this way
- $\nabla_q \mathcal{U}(q) = \left(\dfrac{\partial \mathcal{U}(q)}{\partial q} \right)^T$

$$= \left(\sum_{i=1}^{n} \left(\frac{\partial \mathcal{U}_{i,att}}{\partial O_i^0} + \frac{\partial \mathcal{U}_{i,rep}}{\partial O_i^0} \right) \frac{\partial O_i^0(q)}{\partial q} \right)^T$$

$$= \sum_{i=1}^{n} \left(\frac{\partial O_i^0(q)}{\partial q} \right)^T \left(\nabla \mathcal{U}_{i,att}(O_i^0(q)) + \nabla \mathcal{U}_{i,rep}(O_i^0(q)) \right)$$

  - We have $\nabla \mathcal{U}_{i,att}(O_i^0(q)) + \nabla \mathcal{U}_{i,rep}(O_i^0(q))$ from the previous lecture
  - Now we need $\dfrac{\partial O_i^0(q)}{\partial q}$, which for $i = n$ is the linear velocity Jacobian $J_v(q)$ that we already have
  - Note, for this algorithm we need the Jacobian for every base point, not just the end-effector
  - In practice, when computing $\dfrac{\partial O_i^0(q)}{\partial q}$ for $i < n$, we can do it more efficiently by starting from $J_v(q)$ and zeroing out some columns
- Let $J_{v,O_i}(q) = \dfrac{\partial O_i^0(q)}{\partial q}$, then $J_{v,O_i} = \begin{bmatrix} J_{v,O_i,1} & \cdots & J_{v,O_i,i} & 0_{3\times(n-i)} \end{bmatrix}$
  - $J_{v,O_i,j} = \begin{cases} z_{j-1}^0 & \text{joint } j \text{ is prismatic} \\ z_{j-1}^0 \times (O_i^0 - O_{j-1}^0) & \text{joint } j \text{ is revolute} \end{cases}$
  - This is similar to $J_v(q)$, but notice instead of O_n^0 we have O_i^0, because we are essentially cutting off the manipulator after link i
- The overall algorithm:
  1. Initialize: $q^0 = q^s$
  2. Iterate: $q^{k+1} = q^k + \alpha_k \sum_{i=1}^{n} J_{v,O_i}^T(q^k) \left(F_{i,att}(O_i^0(q^k)) + F_{i,rep}(O_i^0(q^k)) \right)$
  3. Termination condition: $\|q^{k+1} - q^f\| < \varepsilon$ where $\varepsilon > 0$ is a termination threshold
     - In practice, this won't always converge, so often we put a cap on the max iterations and give up if we hit this number
  4. Output: $q^0, q^1, \ldots, q^N$, a set of waypoints in q-space
     - However, these waypoints are often not smooth enough and results in jerky motion
     - Therefore we usually do a spline fit over these waypoints, to get a continuous a second or third-order derivative

Lecture 20, Oct 22, 2025

Spline Interpolation

- Given $q^0, \dots, q^N \in \mathbb{R}^n$ and user-specified times $t_1, \dots, t_N$ where $t_i < t_{i+1}$, the goal is to find a twice-differentiable $q(t)$ such that $q(0) = q^0$ and $q(t_i) = q_i$
  - This is a method to smooth out the trajectories we get from path planning algorithms
- In 1 dimension, a *cubic spline* is a collection of cubic polynomials $P_i(t) = a_3^i t^3 + a_2^i t^2 + a_1^i t + a_0^i, t \in [t_i, t_{i+1}]$ for segments $i = 0, \dots, N-1$
  - There are $4N$ unknowns, $(a_3^i, a_2^i, a_1^i, a_0^i)$
- Each piece of the spline is constrained by the following:
  1. Interpolation constraints: $P_i(t_i) = q^i$ for $i = 0, 1, \dots, N-1$ (all intermediate points) and $P_{N-1}(t_N) = q^N$ (for the final endpoint)
  2. Continuity constraint: $P_i(t_{i+1}) = P_{i+1}(t_{i+1})$ for $i = 0, 1, \dots, N-2$
  3. Differentiability constraint: $\dot{P}_i(t_{i+1}) = \dot{P}_{i+1}(t_{i+1})$ for $i = 0, 1, \dots, N-2$
  4. Twice-differentiability constraint: $\ddot{P}_i(t_{i+1}) = \ddot{P}_{i+1}(t_{i+1})$ for $i = 0, 1, \dots, N-2$
- In total we have $N+1$ constraints from interpolation, and $3(N-1)$ constraints from continuity, giving $4N-2$ constraints
  - To get the same number of constraints as unknowns, we add the constraints that in the beginning and end of motion, the robot has acceleration of 0
  - This translates to $\ddot{P}_0(t_0) = \ddot{P}_{N-1}(t_N) = 0$
- Now we can solve for all the $(a_3^i, a_2^i, a_1^i, a_0^i)$ by solving a linear system
  - Notice that our spline is linear in the parameters, and all constraints are also linear
  - In the end we get a linear system in the form $Ax = b$, very easy to solve
  - e.g. the first constraint translates to $\begin{bmatrix} t_i^3 & t_i^2 & t_i & 1 \end{bmatrix} \begin{bmatrix} a_3^i \\ a_2^i \\ a_1^i \\ a_0^i \end{bmatrix} = q^i$ and so on

Independent Joint Control (Decentralized Robot Control)

- To actually execute the motion, we need to track a reference signal $q^r(t)$, such that $e(t) = q^r(t) - q(t) \to 0$ as $t \to \infty$ by generating inputs $U(t)$
  - One way to do this is to fully model the dynamics of the robot, which is used for high-precision manipulators or large manipulators like industrial robots (*computed torque control*)
    - $M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = U$ where $M(q)$ is mass, $C(q, \dot{q})$ are the Coriolis forces, and $G(q)$ are the gravitational forces
    - This also needs to incorporate the motor models
  - However, a much easier way to achieve this for low-fidelity designs is to use only the motor model, and treat the physics as a disturbance (*independent joint control*)
    - This is known as independent joint control since it controls each joint independently and considers all interactions between them to be disturbances
- Formally, given a twice-differentiable reference signal $q^r(t) = \begin{bmatrix} q_1^r(t) & \cdots & q_n^r(t) \end{bmatrix}^T \in \mathbb{R}^n$, the control problem involves finding feedback control inputs $u_1, \dots, u_n$ to each joint motor, such that $q_i(t) \to q_i^r(t)$ as $t \to \infty$ with desired properties
- We will restrict ourselves to revolute joints for simplicity
- The canonical motor model is $J_m \ddot{\theta}_m + \left(B_m + \frac{k_m k_b}{R} \right) \dot{\theta}_m = \frac{k_m}{R} v - \tau_l$
  - θ_m is the angle of the motor and τ_l is an applied load
    - This contains terms for back EMF, applied load, voltage, etc
  - Let $J = J_m, B = B_m + \frac{k_m k_b}{R}$ and rescaled input $u = \frac{k_m}{R} v$
  - The simplified model is $J\ddot{\theta}_m + B\dot{\theta}_m = u - \tau_l$, which is all we need
- For simplicity, assume that $\theta_m = q \in \mathbb{R}$ for each joint (note there are often offsets, and for a higher fidelity model, there are effects such as backlash and spring/flexibility terms); and also assume $\tau_l = 0$

(so loads are treated as disturbances)

- For a single joint, let $e(t) = q^r(t) - q(t)$
  - Take derivatives: $\dot{e}(t) = \dot{q}^r(t) - \dot{q}(t) \implies \ddot{e}(t) = \ddot{q}^r(t) - \ddot{q}(t) = \ddot{q}^r - \left(-\frac{B}{J}\dot{q} + \frac{1}{J}u\right)$
  - $\ddot{e} = \ddot{q}^r + \frac{B}{J}\dot{q} - \frac{1}{J}u = \ddot{q}^r + \frac{B}{J}\left(\ddot{q}^r - \dot{e}\right) - \frac{1}{J}u$
  - $\ddot{e} + \frac{B}{J}\dot{e} = \ddot{q}^r + \frac{B}{J}\dot{q}^r - \frac{1}{J}u$
    - This is a second-order system, and if we didn't have input, we can see that there is a zero eigenvalue, so the system is unstable
- Using a PD controller: $u = J\ddot{q}^r + B\dot{q}^r + K_p e + K_d \dot{e}$
  - The first terms, $J\ddot{q}^r + B\dot{q}^r$, is a *feedforward* signal that cancels the $\ddot{q}^r$ in our equation; this ensures that even when the (position) error is zero, we still drive the motors enough to achieve the desired velocity and acceleration
  - Substituting this we get: $\ddot{e} + \left(\frac{B}{J} + \frac{K_D}{J}\right)\dot{e} + \frac{K_p}{J}e = 0$
  - Now by choosing K_p and K_d, we can place the poles of this second-order system anywhere we want, using classical control methods

Lecture 21, Oct 24, 2025

Euler-Lagrange – Part 1

- We will develop a model for the dynamics of our manipulator, so we can do higher fidelity control
  - Our goal is to derive the model $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$ presented in the last lecture, which is the standard manipulator model in robotics
- Consider N point masses in $\mathbb{R}^3$, and let $r_i \in \mathbb{R}^3$ denote the position of mass i
  - Each mass satisfies its equations of motion, $m_i\ddot{r}_i - f_i^l - f_i^c = 0$, where f_i^l is the sum of load forces for link i and f_i^c is the sum of constraint forces for link i (i.e. forces that hold the links together)
  - The masses are subject to a set of l holonomic constraints, i.e. their positions are constrained with respect to each other
    - The constraints are expressed as $g(r_1, \ldots, r_N) = 0$ where $g : \mathbb{R}^3 \times \cdots \times \mathbb{R}^3 \mapsto \mathbb{R}^l$ (this is a vector valued function, since we stack all l constraints)
    - To enforce independence, assume that $\frac{\partial g}{\partial r} \in \mathbb{R}^{l \times 3N}$ is such that $\operatorname{rank}\left(\frac{\partial g}{\partial r}\right) = l$, i.e. the Jacobian is full-rank
  - Let $n = 3N - l$ be the degrees of freedom of the system after the constraints are accounted for
    - Assume we have identified n *generalized coordinates*, $(q_1, \ldots, q_n)$, which parametrize the degrees of freedom of the system
    - These turn out to be the exact same as the joint variables
  - The set of allowed states is $\Gamma = \{\, (r_1, \ldots, r_N) \mid g(r_1, \ldots, r_N) = 0 \,\}$, which is parametrized by the n generalized coordinates, i.e. there is a one-to-one mapping between $(q_1, \ldots, q_n)$ and elements of Γ
    - Written explicitly, $r_i = r_i(q_1, \ldots, q_n)$
  - Practically, for a robot, each r_i is taken at the centre of mass of the link (instead of at O_i), so the dynamics work out
- Consider an N-link planar manipulator with all revolute joints
  - For our first link r_1, we have 2 constraints – it stays in the xy plane, and its position on the link stays the same, i.e. its distance from O_0 stays the same
    - The planar constraint is linear, but the distance constraint would be nonlinear since we need to square $r_{i,x}$ and $r_{i,y}$
  - So in total, the number of constraints here is $l = 2N$
  - So our degrees of freedom is $n = 3N - l = 3N - 2N = N$, which is the same as the number of links as we expected
- Let $\delta r = \begin{bmatrix} \delta r_1^T & \cdots & \delta r_N^T \end{bmatrix}^T \in \mathbb{R}^{3N}$ be a *virtual displacement*, i.e. a "virtual" small perturbation

  - We require $\sum_{j=1}^{n} \frac{\partial g}{\partial r_j} \delta r_j = \frac{\partial g}{\partial r} \delta r = 0$ – notice the similarity to the chain rule
    * This can be derived by differentiating $g(r(t)) = 0$ and noting that the result has to be zero for all t
  - Intuitively, this means that if we have infinitesimal perturbations to the point masses, we need the perturbations to satisfy all the constraints; i.e. any movement should stay in Γ by staying on the level set $g(r) = 0$
- Starting from the equations of motion, we have $(m_i \ddot{r}_i - f_i^l - f_i^c)^T \delta r_i = 0$
  - The constraint forces only act in directions orthogonal to the allowable directions (they have no impact in the allowable directions), i.e. $(f_i^c)^T \delta r_i = 0$, so we can ignore the f_i^c term above
  - Over all i, $\sum_{i=1}^{N} (m_i \ddot{r}_i - f_i^l)^T \delta r_i = 0$
  - This is known as the *Lagrange-d'Alembert principle*

Lecture 22, Nov 3, 2025

Euler-Lagrange – Part 2

- Last lecture we derived the Lagrange-d'Alembert principle: $\sum_{i=1}^{N} (m_i \ddot{r}_i - f_i^l)^T \delta r_i = 0$
  - To get to the form we want, we'll get rid of all the r terms and express everything in terms of q
- From the chain rule:
  - $\dot{r}_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \dot{q}_j$
  - $\delta r_i = \sum_{j=1}^{n} \frac{\partial r}{\partial q_j} \mathrm{d}q_j$
    * Note here $\mathrm{d}q_j$ is not a virtual displacement, since unlike δr_i it is not constrained
- The two parts of the equation are then:
  - $$\begin{aligned} \sum_{i=1}^{N} (f_i^l)^T \delta r_i &= \sum_{i=1}^{N} \sum_{j=1}^{n} (f_i^l)^T \frac{\partial r_i}{\partial q_j} \mathrm{d}q_j \\ &= \sum_{j=1}^{n} \varphi_j \mathrm{d}q_j \end{aligned}$$
    * $\varphi_j = \sum_{i=1}^{N} (f_i^l)^T \frac{\partial r_i}{\partial q_j}$ is the jth *generalized force*
  - $\sum_{i=1}^{N} m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^{N} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \mathrm{d}q_j$
- Observe $$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \left(m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right) &= m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} + m_i \dot{r}_i^T \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial r_i}{\partial q_j} \right) \\ &= m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} + m_i \dot{r}_i^T \frac{\partial \dot{r}_i}{\partial q_j} \end{aligned}$$
  - $\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial r_i}{\partial q_j} \right) = \sum_{k=1}^{n} \frac{\partial}{\partial q_k} \left(\frac{\partial r_i}{\partial q_j} \right) \dot{q}_k = \frac{\partial}{\partial q_j} \sum_{k=1}^{n} \frac{\partial r_i}{\partial q_k} \dot{q}_k = \frac{\partial \dot{r}_i}{\partial q_j}$
- Therefore $\sum_{i=1}^{N} m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^{N} \sum_{j=1}^{n} \left(\frac{\mathrm{d}}{\mathrm{d}t} \left(m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right) - m_i \dot{r}_i^T \frac{\partial \dot{r}_i}{\partial q_j} \right) \mathrm{d}q_j$

Lecture 23, Nov 5, 2025

Euler-Lagrange – Part 3

- *Kinetic energy* for a collection of point masses is $T = \sum_{i=1}^{N} \frac{1}{2} m_i \|\dot{r}_i\|^2 = \sum_{i=1}^{N} \frac{1}{2} m_i \dot{r}_i^T \dot{r}_i$
- Notice $\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_j}\right) &= \sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t}\left(m_i \dot{r}_i^T \frac{\partial \dot{r}_i}{\partial \dot{q}_j}\right) \\ &= \sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t}\left(m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j}\right) \end{aligned}$
  - Note $\dot{r}_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \dot{q}_j$, so $\frac{\partial \dot{r}_i}{\partial \dot{q}_j} = \frac{\partial r_i}{\partial q_j}$
  - This is the first term in the summation for $\sum_{i=1}^{N} m_i \ddot{r}_i^T \delta r_i$ from last lecture
  - Also, $\frac{\partial T}{\partial q_j} = \sum_{i=1}^{N} m_i \dot{r}_i^T \frac{\partial \dot{r}_i}{\partial q_j}$ which is the second term
- Combining everything: $\sum_{j=1}^{n} \left(\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j}\right) \mathrm{d}q_j = \sum_{j=1}^{n} \varphi_j \, \mathrm{d}q_j$
  - Since the $\mathrm{d}q_j$ are arbitrary, this means $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} = \varphi_j, j = 1, \ldots, n$
- For each force, $f_i^l = f_i^g + f_i^a = -\nabla_{r_i} \mathcal{U}(r_1, \ldots, r_N) + f_i^a = -\frac{\partial \mathcal{U}}{\partial r_i} + f_i^a$, where the first term is the force caused by gravity
  - $\begin{aligned} \varphi_j &= \sum_{i=1}^{N} (f_i^l)^T \frac{\partial r_i}{\partial q_j} \\ &= \sum_{i=1}^{N} -\frac{\partial \mathcal{U}}{\partial r_i} \frac{\partial r_i}{\partial q_j} + (f_i^a)^T \frac{\partial r_i}{\partial q_j} \\ &= -\frac{\partial \mathcal{U}}{\partial q_j} + \tau_j \end{aligned}$
  - The term τ_j captures all the other forces acting on the joint
- Finally, we get $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial \mathcal{U}}{\partial q_j} = \tau_j$
- Let the *Lagrangian* $\mathcal{L} = T - \mathcal{U}$
  - Note the potential energy is independent of $\dot{q}_j$, so $\frac{\partial T}{\partial \dot{q}_j} = \frac{\partial \mathcal{L}}{\partial \dot{q}_j}$
- We arrive at the *Euler-Lagrange equations*: $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j$

Summary

The Euler-Lagrange equations are a set of n equations of motion for the system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j$$

where the Lagrangian is $\mathcal{L} = T - \mathcal{U}$, n is the number of degrees of freedom ($n = 3N - l$ for l constraints and N particles in 3 dimensions), q are n generalized coordinates which parametrize the set of allowed states, τ_j are the generalized forces:

$$\tau_j = \sum_{i=1}^{N} (f_i^l)^T \frac{\partial r_i}{\partial q_j}$$

where f^l are the applied forces.
A set of l independent holonomic constraints are expressed as

$$g(r_1, \ldots, r_N) = 0 \in \mathbb{R}^l \qquad \operatorname{rank}\left(\frac{\partial g}{\partial r}\right) = l$$

For constraints to be satisified, the virtual displacements $\delta r = \begin{bmatrix}\delta r_1^T & \cdots & \delta r_N^T\end{bmatrix}^T \in \mathbb{R}^{3N}$ must satisfy $\frac{\partial g}{\partial r}\delta r = 0$.

Lecture 24, Nov 7, 2025

Robot Lagrangian

- Now we want to derive the Lagrangian for a robot manipulator, so we can use the Euler-Lagrange equation to model the robot
- For a rigid body, the kinetic energy is the sum of translational kinetic energy and rotational kinetic energy (note in the point mass model we used for derivation, we did not include rotational dynamics)
  - $T = \frac{1}{2} m v^T v + \frac{1}{2} w^T I w$
  - m is the mass of the rigid body, $v \in \mathbb{R}^3$ is the linear velocity vector, $w \in \mathbb{R}^3$ is the angular velocity vector, and $I \in \mathbb{R}^{3\times 3}$ is the *inertia matrix* or *inertia tensor*, a symmetric matrix
  - Note v and w must be expressed in the inertial frame
  - I is expressed in the inertial frame and calculated at the centre of mass of the body
    - Since the joints are moving, this is actually a function of time
    - To make the computations easier, use a frame attached to the body at the centre of mass so the matrix becomes constant
- Consider an n-link robot, consisting of n rigid bodies (links); $r_1^0, \ldots, r_n^0$ are the positions of the centre of masses of each link, which are completely specified by q, and therefore it is appropriate to use q as the generalized coordinates of the system
- The robot has kinetic energy $T = \sum_{i=1}^{n} \frac{1}{2} m_i (\dot{r}_i^0)^T (\dot{r}_i^0) + \frac{1}{2} (w_i^0)^T I_i (w_i^0)$
  - Note I_i is the inertia matrix of link i in an inertial frame attached at the centre of mass of link i
- Define the partial Jacobian J_{v_i}, where we regard r_i^0 as a virtual "end-effector"
  - For revolute joints, $J_{v_i} = \begin{bmatrix} z_0^0 \times (r_i^0) & z_1^0 \times (r_i^0 - O_1^0) & \cdots & z_{i-1}^0 \times (r_i^0 - O_{i-1}^0) & 0_{3\times 3(n-i)}\end{bmatrix}$
  - $J_{w_i} = \begin{bmatrix}\rho_1 z_0^0 & \rho_2 z_1^0 & \cdots & \rho_i z_{i-1}^0 & 0_{3\times 3(n-i)}\end{bmatrix}$
  - We saw a similar concept when we did potential fields, except in that case we treat O_i as the end-effector (see those notes for the formula for prismatic joints)

- We can now rewrite $T = \sum_{i=1}^{n} \frac{1}{2} m_i \dot{q}^T J_{v_i}^T J_{v_i} \dot{q} + \frac{1}{2} \dot{q}^T J_{w_i}^T I_i(q) J_{w_i} \dot{q}$

$$= \frac{1}{2} \dot{q}^T \left(\sum_{i=1}^{n} m_i J_{v_i}^T(q) J_{v_i}(q) + J_{w_i}^T(q) I_i(q) J_{w_i}(q) \right) \dot{q}$$

$$= \frac{1}{2} \dot{q}^T D(q) \dot{q}$$

  - $D(q) = \sum_{i=1}^{n} m_i J_{v_i}^T(q) J_{v_i}(q) + J_{w_i}^T(q) I_i(q) J_{w_i}(q)$ is a symmetric positive definite matrix
- The (gravitational) potential energy is $\mathcal{U}(q) = -\sum_{i=1}^{n} m_i \bar{g}^T r_i^0(q)$
  - $\bar{g} = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} \in \mathbb{R}^3$ is the gravitational acceleration
- Our Lagrangian is then $\mathcal{L}(q, \dot{q}) = \frac{1}{2} \dot{q}^T D(q) \dot{q} + \sum_{i=1}^{n} m_i \bar{g}^T r_i^0(q)$
  - Next time, we will compute $\frac{\partial L}{\partial \dot{q}_j}, \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{q}_j} \right), \frac{\partial \mathcal{L}}{\partial q_j}$

Lecture 25, Nov 10, 2025

Canonical Robot Dynamics Model

- In our previous formula for kinetic energy, our $I_i(q)$ was expressed in the inertial frame, making it a function of q; we can instead write it in a body-fixed frame, which makes it constant, and relate it to the inertial frame inertia tensor via a similarity transform
  - $I_i = R_i^0 \bar{I}_i (R_i^0)^T$ where $\bar{I}_i$ is a constant inertia matrix expressed in a frame centred at the COM of link i
- Using this, $D(q) = \sum_{i=1}^{n} m_i J_{v_i}^T(q) J_{v_i}(q) + J_{w_i}^T(q) R_i^0 \bar{I}_i (R_0^i)^T J_{w_i}^T(q)$
  - $T = \frac{1}{2} \dot{q}^T D(q) \dot{q} = \frac{1}{2} \sum_{i,j}^{n} d_{ij}(q) \dot{q}_i \dot{q}_j$
  - Note $d_{ij}(q) = d_{ji}(q)$ due to symmetry
- $\frac{\partial \mathcal{L}}{\partial \dot{q}_k} = \frac{\partial T}{\partial \dot{q}_k} = \sum_{j=1}^{n} d_{kj}(q) \dot{q}_j$
  - Note the $\frac{1}{2}$ disappears due to the symmetry of $D(q)$, so each term gets summed twice
- $\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}_k} \right) = \sum_{j=1}^{n} d_{kj}(q) \ddot{q}_j + \sum_{j=1}^{n} \frac{\mathrm{d}}{\mathrm{d}t} \left(d_{kj}(q) \right) \dot{q}_j$

$$= \sum_{j=1}^{n} d_{kj}(q) \ddot{q}_j + \sum_{i,j} \frac{\partial d_{kj}(q)}{\partial q_i} \dot{q}_i \dot{q}_j$$

- $\frac{\partial \mathcal{L}}{\partial q_k} = \frac{1}{2} \sum_{j=1}^{n} \frac{\partial d_{ij}(q)}{\partial q_k} \dot{q}_i \dot{q}_j - \frac{\partial \mathcal{U}}{\partial q_k}$
- Substitute into Euler-Lagrange: $\sum_{j=1}^{n} d_{kj}(q) \ddot{q}_j + \sum_{i,j} \left(\frac{\partial d_{kj}(q)}{\partial q_i} - \frac{1}{2} \frac{\partial d_{ij}(q)}{\partial q_k} \right) \dot{q}_i \dot{q}_j + \frac{\partial \mathcal{U}}{\partial q_k} = \tau_k$
  - It can be shown that $\sum_{i,j} \frac{\partial d_{kj}(q)}{\partial q_i} \dot{q}_i \dot{q}_j = \frac{1}{2} \left(\frac{\partial d_{kj}(q)}{\partial q_i} + \frac{\partial d_{ki}(q)}{\partial q_j} \right) \dot{q}_i \dot{q}_j$

- Let $c_{ijk} = \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)$, then $\sum_j d_{kj}(q)\ddot{q}_j + \sum_{i,j} c_{ijk}\dot{q}_i\dot{q}_j + \frac{\partial \mathcal{U}}{\partial q_k} = \tau_k$
  - These are known as the *Christoffel symbols* (of the first kind)
- Organized in matrix form, $D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)$
  - $D(q) \in \mathbb{R}^{n\times n}$ is the "mass matrix" for kinetic energy; this is the inertial term
  - $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ where $[C(q,\dot{q})]_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i$ contains the Coriolis and centrifugal forces
  - $G(q) = \nabla_q \mathcal{U}(q) \in \mathbb{R}^n$ contains the forces due to gravity (or more generally, a potential)
- In general, for a robot modelling problem, we have 2 approaches:
  1. Using Euler-Lagrange: Writing out the kinetic and potential energies, computing the derivatives and substituting into the Euler-Lagrange equation
     - $T = \sum_{i=1}^{n} \frac{1}{2} m_i \|\dot{r}_i^0\|^2 + \frac{1}{2}(w_i^0)^T R_i^0 \bar{I}_i (R_0^i)^T (w_i^0)$
     - $\mathcal{U} = -\sum_{i=1}^{n} m_i \bar{g}_i^T r_i^0$
     - This is suitable for simple problems, where we can often obtain the r_i^0 by inspection and differentiate them
  2. Find $T = \frac{1}{2}\dot{q}^T D(q)\dot{q}$, then $C(q,\dot{q})$ using the formulas (which require $D(q)$ to be known), then $G(q) = \left(\frac{\partial \mathcal{U}}{\partial q}\right)^T$
     - This will work for any system

Summary

The canonical robot dynamics model is given by

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

where the terms contain the inertial, Coriolis/centrifugal, and gravitational terms respectively:

- $D(q) = \sum_{i=1}^{n} \left(m_i J_{v_i}^T(q) J_{v_i}(q) + J_{w_i}^T(q) R_i^0 \bar{I}_i (R_0^i)^T J_{w_i}^T(q)\right)$
- $[C(q,\dot{q})]_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i$ where $c_{ijk} = \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)$
- $G(q) = \nabla_q \mathcal{U}(q)$ where $\mathcal{U}(q) = -\sum_{i=1}^{n} m_i \bar{g}^T r_i^0(q)$

where $\bar{I}_i$ are the inertias of each link measured about its centre of mass in a body-fixed frame, r_i^0 are the centres of mass, $\bar{g} = \begin{bmatrix}0 & 0 & -g\end{bmatrix}^T$ points in the direction of gravitational acceleration, and the Jacobians are given by

$$J_{v_i} = \begin{cases} \begin{bmatrix} z_0^0 \times (r_i^0) & z_1^0 \times (r_i^0 - O_1^0) & \cdots & z_{i-1}^0 \times (r_i^0 - O_{i-1}^0) & 0_{3\times 3(n-i)} \end{bmatrix} & \text{joint } i \text{ is revolute} \\ \begin{bmatrix} z_0^0 & z_1^0 & \cdots & z_{i-1}^0 & 0_{3\times 3(n-i)} \end{bmatrix} & \text{joint } i \text{ is prismatic} \end{cases}$$

$$J_{w_i} = \begin{bmatrix} \rho_1 z_0^0 & \rho_2 z_1^0 & \cdots & \rho_i z_{i-1}^0 & 0_{3\times 3(n-i)} \end{bmatrix}$$

Lecture 26, Nov 12, 2025

Modelling Example

- Example: Pendulum on a cart: a cart with mass m_1 is accelerated by a horizontal force u; there is a pendulum of mass m_2 attached to the cart, where the centre of mass is l units away from the pivot; the inertia of the pendulum about its centre of mass is I_{zz} (along the axis the pendulum will swing)
  - We can model this as a manipulator with a prismatic joint (for the cart) and revolute joint (for the pendulum)
    * According to DH frame rules, z_0 points in the direction of the cart track and z_1 points along the axis of rotation of the pendulum
  - Let x be the position of the cart and ϕ be the angle of the pendulum from vertical
    * In the DH table, $x = d_1$ and $\phi = \theta_2$
  - DH table:

Link	a	α	d	θ
1	0	$-\pi/2$	x	0
1	a_2	0	0	ϕ

  - Assume $r_1^0 = O_1^0$, so now we can write r_1^0, r_2^0 in terms of the generalized coordinates x and ϕ
    * $r_1^0 = \begin{bmatrix} 0 \\ 0 \\ x \end{bmatrix} \implies \dot{r}_1^0 = \begin{bmatrix} 0 \\ 0 \\ \dot{x} \end{bmatrix}$
    * $r_2^0 = \begin{bmatrix} l\cos\phi \\ 0 \\ x - l\sin\phi \end{bmatrix} \implies \dot{r}_2^0 = \begin{bmatrix} -l\sin(\phi)\dot{\phi} \\ 0 \\ \dot{x} - l\cos(\phi)\dot{\phi} \end{bmatrix}$
  - First find the kinetic energy
    * $T_1 = \frac{1}{2} m_1 \|\dot{r}_1^0\|^2 = \frac{1}{2} m_1 \dot{x}^2$ for link 1 since it is not rotating
    * $T_2 = \frac{1}{2} m_2 \|\dot{r}_2^0\|^2 + \frac{1}{2} (w_2^0)^T I_2 w_2^0$
      - $\|\dot{r}_2^0\|^2 = (l\sin(\phi)\dot{\phi})^2 + (\dot{x} - l\cos(\phi)\dot{\phi})^2$
      - $w_2^0 = w_1^0 + R_1^0 w_2^1 = R_1^0 w_2^1 = R_1^0 \dot{\phi} z_1^1 = \dot{\phi} z_1^0 = \begin{bmatrix} 0 \\ \dot{\phi} \\ 0 \end{bmatrix}$
      - $I_2 = R_2^0 \bar{I}_2 (R_2^0)^T$
      - $(R_2^0)^T w_2^0 = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ 0 & 0 & 1 \\ \sin\phi & -\cos\phi & 0 \end{bmatrix}^T \begin{bmatrix} 0 \\ \dot{\phi} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\phi} \end{bmatrix}$
      - $(w_2^0)^T I_2 w_2^0 = \begin{bmatrix} 0 & 0 & \dot{\phi} \end{bmatrix} \bar{I}_2 \begin{bmatrix} 0 \\ 0 \\ \dot{\phi} \end{bmatrix} = I_{zz}\dot{\phi}^2$
    * $T = \frac{1}{2} m_1 \dot{x}^2 + \frac{1}{2} m_2 \left((l\sin(\phi)\dot{\phi})^2 + (\dot{x} - l\cos(\phi)\dot{\phi})^2 \right) + \frac{1}{2} I_{zz}\dot{\phi}^2$
      $= \frac{1}{2} m \dot{x}^2 + \frac{1}{2} (I_{zz} + m_2 l^2)\dot{\phi}^2 - m_2 l\cos(\phi)\dot{x}\dot{\phi}$
      - $m = m_1 + m_2$ is the total mass
  - Now for the potential energy
    * $\bar{g} = \begin{bmatrix} -g \\ 0 \\ 0 \end{bmatrix}$
    * $\mathcal{U}(q) = -(m_1 \bar{g}^T r_1^0 + m_2 \bar{g}^T r_2^0) = m_2 g l \cos\phi$
  - Lagrangian: $\mathcal{L} = T - \mathcal{U} = \frac{1}{2} m \dot{x}^2 + \frac{1}{2} (I_{zz} + m_2 l^2)\dot{\phi}^2 - m_2 l\cos(\phi)\dot{x}\dot{\phi} - m^2 g l \cos\phi$

- Find derivatives... (exercise to the reader)
- Substitute into Euler-Lagrange to get the equations of motion:
  * $m\ddot{x} - m_2 l\cos(\phi)\ddot{\phi} + m_2 l\sin(\phi)\dot{\phi}^2 = u$
  * $(I_{zz} + m_2 l^2)\ddot{\phi} - m_2 l\cos(\phi)\ddot{x} - m_2 lg\cos(\phi) = 0$
- Put this in canonical form $D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$
  * $q = \begin{bmatrix} x \\ \phi \end{bmatrix}$
  * $D(q) = \begin{bmatrix} m & -m_2 l\cos\phi \\ -m_2 l\cos\phi & I_{zz} + m_2 l^2 \end{bmatrix}$
    - Notice that, as expected, $D(q)$ is symmetric (if not, we made a calculation mistake!)
  * $G(q) = \begin{bmatrix} 0 \\ -m_2 lg\cos\phi \end{bmatrix}$
  * $C(q,\dot{q}) = \begin{bmatrix} 0 & m_2 l\sin(\phi)\dot{\phi} \\ 0 & 0 \end{bmatrix}$
    - This is the only term that's left
    - Notice the nonzero entry only has $\dot{\phi}$ instead of $\dot{\phi}^2$ since in the canonical form we multiply it again by $\dot{q}$ in the equation

Lecture 27, Nov 14, 2025

Modelling Example – Matrix and Hybrid Methods

- Using the same cart example from last lecture, this time we will use the formulas to go for the canonical form directly
  1. Compute $D(q)$
     - $D(q) = m_1 J_{v_1}^T J_{v_1} + J_{w_1}^T R_1^0 \bar{I}_1 (R_1^0)^T J_{w_1} + m_2 J_{v_2}^T J_{v_2} + J_{w_2}^T R_2^0 \bar{I}_2 (R_2^0)^T J_{w_2}$
     - Compute Jacobians
       * $J_{w_1} = \begin{bmatrix} \rho_1 z_0^0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$
       * $J_{v_1} = \begin{bmatrix} z_0^0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$
       * $J_{w_2} = \begin{bmatrix} \rho_1 z_0^0 & \rho_2 z_1^0 \end{bmatrix} = \begin{bmatrix} 0 & z_1^0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$
       * $J_{v_2} = \begin{bmatrix} z_0^0 & z_1^0 \times (r_2^0 - O_1^0) \end{bmatrix} = \begin{bmatrix} 0 & -l\sin\phi \\ 0 & 0 \\ 1 & -l\cos\phi \end{bmatrix}$
         - Recall $r_2^0 = \begin{bmatrix} l\cos\phi \\ 0 \\ x - l\sin\phi \end{bmatrix} \implies r_2^0 - O_1^0 = \begin{bmatrix} l\cos\phi \\ 0 \\ -l\sin\phi \end{bmatrix}$
     - Compute rotation matrices
       * R_1^0 does not matter since it's multiplied by J_{w_1} which is zero
       * $R_2^0 = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ 0 & 0 & 1 \\ -\sin\phi & -\cos\phi & 0 \end{bmatrix}$ (from DH table)
     - Compute inertia matrices
       * $\bar{I}_1$ does not matter since again it is multiplied by a zero matrix
       * $\bar{I}_2 = \begin{bmatrix} I_{xx} & * & * \\ * & I_{yy} & * \\ * & * & I_{zz} \end{bmatrix}$ where the $*$ entries don't matter
         - One can verify that $J_{w_2}^T R_2^0 \bar{I}_2 (R_2^0)^T J_{w_2}^T = \begin{bmatrix} 0 & 0 \\ 0 & I_{zz} \end{bmatrix}$

- $D(q) = m_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + m_2 \begin{bmatrix} 1 & -l\cos\phi \\ -l\cos\phi & l^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & I_{zz} \end{bmatrix} = \begin{bmatrix} m & -m_2 l\cos\phi \\ -m_2 l\cos\phi & m_2 l^2 + I_{zz} \end{bmatrix}$
  - Check that $D(q)$ is symmetric
  - Compare with the one we derived at the end of the last lecture and we see that they are the same
2. Compute Christoffel symbols and $C(q, \dot{q})$
   - $c_{ijk} = \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)$
     - e.g. $c_{211} = \frac{1}{2}\left(\frac{\partial d_{11}}{\partial q_2} + \frac{\partial d_{12}}{\partial q_1} - \frac{\partial d_{21}}{\partial q_1}\right) = \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = 0$
     - Many terms cancel since $D(q)$ is symmetric
     - For the 2×2 case, we need to compute 8 Christoffel symbols
   - $[C(q, \dot{q})]_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i$
3. Compute $G(q)$
   - $\mathcal{U}(q) = -(m_1 \bar{g}^T r_1^0 + m_2 \bar{g}^T r_2^0) = m_2 g l \cos\phi\$$
   - $G(q) = \nabla \mathcal{U}(q)$

- Example: planar RR manipulator, with link lengths l_1, l_2 and joint angles θ_1, θ_2; links have masses m_1, m_2 and distances to COM of l_{c_1}, l_{c_2} from the previous joint; links have inertias I_{z_1}, I_{z_2}
1. Find the kinetic energy
   - $T = \frac{1}{2} m_1 \|\dot{r}_1^0\|^2 + \frac{1}{2}(w_1^0)^T I_1 w_1^0 + \frac{1}{2} m_2 \|\dot{r}_2^0\|^2 + \frac{1}{2}(w_2^0)^T I_2 w_2^0$
   - From geometry $r_1^0 = \begin{bmatrix} l_{c_1}\cos q_1 \\ l_{c_1}\sin q_1 \\ 0 \end{bmatrix} \implies \dot{r}_1^0 = \begin{bmatrix} -l_{c_1}\sin(q_1)\dot{q}_1 & l_{c_1}\cos(q_1)\dot{q}_1 \end{bmatrix}$
   - $r_2^0 = \begin{bmatrix} l_1\cos q_1 + l_{c_2}\cos(q_1+q_2) \\ l_1\sin q_1 + l_{c_2}\sin(q_1+q_2 \\ 0 \end{bmatrix} \implies \dot{r}_2^0 = \begin{bmatrix} -l_1\sin(q_1)\dot{q}_1 - l_{c_2}\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2) \\ l_1\cos(q_1)\dot{q}_1 + l_{c_2}\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2) \\ 0 \end{bmatrix}$
   - $w_1^0 = \dot{q}_1 z_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{q}_1$
   - $w_2^0 = w_1^0 + R_1^0 w_2^1 = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix} + \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 \\ \sin q_1 & \cos q_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{bmatrix}$
   - Rotational terms:
     - $(R_1^0)^T w_1^0 = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix} = w_1^0$ so the rotation matrices have no effect
     - Therefore $(w_1^0)^T R_1^0 \bar{I}_1 (R_1^0)^T w_1^0 = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix}^T \bar{I}_1 \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix} = I_{z_1}\dot{q}_1^2$ where I_{z_1} is the bottom-right entry of $\bar{I}_1$
   - $T = \frac{1}{2} m_1 l_{c_1}^2 \dot{q}_1^2 + \frac{1}{2} m_2 \left(l_1^2\dot{q}_1^2 + l_{c_2}^2(\dot{q}_1+\dot{q}_2)^2 + 2l_1 l_{c_2}\cos(q_2)\dot{q}_1(\dot{q}_1+\dot{q}_2)\right) + \frac{1}{2} I_{z_1}\dot{q}_1^2 + \frac{1}{2} I_{z_2}(\dot{q}_1+\dot{q}_2)^2$
2. Factor the kinetic energy into the form $T(q, \dot{q}) = \frac{1}{2}\dot{q}^T D(q)\dot{q}$
   - $\frac{1}{2} q^T D(q)\dot{q} = \frac{1}{2}(d_{11}\dot{q}_1^2 + d_{22}\dot{q}_2^2 + 2d_{12}\dot{q}_1\dot{q}_2)$; match terms to get the entries
   - $D(q) = \begin{bmatrix} m_1 l_{c_1}^2 + m_2(l_1^2 + l_{c_2}^2 + 2l_1 l_{c_2}\cos(q_2)) + I_{z_1} + I_{z_2} & m_2(l_{c_2}^2 + l_1 l_{c_2}\cos(q_2)) + I_{z_2} \\ m_2(l_{c_2}^2 + l_1 l_{c_2}\cos(q_2)) + I_{z_2} & I_{z_2} + m_2 l_{c_2}^2 \end{bmatrix}$
3. Compute $\mathcal{U}(q), G(q)$
   - $\mathcal{U}(q) = m_1 g l_{c_1}\sin(q_1) + m_2 g(l_1\sin(q_1) + l_{c_2}\sin(q_1+q_2))$
   - $G(q) = \nabla_q \mathcal{U}(q) = \begin{bmatrix} (m_1 l_{c_1} + m_2 l_1) g\cos(q_1) + m_2 l_{c_2} g\cos(q_1+q_2) \\ m_2 l_{c_2} g\cos(q_1+q_2) \end{bmatrix}$

4. Compute the Christoffel symbols
   - $c_{ijk} = \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)$
   - Due to the symmetry, we can save some work by noting $c_{ijk} = c_{jik}$ due to the symmetry of the first two terms
   - $c_{111} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_1} = 0$
   - $c_{121} = \frac{1}{2}\left(\frac{\partial d_{12}}{\partial q_1} + \frac{\partial d_{11}}{\partial q_2} - \frac{\partial d_{12}}{\partial q_1}\right) = -m_2 l_1 l_{c_2}\sin(q_2) = c_{211}$
   - $c_{221} = \frac{1}{2}\left(\frac{\partial d_{12}}{\partial q_1} + \frac{\partial d_{12}}{\partial q_2} - \frac{\partial d_{22}}{\partial q_1}\right) = c_{121}$
   - $c_{112} = -c_{121}$
   - $c_{122} = c_{222} = 0$
   - $[C(q,\dot{q})]_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \implies C(q,\dot{q}) = \begin{bmatrix} c_{111}\dot{q}_1 + c_{211}\dot{q}_2 & c_{121}\dot{q}_1 + c_{221}\dot{q}_2 \\ c_{112}\dot{q}_1 + c_{212}\dot{q}_2 & c_{122}\dot{q}_1 + c_{222}\dot{q}_2 \end{bmatrix}$
   - $C(q,\dot{q}) = -m_2 l_1 l_{c_2}\sin(q_2)\begin{bmatrix} \dot{q}_2 & \dot{q}_1 + \dot{q}_2 \\ -\dot{q}_1 & 0 \end{bmatrix}$

Lecture 28, Nov 17, 2025

Control Design

- As we've previously seen, for control design we have the decentralized view (i.e. independent joint control, where robot dynamics are treated as a disturbance) or the centralized view (considers nonlinear coupling of robot dynamics)
- For control design with robot dynamics, we can use 2 classes of models:
  - Ignore motor dynamics: $D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$
    - τ are torques (generalized forces) applied to the joints
  - Include motor dynamics: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B(q)\dot{q} + G(q) = u$
    - u are the motor voltages
    - $B(q)$ models back-EMF
    - $M(q) = D(q) + J$ where J is a diagonal matrix containing the J_m of each motor (see previous lecture with motor model); note $M(q)$ is still symmetric positive definite
    - This is known as the *augmented model*
  - In the end, the algorithms we use for control aren't very different
- Several methods have been developed over the years:
  - *Feedback linearization* (aka *computed torque method*)
  - PD control with gravity compensation
  - *Passivity-based control*
  - Passivity-based control with *parameter adaptation*
    - Unlike the previous methods, we don't actually need to know the system parameters
    - This algorithm uses an adaptive controller where the system parameters are learned
    - This results in a controller that can operate without system parameters, but results in bad transient behaviour

Feedback Linearization

- Consider the augmented model (with motor dynamics): $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B(q)\dot{q} + G(q) = u$, and assume a *fully-actuated* system, i.e. a motor at each joint (as opposed to an *underactuated* system)
- Given the model with all (nominal) parameters known, and a twice-differentiable $r(t) \in \mathbb{R}^n$ (note this is in q-space), the control problem involves finding a control input $u = f(q,\dot{q},r,\dot{r},\ddot{r})$ such that $q(t) \to r(t)$ as $t \to \infty$, for all initial conditions
  - We are designing a state feedback controller
  - Note u is not a function of $\ddot{q}$; practically accelerometers are too noisy for us to get useful

measurements

- *Feedback linearization* involves using a u that cancels out all the nonlinear terms, so we end up with a linear model
  - Choose $u = M(q)v + C(q,\dot{q})\dot{q} + B(q)\dot{q} + G(q)$, where $v(t) \in \mathbb{R}^n$ is an exogenous input (i.e. a "new input" which we will determine)
  - This cancels out most of the dynamics, so we end up with simply $M(q)\ddot{q} = M(q)v$, and since $M(q)$ is invertible, $\ddot{q} = v$
  - Now let the states $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix}$, so $\dot{x} = \begin{bmatrix} x_2 \\ v_1 \end{bmatrix}$, so now we can design a controller $v_1 = k_1 x_1 + k_2 x_2 = k_1 q_1 + k_2 \dot{q}_1$ to stabilize the system – which is a PD controller on this joint
  - Now do this for every joint to track the reference, and we end up with a system with $2n$ states
- Feedback linearization essentially constructs u such that nonlinear dynamics are cancelled out, so the resulting linear system is decoupled and we can control it by independently controlling each joint
- For tracking the reference $r(t)$, define the tracking errors $e_i(t) = r_i(t) - q_i(t), i = 1, \ldots, n$ which we will drive to zero
  - Differentiate the error until we get the input: $\dot{e}_i = \dot{r}_i - \dot{q}_i, \ddot{e}_i = \ddot{r}_i - \ddot{q}_i = \ddot{r}_i - v_i$
  - Choose $v_i = \ddot{r}_i + k_{p,i} e_i + k_{d,i} \dot{e}_i$, i.e. a PD controller with a feedforward term
  - This results in the dynamics $\ddot{e}_i + k_{d,i}\dot{e}_i + k_{p,i} e_i = 0$
    - Any choice of a positive $k_{d,i}, k_{p,i}$ results in poles in the open left half plane, so the error converges to 0
- This is the simplest and worst performing of the centralized methods we will talk about, since we need to know all the system parameters exactly

Lecture 29, Nov 19, 2025

Lyapunov Stability

- Consider a general nonlinear system $\dot{x} = f(x)$ where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is continuously differentiable
- An *equilibrium* state $\bar{x} \in \mathbb{R}^n$ is a state where $f(\bar{x}) = 0$, i.e. $x(t) = \bar{x}, \forall t \geq 0$

Definition

An equilibrium is *stable* if $\forall \varepsilon > 0, \exists \delta > 0$ such that $\|x(0)\| < \delta \implies \|x(t)\| < \varepsilon, \forall t \geq 0$.

- In other words, given any positive radius ε around the equilibrium, we can find another radius δ, where if the initial state starts within δ of the equilibrium, then it will remain within a distance of ε of the equilibrium forever
  - This is the formal definition, which is very hard to work with, so in practice we use equivalent statements of the definition

Definition

An equilibrium is *asymptotically stable* if it is stable, and

$$\exists \delta_0 > 0 \text{ s.t. } \|x(0)\| < \delta_0 \implies \lim_{t \to \infty} x(t) = 0$$

- In other words, there exists a positive radius δ_0 such that if we start within this distance of the equilibrium, we always converge to the equilibrium
- A continuously differentiable function $U : \mathbb{R}^n \mapsto \mathbb{R}$ is *positive definite* at $x = 0$ if $U(0) = 0$ and $U(x) > 0$ for all $x \neq 0$; *negative definite* if $-U(x)$ is positive definite at 0 (i.e. $\forall x \neq 0, U(x) < 0$ and $U(0) = 0$)

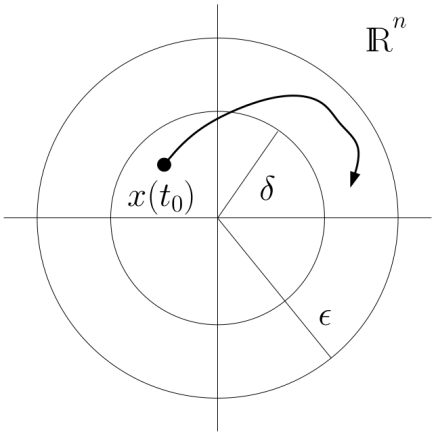

Figure 14: Illustration of the definition of Lyapunov stability.

Theorem

Lyapunov's Theorem: For a system where $f(0) = 0$, if we can find a continuously differentiable *Lyapunov function* $V : \mathbb{R}^n \mapsto \mathbb{R}$ which is positive definite at $x = 0$, then:

1. If $\dot{V}(x) = \dfrac{\partial V}{\partial x} f(x) \leq 0, \forall x \in \mathbb{R}$, then $x = 0$ is stable.
2. If $\dot{V}(x) = \dfrac{\partial V}{\partial x} f(x)$ is negative definite, then $x = 0$ is asymptotically stable.

- The Lyapunov function can be thought of as a measure of energy
  - Consider a solution $x(t)$, then if $\dfrac{\mathrm{d}}{\mathrm{d}t} V(x(t)) = \dfrac{\partial V}{\partial x}\dot{x} = \dfrac{\partial V}{\partial x} f(x) \leq 0$, then this "energy" either stays the same or decreases to zero
  - If $V(x)$ is positive definite, then $V(x) = 0 \implies x = 0$, so as $V(x)$ converges to 0, x must also converge to its equilibrium
  - We essentially converted all the high-dimensional dynamics of the system in x to the dynamics of a single scalar measure $V(x)$

Lecture 30, Nov 21, 2025

Lyapunov Stability Example and LaSalle Invariance Principle

- Example: dampened mass-spring system with $\dot{x}_1 = x_2, \dot{x}_2 = -\dfrac{k}{m}x_1 - \dfrac{b}{m}x_2$, where we take $k = m = b = 1$; prove that the equilibrium $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$ is asymptotically stable
  - For linear systems, we always have a quadratic Lyapunov function
  - Take the Lyapunov function to be $V(x) = x^T P x$ where $P \in \mathbb{R}^{2\times 2}$ is symmetric positive definite
    - For this type of linear system, there is a formula for computing the P matrix
  - Choose $P = \dfrac{1}{2}\begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}$, and expanding the Lyapunov function we get $V(x) = \dfrac{1}{2}(x^2 + x_1x_2 + x_2^2)$
    - We can verify that P is positive definite, which makes $x^T P x$ positive definite at $x = 0$
  - For the derivative, $\dot{V}(x) = \dfrac{\partial V}{\partial x} f(x) = \dfrac{\partial V}{\partial x} Ax = 2x^T PAx$
  - Expanding this gets us $-\dfrac{1}{2}(x_1^2 + x_1x_2 + x_2^2) = -V(x)$ which is negative definite since V is positive definite
  - By Lyapunov's theorem, this system is asymptotically stable

Theorem

$P = P^T$ is positive definite if and only if all leading principal minors are positive, where the nth leading principal minor is the determinant of the $n \times n$ sub-matrix formed by taking the first n rows and columns.

- Can we use our intuitive notion of "energy" from physics for this mass-spring system instead?
  - $V(x) = \frac{1}{2}kx_1^2 + \frac{1}{2}mx_2^2$ would be the total energy, consisting of the spring potential energy and kinetic energy
  - Clearly this is positive definite at 0 due to it containing only squared terms
  - $\dot{V}(x) = kx_1\dot{x}_1 + mx_2\dot{x}_2 = kx_1x_2 + mx_2\left(-\frac{k}{m}x_1 - \frac{b}{m}x_2\right) = -bx_2^2$
  - This is negative for $x_2 \neq 0$, but it does not say anything about x_1, so it's not negative definite!

Theorem

LaSalle Invariance Principle: Suppose there exists $V : \mathbb{R}^n \mapsto \mathbb{R}$ positive definite at the equilibrium $\bar{x}$, and $\dot{V}(x) = \frac{\partial V}{\partial x}f(x) \leq 0$, then $\dot{V}(x(t)) \to 0$ as $t \to \infty$.
Furthermore, if $\dot{V}(x) = 0, \forall t \implies x(t) = \bar{x}, \forall t$, then the equilibrium $\bar{x}$ is asymptotically stable.

- The first part says that as we are going down the level sets of V, we are going to either hit zero or reach a point where the derivative is flat and get stuck; the second part says that if the only place we can get stuck forever is at the equilibrium, then we will always end up at the equilibrium
- Returning to our mass-spring example, we can use the LaSalle invariance principle to show that the equilibrium is asymptotically stable
  - If $\dot{V} = 0$ for all t, then $-bx_2(t)^2 = 0 \implies x_2(t) = 0 \implies \dot{x}_2(t) = 0$
  - Recall our equations of motion: $\dot{x}_1 = x_2, \dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2$
  - Therefore we have $-\frac{k}{m}x_1(t) = 0$ from the second equation, so it must be that $x_1(t) = 0$
  - We have shown that $\dot{V} = 0 \implies x = \bar{x}$, so by the LaSalle invariance principle this equilibrium is indeed asymptotically stable
- As with the example, often when we use a physically meaningful Lyapunov function (e.g. potential + kinetic energy), we end up with $\dot{V}$ being only negative semidefinite, so we need to use the LaSalle invariance principle to make it work

Lecture 31, Nov 24, 2025

PD Control With Gravity Compensation

- Again starting with the augmented model, $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B(q)\dot{q} + G(q) = u$
- Previously, in feedback linearization, we assumed full knowledge of the model; but often this is not realistic, so what if we don't know the full dynamics?
  - In PD control with gravity compensation, we only assume knowledge of $G(q)$, which is much easier to obtain
  - On the other hand, this method only works for a constant reference
- Suppose the reference $q^r(t) \equiv q^r$, i.e. it is constant for all time; choose a controller $u = K_p\tilde{q} + K_d\dot{\tilde{q}} + G(q)$, where $\tilde{q} = q^r - q$ is the tracking error
  - K_p, K_d are symmetric positive definite gain matrices, which are often (but don't have to be) diagonal
- We want to study the closed-loop equilibrium $(\tilde{q}, \dot{\tilde{q}}) = (0,0) \in \mathbb{R}^{2n}$ and show that it is asymptotically stable using Lyapunov and LaSalle
- We will make use of the following facts:

  - $M(q)$ is symmetric positive definite, so it is invertible for all q
  - We can show that $\dot{M}(q,\dot{q}) - 2C(q,\dot{q})$ is skew symmetric
  - $B(q)$ is symmetric positive semi-definite
- The closed-loop system is $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B(q)\dot{q} = K_p\tilde{q} + K_d\dot{\tilde{q}}$, with gravity terms cancelled
- Take the Lyapunov function $V(q,\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}(q-q^r)^T K_p(q-q^r)$
  - Note that we do not need to include $\dot{\tilde{q}}$ explicitly since it equals $-\dot{q}$ as q^r is a constant; we can also say that the Lyapunov function is a function of $\tilde{q}, \dot{\tilde{q}}$
  - Notice that the first term is the kinetic energy (augmented with motor mass terms)
  - The second term can be thought of as a "virtual potential energy" which pulls the state towards the desired state (recall gravity was cancelled so there is no other source of potential energy)
  - This is clearly positive definite at the equilibrium $(q,\dot{q}) = (q^r, 0)$
- $$\begin{aligned}\dot{V} &= \frac{1}{2}\ddot{q}^T M(q)\dot{q} + \frac{1}{2}\dot{q}^T\dot{M}(q,\dot{q})\dot{q} + \frac{1}{2}\dot{q}^T M(q)\ddot{q} + \frac{1}{2}\dot{q}^T K_p(q-q^r) + \frac{1}{2}(q-q^r)^T K_p\dot{q} \\ &= \dot{q}^T M(q)\ddot{q} + (q-q^r)^T K_p\dot{q} + \frac{1}{2}\dot{q}^T\dot{M}(q,\dot{q})\dot{q} \\ &= \dot{q}^T\left(-C(q,\dot{q})\dot{q} - B(q) + K_p\tilde{q} + K_d\dot{\tilde{q}}\right) - \tilde{q}^T K_p\dot{q} + \frac{1}{2}\dot{q}^T\dot{M}(q,\dot{q})\dot{q} \\ &= \dot{q}^T\left(-C(q,\dot{q})\dot{q} - B(q)\dot{q} + K_d\dot{\tilde{q}}\right) + \frac{1}{2}\dot{q}^T\dot{M}(q,\dot{q})\dot{q} \\ &= \frac{1}{2}\dot{q}^T(\dot{M}(q,\dot{q}) - 2C(q,\dot{q}))\dot{q} - \dot{q}^T B(q)\dot{q} + \dot{q}^T K_d\dot{\tilde{q}} \\ &= -\dot{q}^T B(q)\dot{q} + \dot{q}^T K_d\dot{\tilde{q}} \\ &= -\dot{q}^T B(q)\dot{q} - \dot{q}^T K_d\dot{q} \\ &= -\dot{q}^T(B(q) + K_d)\dot{q}\end{aligned}$$
  - Note that since $\dot{V}$ is a scalar, all terms are equal to their transpose
  - Because $\dot{M} - 2C$ is skew-symmetric, $\dot{q}^T(\dot{M}-2C)\dot{q} = \dot{q}^T(\dot{M}-2C)^T\dot{q} = -\dot{q}^T(\dot{M}-2C)\dot{q}$ which means the whole term is zero
  - Now because $B(q)$ is positive semidefinite, for any positive definite K_d we have a $\dot{V}$ negative definite in $\dot{q}$ (but not q!)
- We need to apply LaSalle and show that $\dot{V}(t) = 0, \forall t$ forces $q(t) = 0$
  - $\dot{V} = 0 \implies \dot{q}(t) = 0 \implies \ddot{q}(t) = 0$
  - Substituting into the equation of motion for the closed-loop system, $K_p\tilde{q} + K_d\dot{\tilde{q}} = 0$
  - Since K_p is invertible and $\dot{\tilde{q}} = -\dot{q}$, this means $\tilde{q} = 0$ and so $q = q^r$ is the only solution
  - By the LaSalle invariance principle, we conclude that this closed-loop system is asymptotically stable

Lecture 32, Nov 26, 2025

Passivity-Based Control

- As usual, consider the augmented robot model with a twice-differentiable reference signal $q^r(t)$ (which may not be constant)
- Choose a controller $u = M(q)\ddot{q}^r + C(q,\dot{q})\dot{q}^r + B(q)\dot{q} + G(q) + K\dot{\tilde{q}}$ where K is symmetric positive definite, $\tilde{q} = q^r - q$, $\dot{\tilde{q}} = \dot{q}^r - \dot{q}$
- This results in the closed-loop system $M(q)\ddot{\tilde{q}} + (C(q,\dot{q}) + K)\dot{\tilde{q}} = 0$
- Let $r = \dot{\tilde{q}}$, so the model becomes $M(q)\dot{r} + (C(q,\dot{q}) + K)r = 0$
  - What if we try the Lyapunov function $V = \frac{1}{2}r^T M(q)r$?

* $\dot{V} = r^T M(q)\dot{r} + \frac{1}{2} r^T \dot{M}(q,\dot{q}) r$
$= r^T(-C(q,\dot{q})r - Kr) + \frac{1}{2} r^T \dot{M} r$
$= -r^T K r + \frac{1}{2} r^T (\dot{M}(q,\dot{q}) - 2C) r$
$= -r^T K r$
    * This is negative definite in r, but we still need $\tilde{q}$, so this is still insufficient
- If we define $r = \dot{\tilde{q}} + \Lambda \tilde{q}$, where Λ is a diagonal positive definite matrix, then if $r = 0$ for all time, then $\dot{\tilde{q}} + \Lambda \tilde{q} = 0$, which means $\dot{\tilde{q}} = -\Lambda \tilde{q} \implies \tilde{q}(t) = e^{-\Lambda t}\tilde{q}(0)$ which goes to zero
- Choose a new controller $u = M(q)(\ddot{q}^r + \Lambda \dot{\tilde{q}}) + C(q,\dot{q})(\dot{q}^r + \Lambda \tilde{q}) + B(q)\dot{q} + G(q) + K(\dot{\tilde{q}} + \Lambda \tilde{q})$
- The new equations of motion are $M(q)(\ddot{\tilde{q}} + \Lambda \dot{\tilde{q}}) + C(q,\dot{q})(\dot{\tilde{q}} + \Lambda \tilde{q}) + K(\dot{\tilde{q}} + \Lambda \tilde{q}) = 0$ which, using the new definition of r, is $M(q)\dot{r} + (C(q,\dot{q}) + K)r = 0$
- Try the Lyapunov function $V = \frac{1}{2} r^T M(q) r + \tilde{q}^T P \tilde{q}$, where P is a symmetric positive definite matrix that is to be determined
    - V is positive definite at $(\tilde{q}, r) = (0,0)$, and $\tilde{q} = 0 \implies r = \dot{\tilde{q}} + \Lambda \tilde{q} = \Lambda \tilde{q}$, so this is equivalent to being positive definite at $(\tilde{q}, \dot{\tilde{q}}) = 0$
    - $\dot{V} = -r^T K r + 2\tilde{q}^T P \dot{\tilde{q}}$
$= -(\dot{\tilde{q}} + \Lambda \tilde{q})^T K (\dot{\tilde{q}} + \Lambda \tilde{q}) + 2\tilde{q}^T P \dot{\tilde{q}}$
$= -\dot{\tilde{q}}^T K \dot{\tilde{q}} - \dot{\tilde{q}}^T K \Lambda \tilde{q} - \tilde{q}^T \Lambda K \dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} + 2\tilde{q}^T P \dot{\tilde{q}}$
$= -\dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} - 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + 2\tilde{q}^T P \dot{\tilde{q}}$
    - Now if we choose $P = \Lambda K$, then $\dot{V} = -\dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} = -\begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}^T \begin{bmatrix} \Lambda K \Lambda & 0 \\ 0 & K \end{bmatrix} \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}$
    - Therefore we have $\dot{V}$ negative definite at the equilibrium, so the closed-loop system is asymptotically stable

Lecture 33, Nov 28, 2025

Linear Parametrization of the Robot Model

- How can we learn the parameters of the system?
- The robot model can be "factorized" as $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B(q)\dot{q} + G(q) = Y(q,\dot{q},\ddot{q})\Phi$, where $Y(q,\dot{q},\ddot{q})$ is a *regressor* matrix that models the structure of the robot, and Φ is a vector containing all the physical parameters of the robot
    - $Y(q,\dot{q},\ddot{q})$ is considered known, since it's based on the structure of the robot, which we have exactly even if we don't know the exact system parameters
    - Φ is unknown, and an adaptive controller would need to learn this parameter vector
- Example: for the pendulum cart $D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} M\ddot{q}_1 - m_2 l\cos(q_2)\ddot{q}_2 + m_2 l \sin(q_2)\dot{q}_2^2 \\ -m_2 l\cos(q_2)\ddot{q}_1 + (I + m_2 l^2)\ddot{q}_2 - m_2 l g \sin(q_2) \end{bmatrix}$
    - Identify the independent parameters: $M, m_2 l, I + m_2 l^2 = \bar{I}$
        * The fewer independent parameters we have, the better, so in this case because we never see m_2 and l on their own, we group them into a single parameter
    - We can factor into $\begin{bmatrix} \ddot{q}_1 & -\cos(q_2)\ddot{q}_2 + \sin(q_2)\dot{q}_2^2 & 0 \\ 0 & -\cos(q_2)\ddot{q}_1 - g\sin(q_2) & \ddot{q}_2 \end{bmatrix} \begin{bmatrix} M \\ m_2 l \\ \bar{I} \end{bmatrix}$

Lecture 34, Dec 1, 2025

Passivity-Based Control With Adaptation

- Now we will attempt to use passivity-based control while learning the system dynamics, resulting in the *Slotine-Li controller*

- Let $r = \dot{\tilde{q}} + \Lambda\tilde{q}, v = \dot{q}^r - \Lambda\tilde{q}, a = \ddot{q}^r - \Lambda\dot{\tilde{q}} = \dot{v}$, where $\tilde{q} = q - q^r$ (note the definition of $\tilde{q}$ is reversed compared to last lecture!)
  - Notice that $\dot{q} = r + v, \ddot{q} = \dot{r} + a$
- Substituting the above: $M(q)(\dot{r} + a) + C(q, \dot{q})(r + v) + B(q)\dot{q} + G(q) = u$
  $\implies M(q)\dot{r} + C(q, \dot{q})r = u - (M(q)a + C(q, \dot{q})v + B(q)\dot{q} + G(q))$
  - The left hand side is the same as what we had for the original passivity-based control, but now we have an extra term on the right
- Recall the linear parametrization: $M(q)a + C(q, \dot{q})v + B(q)\dot{q} + G(q) = Y(q, \dot{q}, a, v)\Theta$, where Y is the known regressor matrix, and Θ is the minimal set of parameters for the system
  - The model becomes $M(q)\dot{r} + C(q, \dot{q})r = u - Y(q, \dot{q}, a, v)\Theta$
- Choose a controller $u = u_s + u_a = -Kr + Y(q, \dot{q}, a, v)\hat{\Theta}$ consisting of a stabilizing and an adaptive term, where $\hat{\Theta}$ is our estimate of the system parameters
  - This results in the closed-loop system $M(q)\dot{r} + (C(q, \dot{q}) + K)r = Y(q, \dot{q}, a, v)\tilde{\Theta}$ where $\tilde{\Theta} = \hat{\Theta} - \Theta$ is the parameter estimation error
- We will apply Lyapunov analysis to see what adaptive control policy gives us an asymptotically stable system, using $V = \frac{1}{2}r^T M(q)r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2}\tilde{\Theta}^T \Gamma^{-1}\tilde{\Theta}$ where Γ, Λ, K are symmetric positive definite matrices
  - This is trivially positive definite at the equilibrium $(r, \tilde{q}, \tilde{\Theta}) = (0, 0, 0)$
  - $\dot{V} = r^T M(q)\dot{r} + \frac{1}{2}r^T \dot{M}(q, \dot{q})r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \tilde{\Theta}^T \Gamma^{-1}\dot{\tilde{\Theta}}$
    $= r^T(-C(q, \dot{q})r - Kr) + \frac{1}{2}r^T \dot{M}(q, \dot{q})r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + r^T Y(q, \dot{q}, a, v)\tilde{\Theta} + \tilde{\Theta}^T \Gamma^{-1}\dot{\tilde{\Theta}}$
    $= -r^T Kr + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \tilde{\Theta}^T Y^T(q, \dot{q}, a, v)r + \tilde{\Theta}^T \Gamma^{-1}\dot{\tilde{\Theta}}$
    $= -r^T Kr + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \tilde{\Theta}^T \left(Y^T(q, \dot{q}, a, v)r + \Gamma^{-1}\dot{\tilde{\Theta}}\right)$
  - This suggests that we should choose $\dot{\tilde{\Theta}} = \dot{\hat{\Theta}} = -\Gamma Y^T r$, then we can make the last term disappear
    * This is the classic gradient law for parameter adaptation
    * Γ is the learning rate, which decides how fast our parameter estimates converges
  - If we do this, then $\dot{V} = -r^T Kr + 2\tilde{q}^T \Lambda K \dot{\tilde{q}}$
    $= \cdots$ (see notes from lecture 32)
    $= -\begin{bmatrix}\tilde{q}^T & \dot{\tilde{q}}^T\end{bmatrix}\begin{bmatrix}\Lambda K \Lambda & 0 \\ 0 & K\end{bmatrix}\begin{bmatrix}\tilde{q} \\ \dot{\tilde{q}}\end{bmatrix}$
- Right now, we have $\dot{V} \leq 0$ so we know the equilibrium is stable, however since our state also includes $\tilde{\Theta}$ this is not fully asymptotically stable
  - By LaSalle, we know that $\dot{V} \to 0$, which means that $(\tilde{q}, \dot{\tilde{q}}) \to (0, 0)$, i.e. we can track the reference
  - However, the parameter estimates themselves do not necessarily converge
- We need a *persistence of excitation* condition on Y to get $\tilde{\Theta} \to 0$
  - Intuitively this means that we need to make the robot do "exciting" behaviours, i.e. behaviours which expose all the parameters of the robot, in order to have the parameters converge
  - Otherwise we can track the reference, but we might not have enough information to fully learn the robot model; e.g. we can command the robot to track a reference signal that only moves a single joint, which means we can't learn about the rest of the system
  - This is a graduate level topic

Review – 2024 Final Q4

- A pendulum has equation of motion $mr^2\ddot{\theta} + k\dot{\theta} + mgr\sin\theta = 0$, which has potential energy $U(\theta) = \frac{g}{r}(1 - \cos\theta)$; either prove that the equilibrium $(\theta^*, \dot{\theta}^*) = (0, 0)$ (i.e. the hanging down position) is asymptotically stable, or stable but not asymptotically stable
  - Let the state $x = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}$, then $\dot{x} = \begin{bmatrix}x_2 \\ -\frac{k}{mr^2}x_2 - \frac{g}{r}\sin x_1\end{bmatrix}$

- Since this is a mechanical system, we can try using the total energy: $V(x) = \frac{1}{2}x_2^2 + \frac{g}{r}(1 - \cos\theta)$
  * The first term is a potential energy; nevermind that we don't have m, it doesn't matter
- We can check that around $(0, 0)$ this is positive definite if we restrict the angle to $(-\pi, \pi)$
- $\dot{V} = x_2\dot{x}_2 + \frac{g}{r}\sin(x_1)\dot{x}_1$

  $= x_2\left(-\frac{k}{mr^2}x_2 - \frac{g}{r}\sin x_1\right) + \frac{g}{r}\sin(x_1)x_2$

  $= -\frac{k}{mr^2}x_2^2$
- This is negative semidefinite; by Lyapunov's theorem, this equilibrium is at least stable, but we need LaSalle to conclude that it is asymptotically stable
- $\dot{V} \equiv 0, \forall t \implies \dot{\theta} = 0 \implies \ddot{\theta} = 0 \implies mgr\sin\theta = 0 \implies \theta = 0$
- Therefore the equilibrium is asymptotically stable