PHYSICS OF MOTION: DYNAMICS

Gateway to Robotics

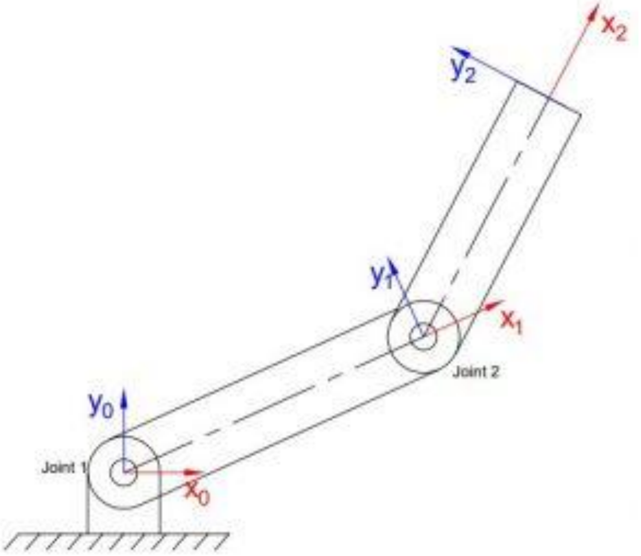
Roberto Martin-Martin

Assistant Professor of Computer Science.

https://pollev.com/robertomartinmartin739

Exercise

- Assuming that the end-effector configuration of a 2 DoF robot has only two variables, x and y
- $FK(q) = \begin{pmatrix} q_0^2 \\ q_0 q_1 \end{pmatrix}$
- Reminder:
  - Inverse of a 2x2 matrix of the form $\mathrm{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is $\mathrm{M}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$



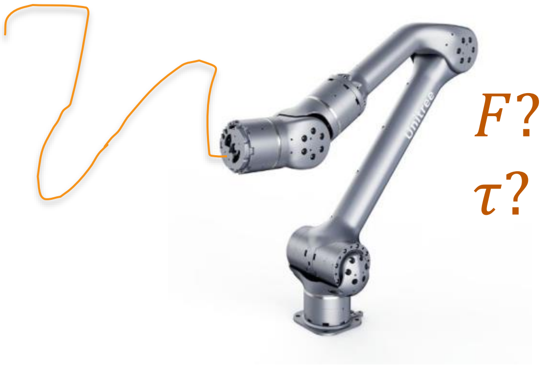
Recap

- Inverse Kinematics
  - Mapping desired end-effector configuration (pose) to joint values to achieve it
  - Not as easy as forward kinematics!
  - It may have 0 solutions, 1 solution, infinite solutions
  - Usually not a closed form → Iterative numerical process

What will you learn today?

- Equations of motion in robots
- Elements of the equation
- Intuition behind each element
- Computing dynamics (bird’s eye view)

Why does this matter?



- Until now:
  - We have been looking at motion (only)
- But we know from Newton's laws that F=ma
- We also saw that motors generate forces (torques)
- What forces/torques do we need to generate to create the desired motion?
- What forces are already acting on the robot and creating motion (disturbances)
  - What disturbances act on the robot in the video?

Equation of Motion of a Robot

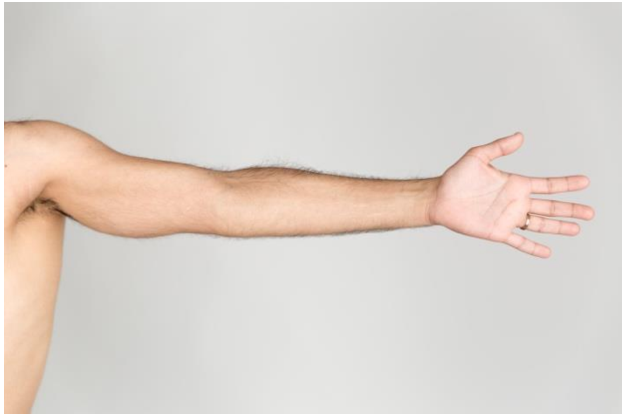
$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

- τ is the applied torque on each joint
- q is our joint coordinates, generally the joint angles of the serial link manipulator.
- $\dot{q}$ the joint velocities, the rates of change of the joint coordinates and
- $\ddot{q}$ is the joint acceleration.
- g is a term which represents the torque that's due to the gravity acting on the robot manipulator and gravity is a function only of the joint coordinates q.
- M is the inertia matrix and it is a function only of the joint coordinates multiply by the joint accelerations
- C is the

The Gravity Term

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

- A vector of length = number of joints
- Depends <u>only</u> on the configuration (pose) of the robot
  - No dependency with velocity/acceleration!
- Can be computed based on the dynamic parameters of the robot



What are the Dynamic Parameters of a Robot?

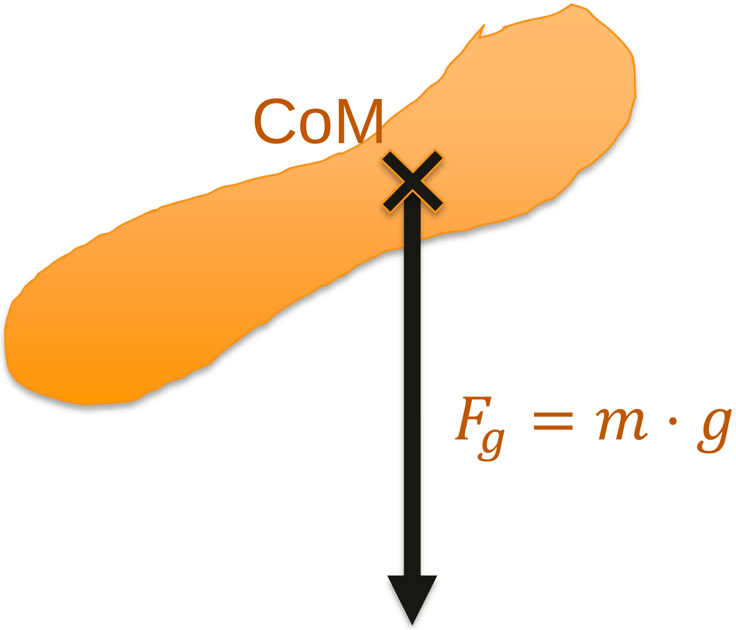
- For each link:
  - Mass of the link
  - Center of mass
  - Inertia matrix
- Hard to obtain!
  - Requires knowing the material, construction, density…
    OR
  - Infer the parameters (dynamics parameter estimation) by applying forces/torques and measuring motion

The Inertia Matrix of a Rigid Body

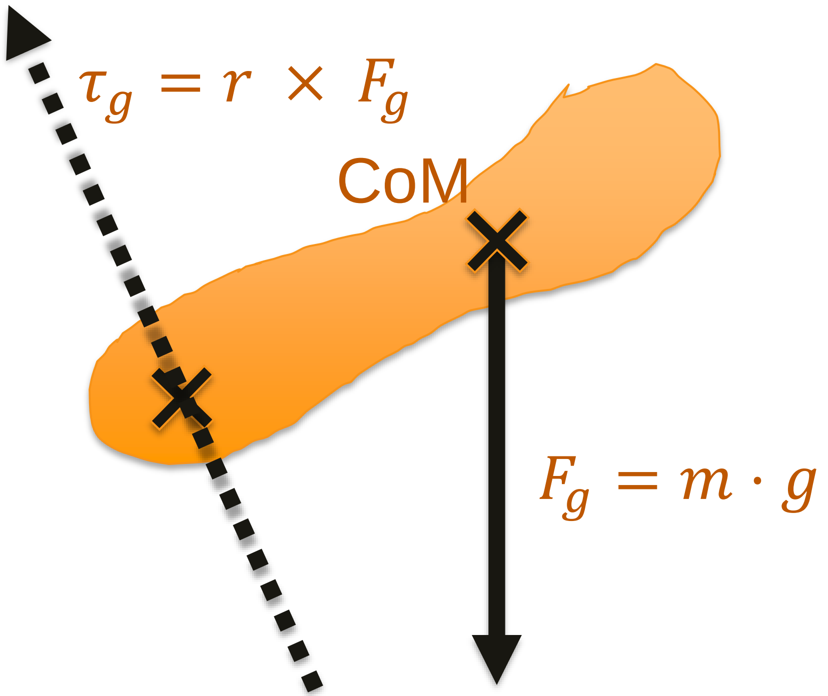
- Moment of inertia: quantity that determines the torque necessary to achieve a desired angular acceleration about an axis

$$\bar{\tau} = I \cdot \bar{\alpha}$$

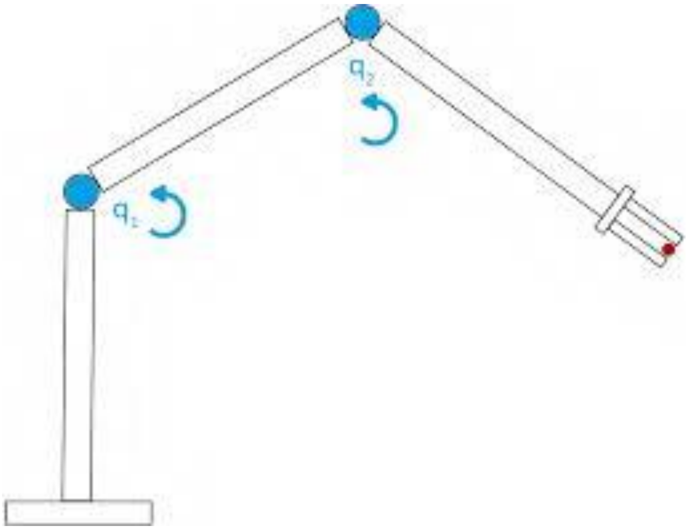
Gravity Force on a Rigid Body



Gravity Torque on a Joint from a Robot Link



Back to the Gravity Term

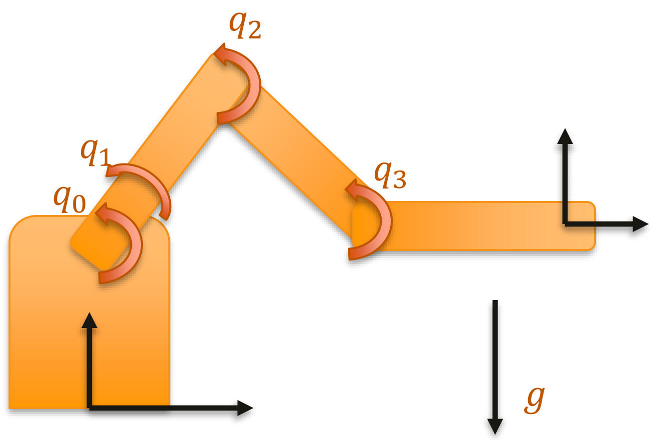


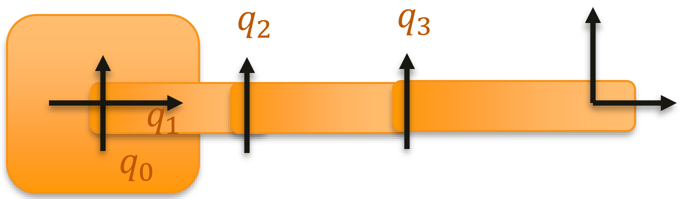
- $g(q)$: Propagate ”up” the contribution of the weight of each link on each of the joints
- $g_i(q)$ will depend on:
  - Masses of the links “after” that joint
  - All joint values

https://pollev.com/robertomartinmartin739

Exercise

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$



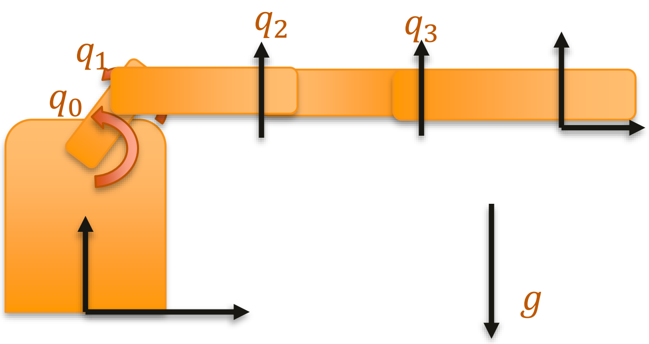


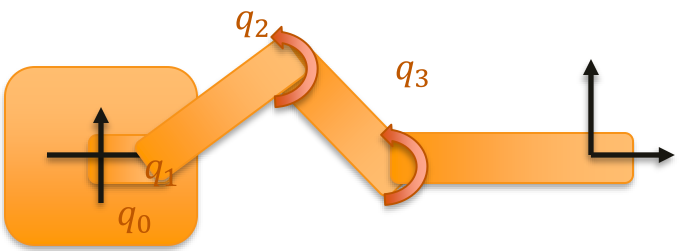
top-down view

https://pollev.com/robertomartinmartin739

Exercise

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

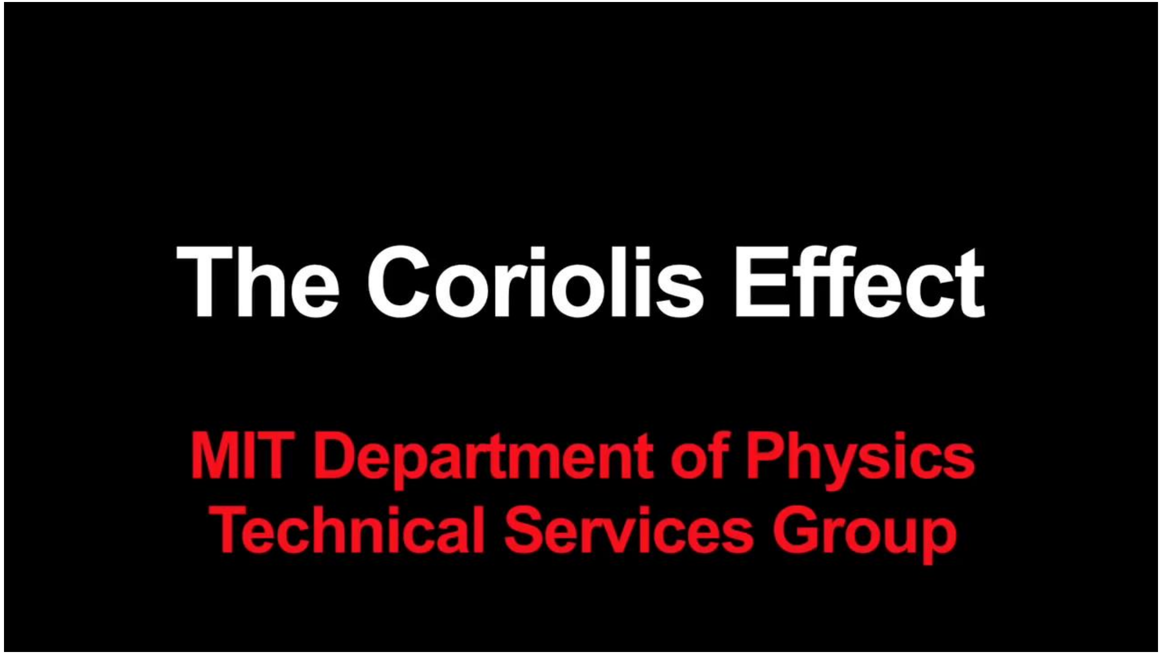




top-down view

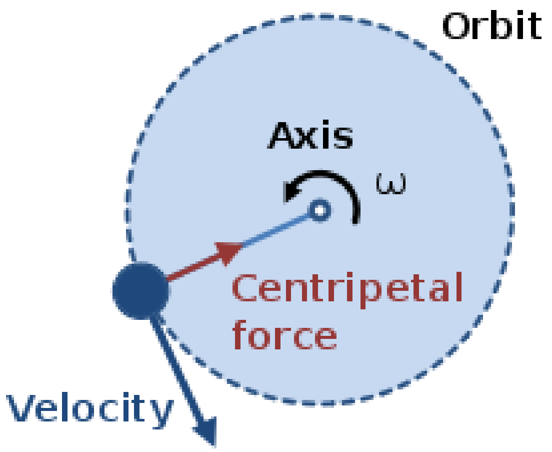
Coriolis Effect (Force)

- $C(q, \dot{q})\dot{q}$
- Fictitious force that act on objects when the reference frame is rotating
- We need to take it into account to compensate for it when moving our robot arm



Centripetal Effect (Force)

- $C(q, \dot{q})\dot{q}$
- Force that keeps a link rotating instead of "flying away"
- It acts on each joint due to other links rotating



The Mass Matrix

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

- Mass that represents the <u>inertia</u> of the robot with respect to motion of each joint
- n x n matrix (n is the number of joints)
- Positive semidefinite and symmetric
- “How much torque is required to accelerate with one joint”
- “How much torque the acceleration of a joint causes on another joint”

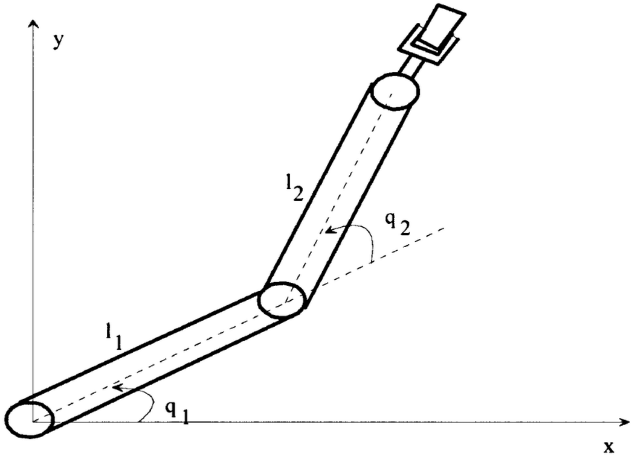
The Mass Matrix – Main Diagonal

- $M(q) = \begin{pmatrix} m_{11} & & \\ & m_{ii} & \\ & & m_{nn} \end{pmatrix}$
- m_{ii}
  - Inertia of a rigid body (how much torque do I need to move that link)
  - Directly related to the mass of each link
    - Increasing the mass → increases that term and consecutive

https://pollev.com/robertomartinmartin739

Exercise

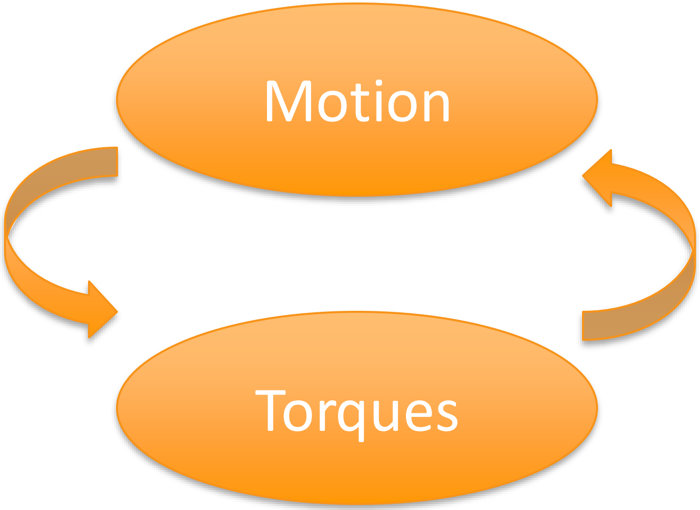
$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g$$



- $M(q) = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$, all non-zero
- $\tau = M(q)\ddot{q}$

Forward Dynamics and Inverse Dynamics

- Forward Dynamics:
  - Compute the acceleration generated by some torques (given pose and velocity)
  - Useful for simulation
- Inverse Dynamics:
  - Compute the torques necessary to create some given accelerations (given pose and velocity)
  - Useful for control



Forward Dynamics

- The robot is at a pose (from previous time step)
- I apply some joint torque
  - This is usually what I can control in my robot/motors!
- What is the acceleration in the next time step (and the pose)?

Forward Dynamics

What are the joint accelerations created by some torque?

$$\ddot{q} = \mathbf{M}^{-1}(q)\{\tau - \mathbf{C}(q,\dot{q})\dot{q} - g(q)\}$$

$$\dot{q} = \int \ddot{q}\, dt$$

$$q = \int \dot{q}\, dt$$

- aka the **forward** dynamics
  - maps torque to motion $\tau \rightarrow (q, \dot{q}, \ddot{q})$
  - used to simulate robot motion

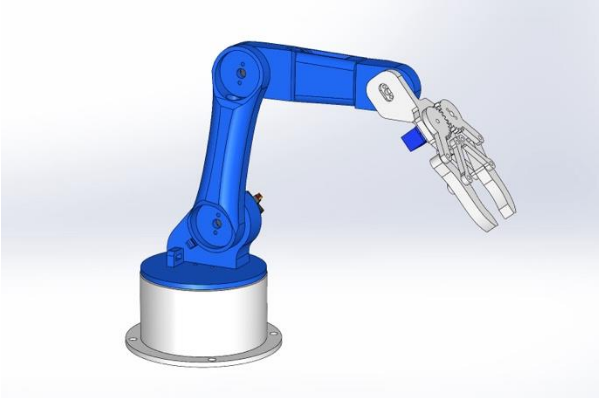
Inverse Dynamics

- The robot is at a pose (from previous time step)
- I know where I want to be in the next time step
- What is the joint torque to achieve the desired acceleration (and pose)?

How do we compute (Inverse) Dynamics?

What are the joint torques necessary to create some joint acceleration?

- Newton-Euler Approach
  - Iterative-recursive process
  - Compute the torques necessary to create some given accelerations (given pose and velocity)
  - **Step 1: Forward iteration**
    - <u>We determine the Cartesian pose, velocity and acceleration of each CoM</u>
    - Given q, $\dot{q}$, $\ddot{q}$ for link i=1 to n:
      - Compute the Cartesian velocity of link i as the composition of the Cartesian velocity of link i-1 and the motion caused by $\dot{q}_i$
      - Compute the Cartesian acceleration of link i as the composition of the Cartesian acceleration of link i-1, the motion caused by $\ddot{q}_i$ and a velocity-product term



How do we compute (Inverse) Dynamics?

What are the joint torques necessary to create some joint acceleration?

- Newton-Euler Approach
  - Iterative-recursive process
  - Compute the torques necessary to create some given accelerations (given pose and velocity)
  - **Step 2: Backward iteration (from n to 1)**
    - We determine the joint torque necessary to create the previously computed Cartesian motion
    - for link i=n to 1:
      - Compute the Cartesian wrench of link i as the composition of the Cartesian wrench of link i+1 and the wrench necessary to create the Cartesian velocity and acceleration of link i
      - Compute the torque τ_i as the component of the Cartesian wrench of link i around the joint axis i