**Topics Covered:**

- Dynamics of Manipulators
- Equations of Motion
- Control

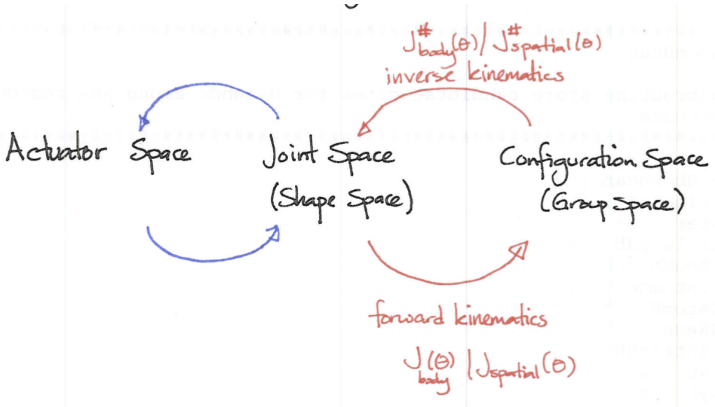
**Additional Reading:**

- Craig 6.8-6.9
- LP 8.1-8.2

# Dynamics of Manipulators

So far in the course, we have only discussed *kinematic* models of manipulators. These models relate joint position/motion to end-effector position/motion.

This was visualized by the diagram:

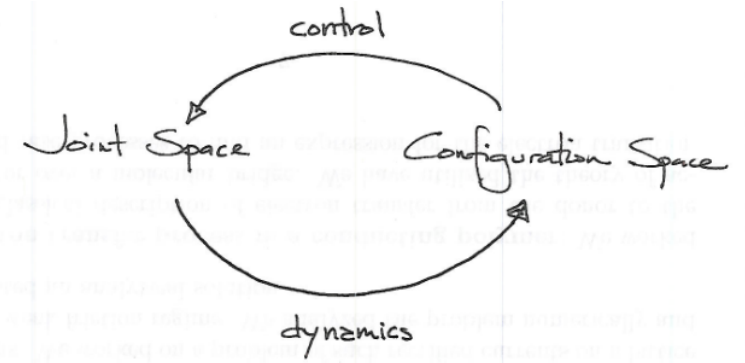



$$\text{positions} = \begin{cases} \text{forward kinematics} \\ \text{inverse kinematics} \end{cases}, \quad \text{velocities} = \begin{cases} J_b(\theta), J_s(\theta) \\ J_b^\dagger(\theta), J_s^\dagger(\theta) \end{cases}$$

But what about accelerations?? Well that will be the focus of this lecture.

**Aside:** We're not going to talk about the left-hand side of this diagram. These arrows deal with the control of servomechanics and linear actuators. This is more of a classic control problem; *how do we realize a desired torque or linear force using a given motor?*.

We are interested in what happens if we consider the second derivative of the red arrows:




By *dynamics* of a manipulator, we mean how the manipulator moves in response to actuator forces. By *control* of a manpulator, we mean how to generate actuator forces to achieve desired motion.

1. Dynamics
    - Equations of Motion
    - Lagrangian mechanics
2. Control
    - Position control
    - Force control

# Equations of Motion

The dynamic equations associated with *robot dynamics* are referred to as the *equations of motion* and are a set of second-order differential equations of the form:

$$\tau = M(\theta)\ddot{\theta} + \underbrace{C(\theta,\dot{\theta})\dot{\theta} + G(\theta)}_{h(\theta,\dot{\theta})}$$

where $\theta = \mathbb{R}^n$ is the vector of joint angles, $\tau = \mathbb{R}^n$ is the vector of joint torques (and forces if linear actuators), $M(\theta) = \mathbb{R}^{n\times n}$ is a symmetric positive-definite mass matrix, $C(\theta,\dot{\theta}) = \mathbb{R}^{n\times n}$ is the Coriolis matrix, and $G(\theta) = \mathbb{R}^n$ is the gravity vector. As shown, the Coriolis matrix and the gravity vector are often lumped together into a generalized vector $h$. As stated in LP, these expressions can be "extraordinarily complex" despite their simple form.

Similar to our notion of forwards and inverse kinematics, we have notions of forwards and inverse *dynamics*:

$$\begin{aligned} \textit{forwards dynamics:} \quad & \ddot{\theta} = M^{-1}(\theta)(\tau - h(\theta, \dot{\theta})) \\ \textit{inverse dynamics:} \quad & \tau = M(\theta)\ddot{\theta} + h(\theta, \dot{\theta}) \end{aligned}$$

There are two typical ways to derive these dynamic equations:

1. Newton-Euler formulation
2. Lagrangian dynamics formulation

## Lagrangian Formulation

**Definition: Lagrangian function**

A Lagrangian function $\mathcal{L}(\theta, \dot{\theta})$ is defined as the overall system's kinetic energy $\mathcal{K}(\theta, \dot{\theta})$ minus the potential energy $\mathcal{P}(\theta)$:

$$\mathcal{L}(\theta, \dot{\theta}) = \mathcal{K}(\theta, \dot{\theta}) - \mathcal{P}(\theta)$$

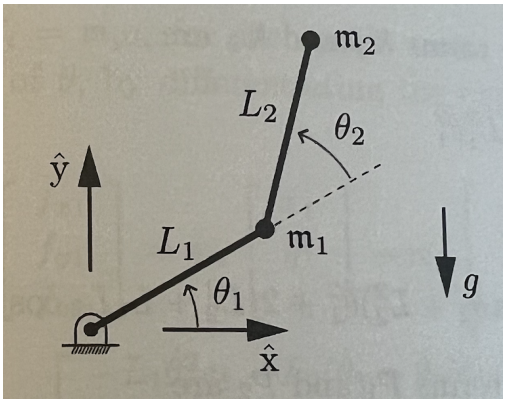
This formulation allows us to derive a system's equations of motion in terms of the Lagrangian:

$$\tau = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta}$$

with $\tau \in \mathbb{R}^n$ being the generalized forces acting on the system (in joint-space).

### Example

Consider the two-link planar manipulator that we've been working with.



We can use our forward kinematics to solve for the position and velocity of each mass. First, for the first mass:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 \\ L_1 \sin\theta_1 \end{bmatrix}, \quad \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} -L_1 \sin\theta_1 \\ L_1 \cos\theta_1 \end{bmatrix} \dot{\theta}_1$$

Now, for the second mass:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{bmatrix},$$
$$\begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

The Lagrangian is then of the form:

$$L(\theta, \dot{\theta}) = (\mathcal{K}_1 - \mathcal{P}_1) + (\mathcal{K}_2 - \mathcal{P}_2)$$

where kinetic energy is captured by the formula $\mathcal{K} = \frac{1}{2}mv^2$ and potential energy is the formula $\mathcal{P} = mgh$. Specifically, for our masses, we have the following kinetic energy terms:

$$\begin{aligned} \mathcal{K}_1 &= \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2) \\ &= \frac{1}{2} m_1 L_1^2 \dot{\theta}_1^2 \end{aligned}$$

$$\begin{aligned} \mathcal{K}_2 &= \frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2) \\ &= \frac{1}{2} m_2 \left( (L_1^2 + 2L_1 L_2 \cos\theta_2 + L_2^2)\dot{\theta}_1^2 + 2(L_2^2 + L_1 L_2 \cos\theta_2)\dot{\theta}_1 \dot{\theta}_2 + L_2^2 \dot{\theta}_2^2 \right) \end{aligned}$$

Lastly, our potential energy terms are:

$$\begin{aligned} \mathcal{P}_1 &= m_1 g y_1 \\ &= m_1 g L_1 \sin\theta_1 \end{aligned}$$

$$\begin{aligned} \mathcal{P}_2 &= m_2 g y_2 \\ &= m_2 g (L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)) \end{aligned}$$

The individual equations of motion for each joint can be represented as:

$$\tau_i = \frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} \right) - \frac{\partial \mathcal{L}}{\partial \theta_i}$$

Thus, plugging in our terms for the Lagrangian, and taking the partial derivatives yields:

$$\begin{aligned} \tau_1 &= (m_1 L_1^2 + m_2 (L_1^2 + 2L_1 L_2 \cos\theta_2 + L_2^2))\ddot{\theta}_1 + m_2 (L_1 L_2 \cos\theta_2 + L_2^2)\ddot{\theta}_2 \\ &\quad - m_2 L_1 L_2 \sin\theta_2 (2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^2) + (m_1 + m_2) L_1 g \cos\theta_1 + m_2 g L_2 \cos(\theta_1 + \theta_2), \\ \tau_2 &= m_2 (L_1 L_2 \cos\theta_2 + L_2^2)\ddot{\theta}_1 + m_2 L_2^2 \ddot{\theta}_2 + m_2 L_1 L_2 \sin\theta_2 \dot{\theta}_1^2 + m_2 g L_2 \cos(\theta_1 + \theta_2) \end{aligned}$$

We can rewrite these equations to match the general form:

$$\tau = M(\theta)\ddot{\theta} + \underbrace{c(\theta,\dot{\theta}) + g(\theta)}_{h(\theta,\dot{\theta})}$$

$$\tau = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix} m_1 L_1^2 + m_2(L_1^2 + 2L_1 L_2 \cos\theta_2 + L_2^2) & m_2(L_1 L_2 \cos\theta_2 + L_2^2) \\ m_2(L_1 L_2 \cos\theta_2 + L_2^2) & m_2 L_2^2 \end{bmatrix} \begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix} -m_2 L_1 L_2 \sin\theta_2 (2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \\ m_2 L_1 L_2 \sin\theta_2 \dot{\theta}_1^2 \end{bmatrix} + \begin{bmatrix} (m_1 + m_2) L_1 g \cos\theta_1 + m_2 g L_2 \cos(\theta_1 + \theta_2) \\ m_2 g L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

# Control

## Feedforward Control

As mentioned earlier, one method for controlling a manipulator is to design the desired motions in terms of position, velocity, and acceleration, and then calculate the necessary torques to achieve these motions using our equations of motion. This is known as *feedforward control* and has the following pseudocode:

```
time = 0                                                    // dt =  cycle time
loop
    [qd, qdotd, qdotdotd] = trajectory(time)                // trajectory gen.
    tau = M(qd)*qdotdotd + C(qd, qdotd)*qdotd + G(qd)       // calculate dynamics
    commandTorque(tau)
    time = time + dt
end loop
```

## Feedback Control

Since our equations of motion are often complex and inaccurate, we can use feedback control to correct for these inaccuracies. This is done by measuring the actual joint angles and velocities, and then using a PID controller to generate the necessary torques to correct for any errors. This is known as *feedback control* and has the following pseudocode:

```
time = 0                                                    // dt =  cycle time
eint = 0                                                    // integral error
qprev = readJointAngles()                                   // init. joint angle q
loop
    [qd, qdotd] = trajectory(time)                          // trajectory gen.
    q = readJointAngles()                                   // read joint angle q
    qdot = (q-qprev)/dt                                     // calculate joint vel
```

```
    qprev = q

    e = qd - q                                          // position error
    edot = qdotd - qdot                                 // velocity error
    eint = eint + e*dt                                  // integral error

    tau = Kp*e + Ki*eint + Kd*edot                      // PID control
    commandTorque(tau)

    time = time + dt
end loop
```

PID control is extremely common in the field of robotics research, and is summarized by the following torque law:

$$\tau = K_p e + K_i \int e(t)dt + K_d \dot{e}$$

with the error dynamics:

$$e = q_d - q, \quad \dot{e} = \dot{q}_d - \dot{q}$$

Here, $K_p$, $K_d$ and $K_i$ are positive control gains. The proportional gain $K_p$ acts as a virtual spring that pulls the system towards the desired position, the derivative gain $K_d$ acts as a virtual damper that reduces the velocity error, and the integral gain $K_i$ acts as a virtual integrator that corrects for steady-state errors.

There is a large literature of work that explores how to select these three gains to avoid overshoot, oscillations, and other undesirable behaviors. In general, the tuning procedure is as follows: