Robot Modeling

Optimization-based Robot Control

Andrea Del Prete

University of Trento

Schedule

Classroom Code: 2ym4lka

First week:

1. Modeling ($\approx$ 1 hour)
2. Joint-Space Control ($\approx$ 1 hour)
3. Task-Space Control ($\approx$ 1 hour)
4. Implementation ($\approx$ 1 hour)
5. Coding ($\approx$ 2 hours)

Schedule

Classroom Code: 2ym4lka

First week:

1. Modeling ($\approx$ 1 hour)
2. Joint-Space Control ($\approx$ 1 hour)
3. Task-Space Control ($\approx$ 1 hour)
4. Implementation ($\approx$ 1 hour)
5. Coding ($\approx$ 2 hours)

Second week:

1. Limits of Reactive Control ($\approx$ 0.5 hour)
2. Linear Inverted Pendulum Model $\approx$ 0.5 hour)
3. Center of Mass Trajectory Generation ($\approx$ 1 hour)
4. Implementation ($\approx$ 1 hour)
5. Coding: CoM trajectory optimization ($\approx$ 1 hour)
6. Coding: walking with TSID ($\approx$ 2 hours)

Options for coding

- use my 11 GB VM (*VMware Fusion*, compatible with *VirtualBox*)

Options for coding

- use my 11 GB VM (*VMware Fusion*, compatible with *VirtualBox*)
- install TSID and dependencies (available on *github.com*):
    - TSID (branch *devel*)
    - Pinocchio
    - Gepetto-viewer
    - Gepetto-viewer-corba

Table of contents

Notation & Definitions

State $\triangleq x$.

Control $\triangleq u$.

Notation & Definitions

State $\triangleq x$.

Control $\triangleq u$.

Identity matrix $\triangleq I$.
Zero matrix $\triangleq 0$.
Matrix size written as index (when needed), e.g., I_3.

Notation & Definitions

State $\triangleq x$.

Control $\triangleq u$.

Identity matrix $\triangleq I$.
Zero matrix $\triangleq 0$.
Matrix size written as index (when needed), e.g., I_3.

Fully actuated system: number of actuators = number of degrees of freedom (e.g., manipulator).

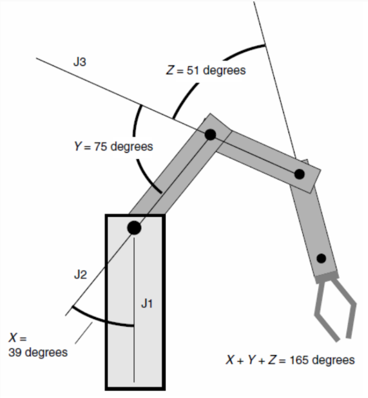
Under actuated system: number of actuators < number of degrees of freedom (e.g., legged robot, quadrotor).

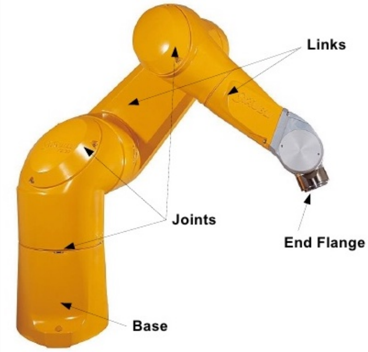
Modeling Robot Manipulators

Robot Manipulators: Fixed-base Robots

Robot base is (typically) fixed (e.g., attached to the ground).

Configuration represented by vector $q \in \mathbb{R}^{n_q}$ of (relative) joint angles.
Velocity represented by vector $v = \dot{q} \in \mathbb{R}^{n_v}$ of (relative) joint velocities.





Actuation Models

Typically each joint driven by 1 actuator (e.g., electric, hydraulic, pneumatic).

Actuation Models

Typically each joint driven by 1 actuator (e.g., electric, hydraulic, pneumatic).

Actuator models:

- velocity source
- acceleration source
- torque source
- ...

Actuation Models

Typically each joint driven by 1 actuator (e.g., electric, hydraulic, pneumatic).

Actuator models:

- velocity source
- acceleration source
- torque source
- ...

Appropriate model depends on robot and task.

Velocity Input

Model actuators as velocity sources.

- Good for hydraulic.
- Good for electric in certain conditions (e.g., manipulators).

Velocity Input

Model actuators as velocity sources.

- Good for hydraulic.
- Good for electric in certain conditions (e.g., manipulators).

$$x \triangleq q$$

$$u \triangleq v$$

Dynamics is simple integrator:

$$\dot{x} = u$$

Acceleration Input

Model actuators as acceleration sources.

- Good for electric w/o large contact forces.

Acceleration Input

Model actuators as acceleration sources.

- Good for electric w/o large contact forces.

$$x \triangleq (q, v)$$

$$u \triangleq \dot{v}$$

Dynamics is double integrator:

$$\begin{bmatrix} \dot{q} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u$$

Torque Input

Model actuators as torque sources.

Good for electric w/o high-friction gear box—rarely the case (unfortunately).

$$x \triangleq (q, v) \qquad u \triangleq \tau$$

Torque Input

Model actuators as torque sources.

Good for electric w/o high-friction gear box—rarely the case (unfortunately).

$$x \triangleq (q, v) \qquad u \triangleq \tau$$

Dynamics of fully-actuated mechanical system (e.g., manipulator):

$$M(q)\dot{v} + h(q, v) = \tau,$$

where

- $M(q) \in \mathbb{R}^{n_v \times n_v} \triangleq$ (positive-definite) mass matrix,
- $h(q, v) \in \mathbb{R}^{n_v} \triangleq$ bias forces,
- $\tau \in \mathbb{R}^{n_v} \triangleq$ joint torques.

Torque Input: Fully-Actuated Dynamics

Bias forces sometimes decomposed as:

$$h(q, v) = C(q, v)v + g(q)$$

- $C(q, v)v \triangleq$ Coriolis and centrifugal effects
- $g(q) \triangleq$ gravity forces

Torque Input: Fully-Actuated Dynamics

Bias forces sometimes decomposed as:

$$h(q, v) = C(q, v)v + g(q)$$

- $C(q, v)v \triangleq$ Coriolis and centrifugal effects
- $g(q) \triangleq$ gravity forces

Nonlinear state-space dynamics:

$$\begin{bmatrix} \dot{q} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -M(q)^{-1} h(q, v) \end{bmatrix} + \begin{bmatrix} 0 \\ M(q)^{-1} \end{bmatrix} u$$

Inverse VS Forward Dynamics

Forward Dynamics

Given q, v, τ compute $\dot{v}$:

$$\dot{v} = M(q)^{-1}(\tau - h(q, v))$$

Problem solved by simulators.

Inverse VS Forward Dynamics

Forward Dynamics

Given q, v, τ compute $\dot{v}$:

$$\dot{v} = M(q)^{-1}(\tau - h(q, v))$$

Problem solved by simulators.

Inverse Dynamics

Given $q, v, \dot{v}$ compute τ:

$$\tau = M(q)\dot{v} + h(q, v)$$

Problem solved by controllers.

Modeling Robots in Contact

Adding Contact Forces

If robot in contact with surrounding $\rightarrow$ contact forces $f \in \mathbb{R}^{n_f}$:

$$M(q)\dot{v} + h(q, v) = \tau + J(q)^\top f,$$

where $J(q) \in \mathbb{R}^{n_f \times n_v} \triangleq$ contact Jacobian:

Adding Contact Forces

If robot in contact with surrounding $\rightarrow$ contact forces $f \in \mathbb{R}^{n_f}$:

$$M(q)\dot{v} + h(q, v) = \tau + J(q)^\top f,$$

where $J(q) \in \mathbb{R}^{n_f \times n_v} \triangleq$ contact Jacobian:

$$J(q) = \frac{\partial c(q)}{\partial q},$$

where $c(q) : \mathbb{R}^{n_q} \rightarrow \mathbb{R}^{n_f} \triangleq$ forward geometry of contact points (i.e. function mapping joint angles to contact point positions).

Robots in Rigid Contact

Rigid contacts constrain motion.

Robots in Rigid Contact

Rigid contacts constrain motion.

$$c(q) = \text{const} \qquad \iff \qquad \text{Contact points do not move}$$

Robots in Rigid Contact

Rigid contacts constrain motion.

$$c(q) = \text{const} \qquad \Longleftrightarrow \qquad \text{Contact points do not move}$$

Differentiate:

$$J v = 0 \qquad \Longleftrightarrow \qquad \text{Contact point velocities are null}$$

$$J \dot{v} + \dot{J} v = 0 \qquad \Longleftrightarrow \qquad \text{Contact point accelerations are null}$$

Robots in Rigid Contact

Rigid contacts constrain motion.

$$c(q) = \text{const} \qquad \Longleftrightarrow \qquad \text{Contact points do not move}$$

Differentiate:

$$\begin{aligned} Jv = 0 \qquad &\Longleftrightarrow \qquad \text{Contact point velocities are null} \\ J\dot{v} + \dot{J}v = 0 \qquad &\Longleftrightarrow \qquad \text{Contact point accelerations are null} \end{aligned}$$

Introduce constraints in dynamics:

$$\begin{bmatrix} M & -J^\top & -I \\ J & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{v} \\ f \\ \tau \end{bmatrix} = \begin{bmatrix} -h \\ -\dot{J}v \end{bmatrix} \tag{1}$$

Inverse VS Forward Dynamics with Rigid Contacts

Forward Dynamics (with constraints)

Given q, v, τ compute $\dot{v}$ and f:

$$\begin{bmatrix} \dot{v} \\ f \end{bmatrix} = \begin{bmatrix} M & -J^\top \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tau - h \\ -\dot{J}v \end{bmatrix}$$

Problem solved by (bilateral) rigid contact simulators.

Inverse VS Forward Dynamics with Rigid Contacts

Forward Dynamics (with constraints)

Given q, v, τ compute $\dot{v}$ and f:

$$\begin{bmatrix} \dot{v} \\ f \end{bmatrix} = \begin{bmatrix} M & -J^\top \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tau - h \\ -\dot{J}v \end{bmatrix}$$

Problem solved by (bilateral) rigid contact simulators.

Inverse Dynamics (with constraints)

Given $q, v, \dot{v}$ compute τ and f:

$$\begin{bmatrix} \tau \\ f \end{bmatrix} = \begin{bmatrix} I & J^\top \end{bmatrix}^\dagger (M\dot{v} + h),$$

where $\dagger$ represents pseudo-inverse.

Inverse VS Forward Dynamics with Rigid Contacts

Forward Dynamics (with constraints)

Given q, v, τ compute $\dot{v}$ and f:

$$\begin{bmatrix} \dot{v} \\ f \end{bmatrix} = \begin{bmatrix} M & -J^\top \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tau - h \\ -\dot{J}v \end{bmatrix}$$

Problem solved by (bilateral) rigid contact simulators.

Inverse Dynamics (with constraints)

Given $q, v, \dot{v}$ compute τ and f:

$$\begin{bmatrix} \tau \\ f \end{bmatrix} = \begin{bmatrix} I & J^\top \end{bmatrix}^\dagger (M\dot{v} + h),$$

where $\dagger$ represents pseudo-inverse.

Implicit assumption: $\dot{v}$ satisfies constraints.

Inverse VS Forward Dynamics with Rigid Contacts

Forward Dynamics (with constraints)

Given q, v, τ compute $\dot{v}$ and f:

$$\begin{bmatrix} \dot{v} \\ f \end{bmatrix} = \begin{bmatrix} M & -J^\top \\ J & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tau - h \\ -\dot{J}v \end{bmatrix}$$

Problem solved by (bilateral) rigid contact simulators.

Inverse Dynamics (with constraints)

Given $q, v, \dot{v}$ compute τ and f:

$$\begin{bmatrix} \tau \\ f \end{bmatrix} = \begin{bmatrix} I & J^\top \end{bmatrix}^\dagger (M\dot{v} + h),$$

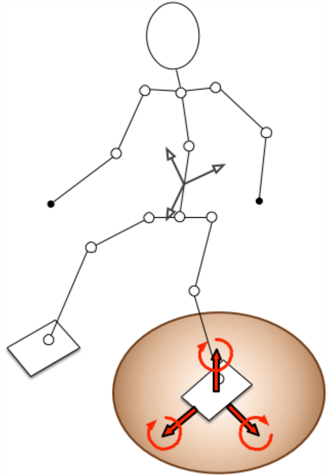
where $\dagger$ represents pseudo-inverse.

Implicit assumption: $\dot{v}$ satisfies constraints.

Primitive version of inverse-dynamics control with rigid contacts.

Modeling Legged Robots

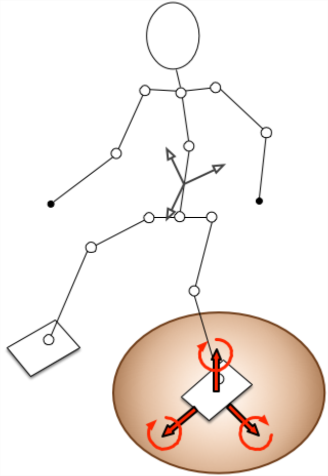
Modeling Legged (Floating-Base) Robots



PROBLEM

Joint angles not enough to describe robot configuration.

Modeling Legged (Floating-Base) Robots



PROBLEM

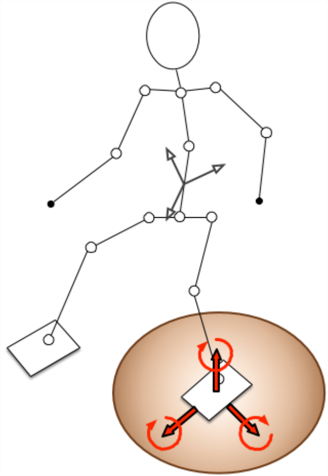
Joint angles not enough to describe robot configuration.

SOLUTION

Add pose (position + orientation) of one link (called base) w.r.t. inertial frame:

$$q = (\underbrace{x_b}_{\text{Base pose}} , \underbrace{q_j}_{\text{Joint angles}})$$

Modeling Legged (Floating-Base) Robots



PROBLEM

Joint angles not enough to describe robot configuration.

SOLUTION

Add pose (position + orientation) of one link (called base) w.r.t. inertial frame:

$$q = (\underbrace{x_b}_{\text{Base pose}} , \underbrace{q_j}_{\text{Joint angles}})$$

Now q sufficient to describe robot configuration in space.

Base Pose

$x_b \in \mathrm{SE}(3) \triangleq$ special Euclidian group, comprising any combination of

- translations: elements of $\mathbb{R}^3$,
- rotations: elements of $\mathrm{SO}(3) \triangleq$ special orthogonal group

Base Pose

$x_b \in \mathrm{SE}(3) \triangleq$ special Euclidian group, comprising any combination of

- translations: elements of $\mathbb{R}^3$,
- rotations: elements of $\mathrm{SO}(3) \triangleq$ special orthogonal group

Can represent SO(3) elements with:

- minimal representations: 3 elements but suffer from singularities (e.g., Euler angles, roll-pitch-yaw)
- redundant representations: $\geq$4 elements but free from singularities (e.g., quaternions, rotation matrices)

Base Pose

$x_b \in \mathrm{SE}(3) \triangleq$ special Euclidian group, comprising any combination of

- translations: elements of $\mathbb{R}^3$,
- rotations: elements of $\mathrm{SO}(3) \triangleq$ special orthogonal group

Can represent SO(3) elements with:

- minimal representations: 3 elements but suffer from singularities (e.g., Euler angles, roll-pitch-yaw)
- redundant representations: $\geq$4 elements but free from singularities (e.g., quaternions, rotation matrices)

We represent SE(3) elements as 7d vectors: 3d for position, 4d for orientation (quaternion).

Quaternions and Spatial Rotations

Unit quaternions: convenient notation for rotations in 3d.

Quaternions and Spatial Rotations

Unit quaternions: convenient notation for rotations in 3d.

Compared to Euler angles: simpler to compose and avoid gimbal-lock problem.

Quaternions and Spatial Rotations

Unit quaternions: convenient notation for rotations in 3d.

Compared to Euler angles: simpler to compose and avoid gimbal-lock problem.

Compared to rotation matrices: more compact, numerically stable, and efficient.

Quaternions and Spatial Rotations

Unit quaternions: convenient notation for rotations in 3d.

Compared to Euler angles: simpler to compose and avoid gimbal-lock problem.

Compared to rotation matrices: more compact, numerically stable, and efficient.

Any 3d rotation equivalent to single rotation by angle θ about fixed axis (unit vector $u = (u_x, u_y, u_z)$).

$$\text{quaternion} = (u_x s, u_y s, u_z s, c)$$

where $c = \cos\frac{\theta}{2}$ and $s = \sin\frac{\theta}{2}$. Note that $||\text{quaternion}|| = 1 \quad \forall\, \theta, u$.

Base Velocity

Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$.

Base Velocity

Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$.
Robot velocity is $v = (\nu_b, \dot{q}_j)$, where $\nu_b = (\dot{p}_b, \omega_b) \in \mathbb{R}^6$.

Base Velocity

Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$.
Robot velocity is $v = (\nu_b, \dot{q}_j)$, where $\nu_b = (\dot{p}_b, \omega_b) \in \mathbb{R}^6$.

Angular velocity $\omega_b \in \mathbb{R}^3$ related to time derivative of associated rotation matrix $R_b \in \mathbb{R}^{3\times 3}$ by:

$$\dot{R}_b = \hat{\omega}_b R_b \quad \rightarrow \quad R_b(t) = e^{\hat{\omega}_b t} R_b(0)$$

where $\hat{\omega}_b \in \mathbb{R}^{3\times 3}$ is skew-symmetric matrix associated to ω_b.

Base Velocity

Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$.
Robot velocity is $v = (\nu_b, \dot{q}_j)$, where $\nu_b = (\dot{p}_b, \omega_b) \in \mathbb{R}^6$.

Angular velocity $\omega_b \in \mathbb{R}^3$ related to time derivative of associated rotation matrix $R_b \in \mathbb{R}^{3\times 3}$ by:

$$\dot{R}_b = \hat{\omega}_b R_b \quad \rightarrow \quad R_b(t) = e^{\hat{\omega}_b t} R_b(0)$$

where $\hat{\omega}_b \in \mathbb{R}^{3\times 3}$ is skew-symmetric matrix associated to ω_b.

So q and v have different sizes $(n_q = n_v + 1)$

Underactuated Systems

Underactuated systems: less actuators than DoFs:

$$\underbrace{n_{va}}_{\text{number of actuators}} < \underbrace{n_v}_{\text{number of DoFs}}$$

Underactuated Systems

Underactuated systems: less actuators than DoFs:

$$\underbrace{n_{va}}_{\text{number of actuators}} < \underbrace{n_v}_{\text{number of DoFs}}$$

Assume ordered elements of $q \triangleq (q_u, q_a)$:

- $q_u \in \mathbb{R}^{n_{qu}}$: passive (unactuated) joints,
- $q_a \in \mathbb{R}^{n_{qa}}$: actuated joints.

Similarly, $v \triangleq (v_u, v_a)$, $v_u \in \mathbb{R}^{n_{vu}}$, $v_a \in \mathbb{R}^{n_{va}}$.

Underactuated Systems

Underactuated systems: less actuators than DoFs:

$$\underbrace{n_{va}}_{\text{number of actuators}} < \underbrace{n_v}_{\text{number of DoFs}}$$

Assume ordered elements of $q \triangleq (q_u, q_a)$:

- $q_u \in \mathbb{R}^{n_{qu}}$: passive (unactuated) joints,
- $q_a \in \mathbb{R}^{n_{qa}}$: actuated joints.

Similarly, $v \triangleq (v_u, v_a)$, $v_u \in \mathbb{R}^{n_{vu}}$, $v_a \in \mathbb{R}^{n_{va}}$.

$S \triangleq \begin{bmatrix} 0_{n_{va} \times n_{vu}} & I_{n_{va}} \end{bmatrix}$ is selection matrix:

$$v_a = Sv$$

Underactuated Systems

Underactuated systems: less actuators than DoFs:

$$\underbrace{n_{va}}_{\text{number of actuators}} < \underbrace{n_v}_{\text{number of DoFs}}$$

Assume ordered elements of $q \triangleq (q_u, q_a)$:

- $q_u \in \mathbb{R}^{n_{qu}}$: passive (unactuated) joints,
- $q_a \in \mathbb{R}^{n_{qa}}$: actuated joints.

Similarly, $v \triangleq (v_u, v_a)$, $v_u \in \mathbb{R}^{n_{vu}}$, $v_a \in \mathbb{R}^{n_{va}}$.

$S \triangleq \begin{bmatrix} 0_{n_{va} \times n_{vu}} & I_{n_{va}} \end{bmatrix}$ is selection matrix:

$$v_a = Sv$$

For legged robots typically $q_u = x_b$ (all joints are actuated).

Under-Actuated Dynamic

Dynamics of under-actuated mechanical system:

$$M(q)\dot{v} + h(q, v) = S^\top \tau + J(q)^\top f$$

Contrary to fully-actuated case: $\tau \in \mathbb{R}^{n_{va}}$.

Under-Actuated Dynamic

Dynamics of under-actuated mechanical system:

$$M(q)\dot{v} + h(q, v) = S^\top \tau + J(q)^\top f$$

Contrary to fully-actuated case: $\tau \in \mathbb{R}^{n_{va}}$.

Often decompose into unactuated and actuated parts:

$$\begin{aligned} M_u(q)\dot{v} + h_u(q, v) &= J_u(q)^\top f \\ M_a(q)\dot{v} + h_a(q, v) &= \tau + J_a(q)^\top f \end{aligned} \tag{2}$$

where

$$M = \begin{bmatrix} M_u \\ M_a \end{bmatrix} \quad h = \begin{bmatrix} h_u \\ h_a \end{bmatrix} \quad J = \begin{bmatrix} J_u & J_a \end{bmatrix} \tag{3}$$

Recap

Manipulator:

$$M(q)\dot{v} + h(q, v) = \tau$$

Manipulator in contact:

$$M(q)\dot{v} + h(q, v) = \tau + J(q)^\top f$$

Legged robot (in contact):

$$M(q)\dot{v} + h(q, v) = S^\top \tau + J(q)^\top f$$

If contacts are rigid:

$$J\dot{v} = -\dot{J}v$$