Robot Modeling

Optimization-based Robot Control

Andrea Del Prete

University of Trento
Schedule

Classroom Code: 2ym4lka

First week:

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

Second week:

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

1. Modeling Robot Manipulators
2. Modeling Robots in Contact
3. Modeling Legged Robots
State $\triangleq x$.

Control $\triangleq u$. 
Notation & Definitions

\textbf{State} \triangleq x.
\textbf{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.
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
- ...

Appropriate model depends on robot and task.
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).
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:

\[ \dot{x} = u \]
Model actuators as acceleration sources.

- Good for electric w/o large contact forces.
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:

\[
\begin{bmatrix}
\dot{q} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
0 & l \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
q \\
v
\end{bmatrix} +
\begin{bmatrix}
0 \\
l
\end{bmatrix}
u
\]
Model actuators as **torque** sources.

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

\[ x \triangleq (q, v) \quad u \triangleq \tau \]
Model actuators as \textit{torque} sources.

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

\begin{equation}
\begin{aligned}
x & \triangleq (q, v) \\
u & \triangleq \tau
\end{aligned}
\end{equation}

Dynamics of \textit{fully-actuated} mechanical system (e.g., manipulator):

\begin{equation}
M(q)\ddot{v} + h(q, v) = \tau,
\end{equation}

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.
**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
Bias forces sometimes decomposed as:

\[ h(q, v) = C(q, v)v + g(q) \]

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

Nonlinear state-space dynamics:

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

Given $q, v, \tau$ compute $\dot{v}$:

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

Problem solved by simulators.
Inverse VS Forward Dynamics

**Forward Dynamics**
Given $q$, $v$, $\tau$ compute $\dot{v}$:

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

Problem solved by simulators.

**Inverse Dynamics**
Given $q$, $v$, $\dot{v}$ compute $\tau$:

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

Problem solved by controllers.
Modeling Robots in Contact
Adding Contact Forces

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

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

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 $f \in \mathbb{R}^{n_f}$:

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

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).
Rigid contacts constrain motion.
Rigid contacts constrain motion.

\[ c(q) = \text{const} \iff \text{Contact points do not move} \]
Rigid contacts constrain motion.

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

Differentiate:

\[ Jv = 0 \iff \text{Contact point velocities are null} \]
\[ J\dot{v} + \dot{J}v = 0 \iff \text{Contact point accelerations are null} \]
Robots in Rigid Contact

Rigid contacts constrain motion.

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

Differentiate:

\[ Jv = 0 \iff \text{Contact point velocities are null} \]
\[ J\dot{v} + \dot{J}v = 0 \iff \text{Contact point accelerations are null} \]

Introduce constraints in dynamics:

\[
\begin{bmatrix}
M & -J^T & -I \\
J & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\dot{v} \\
f \\
\tau
\end{bmatrix}
= 
\begin{bmatrix}
-h \\
-Jv
\end{bmatrix}
\] (1)
Forward Dynamics (with constraints)

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

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

Problem solved by (bilateral) rigid contact simulators.
Inverse VS Forward Dynamics with Rigid Contacts

Forward Dynamics (with constraints)
Given $q, v, \tau$ compute $\dot{v}$ and $f$:

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

Problem solved by (bilateral) rigid contact simulators.

Inverse Dynamics (with constraints)
Given $q, v, \dot{v}$ compute $\tau$ and $f$:

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

where $\dagger$ represents pseudo-inverse.
Inverse VS Forward Dynamics with Rigid Contacts

**Forward Dynamics (with constraints)**
Given $q, v, \tau$ compute $\dot{v}$ and $f$:

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

Problem solved by (bilateral) rigid contact simulators.

**Inverse Dynamics (with constraints)**
Given $q, v, \dot{v}$ compute $\tau$ and $f$:

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

where $^\dagger$ represents pseudo-inverse.

Implicit assumption: $\dot{v}$ satisfies constraints.
Inverse VS Forward Dynamics with Rigid Contacts

**Forward Dynamics (with constraints)**

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

\[
\begin{bmatrix}
\dot{v} \\
f
\end{bmatrix} = \begin{bmatrix}
M & -J^T \\ J & 0
\end{bmatrix}^{-1} \begin{bmatrix}
\tau - h \\ -J\dot{v}
\end{bmatrix}
\]

Problem solved by (bilateral) rigid contact simulators.

**Inverse Dynamics (with constraints)**

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

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

where $^\dagger$ represents pseudo-inverse.

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

Primitive version of inverse-dynamics control with rigid contacts.
Modeling Legged Robots
PROBLEM
Joint angles not enough to describe robot configuration.
PROBLEM
Joint angles not enough to describe robot configuration.

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

\[ q = (x_b, q_j) \]

Base pose, Joint angles
PROBLEM
Joint angles not enough to describe robot configuration.

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

\[ q = (x_b, q_j) \]

Base pose Joint angles

Now \( q \) sufficient to describe robot configuration in space.
$x_b \in \text{SE}(3) \triangleq \text{special Euclidian group, comprising any combination of}$

- translations: elements of $\mathbb{R}^3$,
- rotations: elements of $\text{SO}(3) \triangleq \text{special orthogonal group}$
$x_b \in SE(3) \triangleq$ special Euclidian group, comprising any combination of

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

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

- translations: elements of \( \mathbb{R}^3 \),
- rotations: elements of \( SO(3) \triangleq \) 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).
Unit quaternions: convenient notation for rotations in 3d.
Unit quaternions: convenient notation for rotations in 3d.

Compared to Euler angles: simpler to compose and avoid gimbal-lock problem.
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.
**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 $\theta$ 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 $||\text{quaternion}|| = 1 \quad \forall \theta, u$. 
Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$. 
Robot configuration is $q = (x_b, q_j)$, where $x_b = (p_b, o_b) \in \mathbb{R}^7$. Robot velocity is $\nu = (\nu_b, \dot{q}_j)$, where $\nu_b = (\dot{p}_b, \omega_b) \in \mathbb{R}^6$. 
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 \( \omega_b \).
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 \( \omega_b \).

So \( q \) and \( v \) have different sizes \( (n_q = n_v + 1) \).
Underactuated systems: less actuators than DoFs:

\[ n_{va} < n_v \]

number of actuators       number of DoFs
Underactuated systems: less actuators than DoFs:

\[ n_{va} \lessdot n_v \]

number of actuators \hspace{2cm} 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:

\[
\begin{array}{c}
\overset{\text{number of actuators}}{n_{va}} < \overset{\text{number of DoFs}}{n_v}
\end{array}
\]

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: less actuators than DoFs:

\[ n_{va} < n_v \]

number of actuators

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).
Dynamics of under-actuated mechanical system:

\[ M(q)\dot{v} + h(q, v) = S^T \tau + J(q)^T f \]

Contrary to fully-actuated case: \( \tau \in \mathbb{R}^{n_{va}} \).
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 decomposed into unactuated and actuated parts:

\[
M_u(q)\dot{v} + h_u(q, v) = J_u(q)^\top f
\]

\[
M_a(q)\dot{v} + h_a(q, v) = \tau + J_a(q)^\top f
\]  

(2)

where

\[
M = \begin{bmatrix} M_u \\ M_a \end{bmatrix} \quad h = \begin{bmatrix} h_u \\ h_a \end{bmatrix} \quad J = \begin{bmatrix} J_u & J_a \end{bmatrix}
\]  

(3)
Recap

Manipulator:

\[ M(q)\dot{\nu} + h(q, \nu) = \tau \]

Manipulator in contact:

\[ M(q)\dot{\nu} + h(q, \nu) = \tau + J(q)^\top f \]

Legged robot (in contact):

\[ M(q)\dot{\nu} + h(q, \nu) = S^\top \tau + J(q)^\top f \]

If contacts are rigid:

\[ J\dot{\nu} = -\dot{J}\nu \]