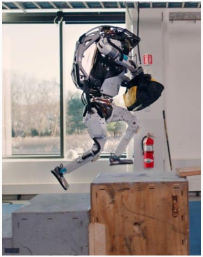
AuSRoS 2024
Robotic Control and Estimation

Lecture 3: Multi-Body System Control

Ian Manchester
with material by Damian Abood

Australian Centre for Robotics, The University of Sydney

Motivation: High-Performance Humanoid Robots



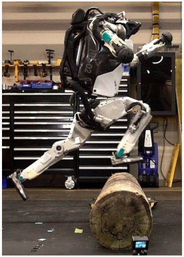
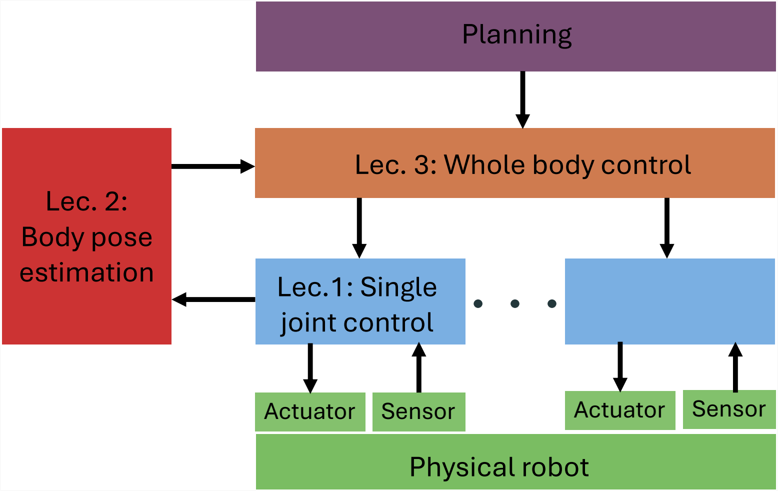


Image: bostondynamics.com

Hierarchy of Control System



What We Have Achieved so Far

In **Lecture 1** we

- Looked at the dynamics of systems with one degree of freedom
- Showed how PID control can achieve a desired stable response.
- Discussed some fundamental limitations of feedback control

In **Lecture 2** we:

- Presented the pose kinematics of rigid bodies with transformation matrices T
- Derived Kalman filters to estimate the state of a system given measurements.

In this final lecture, we will:

- Discuss control of **multi-body systems**
- Introduce the basic concepts of **model-predictive control (MPC)**

Multi-Body Kinematics and Dynamics

Rigid Body Kinematics: A Review

The pose of a rigid body with body frame O_B with respect to a frame O_I can be represented with

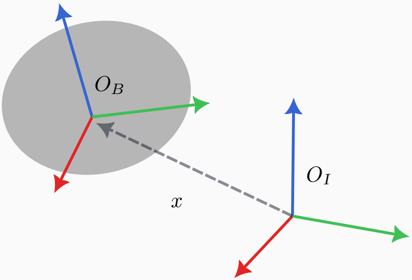
- Position ${}^I x_B \in \mathbb{R}^3$
- Orientation ${}^I R_B \in \mathcal{SO}3$

and represented compactly as transformation matrix

$$T = \begin{bmatrix} {}^I R_B & {}^I x_B \\ 0_{3\times 1} & 1 \end{bmatrix}$$

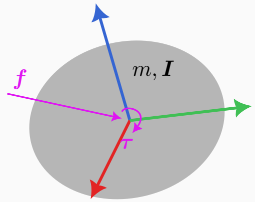
The kinematics of rigid body with body velocities (v, ω) can then be expressed as

$$\dot{T} = T \begin{bmatrix} \omega_\times & v \\ 0_{3\times 1} & 0 \end{bmatrix}$$



Rigid Body Dynamics

Rigid bodies have mass m and rotational inertia $\boldsymbol{I}$ about their body frames.



Due to these properties, the dynamics of a rigid body (in the body frame) are

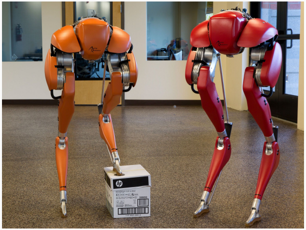
$$\boldsymbol{f} = m\boldsymbol{a} \tag{1}$$

$$\boldsymbol{\tau} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \tag{2}$$

Where $\boldsymbol{f}$ and $\boldsymbol{\tau}$ are the net translational and rotational forces acting on the body respectively.

Multi-Body Systems

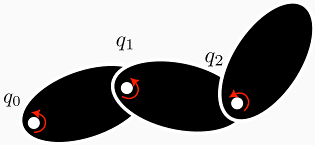
- The dynamics of complex robotic systems (e.g. walking robots, manipulator arms) can not be described as single rigid-body.
- They should be viewed as a collection of inter-connected rigid-bodies.



Generalised Coordinates for Rigid Body Systems

The configuration of a robotic system is defined by the state of its joints, which we represent as a vector $\boldsymbol{q}$.

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix}$$



All joint transforms $\boldsymbol{T}(\boldsymbol{q})$, velocities $\boldsymbol{v}_J(\dot{\boldsymbol{q}})$ and accelerations $\boldsymbol{a}_J(\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ functions of $\boldsymbol{q}$ and its rates. The kinematic and dynamics from before can thus be expressed fully in terms of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$.

The dynamics of all bodies can be grouped together to form the following expression

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{3}$$

this is referred to as the manipulator equation

The Manipulator Equation

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{4}$$

- $\boldsymbol{M}(\boldsymbol{q})$ - Inertial matrix (effects of mass and inertia)
- $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ - Coriolis matrix (effects of centripetal acceleration)
- $\boldsymbol{G}(\boldsymbol{q})$ - Potential vector (gravitational effects)
- $\boldsymbol{\tau}$ - Generalised input (forces acting at each joint specified by $\boldsymbol{q}$)

Can be computed analytically for small systems (Euler-Lagrange equations), but quickly gets extremely complicated.

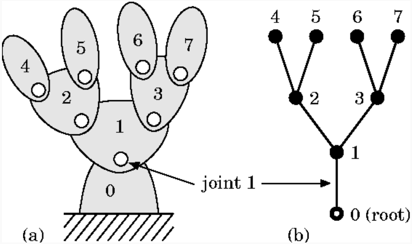
Connecting Rigid Bodies Together

We can represent multi-body systems as kinematic trees.

For the i-th body in the tree we provide the following information:

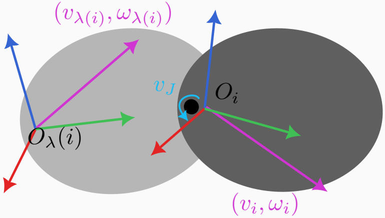
- Inertial data $(m_i, \boldsymbol{I}_i)$
- Which link it is connected to (its parent $\lambda(i)$)
- The type of joint connecting it (e.g. hinge, prismatic)
- Transform ${}^{i}T_{\lambda(i)}$ to express its body frame relative to its parent

This information is typically stored in Unified Robot Description Format (URDF) as a .urdf file



Multi-Body Kinematics

This tree structure can lead to recursively computing the kinematics of the system



For body velocities $\boldsymbol{v} = (v, \omega)$ and joint velocities $\boldsymbol{v}_J$,

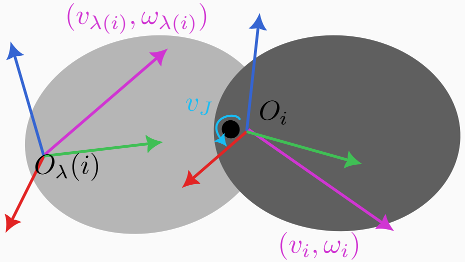
$$\boldsymbol{v}_i =^i \boldsymbol{T}_{\lambda(i)} \boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_J$$

For body accelerations $\boldsymbol{a} = (a, \dot{\omega})$ and joint accelerations $\boldsymbol{a}_J$,

$$\boldsymbol{a}_i =^i \boldsymbol{T}_{\lambda(i)} \boldsymbol{a}_{\lambda(i)} + \boldsymbol{a}_J$$

with ${}^i\boldsymbol{T}_{\lambda(i)} \in \mathbb{R}^{6\times 6}$ transforming both linear and rotational parts

Multi-Body Dynamics



Given body velocities $\boldsymbol{v}_i$ and accelerations $\boldsymbol{a}_i$ for each body, we can then compute the net forces of each body

$$\boldsymbol{f}_i = m_i \boldsymbol{a}_i, \qquad \boldsymbol{\tau}_i = \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{I}_i \boldsymbol{\omega}_i \tag{5}$$

We then traverse the tree **backwards** adding up all the reaction forces on each body.

This is the basic idea of the **Recursive Newton Euler Algorithm** (RNEA)

Algorithmic Approaches to Rigid-Body Dynamics

Efficient algorithms have been developed using these recursive ideas to compute components of

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{6}$$

which include:

- **Forward Kinematics** - Computes all transforms ${}^{i}T_{I}$, velocities and accelerations for each body w.r.t the global frame.
- **Articulated Body Algorithm (ABA)** - Computes forward dynamics $\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q}))$
- **Recursive Newton-Euler Algorithm (RNEA)** - Computes inverse dynamics $\tau = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q})$
- **Composite Rigid-Body Algorithm (CRBA)** - Computes inertial matrix $\boldsymbol{M}(\boldsymbol{q})$

Controlling Rigid Body Systems

Motivation

We now have the ability to compute the dynamics of complex, multi-body systems, which can be expressed under the form

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

Can we make use of these dynamics to control these systems?

Isn't PID Control Enough?

Given a reference state $\boldsymbol{q}_r$, can we achieve $\boldsymbol{q} \approx \boldsymbol{q}_r$ with the input

$$\boldsymbol{\tau} = K_P(\boldsymbol{q}_r - \boldsymbol{q}) + K_D(\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}}) := K_P\boldsymbol{e} + K_D\dot{\boldsymbol{e}}$$

The system dynamics become

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = K_P\boldsymbol{e} + K_D\dot{\boldsymbol{e}} \tag{7}$$

The error dynamics are dependent on the dynamics of the system, difficult to tune and use our methods from **Lecture 1**.

Integral control can try to correct errors, but ignores prior knowledge, stability can be difficult to establish over full range of motion.

Computed Torque Control

Instead of setting the input $\boldsymbol{\tau}$ proportional to the error, make the generalised acceleration $\ddot{\boldsymbol{q}}$ proportional to the error.

That is, drive the system to behave such that

$$\ddot{\boldsymbol{q}}_d = \ddot{\boldsymbol{q}}_r + K_P(\boldsymbol{q}_r - \boldsymbol{q}) + K_D(\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}})$$

Using **inverse dynamics** (RNEA algorithm), we can find the value for $\boldsymbol{\tau}$ given the state $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d$

$$\boldsymbol{\tau} = \texttt{invdyn}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) \tag{8}$$

Computed Torque Control - Error Dynamics

Using this input, we achieve for the system

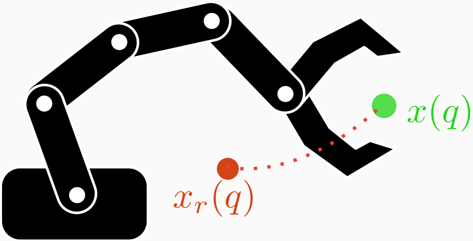
$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{G} = \boldsymbol{\tau} &= \boldsymbol{M}\ddot{\boldsymbol{q}}_d + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{G} \\ \boldsymbol{M}(\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_d) &= 0 \\ \Rightarrow \ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_d &= 0 \\ \ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_r + K_P(\boldsymbol{q}) + K_D(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_r) &= 0 \\ \ddot{\boldsymbol{e}} + K_P\boldsymbol{e} + K_D\dot{\boldsymbol{e}} &= 0 \end{aligned}$$

The error dynamics do not have the dynamics of the system in them, as we have effectively "cancelled" them out.

Operational-Space Control

What if a higher level of control is required that is not easy to describe from a joint-based perspective?

Represent tasks (e.g. positions, orientations) as vectors $\boldsymbol{x}(\boldsymbol{q}) \in \mathcal{X}$ in the **operational** space $\mathcal{X}$.

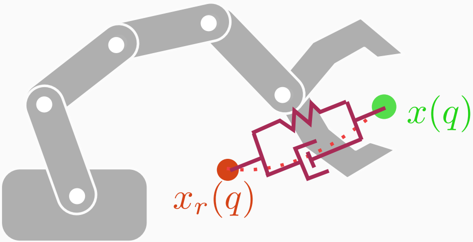


Operational-Space Control

We can design references $\boldsymbol{x}_r(\boldsymbol{q})$ within task space (e.g. end-effector position and orientations, centre-of-mass tracking).

Similar to computed torque control, create a control law that creates a desired **task** acceleration, such that

$$\ddot{\boldsymbol{x}}_{\boldsymbol{d}} = \ddot{\boldsymbol{x}}_{\boldsymbol{r}} + K_P(\boldsymbol{x}_r - \boldsymbol{x}) + K_D(\dot{\boldsymbol{x}}_{\boldsymbol{r}} - \dot{\boldsymbol{x}})$$



Operational-Space Control

Map these task accelerations to generalised accelerations $\ddot{\boldsymbol{q}}$ through the relationship

$$\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{9}$$

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{10}$$

Where $J(\boldsymbol{q}) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}$ is the **task Jacobian**

There are many feasible solutions $(\ddot{\boldsymbol{q}}, \boldsymbol{\tau})$ that jointly satisfy

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

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

If joint-torques are unconstrained, one solution can be computed with the pseudo-inverse with null-space PD:

$$\ddot{\boldsymbol{q}} = J^{+}(\ddot{\boldsymbol{x}} - \dot{J}\dot{\boldsymbol{q}}) + (I - J^{+}J)(K_P(\boldsymbol{q}_r - \boldsymbol{q}) + K_D(\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}}))$$

and then solve for τ with RNEA (inverse dynamics)

Optimal Operational-Space Control

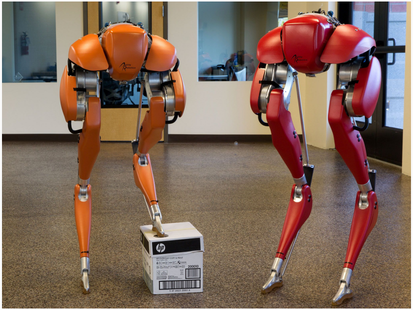
When there are constraints, solutions can be found by framing the problem as an optimal control problem, e.g. with different weightings w_i for different tasks:

$$\begin{aligned} \text{minimise} \quad & \sum_i w_i ||J_i \ddot{\boldsymbol{q}} + \dot{J}_i \dot{\boldsymbol{q}} - \ddot{\boldsymbol{x}}_{\boldsymbol{i}}||^2 \\ \text{subject to} \quad & \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \\ & \boldsymbol{\tau} \in \mathcal{U} \end{aligned}$$

Despite the nonlinear dynamics, this is a convex **quadratic program** in the variables $(\ddot{\boldsymbol{q}}, \boldsymbol{\tau})$, since they appear linearly in the constraints.

Can also optimize over $\ddot{\boldsymbol{x}}$, e.g. based on LQR cost-to-go.

Floating Base Systems



- Floating-base systems (legged and aerial robotics)

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix} + J(\boldsymbol{q})^T \boldsymbol{\lambda}$$

- λ are environmental reaction forces, satisfy complementarity constraints (i.e. non-zero if in contact)

Model Predictive Control

Model-Predictive Control

- Operational space control plan "in-the-moment", it performs an instantaneous action to minimise an objective at each instant.
- What if we want to take actions that account and impact the future behaviour of the system (i.e. thinking ahead)?

Model Predictive Control: The Problem

We can formulate this as a constrained optimal control problem:

$$\begin{aligned} \text{minimise} \quad & J(u(\cdot), x(\cdot)) := \sum_{t=0}^{\infty} g(x(t), u(t)) \\ \text{subject to} \quad & x(t+1) = f(x(t), u(t)) \quad \forall t, \\ & x(0) = x_0, \\ & u(t) \in \mathcal{U} \quad \forall t, \\ & x(t) \in \mathcal{X} \quad \forall t. \end{aligned}$$

In fact, we want a **feedback controller** (a.k.a. policy), i.e. a solution of this problem for **all possible** x_0

Model-Predictive Control

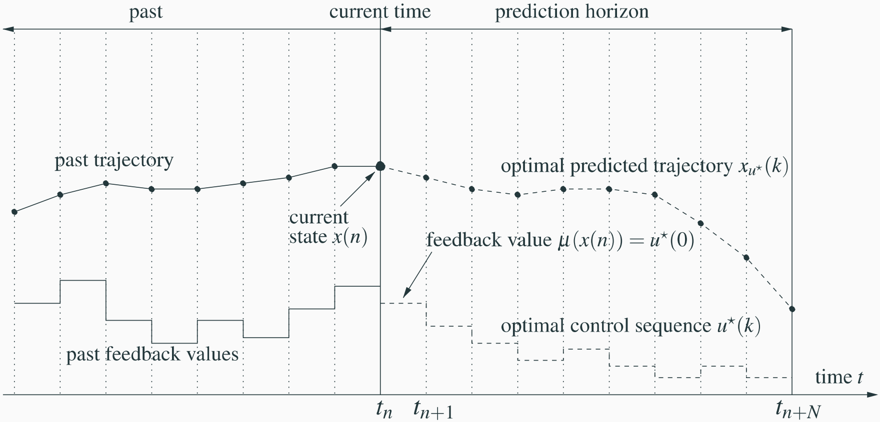


Image: Grune & Pannek (2010), Nonlinear Model Predictive Control: Theory & Algorithms.

Model Predictive Control

The MPC strategy is as follows. At each time step t:

- Measure (or estimate) the true state $x(t)$.
- Solve the following finite-dimensional optimisation problem:

$$\begin{aligned} \text{minimise } & \hat{J}(\breve{u}, \breve{x}) = \sum_{k=0}^{h-1} g(\breve{x}(k), \breve{u}(k)) + \hat{V}(x(h)) \\ \text{subject to } & \breve{x}(k+1) = f(\breve{x}(k), \breve{u}(k)), \quad \forall k = 0, ..., h-1 \\ & \breve{u}(k) \in \mathcal{U} \quad \forall k = 0, 1, ..., h-1, \\ & \breve{x}(k) \in \mathcal{X} \quad \forall k = 0, 1, ..., h-1, \\ & \breve{x}(0) = x(t), \\ & \breve{x}(h) \in \mathcal{X}_h. \end{aligned}$$

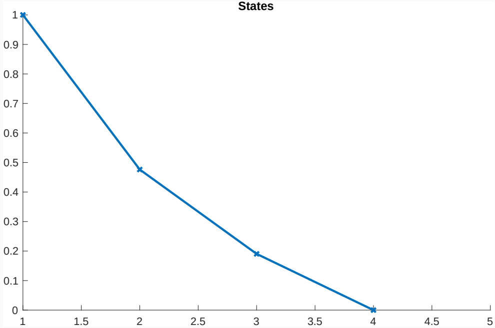
- Apply the control $u^\star(t) = \breve{u}(0)$, and discard the rest of $\breve{u}$ and $\breve{x}$

Here $\hat{V}$ is an estimate of the value function, "tail cost", valid in some region $\mathcal{X}_h$.

Model Predictive Control

- Note that the variables $\breve{x}, \breve{u}$ are predictions of possible future x and u. They are just *internal variables* in the optimization process.
- The specification of h, $\hat{V}$ and $\mathcal{X}_h$ give some freedom for design:
  - Generally, the longer the horizon h is the closer to optimality, at the expense of computational cost, and $\hat{V}, \mathcal{X}_h$ become irrelevant as $h \to \infty$.
  - If $\hat{V} \approx V$ closely and $\mathcal{X}_h$ is large, then a short horizon h can be used for faster online computation.
  - Computation of $\hat{V}, \mathcal{X}_h$ is usually done offline, and can be very computationally intensive (RL, approximate dynamic programming, HJB equations).

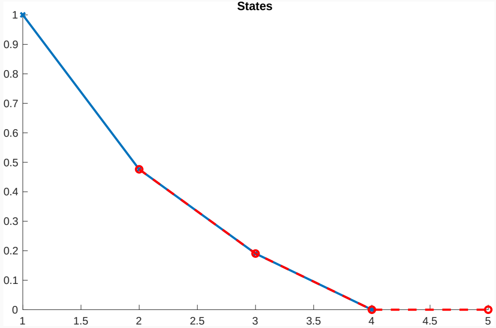
Stabilizing Constraints



Terminal constraint: states enters a "safe set" it can stay in forever for zero future cost (e.g. an equilibrium). Assume all other states have positive cost.

- Let $V(x_1)$ be the cost of the MPC-computed sequence at time-step 1. I.e. $V(x_1) = g(x_1, u_1) + g(x_2, u_2) + g(x_3, u_3)$.

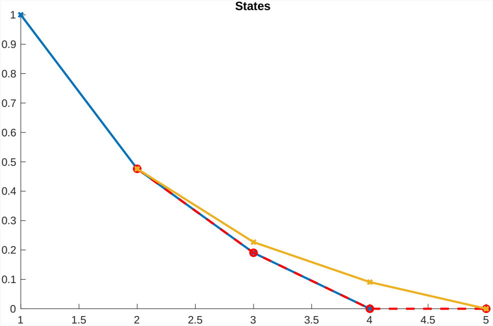
Stabilizing Constraints



Terminal constraint: states enters a "safe set" it can stay in forever for zero future cost (e.g. an equilibrium). Assume all other states have positive cost.

- Let $V(x_1)$ be the cost of the optimal MPC-computed sequence at $t = 1$. I.e. $V(x_1) = g(x_1, u_1) + g(x_2, u_2) + g(x_3, u_3)$.
- Let $\tilde{V}(x_2)$ be the cost of starting at $t = 2$, but following the same path, then "doing nothing". $\tilde{V}(x_2) = g(x_2, u_2) + g(x_3, u_3) < V(x_1)$. Why?

Stabilizing Constraints



Terminal constraint: states enters a "safe set" it can stay in forever for zero future cost (e.g. an equilibrium). Assume all other states have positive cost.

- Let $V(x_1)$ be the cost of the MPC-computed path at $t = 1$. I.e. $V(x_1) = g(x_1, u_1) + g(x_2, u_2) + g(x_3, u_3)$.
- Let $\tilde{V}(x_2)$ be the cost of starting at time-step 2, but following the same path, then "doing nothing". $\tilde{V}(x_2) = g(x_2, u_2) + g(x_3, u_3) < V(x_1)$.
- Now let $V(x_2)$ be the cost of the optimal (MPC-computed) path at time step 2. Claim $V(x_2) \leq \tilde{V}(x_2) < V(x_1)$. Why?

Conclusions and Future Directions

We have seen some of the main concepts algorithmic and optimization-based methods for multi-joint control. Many active areas of current research, including:

- Integration of control and planning, e.g. footstep selection for walking robots
- Integration with learning in various forms: learning policies, value functions, models, etc
- Dealing with challenges due to uncertainty, partial observability, etc.

Further Reading:

- Roy Featherstone, **Rigid Body Dynamics Algorithms**, 2008.
- Oussama Khatib, **A unified approach for motion and force control of robot manipulators: The operational space formulation**, 1987.
- Grune & Pannek, **Nonlinear Model Predictive Control: Theory & Algorithms**, 2010.