Lecture 10

Controlling Characters

Libin Liu

School of Intelligence Science and Technology
Peking University

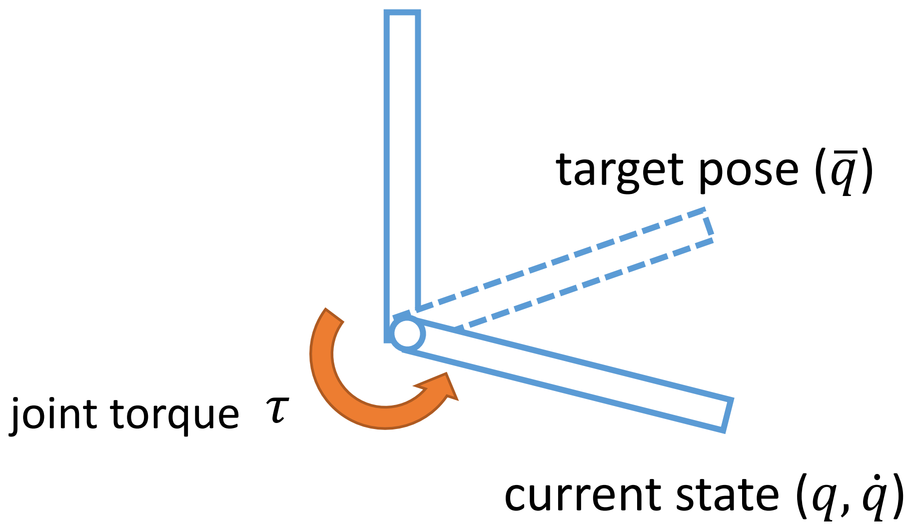
GAMES105 课程交流

VCL @ PKU

Outline

- More about PD (Proportional-Derivative) control
  - Stable PD control
- Feedforward Motion Control
  - Trajectory optimization
- Feedback Motion Control
  - Static balance

PD Control for Characters



target pose

$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

proportional control

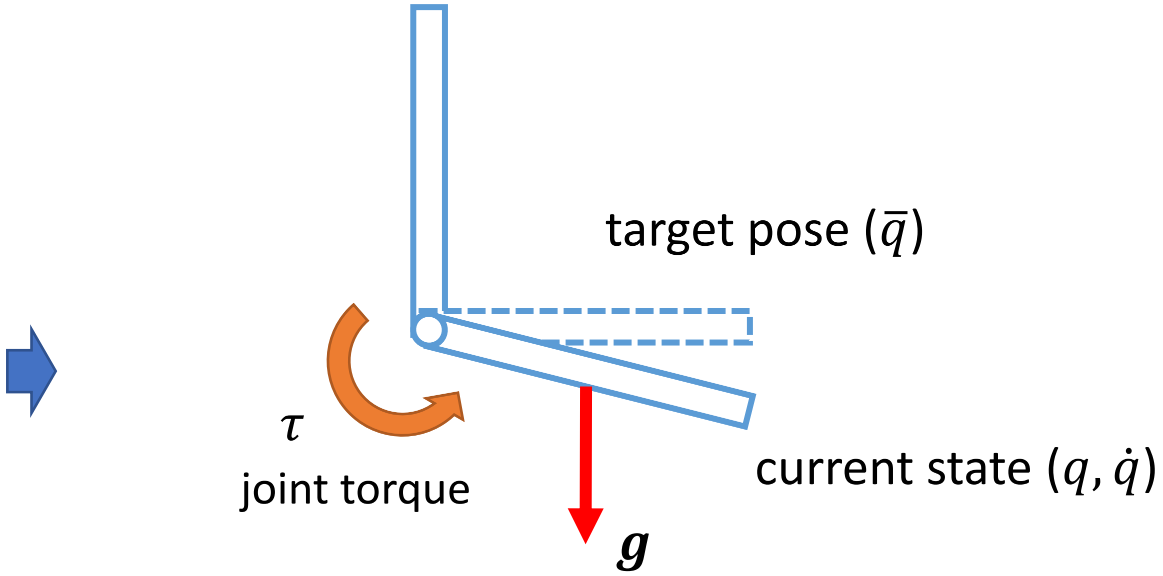
derivative control

Problems with PD Control

PD control computes torques based on errors

- Steady state error

This arm never reaches the target angle under gravity

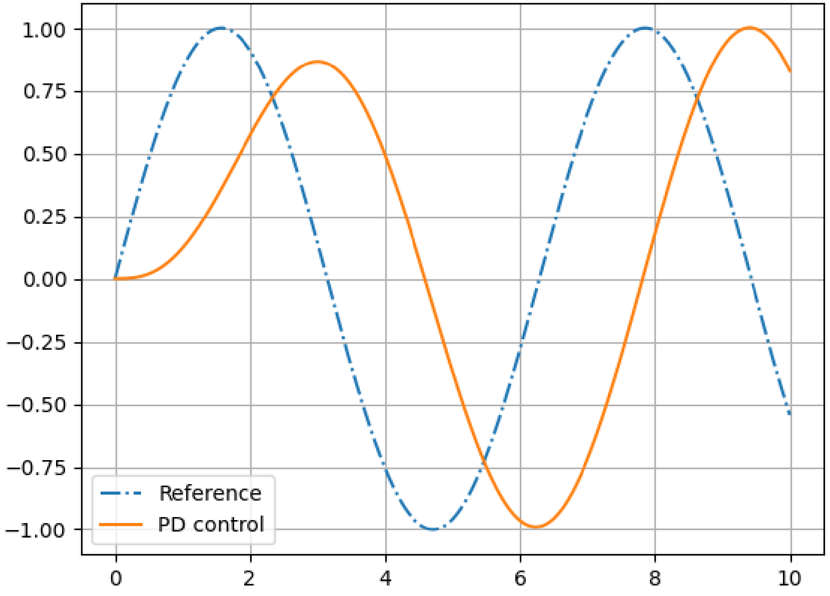


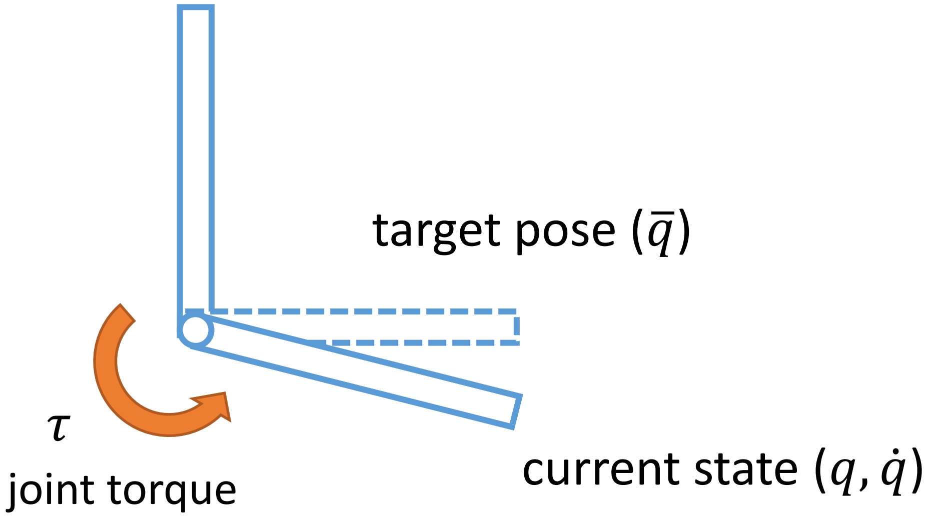
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

Problems with PD Control

PD control computes torques based on errors

- Steady state error
- Motion falls behind the reference





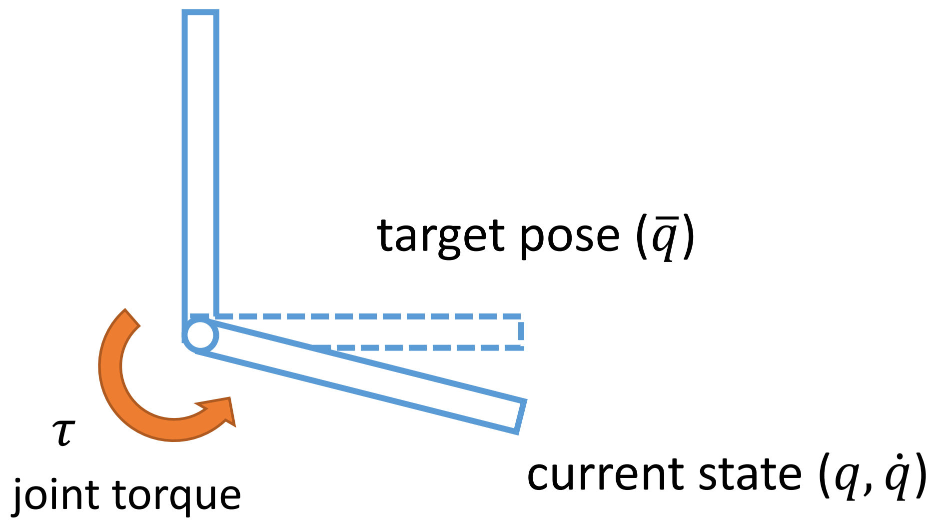
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

Problems with PD Control

PD control computes torques based on errors

- Steady state error
- Motion falls behind the reference

High-gain (k_p) control is more precise…



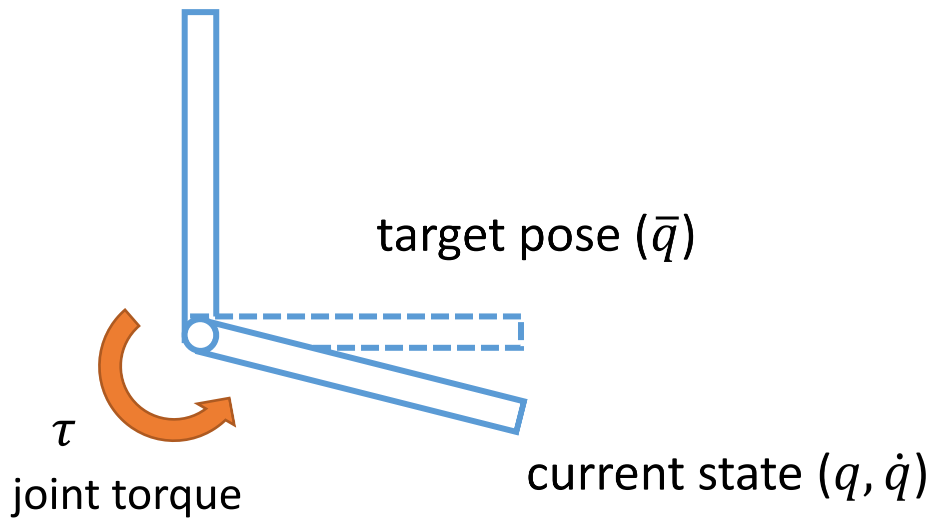
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

Problems with PD Control

PD control computes torques based on errors

- Steady state error
- Motion falls behind the reference

High-gain (k_p) control is more precise but less stable…



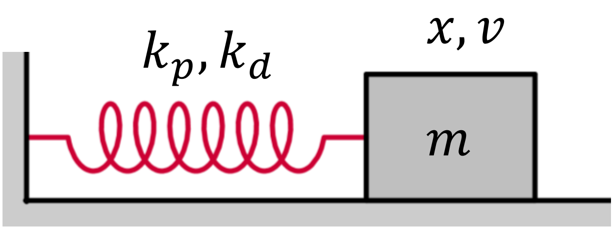
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

Stability of PD Control

Semi-implicit Euler Integration

$$v_{n+1} = v_n + h\frac{f}{m}$$

$$x_{n+1} = x_n + hv_{n+1}$$



$$f = -k_p x - k_d v$$

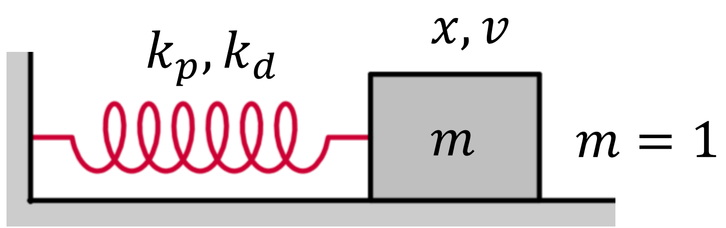
h: simulation time step

Stability of PD Control

Semi-implicit Euler Integration

$$v_{n+1} = v_n + hf$$

$$x_{n+1} = x_n + hv_{n+1}$$



$$f = -k_p x - k_d v$$

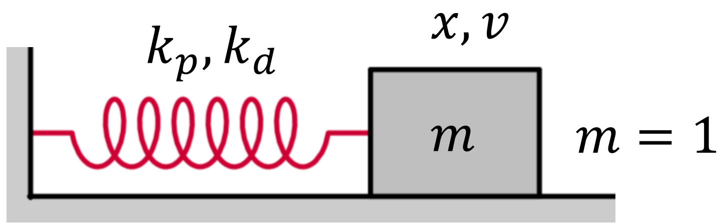
h: simulation time step

Stability of PD Control

Semi-implicit Euler Integration

$$v_{n+1} = v_n + h(-k_p x_n - k_d v_n)$$

$$x_{n+1} = x_n + h v_{n+1}$$



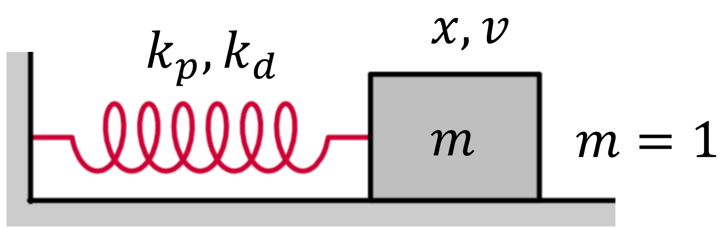
$$f = -k_p x - k_d v$$

h: simulation time step

Stability of PD Control

Semi-implicit Euler Integration

$$\begin{bmatrix} v_{n+1} \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} 1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2 \end{bmatrix} \begin{bmatrix} v_n \\ x_n \end{bmatrix}$$



$$f = -k_p x - k_d v$$

h: simulation time step

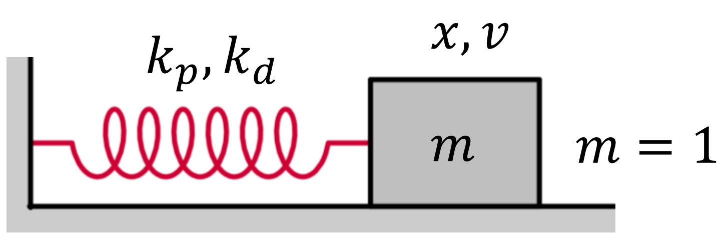
Stability of PD Control

Semi-implicit Euler Integration

$$\begin{bmatrix} v_{n+1} \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} 1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2 \end{bmatrix} \begin{bmatrix} v_n \\ x_n \end{bmatrix}$$

$$\boldsymbol{s}_{n+1} = A \boldsymbol{s}_n$$

$$\boldsymbol{s}_{n+1} = A^n \boldsymbol{s}_1$$



$$f = -k_p x - k_d v$$

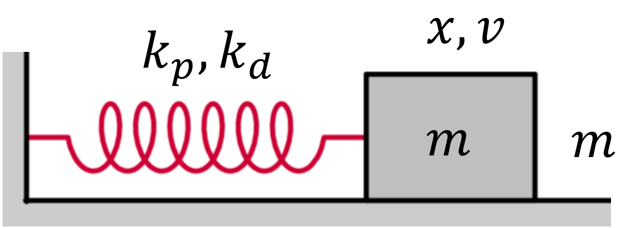
h: simulation time step

Stability of PD Control

$$\boldsymbol{s}_{n+1} = A\boldsymbol{s}_n \quad \Rightarrow \quad \boldsymbol{s}_{n+1} = A^n\boldsymbol{s}_1$$

Assume A has two eigenvalues λ_1, λ_2 and two eigen vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$. Note $\lambda_i \in \mathbb{C}$

$$A = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1} \qquad P = \begin{bmatrix}\vdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ \vdots & \vdots\end{bmatrix}$$



$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix}1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2\end{bmatrix}$$

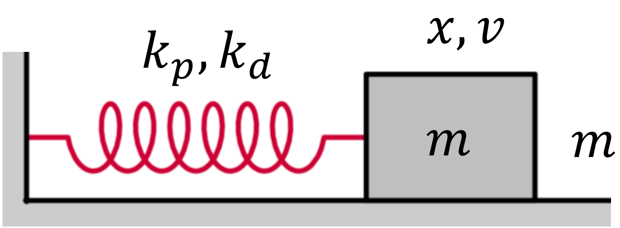
Stability of PD Control

$$\boldsymbol{s}_{n+1} = A\boldsymbol{s}_n \quad \Rightarrow \quad \boldsymbol{s}_{n+1} = A^n\boldsymbol{s}_1$$

Assume A has two eigenvalues λ_1, λ_2 and two eigen vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$. Note $\lambda_i \in \mathbb{C}$

$$A = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1} \qquad P = \begin{bmatrix}\vdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ \vdots & \vdots\end{bmatrix}$$

$$\boldsymbol{s}_{n+1} = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1}\boldsymbol{s}_n$$



$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix}1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2\end{bmatrix}$$

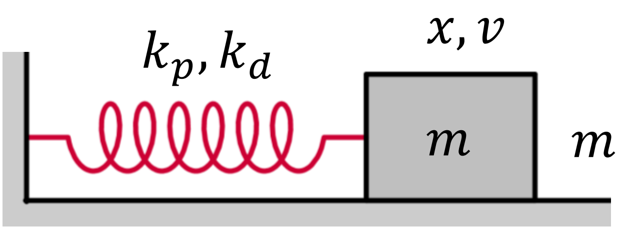
Stability of PD Control

$$\boldsymbol{s}_{n+1} = A\boldsymbol{s}_n \quad \Rightarrow \quad \boldsymbol{s}_{n+1} = A^n\boldsymbol{s}_1$$

Assume A has two eigenvalues λ_1, λ_2 and two eigen vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$. Note $\lambda_i \in \mathbb{C}$

$$A = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1} \qquad P = \begin{bmatrix}\vdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ \vdots & \vdots\end{bmatrix}$$

$$\boldsymbol{z}_{n+1} = \begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}\boldsymbol{z}_n \qquad \boldsymbol{z} = P^{-1}\boldsymbol{s}$$



$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix}1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2\end{bmatrix}$$

Stability of PD Control

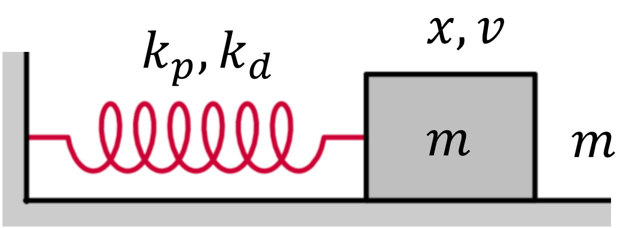
$$\boldsymbol{s}_{n+1} = A\boldsymbol{s}_n \Rightarrow \boldsymbol{s}_{n+1} = A^n\boldsymbol{s}_1$$

Assume A has two eigenvalues λ_1, λ_2 and two eigen vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$. Note $\lambda_i \in \mathbb{C}$

$$A = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1} \qquad P = \begin{bmatrix}\vdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ \vdots & \vdots\end{bmatrix}$$

$$\Downarrow$$

$$\boldsymbol{z}_{n+1} = \begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}\boldsymbol{z}_n \Rightarrow \boldsymbol{z}_{n+1} = \begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}^n\boldsymbol{z}_1$$



$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix}1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2\end{bmatrix}$$

Stability of PD Control

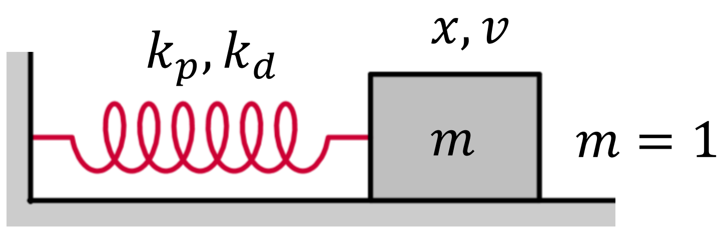
$$\boldsymbol{s}_{n+1} = A\boldsymbol{s}_n \quad \Rightarrow \quad \boldsymbol{s}_{n+1} = A^n\boldsymbol{s}_1$$

Assume A has two eigenvalues λ_1, λ_2 and two eigen vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$. Note $\lambda_i \in \mathbb{C}$

$$A = P\begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}P^{-1} \qquad P = \begin{bmatrix}\vdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ \vdots & \vdots\end{bmatrix}$$

$$\Downarrow$$

$$\boldsymbol{z}_{n+1} = \begin{bmatrix}\lambda_1 & \\ & \lambda_2\end{bmatrix}\boldsymbol{z}_n \quad \Rightarrow \quad \boldsymbol{z}_{n+1} = \begin{bmatrix}\lambda_1^n & \\ & \lambda_2^n\end{bmatrix}\boldsymbol{z}_1$$



$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix}1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2\end{bmatrix}$$

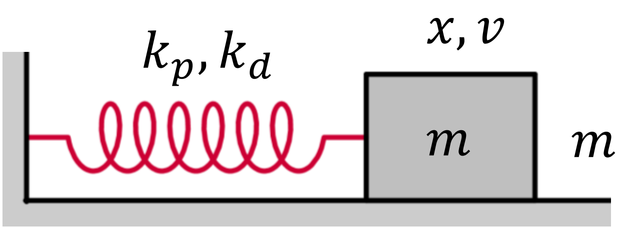
Stability of PD Control

$$\boldsymbol{z}_{n+1} = \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} \boldsymbol{z}_n \Rightarrow \boldsymbol{z}_{n+1} = \begin{bmatrix} \lambda_1^n & \\ & \lambda_2^n \end{bmatrix} \boldsymbol{z}_1$$

$\lambda_1, \lambda_2 \in \mathbb{C}$ are eigenvalues of A

if $|\lambda_1| > 1$ $\Rightarrow$ $\lim_{n\to\infty} \|\boldsymbol{z}_n\| \to \infty$

The system is unstable!



$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix} 1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2 \end{bmatrix}$$

Stability of PD Control

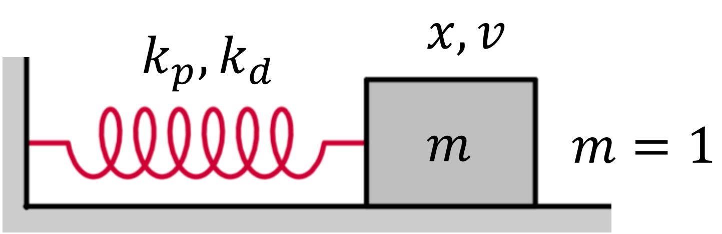
$$\boldsymbol{z}_{n+1} = \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} \boldsymbol{z}_n \Rightarrow \boldsymbol{z}_{n+1} = \begin{bmatrix} \lambda_1^n & \\ & \lambda_2^n \end{bmatrix} \boldsymbol{z}_1$$

$\lambda_1, \lambda_2 \in \mathbb{C}$ are eigenvalues of A

if $|\lambda_1| > 1 \Rightarrow \lim_{n\to\infty} \|\boldsymbol{z}_n\| \to \infty$

The system is unstable!

Condition of stability: $|\lambda_i| \leq 1$ for all λ_i

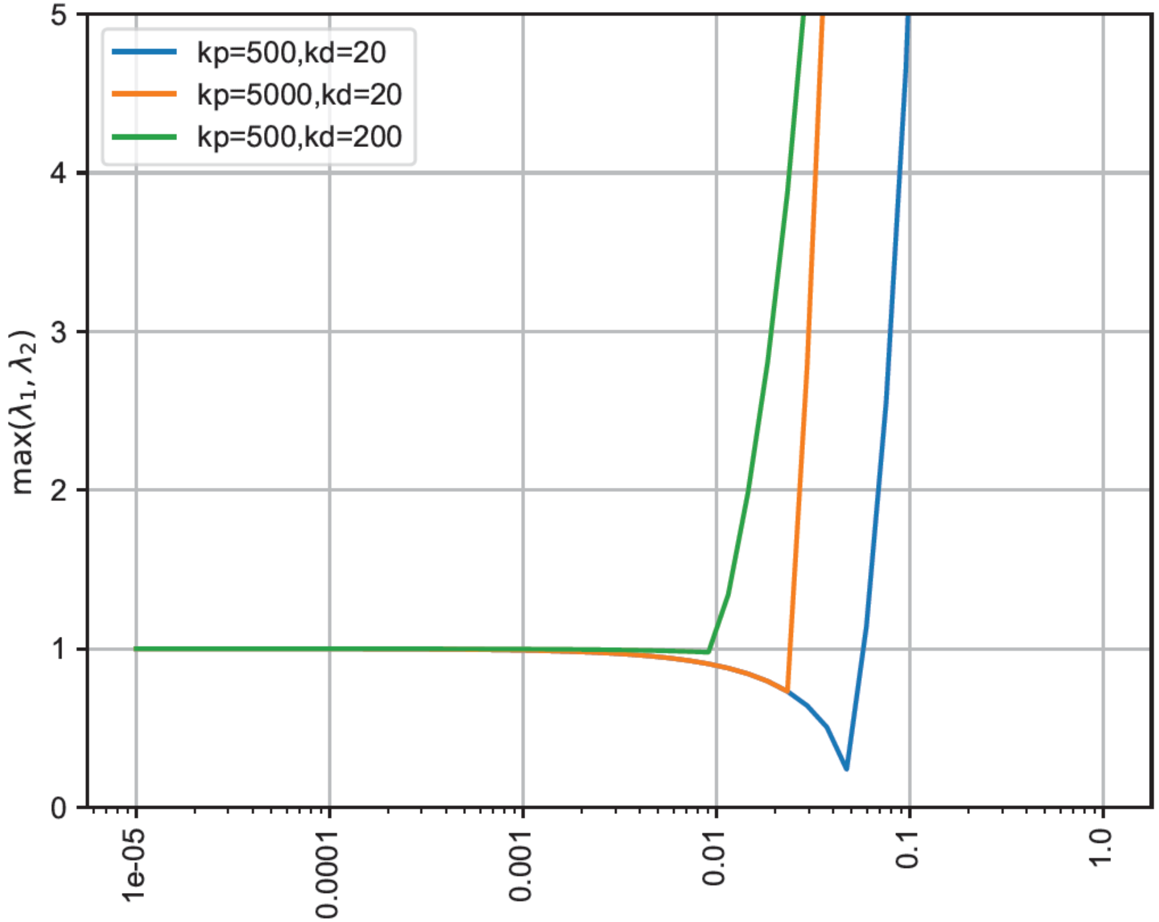


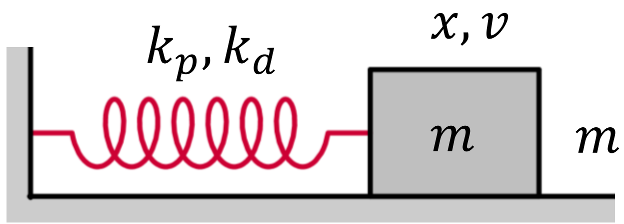
$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix} 1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2 \end{bmatrix}$$

Stability of PD Control





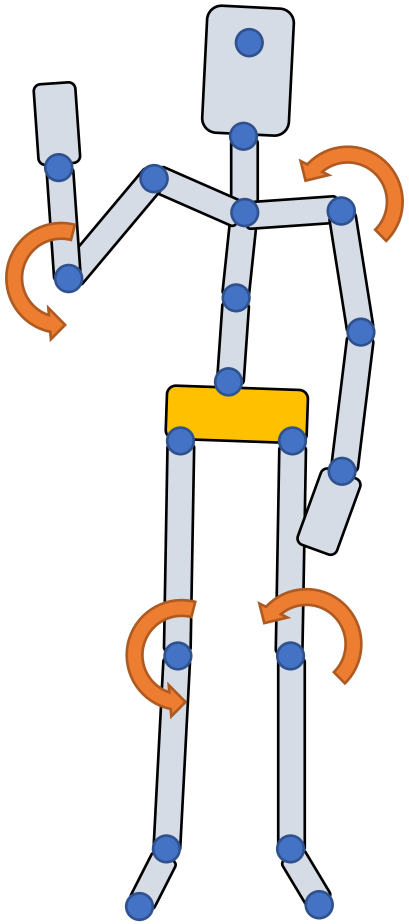
$m = 1$

$$f = -k_p x - k_d v$$

h: simulation time step

$$A = \begin{bmatrix} 1 - k_d h & -k_p h \\ h(1 - k_d h) & 1 - k_p h^2 \end{bmatrix}$$

PD Control for Characters



$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

- Determining gain and damping coefficients can be difficult…
  - A typical setting $k_p = 200$, $k_d = 20$ for a 50kg character
  - Light body requires smaller gains
  - Dynamic motions need larger gains
- High-gain/high-damping control can be unstable, so small times is necessary
  - $h = 0.5{\sim}1\text{ms}$ is often used, or $1000{\sim}2000\text{Hz}$
  - Higher gain/damping requires smaller time step

A More Stable PD Control

Semi-implicit Euler Integration

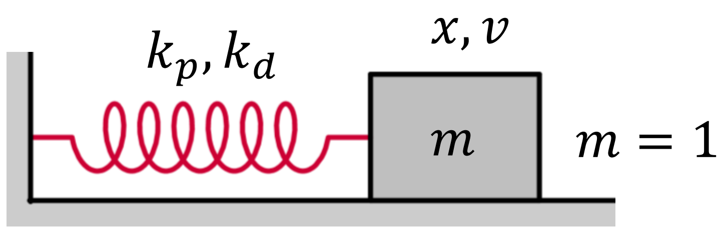
$$v_{n+1} = v_n + h(-k_p x_n - k_d v_n)$$

$$x_{n+1} = x_n + h v_{n+1}$$

$$\Downarrow$$

$$v_{n+1} = v_n + h(-k_p x_n - k_d \mathbf{v_{n+1}})$$

$$x_{n+1} = x_n + h v_{n+1}$$



$$f = -k_p x - k_d v$$

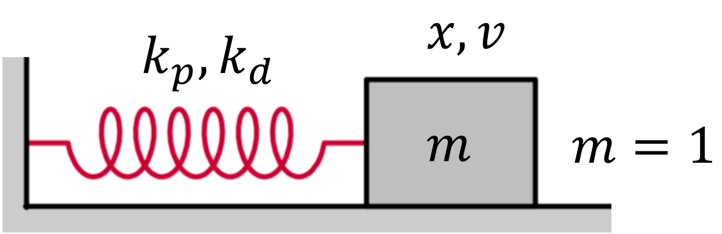
h: simulation time step

A More Stable PD Control

$$v_{n+1} = v_n + h\left(-k_p x_n - k_d \boldsymbol{v_{n+1}}\right)$$

$$x_{n+1} = x_n + h v_{n+1}$$

$$\begin{bmatrix} v_{n+1} \\ x_{n+1} \end{bmatrix} = \frac{1}{1 + hk_d} \begin{bmatrix} 1 & -k_p h \\ h & 1 + k_d h - k_p h^2 \end{bmatrix} \begin{bmatrix} v_n \\ x_n \end{bmatrix}$$

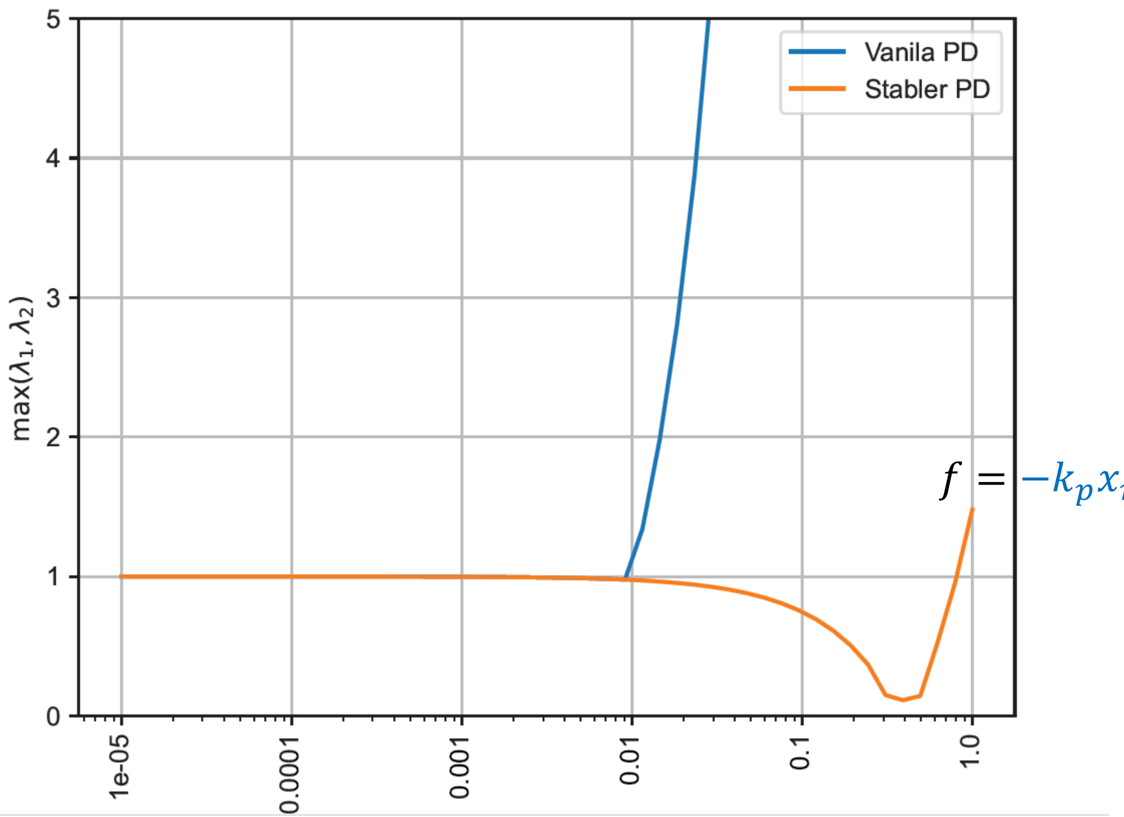


$$f = -k_p x - k_d v$$

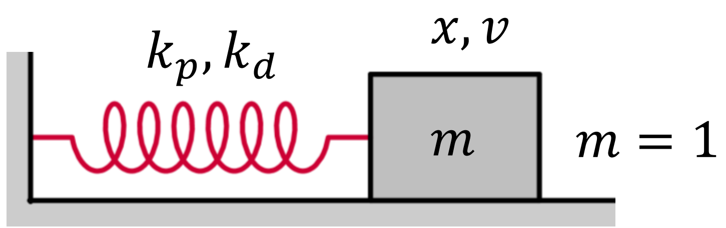
h: simulation time step

A More Stable PD Control

$f = -k_p x_n - k_d \boldsymbol{v_n}$



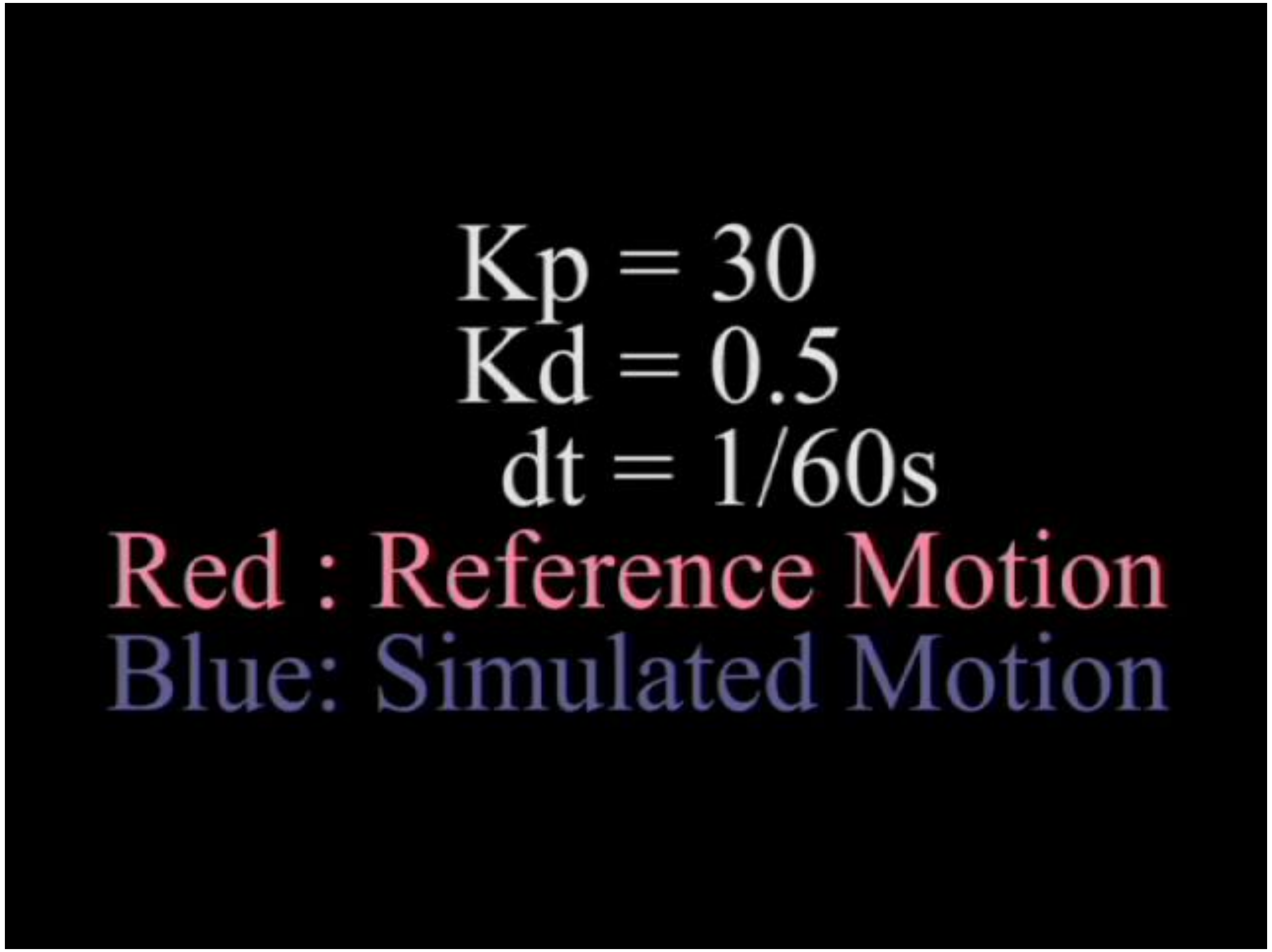
$f = -k_p x_n - k_d \boldsymbol{v_{n+1}}$



$$f = -k_p x - k_d v$$

h: simulation time step

Stable PD Control

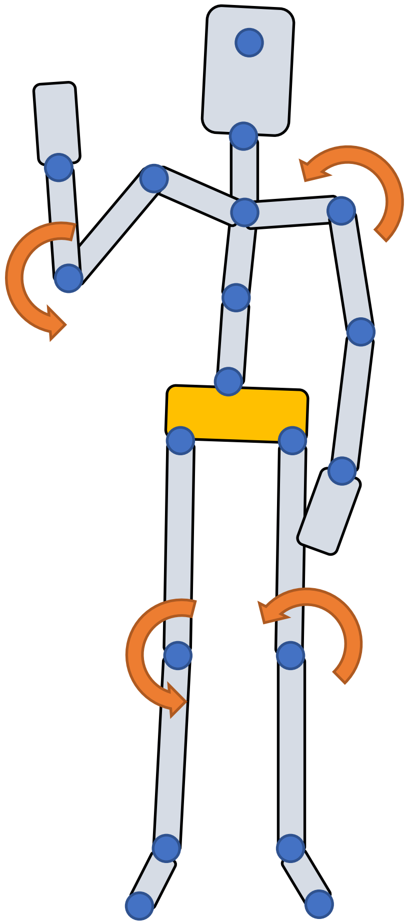


$$\tau_{\text{int}} = -\mathbf{K}_p(\mathbf{q}^n + \dot{\mathbf{q}}^n \Delta t - \bar{\mathbf{q}}^{n+1}) - \mathbf{K}_d(\dot{\mathbf{q}}^n + \ddot{\mathbf{q}}^n \Delta t)$$

Stable Proportional-Derivative Controllers

Jie Tan* Karen Liu† Greg Turk‡
Georgia Institute of Technology

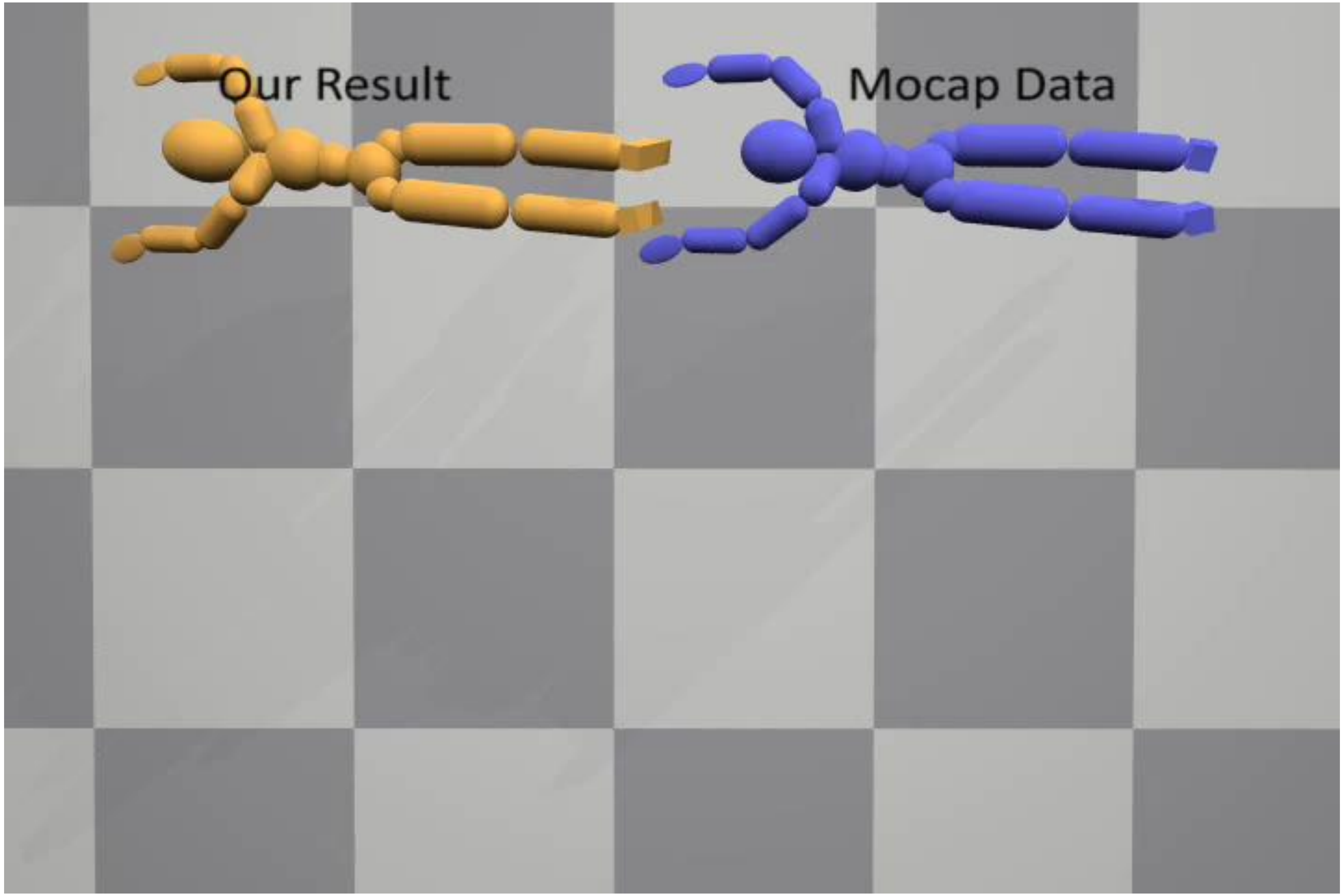
PD Control for Characters



$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

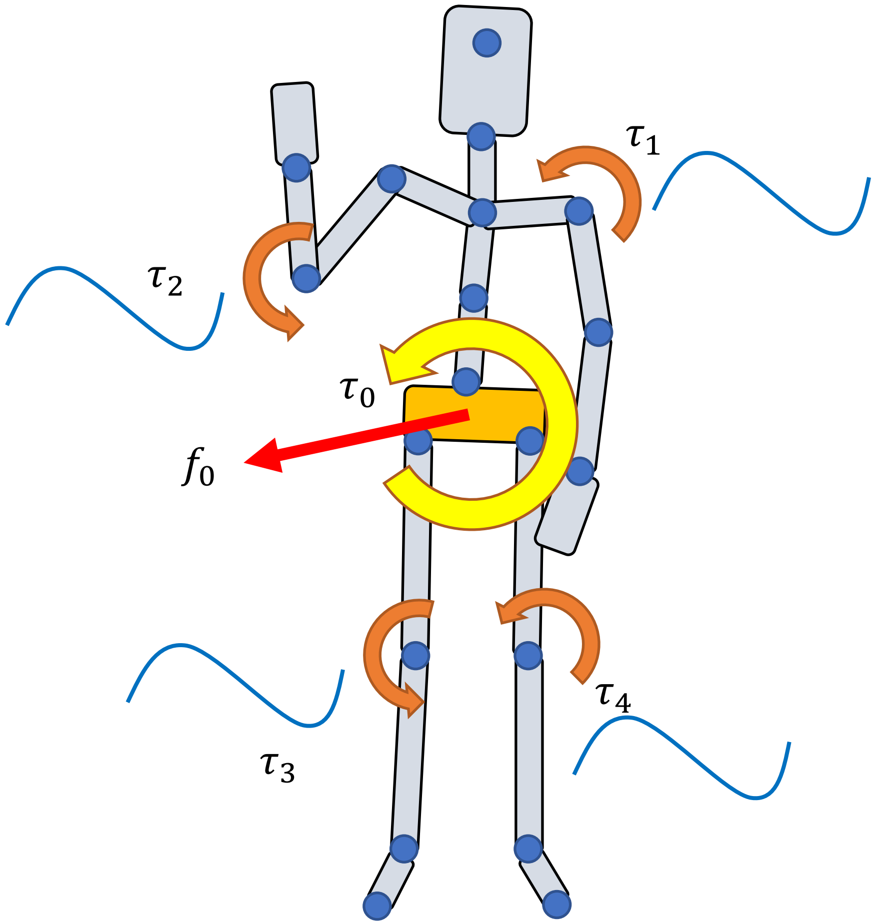
- Determining gain and damping coefficients can be difficult…
  - A typical setting $k_p = 200$, $k_d = 20$ for a $50\mathrm{kg}$ character
  - Light body requires smaller gains
  - Dynamic motions need larger gains
- High-gain/high-damping control can be unstable, so small times is necessary
  - $h = 0.5\sim1\mathrm{ms}$ is often used, or $1000\sim2000\mathrm{Hz}$
  - $h = 1/120\mathrm{s}\sim1/60\mathrm{s}$, or 120Hz/60Hz with Stable PD
  - Higher gain/damping requires smaller time step

Tracking Mocap



[SAMCON – Liu et al 2010]

Tracking Mocap with Root Forces/Torques



τ_j: joint torques

Apply τ_j to “child” body

Apply $-\tau_j$ to “parent” body

All forces/torques sum up to zero

f_0, τ_0: root force / torque

Apply f_0 to the root body

Apply τ_0 to the root body

Non-zero net force/torque on the character!

Physically Plausible Animation

Party Animals

Totally Accurate Battle Simulator
https://www.youtube.com/watch?v=WFKGWfdG3bU

Trajectory Optimization

Find the trajectories:

Simulation trajectory: $s_0, s_1, \dots, s_T$

Control trajectory: $a_0, a_1, \dots, a_{T-1}$

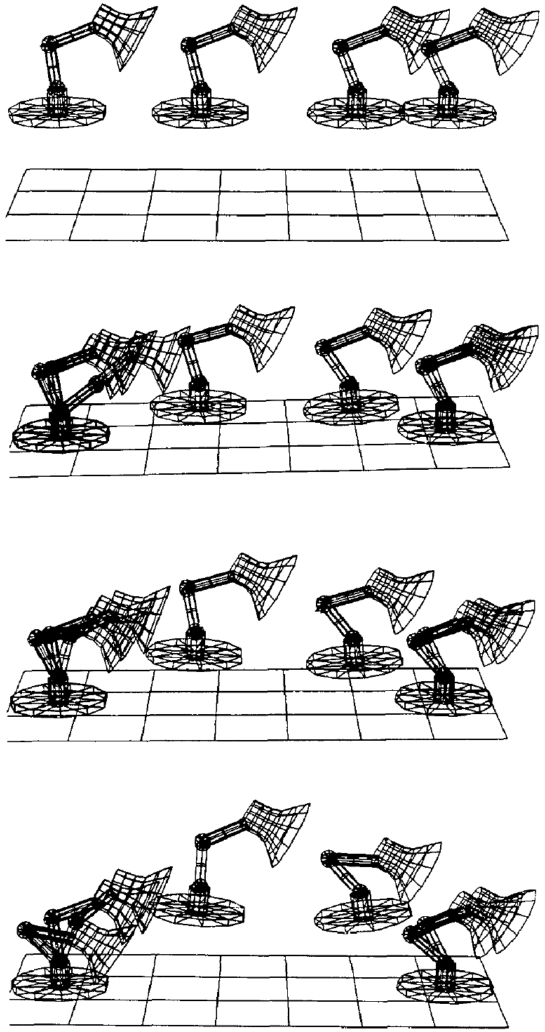
that minimize the objective function

$$\min_{\{s_t, a_t\}} f(s_T) + \sum_{t=0}^{T-1} f(s_t, a_t)$$

and satisfy the constraints:

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda \qquad \text{Equations of motion}$$

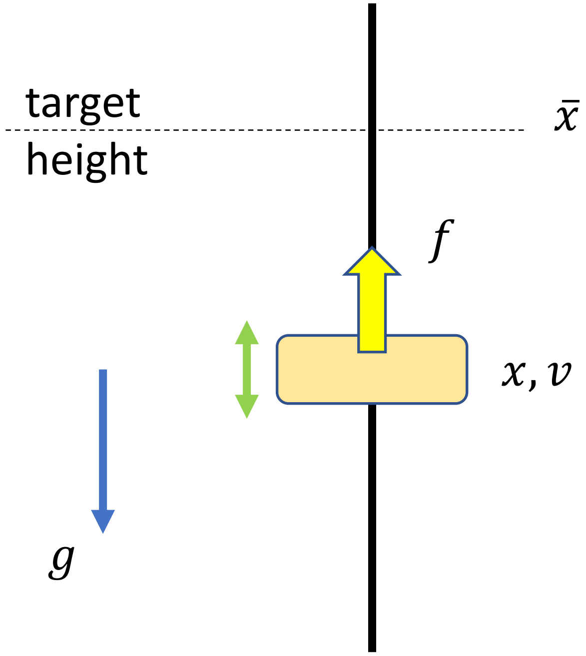
$$g(\boldsymbol{x}, \boldsymbol{v}) \geq 0 \qquad \text{constraints}$$



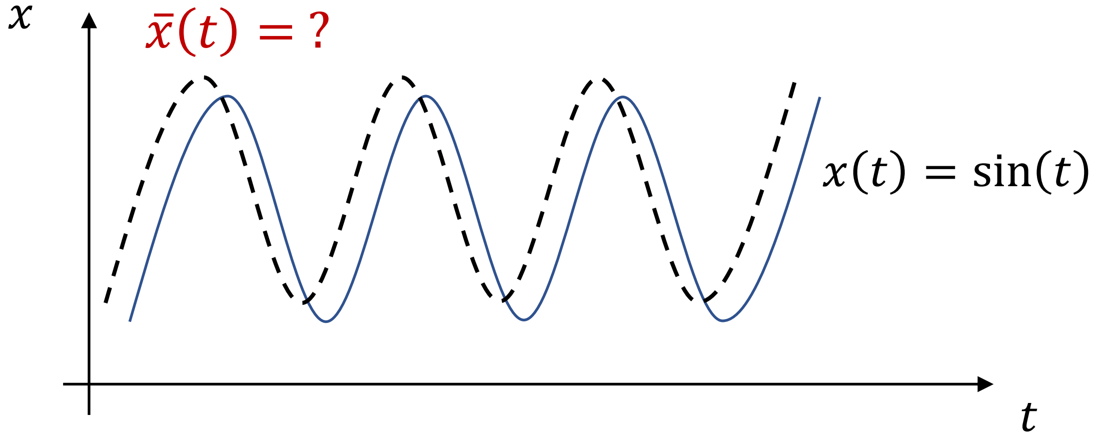
[Witkin and Kass 1988 – Spacetime constraints]

A very simple example

$$f = k_p(\bar{x} - x) - k_d v$$

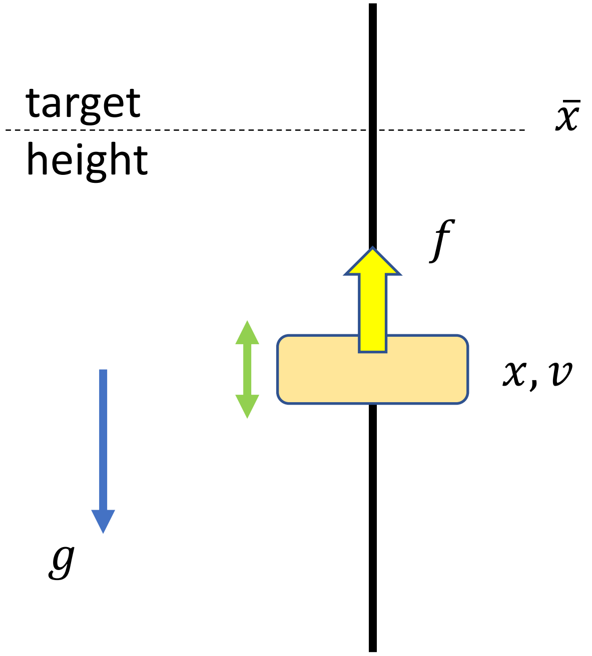


Compute a target trajectory $\bar{x}(t)$
such that the simulated trajectory $x(t)$
is a sine curve.



A very simple example

$$f = k_p(\bar{x} - x) - k_d v$$

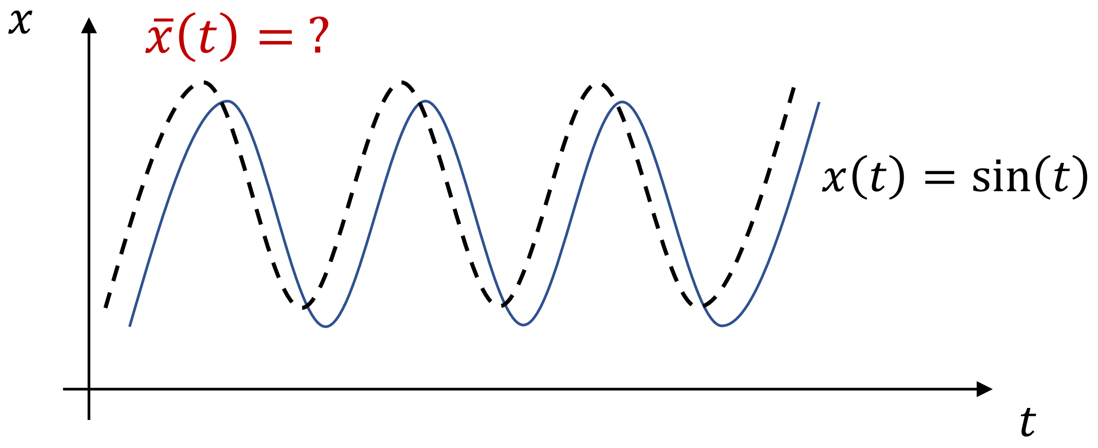


$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

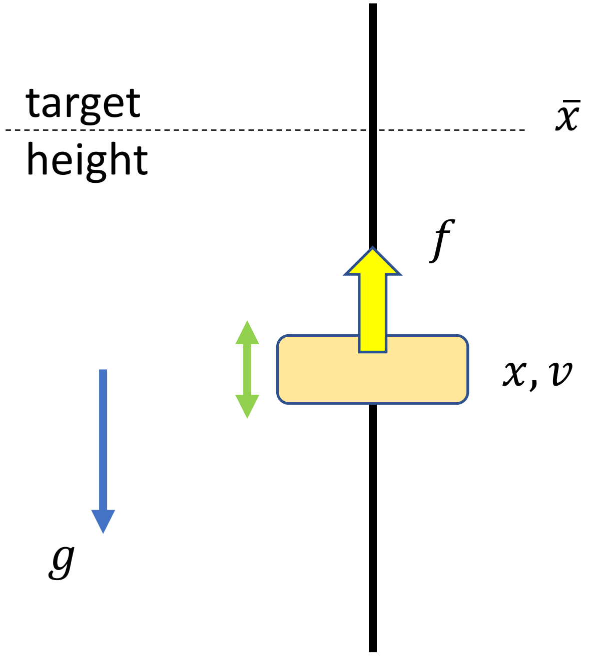
$$x_{n+1} = x_n + h v_{n+1}$$

Compute a target trajectory $\bar{x}(t)$ such that the simulated trajectory $x(t)$ is a sine curve.



A very simple example

$$f = k_p(\bar{x} - x) - k_d v$$

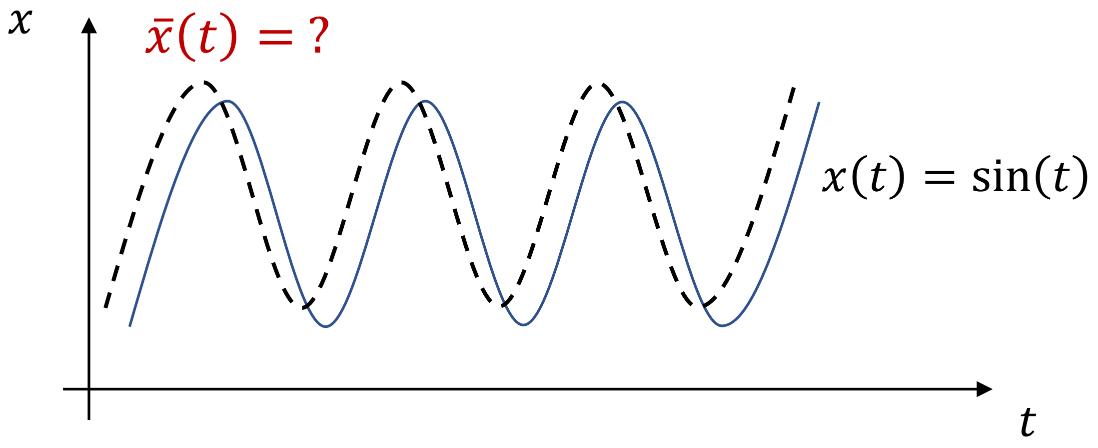


Compute a target trajectory $\bar{x}(t)$
such that the simulated trajectory $x(t)$
is a sine curve.

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + \sum_{n=0}^{N} \bar{x}_n^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$



A very simple example

Hard constraints:

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + w_{\bar{x}} \sum_{n=0}^{N} \bar{x}_n^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$

Soft constraints:

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + w_{\bar{x}} \sum_{n=0}^{N} \bar{x}_n^2$$

$$+ w_v \sum_{n=0}^{N} \Big(v_{n+1} - v_n - h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)\Big)^2$$

$$+ w_x \sum_{n=0}^{N} (x_{n+1} - x_n - h v_{n+1})^2$$

A very simple example

Collocation methods:

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} \left(\sin(t_n;\theta) - x(t_n;\theta)\right)^2 + w_{\bar{x}} \sum_{n=0}^{N} \bar{x}^2(t_n;\theta)$$

$$s.t. \quad v(t_{n+1};\theta) = v(t_n) + hk_p\left(\bar{x}(t_n;\theta) - x(t_n;\theta)\right) - hk_d v(t_n;\theta)$$

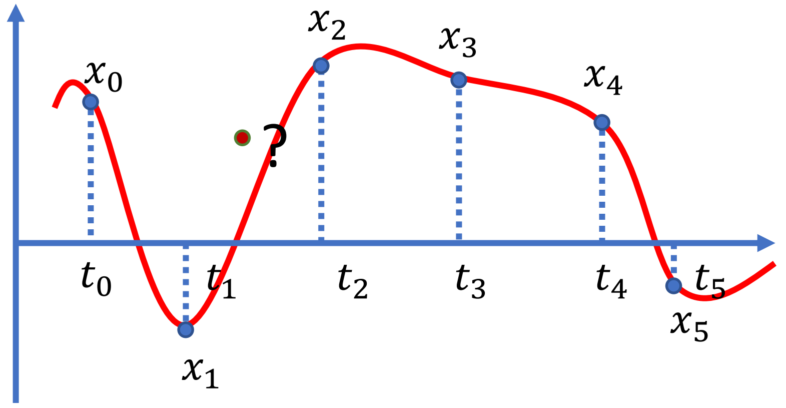
$$x(t_{n+1};\theta) = x(t_n;\theta) + hv(t_{n+1};\theta)$$

Assume the optimization variables $\{x_n, v_n, \bar{x}_n\}$ are values of a set of parametric curves

- typically polynomials or splines

Optimize the parameters of the curves $\boldsymbol{\theta}$ instead

- with smaller number of variables than the original problem



A very simple example

How to solve this optimization problem?

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + \sum_{n=0}^{N} \bar{x}_n^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$

A very simple example

How to solve this optimization problem?

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + \sum_{n=0}^{N} \bar{x}_n^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$

Gradient-based approaches:

- Gradient descent
- Newton's methods
- Quasi-Newton methods
- ……

A very simple example

How to solve this optimization problem?

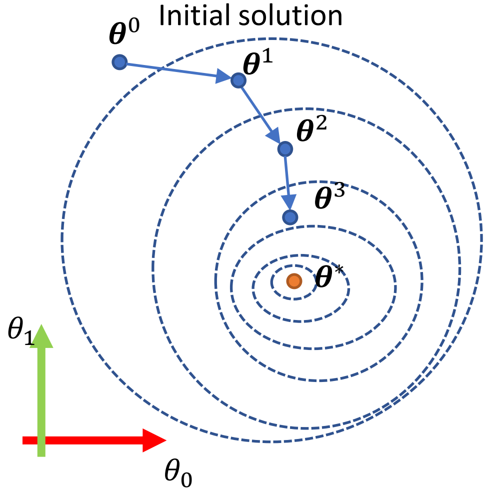
$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + \sum_{n=0}^{N} \bar{x}_n^2$$

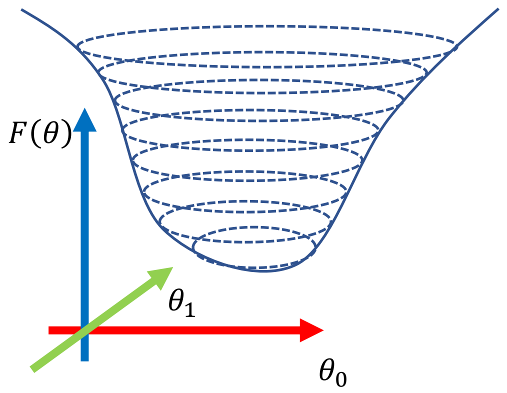
$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$

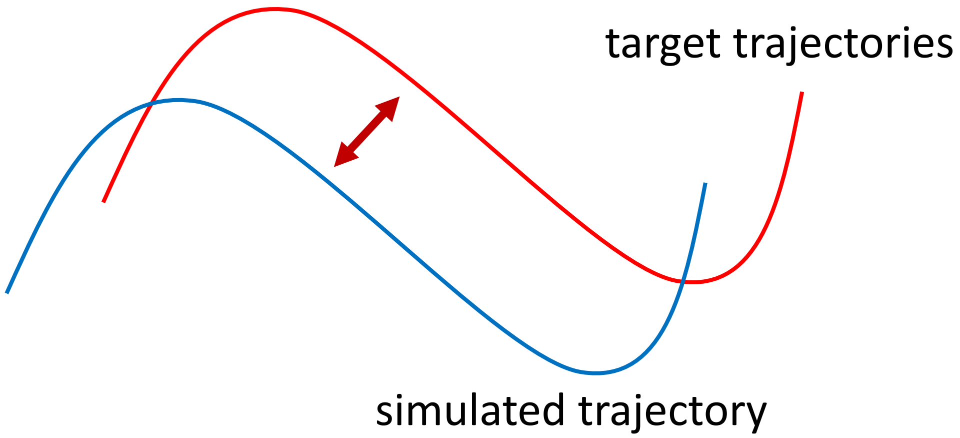
Gradient-based approaches:

- Gradient descent
- Newton's methods
- Quasi-Newton methods
- ……

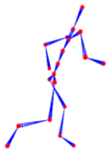




Trajectory Optimization for Tracking Control

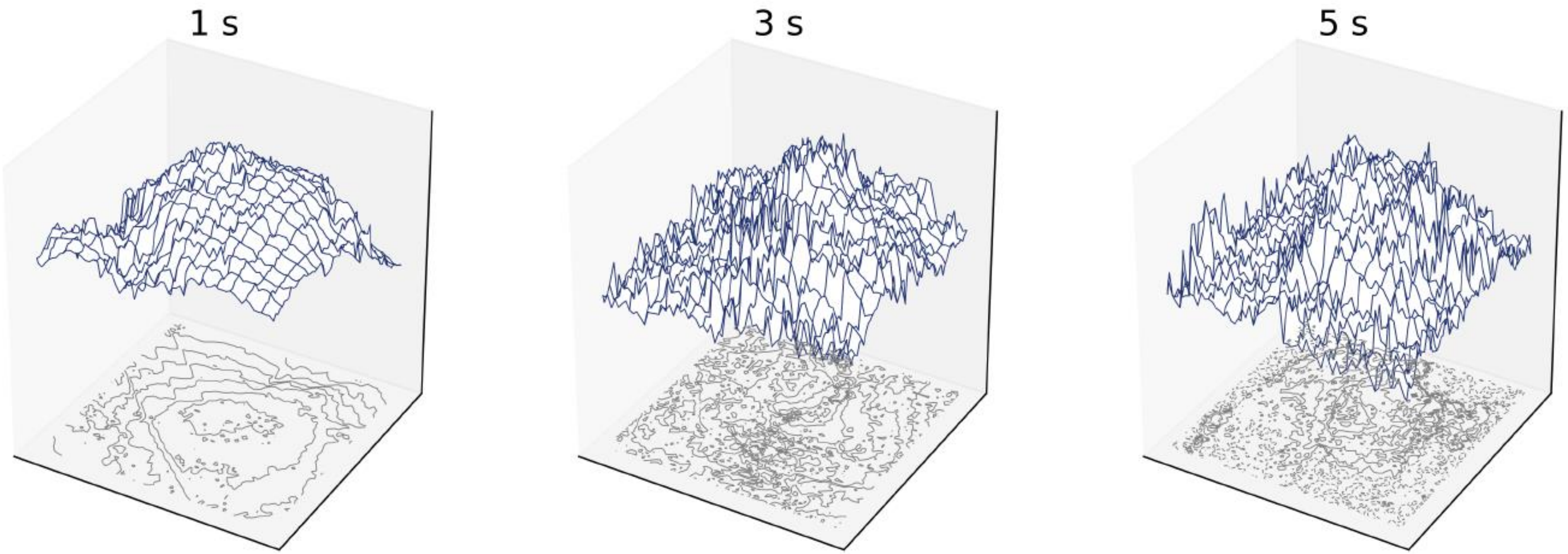


find a target trajectory

minimize ‖  −  ‖ + ⋯

Problem with Gradient-Based Methods

- The optimization problem is usually highly nonlinear, gradients are unreliable
- The system is a black box with unknow dynamics, gradients are not available



Trajectory optimization landscapes of humanoid locomotion

[Hämäläinen et al 2020 - Visualizing Movement Control Optimization Landscapes]

Derivative-Free Optimization

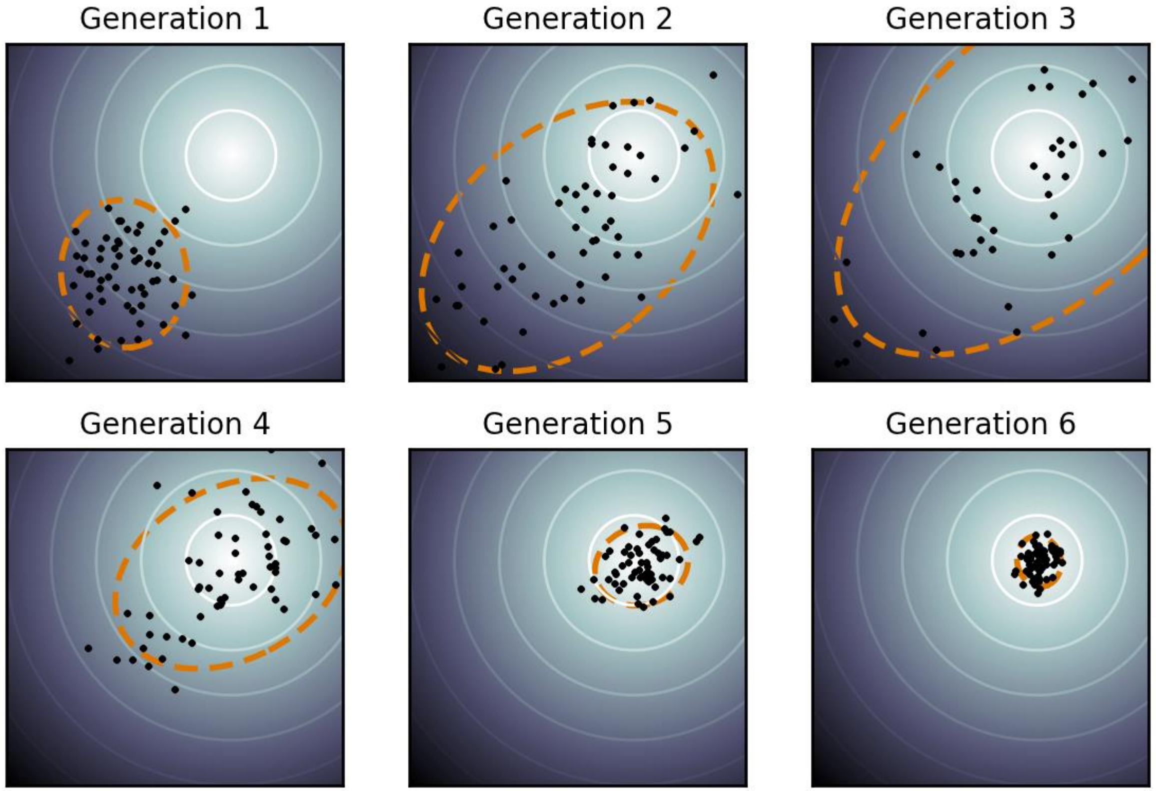
- Iterative methods
  - Goal: find the variables $\boldsymbol{x}$ that optimize $f(\boldsymbol{x})$
  - Determining an initial guess of $\boldsymbol{x}$
  - Repeat:
    - Propose a set of candidate variables $\{\boldsymbol{x}_i\}$ according to $\boldsymbol{x}$
    - Evaluate the objective function $f_i = f(\boldsymbol{x}_i)$
    - Update the estimation for $\boldsymbol{x}$

Derivative-Free Optimization

- Iterative methods
    - Goal: find the variables $\boldsymbol{x}$ that optimize $f(\boldsymbol{x})$
    - Determining an initial guess of $\boldsymbol{x}$
    - Repeat:
        - Propose a set of candidate variables $\{\boldsymbol{x}_i\}$ according to $\boldsymbol{x}$
        - Evaluate the objective function $f_i = f(\boldsymbol{x}_i)$
        - Update the estimation for $\boldsymbol{x}$
- Examples:
    - Bayesian optimization, Evolution strategies (e.g. CMA-ES), Stochastic optimization, Sequential Monte Carlo methods, ……

CMA-ES

- Covariance matrix adaptation evolution strategy (CMA-ES)
  - A widely adopted derivative-free method in character animation

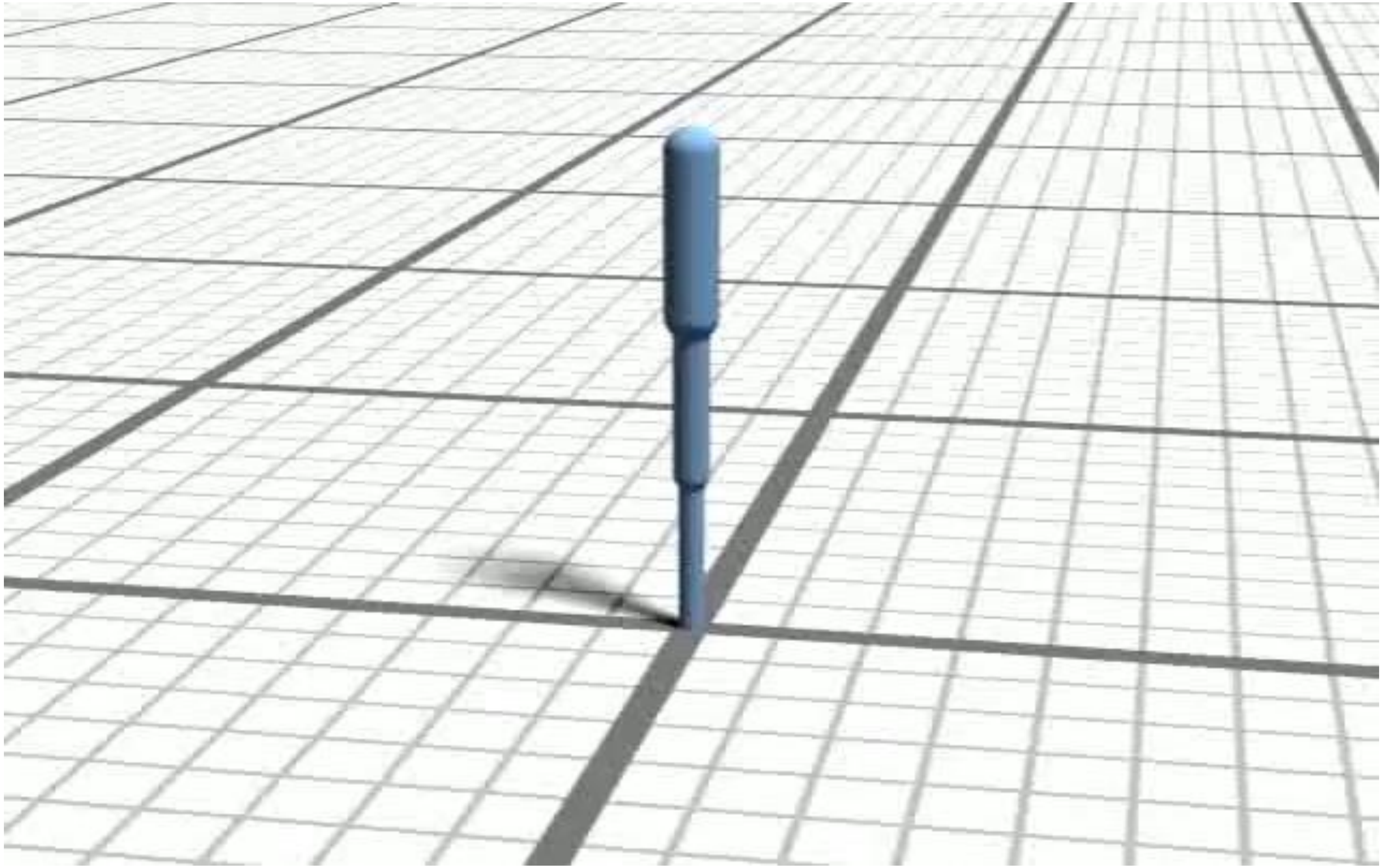


[Hansen 2006, The CMA evolution strategy: a comparing review]

Goal: find the variables $\boldsymbol{x}$ that optimize $f(\boldsymbol{x})$

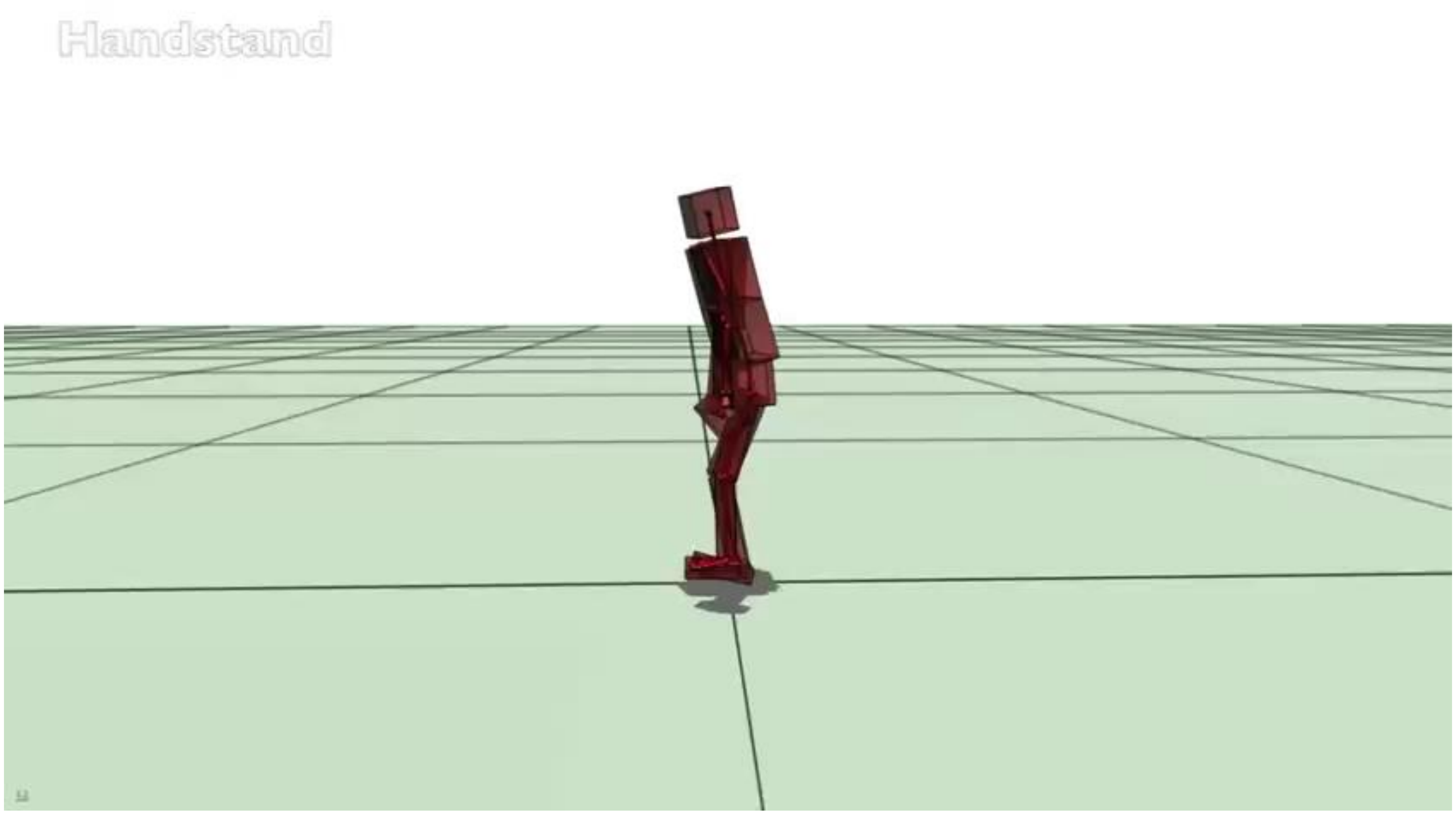
- Initialize Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$
- Repeat:
  - sample candidate variables $\{\boldsymbol{x}_i\} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$
  - Evaluate the objective function $f_i = f(\boldsymbol{x}_i)$
    - Involve simulation and generate simulation trajectories
  - Sort $\{f_i\}$ and keep the top N elite samples
  - Update $\boldsymbol{\mu}, \Sigma$ according to the elite samples

CMA-ES



[Wampler and Popović 2009 - *Optimal Gait and Form for Animal Locomotion*]

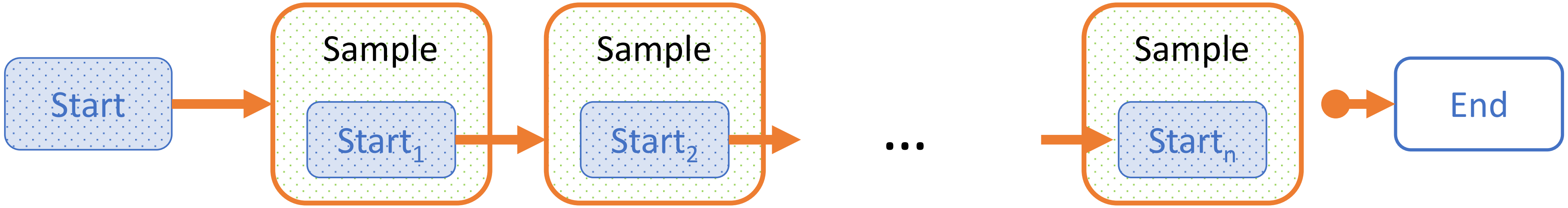
CMA-ES



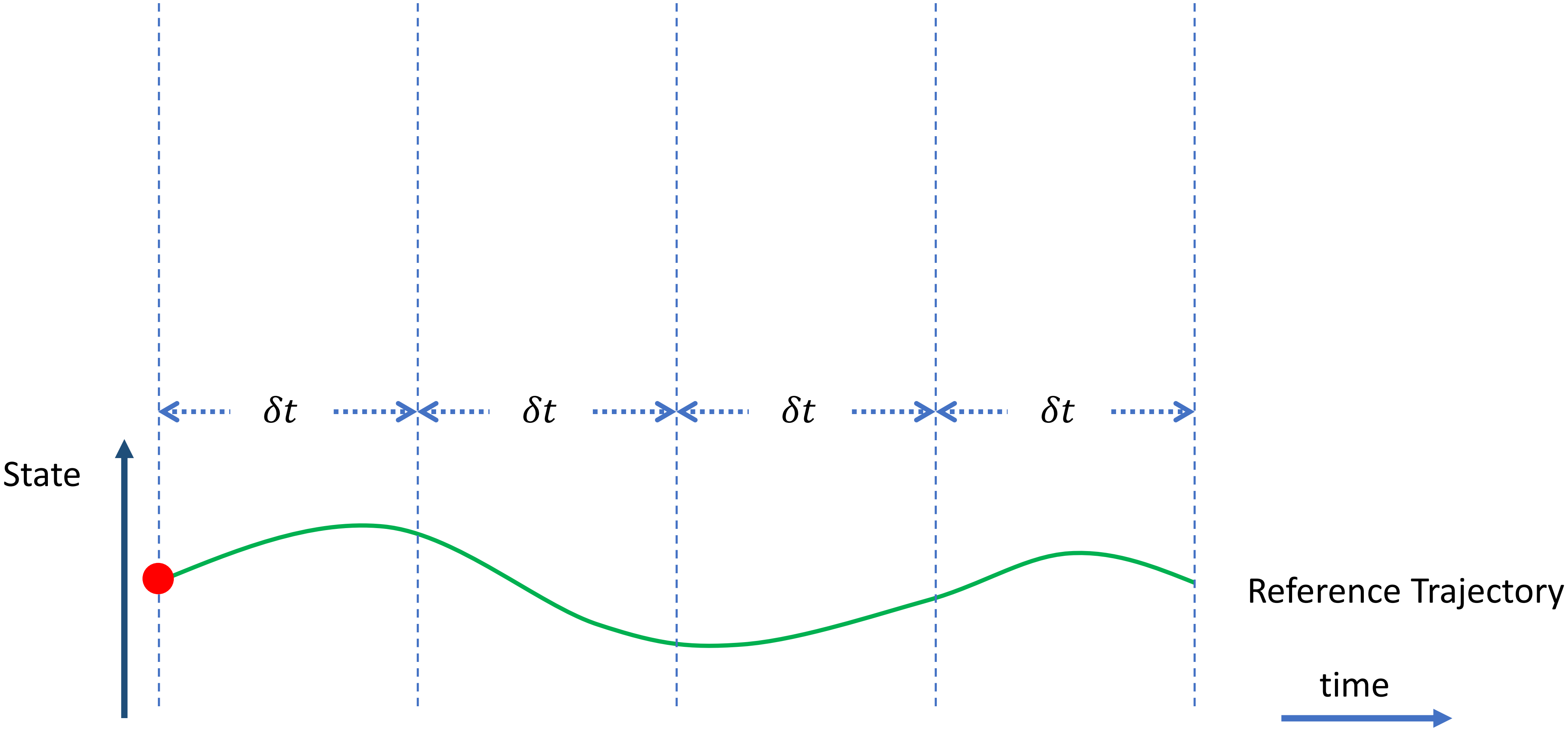
[Al Borno et al. 2013 - *Trajectory Optimization for Full-Body Movements with Complex Contacts*]

SAMCON

- **SA**mpling-based **M**otion **CON**trol [Liu et al. 2010, 2015]
  - Motion Clip → Open-loop control trajectory
  - A sequential Monte-Carlo method

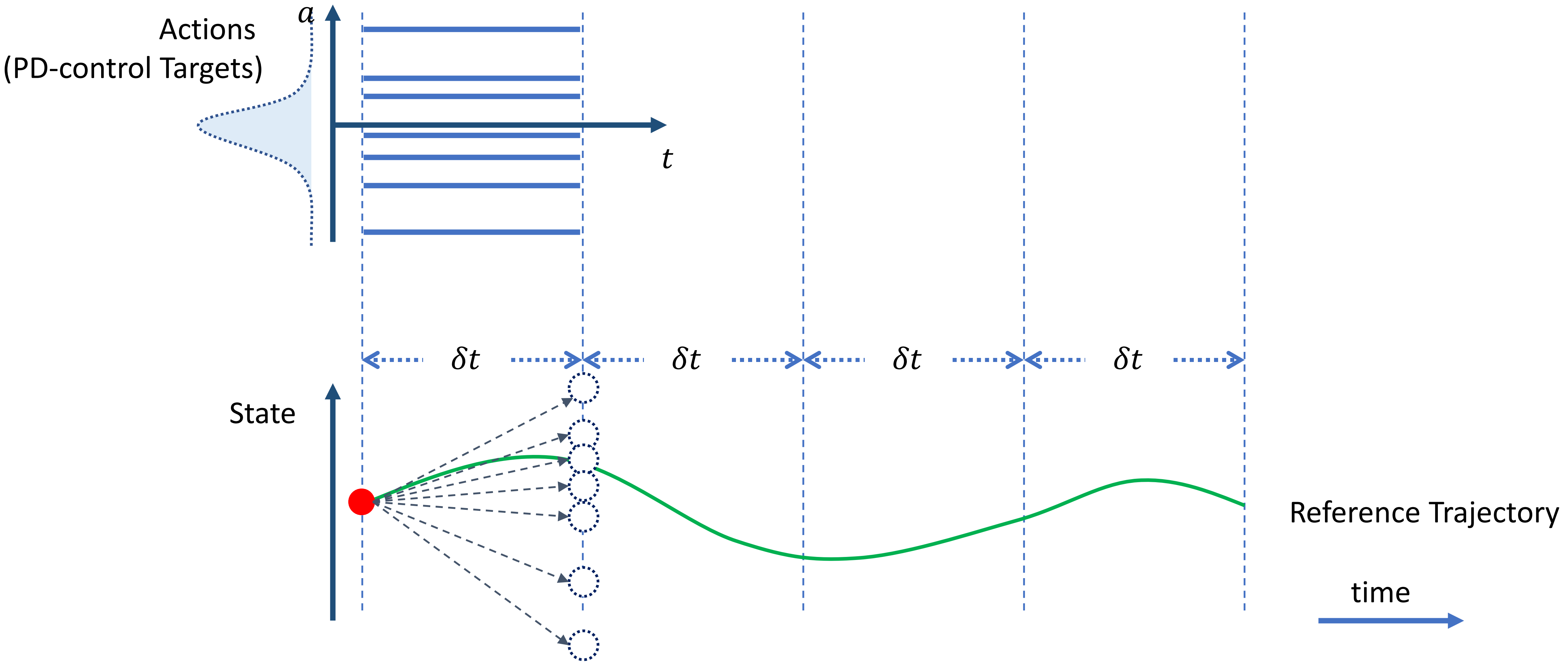


SAMCON

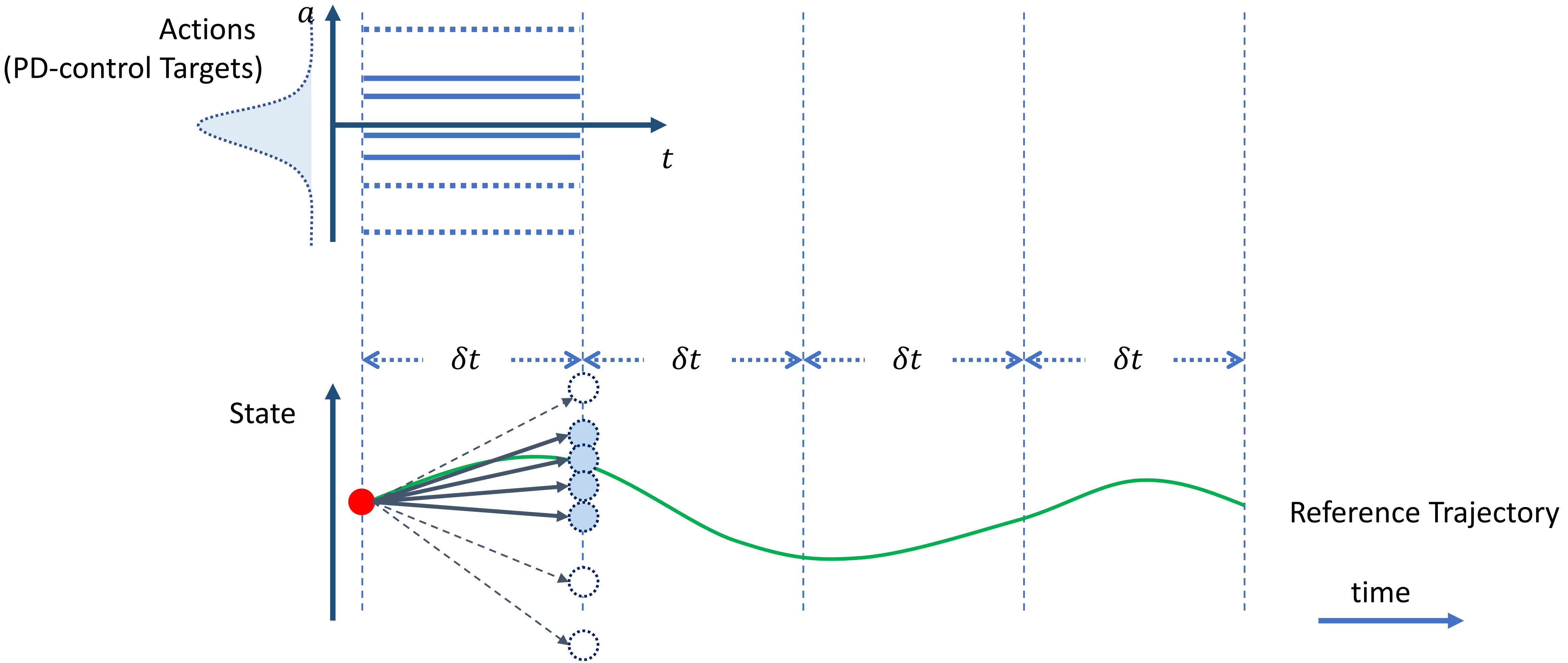


δt
δt
δt
δt
State
Reference Trajectory
time

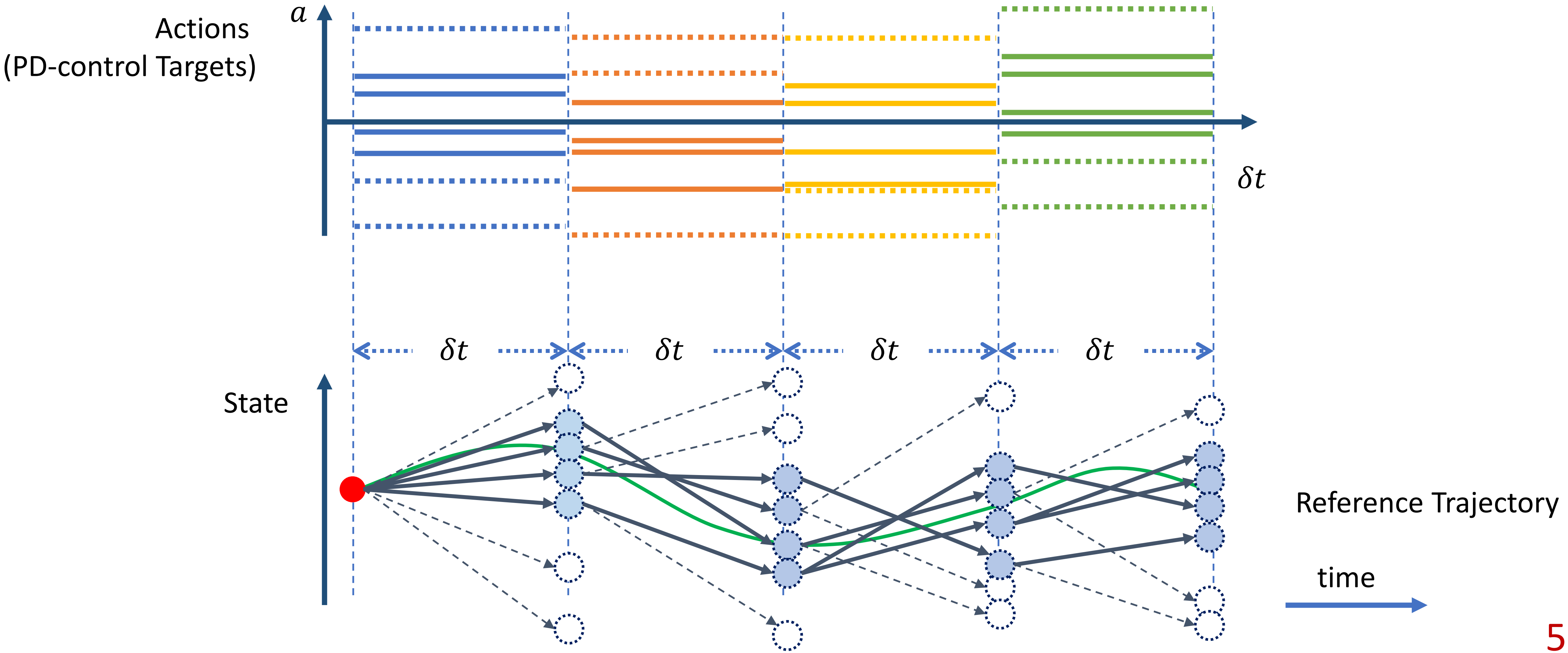
Sampling & Simulation



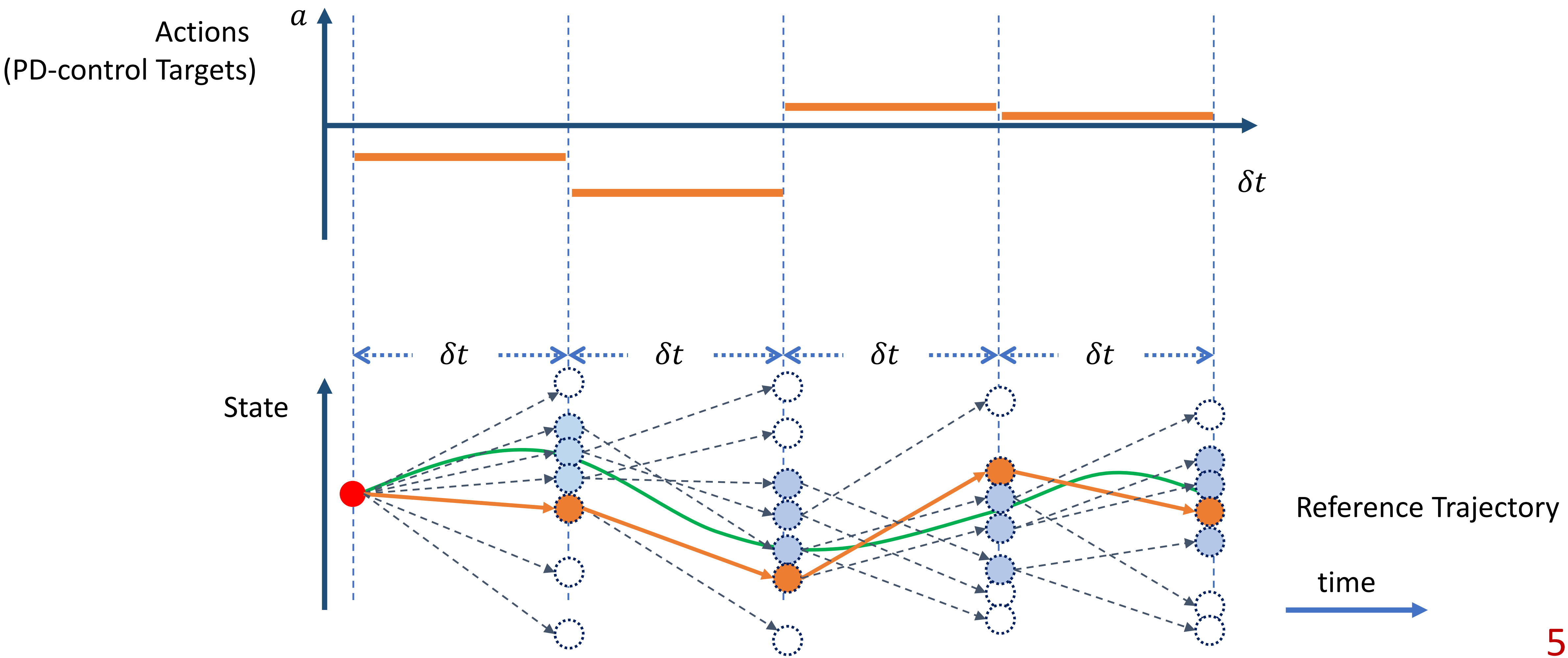
Sample Selection

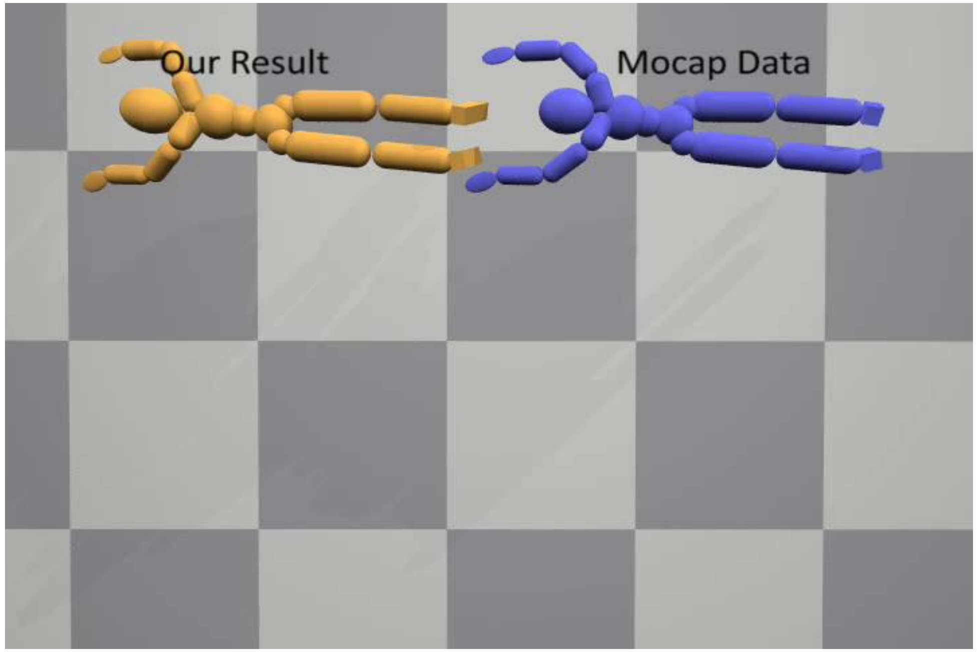


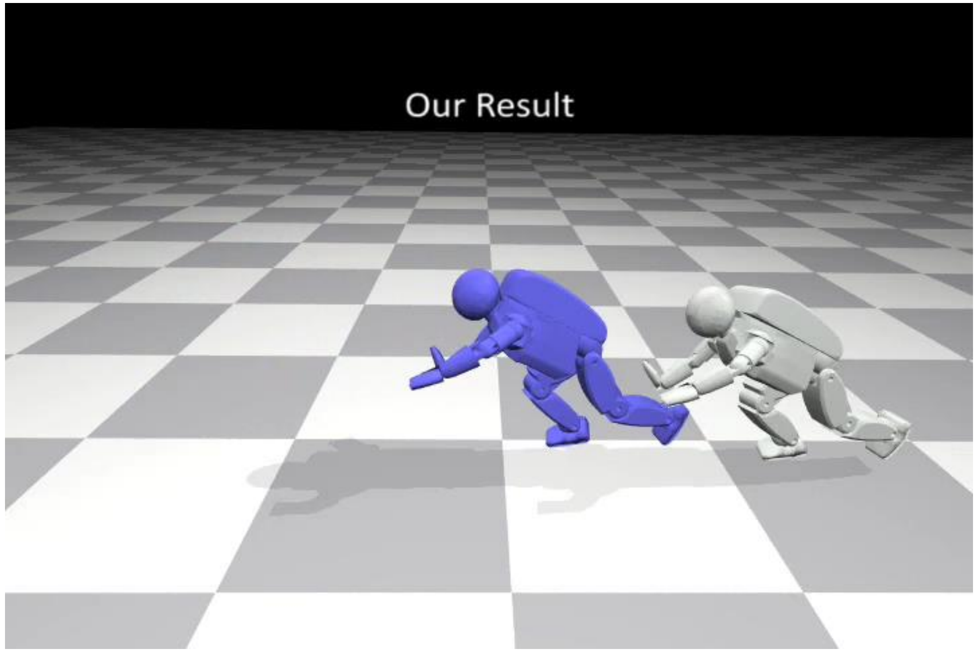
SAMCON Iterations

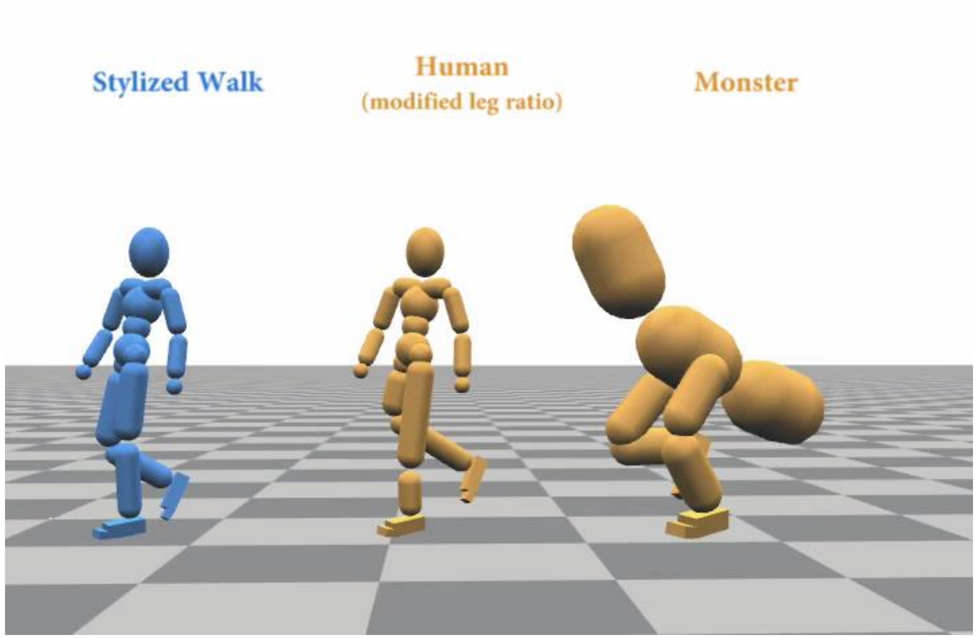


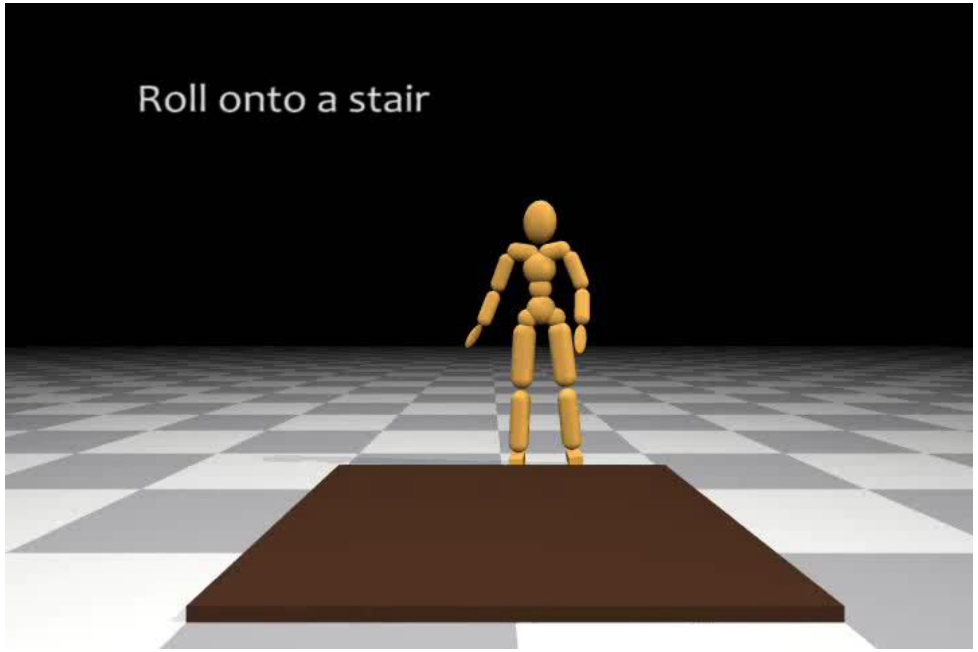
Constructed Open-loop Control Trajectory



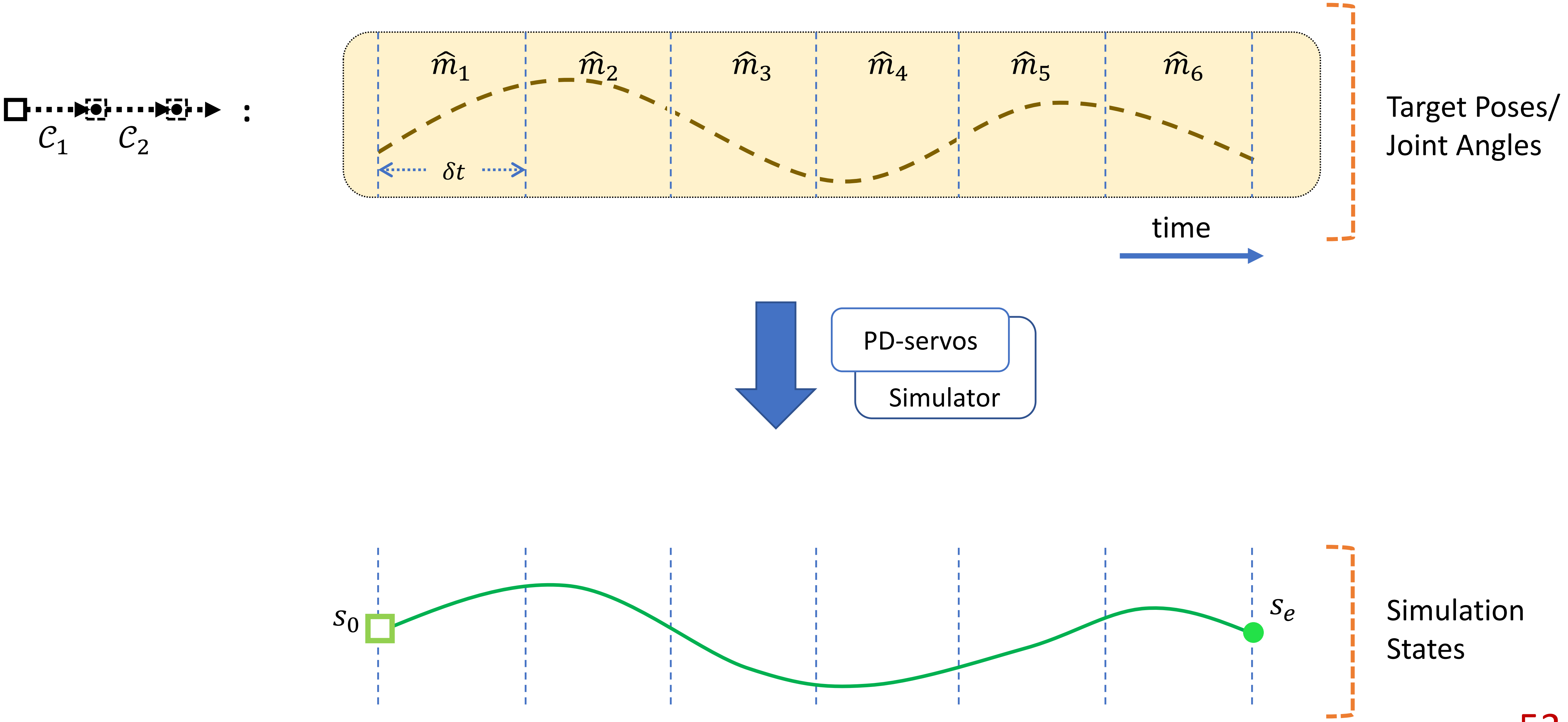

Our Result
Mocap Data


Our Result

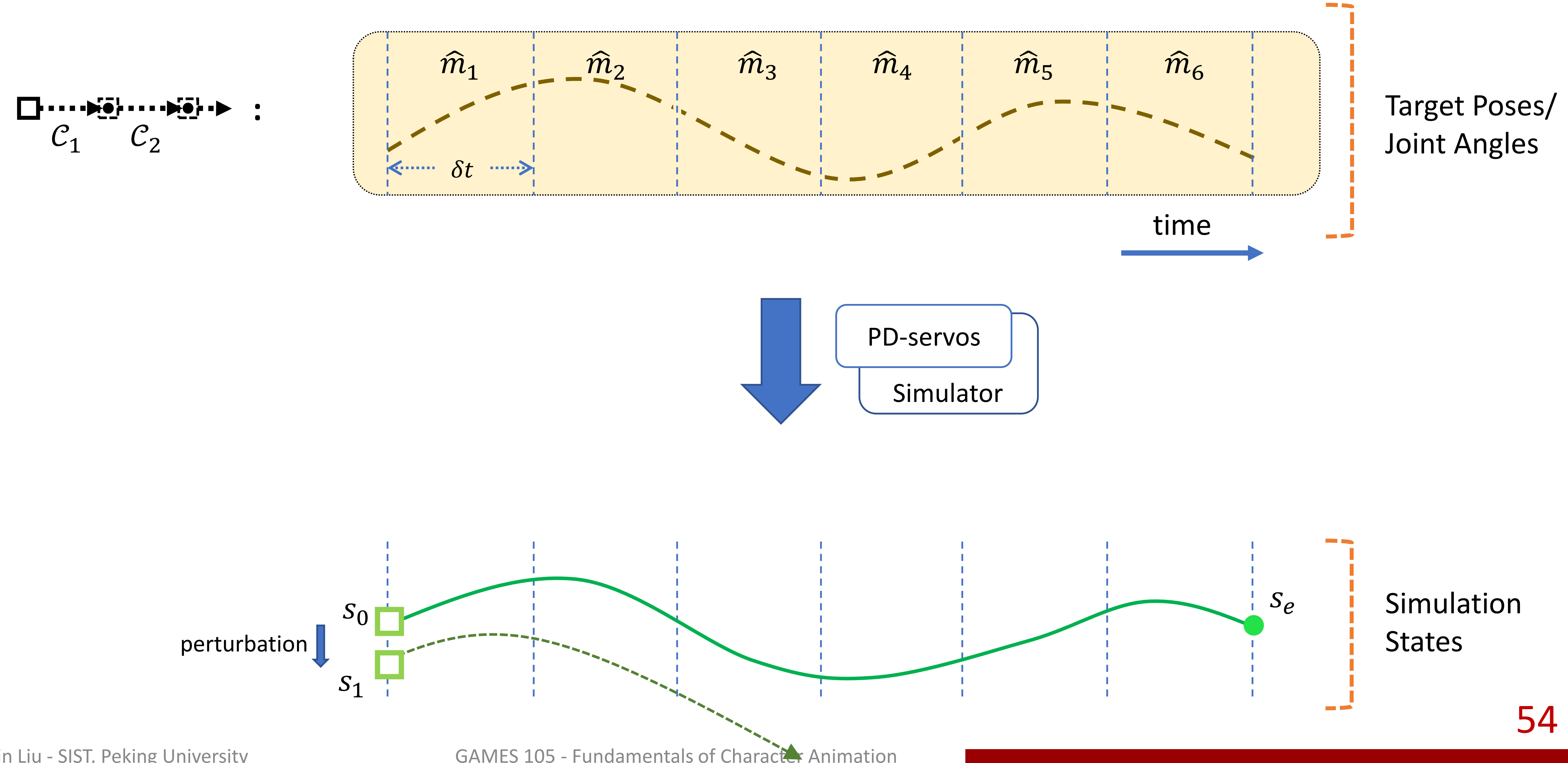

Stylized Walk
Human
(modified leg ratio)
Monster


Roll onto a stair

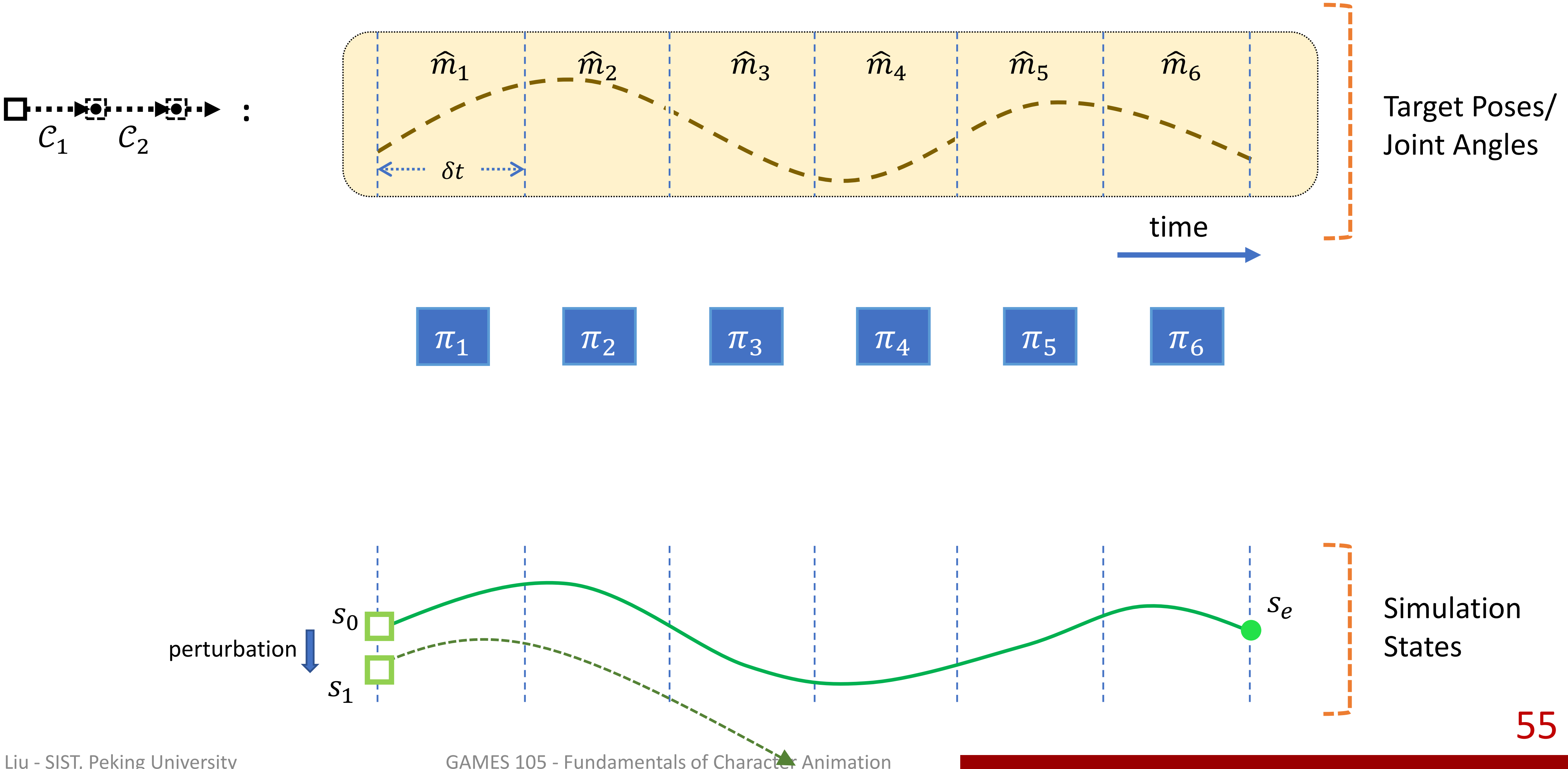
Feedforward Control



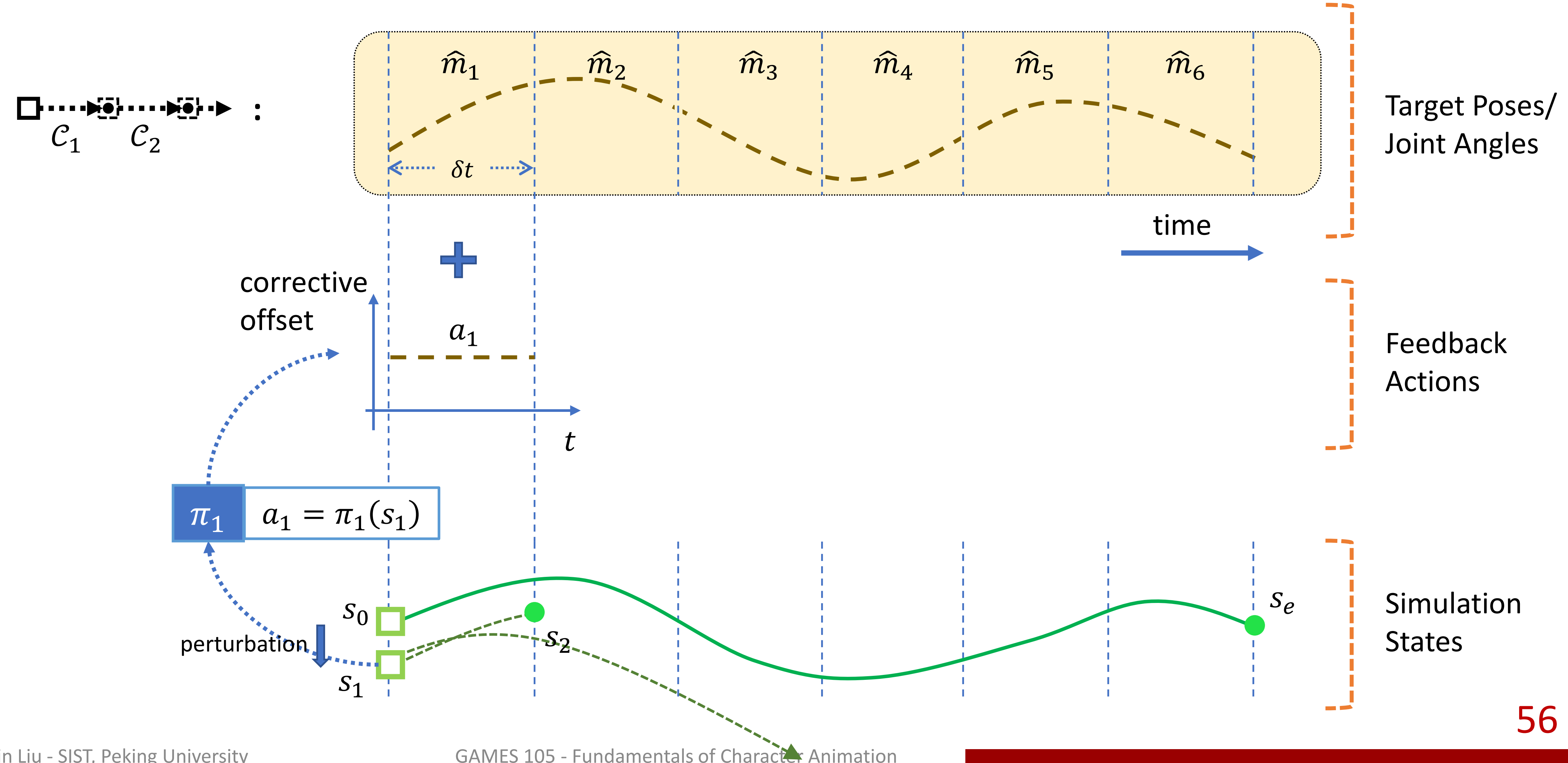
Feedforward Control



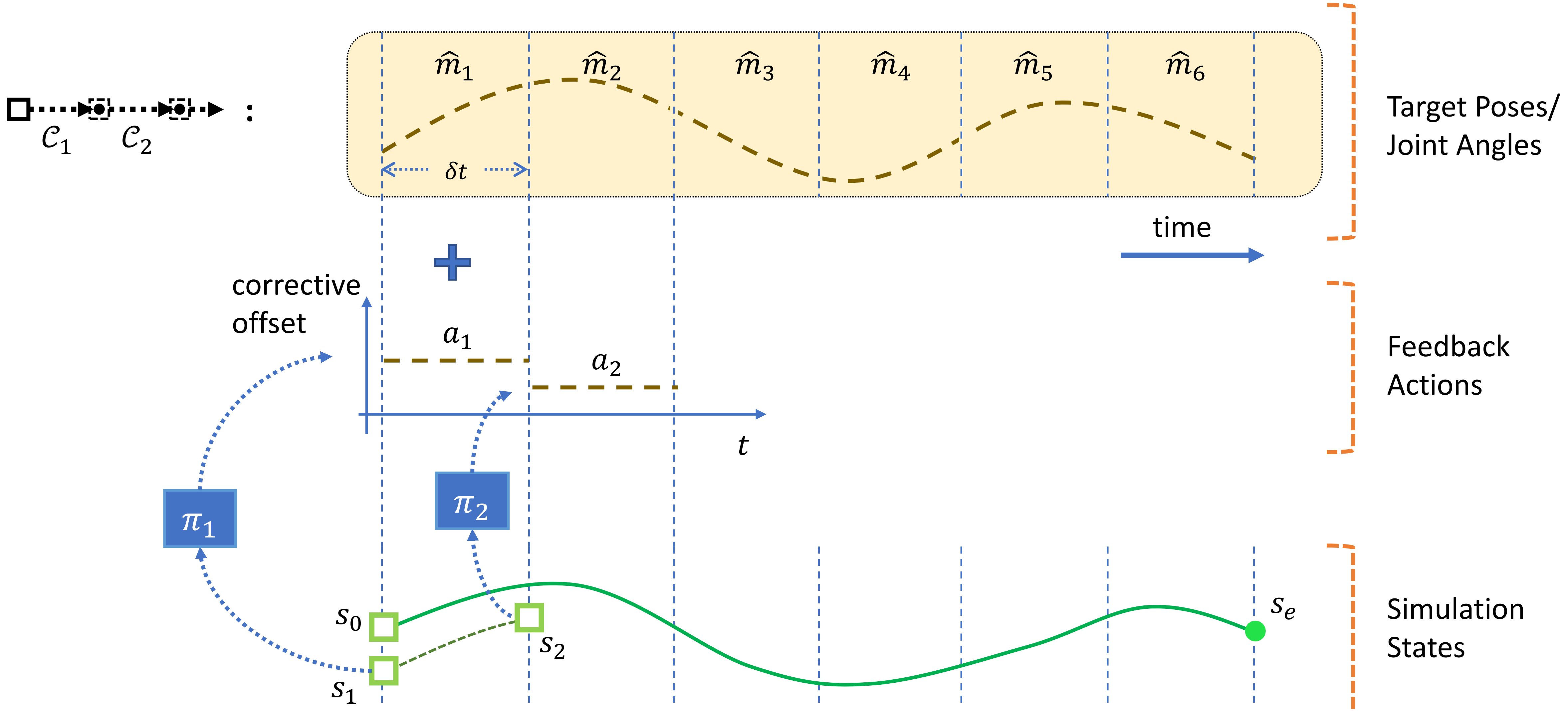
Feedforward Control



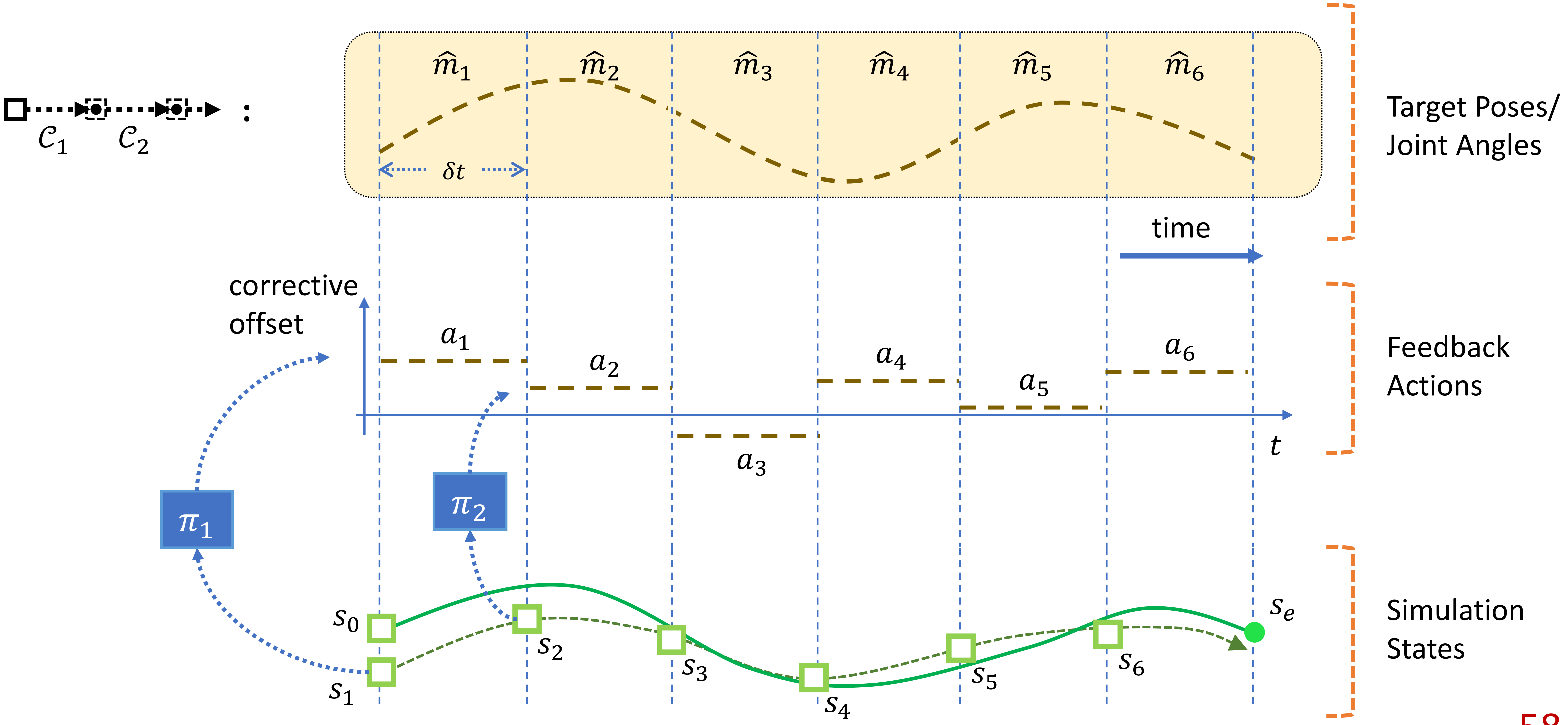
Feedback Control



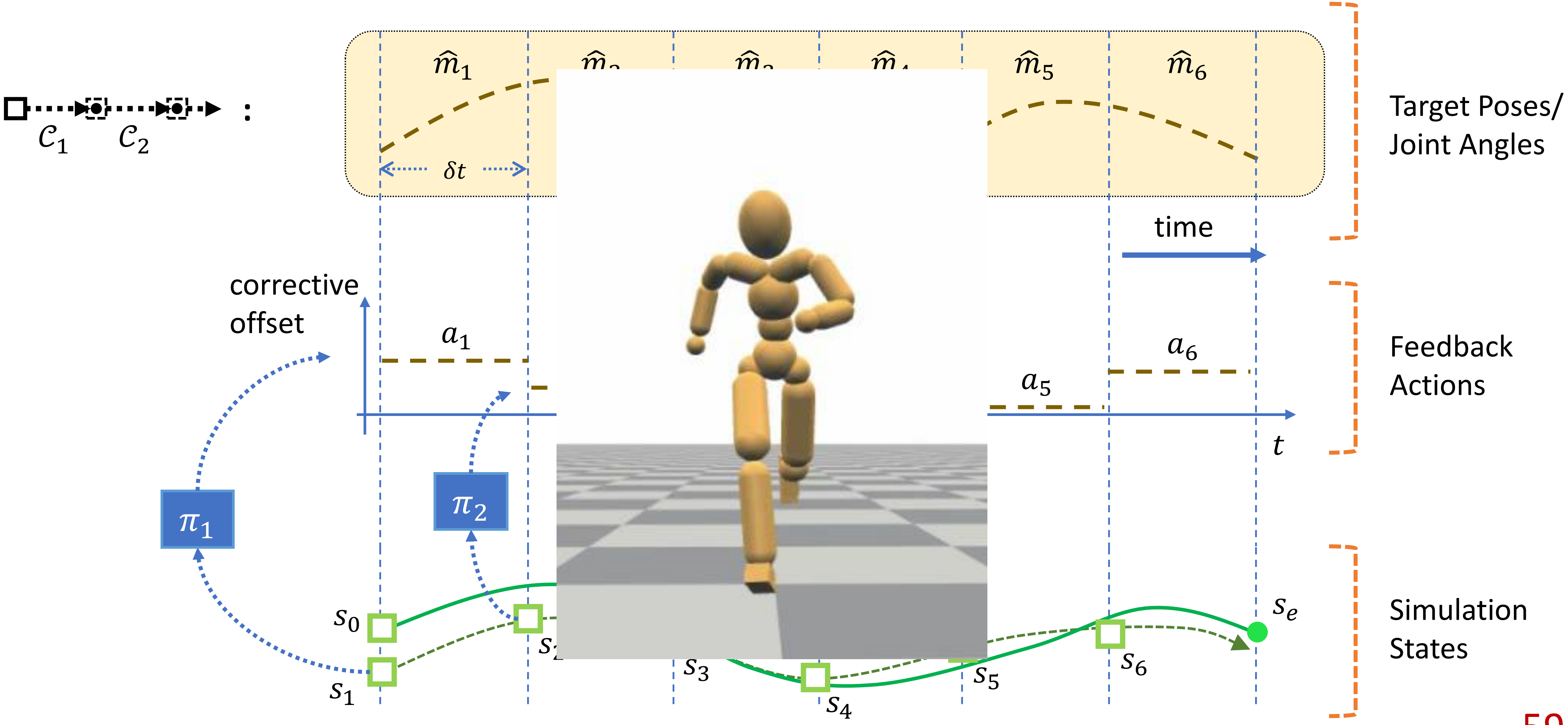
Feedback Control



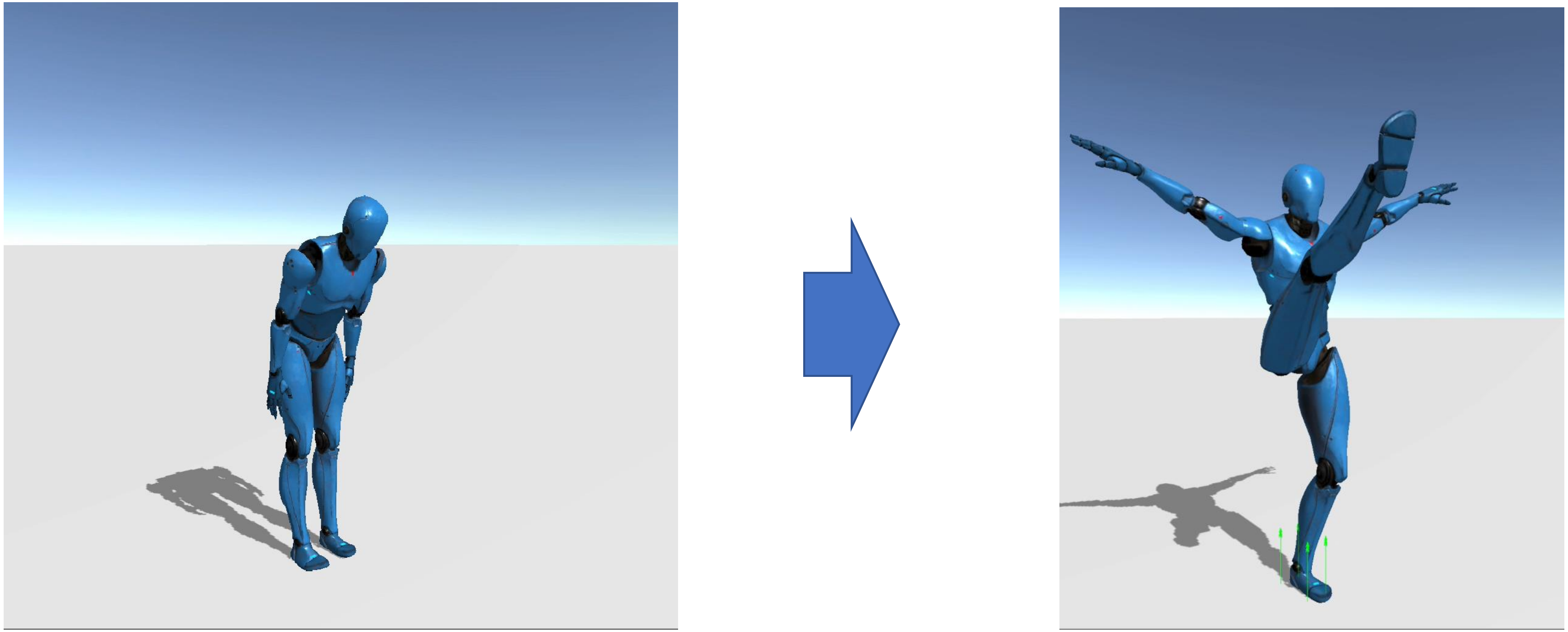
Feedback Control



Feedback Control

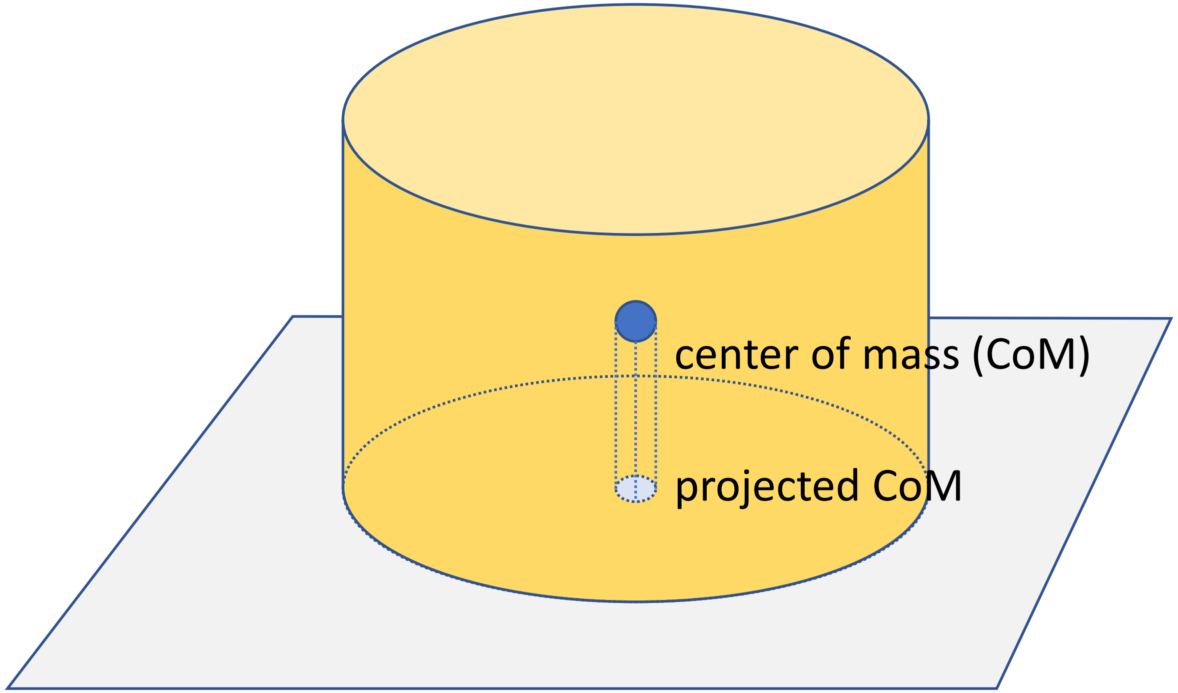


Static Balance



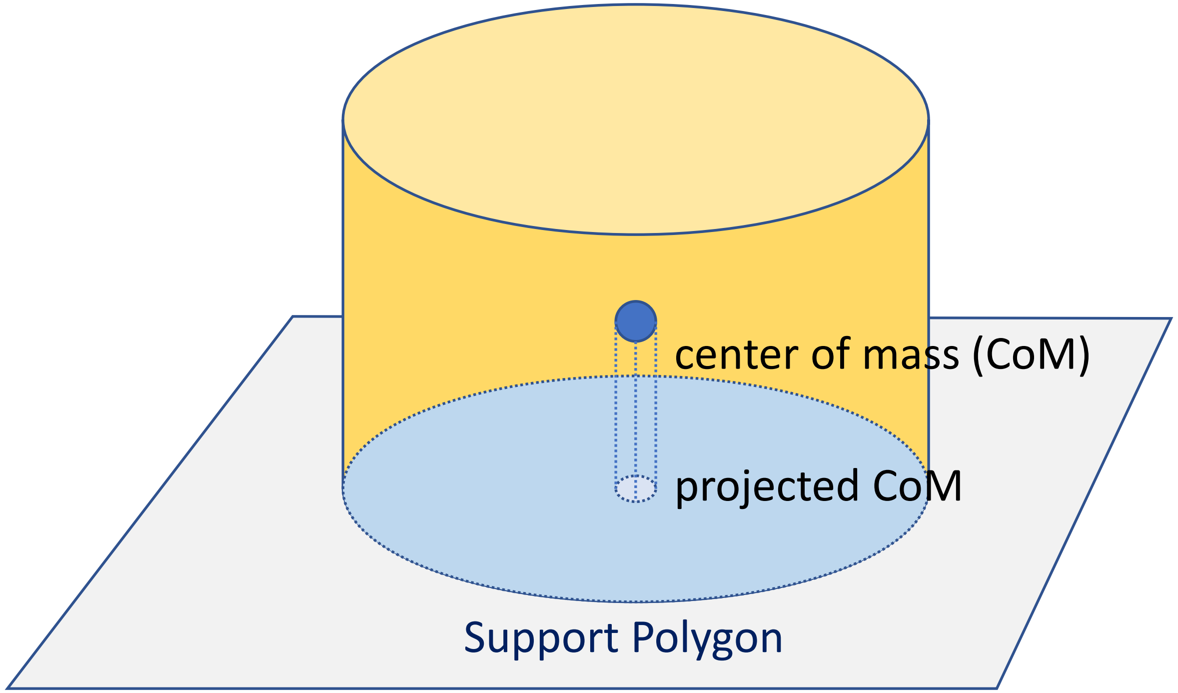
Static Balance

What is balance?



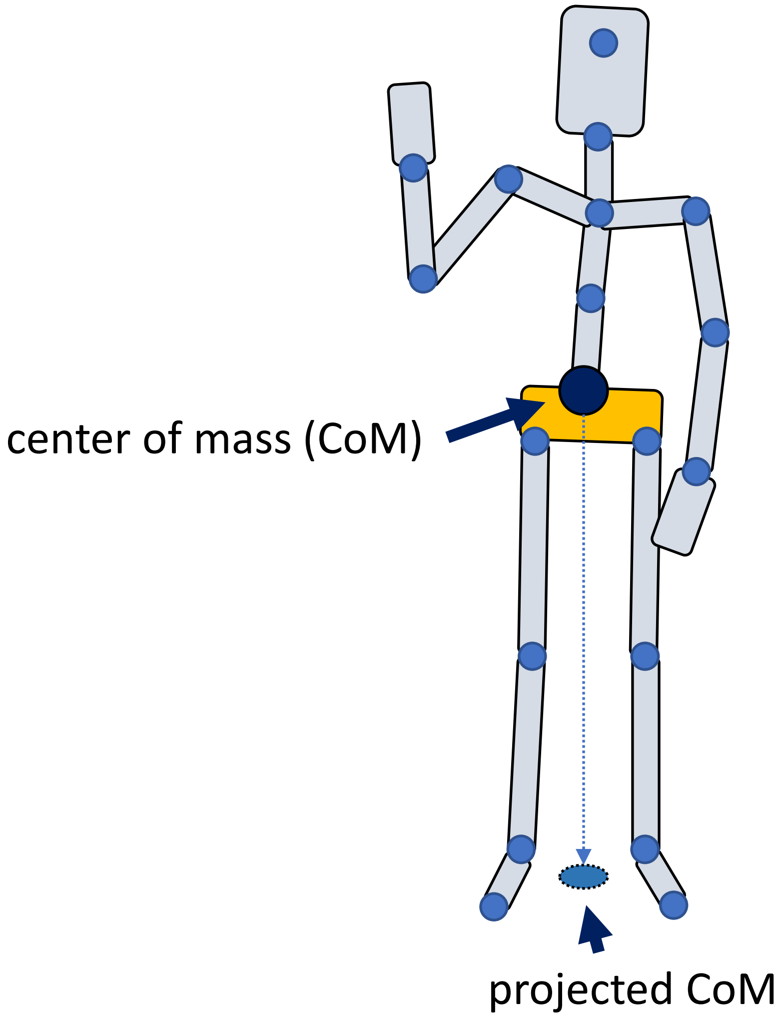
Static Balance

What is balance?



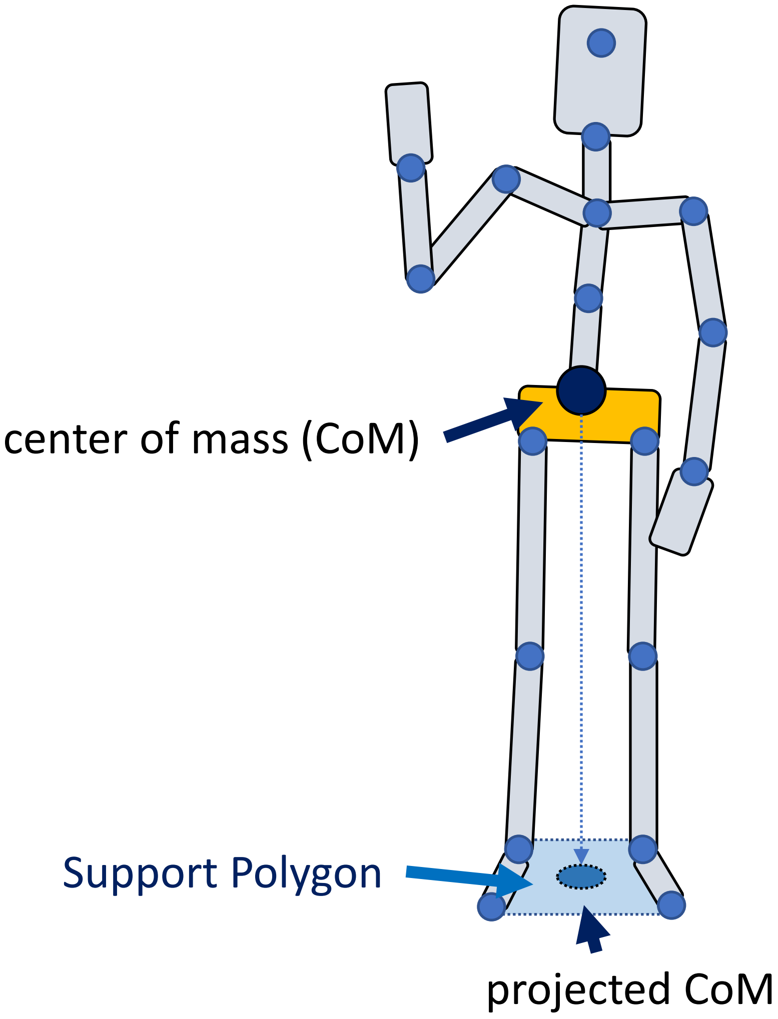
Static Balance

What is balance?



Static Balance

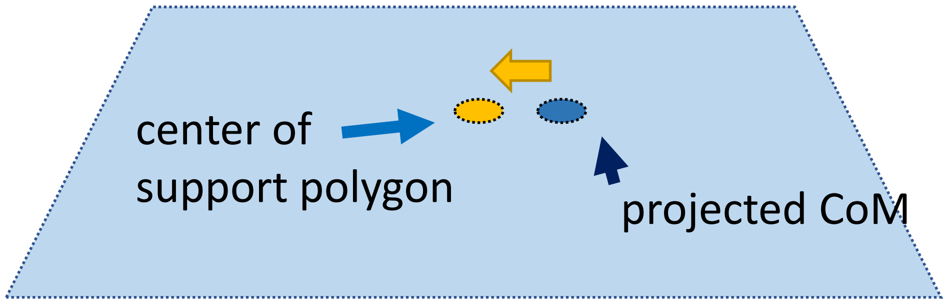
What is balance?

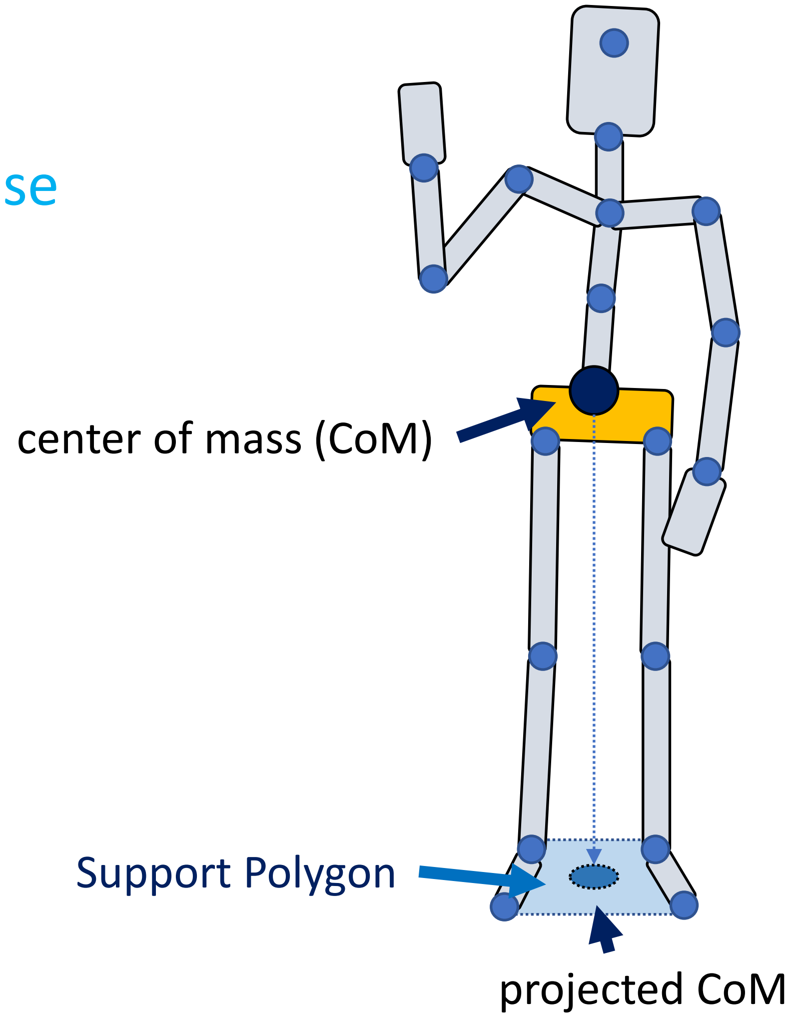


Static Balance

A simple strategy to maintain balance:

- Keep projected CoM close to the center of support polygon while tracking a standing pose





Static Balance

A simple strategy to maintain balance:

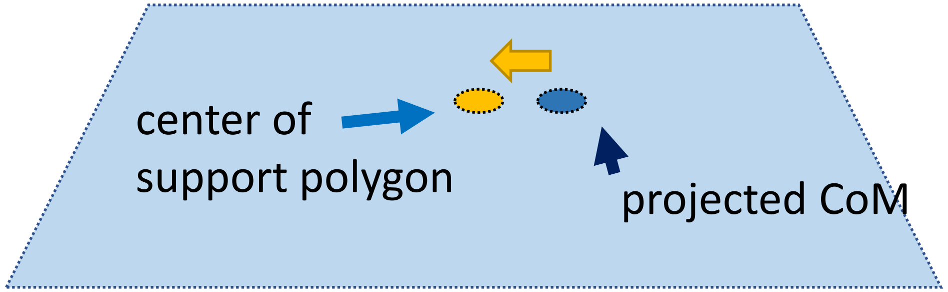
- Keep projected CoM close to the center of support polygon while tracking a standing pose
- Use PD control to compute feedback torque

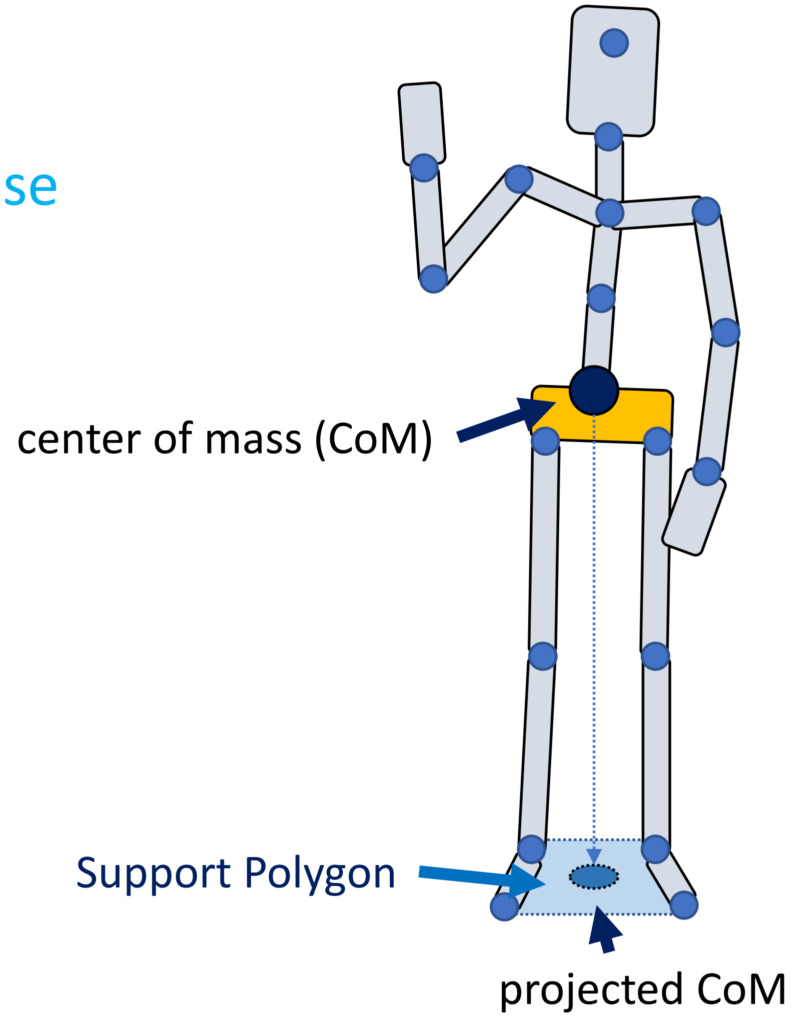
$$\tau = k_p(\bar{\boldsymbol{c}} - \boldsymbol{c}) - k_d\dot{\boldsymbol{c}}$$

CoM velocity

desired CoM position
(e.g. center of support polygon)

CoM position of
the character





Static Balance

A simple strategy to maintain balance:

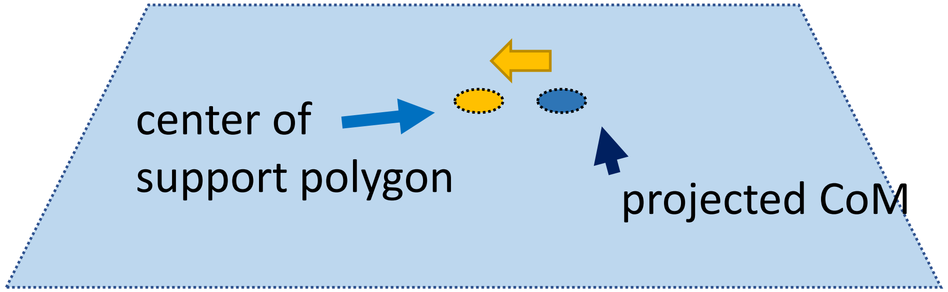
- Keep projected CoM close to the center of support polygon while tracking a standing pose
- Use PD control to compute feedback torque

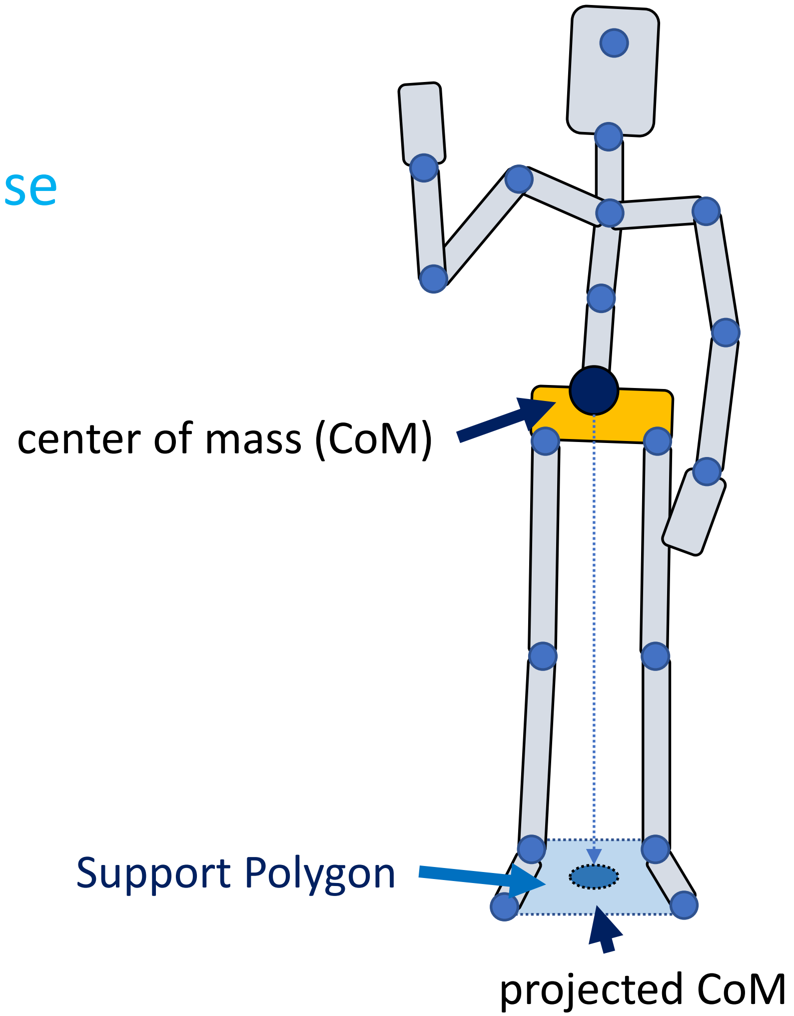
$$\tau = k_p(\bar{\boldsymbol{c}} - \boldsymbol{c}) - k_d\dot{\boldsymbol{c}}$$

gain coefficient

damping coefficient

need to tune manually…





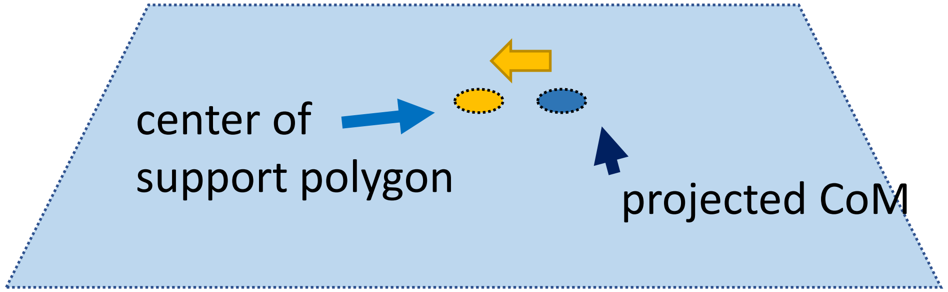
Static Balance

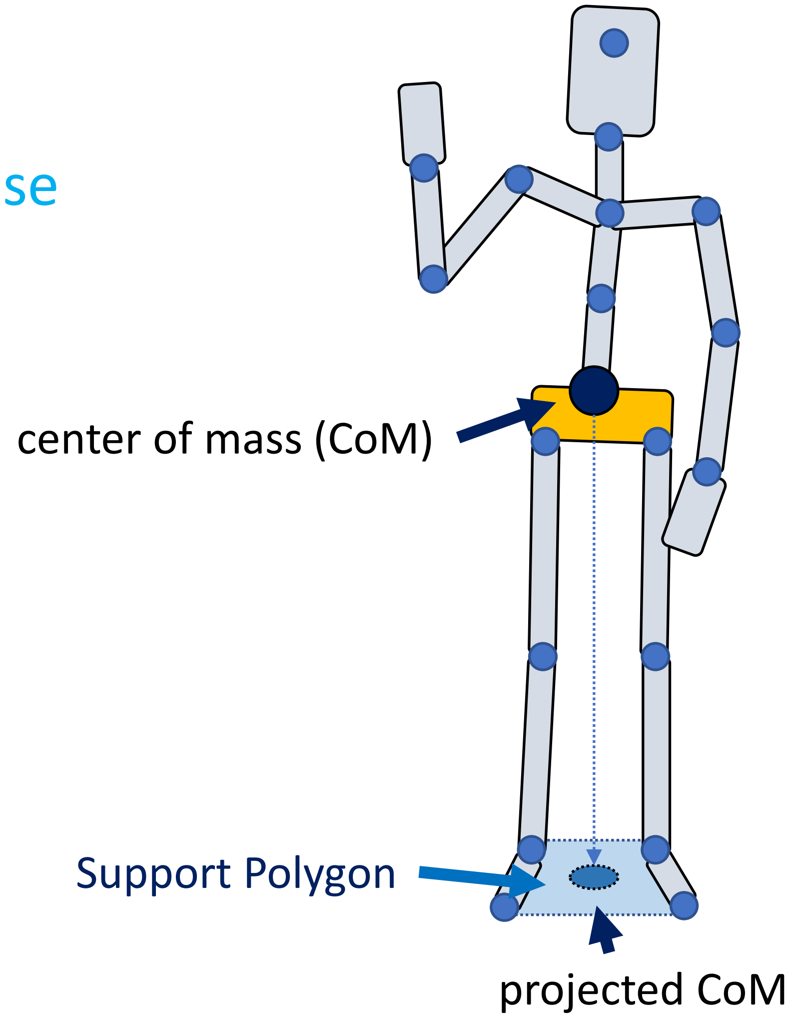
A simple strategy to maintain balance:

- Keep projected CoM close to the center of support polygon while tracking a standing pose
- Use PD control to compute feedback torque

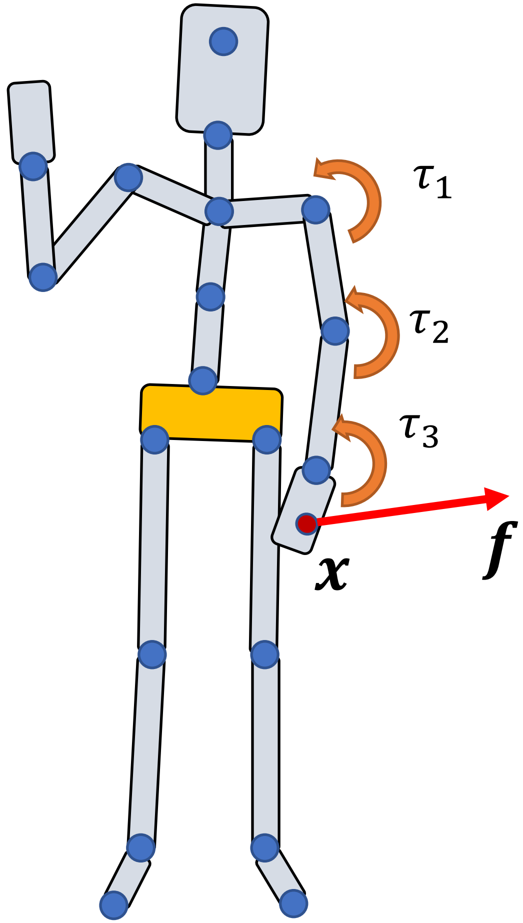
$$\tau = k_p(\bar{\boldsymbol{c}} - \boldsymbol{c}) - k_d\dot{\boldsymbol{c}}$$

- Apply the feedback torque at ankles (ankle strategy) or hips (hip strategy)





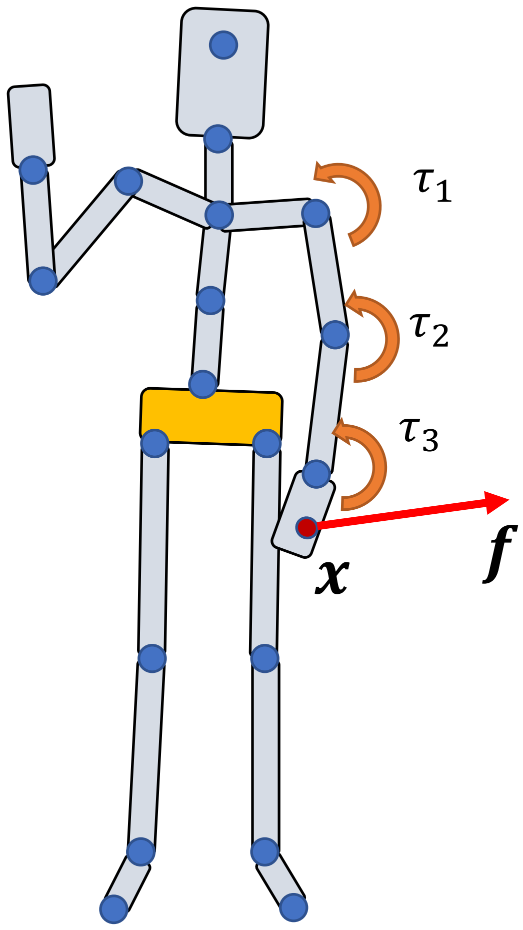
Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the effect of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

- Note that the desired force $\boldsymbol{f}$ is not actually applied
- Also called "virtual force"

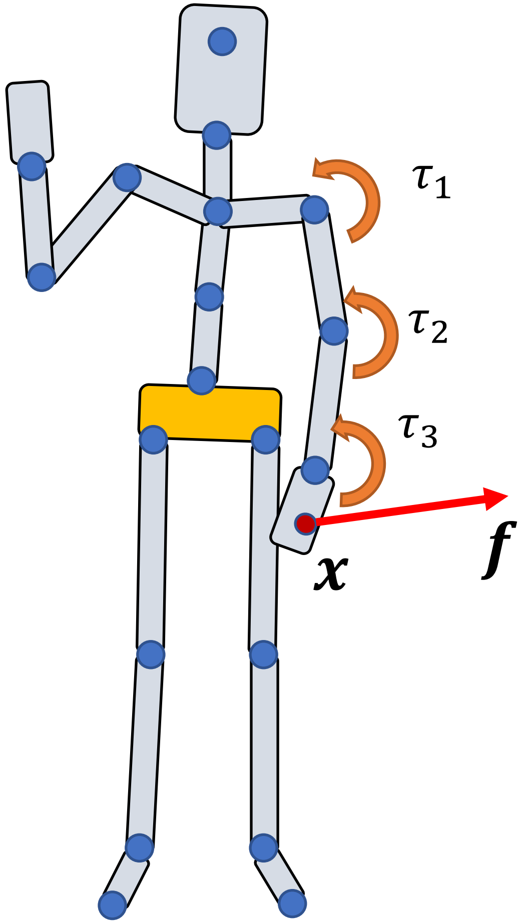
Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same work

Jacobian Transpose Control

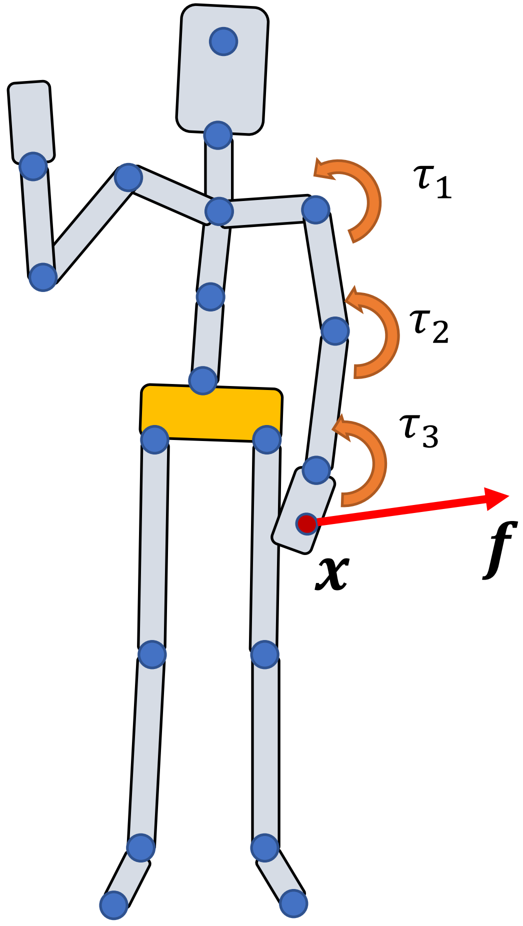


Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same power

$$P = \boldsymbol{f}^T \dot{\boldsymbol{x}} = \boldsymbol{\tau}^T \dot{\boldsymbol{\theta}}$$

Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

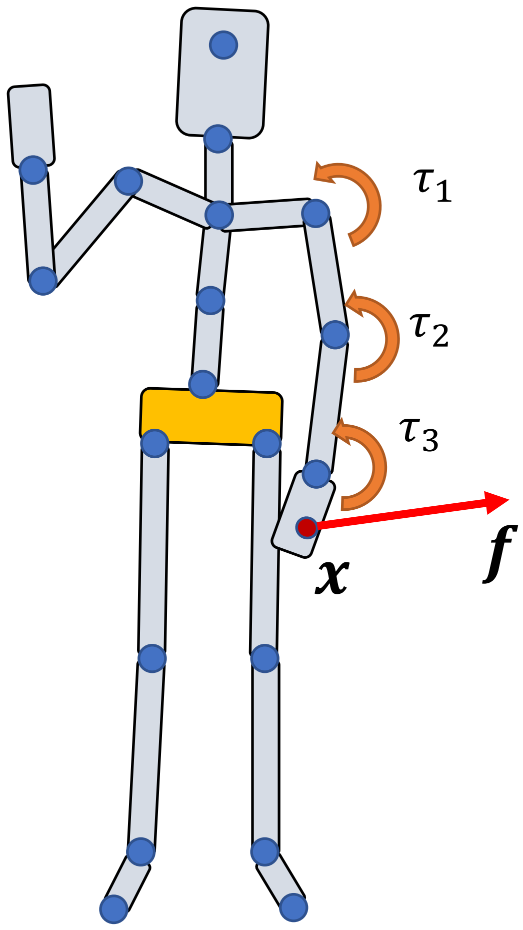
Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same power

$$P = \boldsymbol{f}^T \dot{\boldsymbol{x}} = \boldsymbol{\tau}^T \dot{\boldsymbol{\theta}}$$

Forward kinematics $\boldsymbol{x} = g(\boldsymbol{\theta}) \Rightarrow \dot{\boldsymbol{x}} = J\dot{\boldsymbol{\theta}}$

$$J = \frac{\partial g}{\partial \boldsymbol{\theta}}$$

Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same power

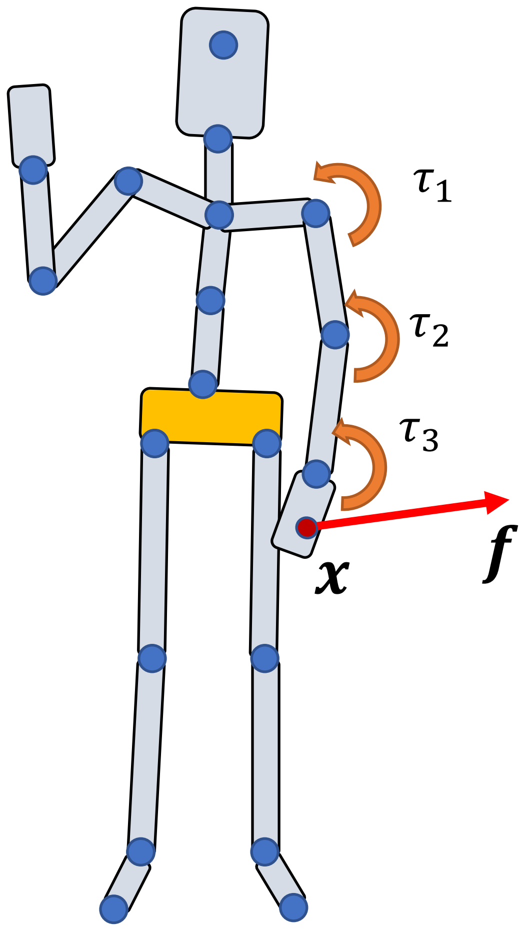
$$P = \boldsymbol{f}^T \dot{\boldsymbol{x}} = \boldsymbol{\tau}^T \dot{\boldsymbol{\theta}}$$

Forward kinematics $\boldsymbol{x} = g(\boldsymbol{\theta}) \Rightarrow \dot{\boldsymbol{x}} = J\dot{\boldsymbol{\theta}}$

$$\boldsymbol{f}^T J \dot{\boldsymbol{\theta}} = \boldsymbol{\tau}^T \dot{\boldsymbol{\theta}}$$

$$J = \frac{\partial g}{\partial \boldsymbol{\theta}}$$

Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same power

$$P = \boldsymbol{f}^T \dot{\boldsymbol{x}} = \boldsymbol{\tau}^T \dot{\boldsymbol{\theta}}$$

Forward kinematics $\boldsymbol{x} = g(\boldsymbol{\theta}) \Rightarrow \dot{\boldsymbol{x}} = J\dot{\boldsymbol{\theta}}$

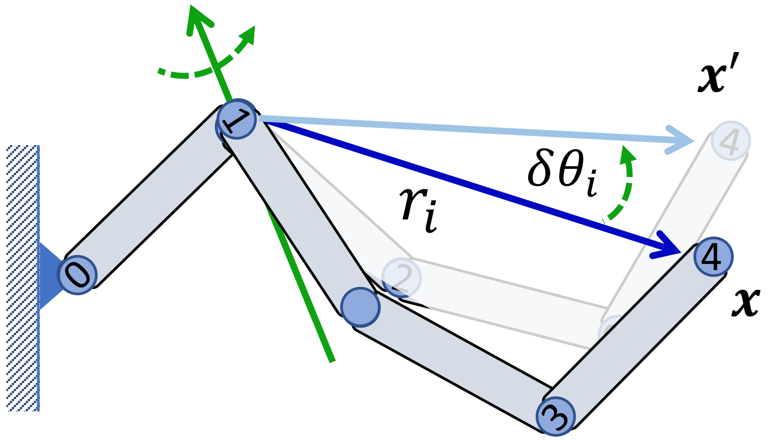
$$\boldsymbol{\tau} = J^T \boldsymbol{f}$$

$$J = \frac{\partial g}{\partial \boldsymbol{\theta}}$$

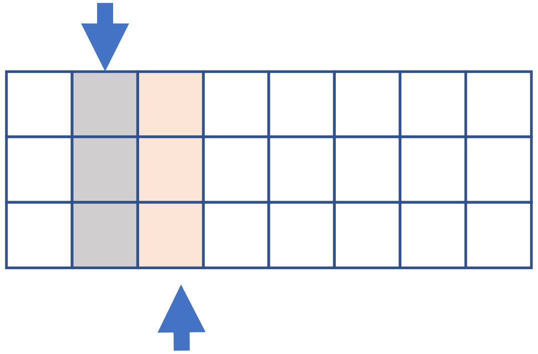
Recall: Geometric Approach

$$J = \frac{\partial g}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial g}{\partial \theta_0} & \frac{\partial g}{\partial \theta_1} & \cdots & \frac{\partial g}{\partial \theta_n} \end{pmatrix}$$

Assuming all joints are hinge joint

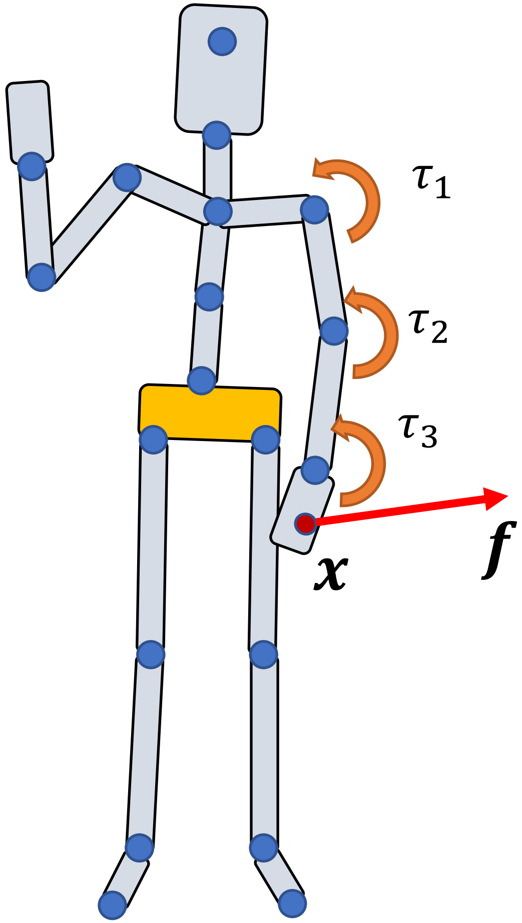


$$\frac{\partial g}{\partial \theta_1} = \boldsymbol{a}_1 \times \boldsymbol{r}_1$$



$$\frac{\partial f}{\partial \theta_2} = \boldsymbol{a}_2 \times \boldsymbol{r}_2$$

Jacobian Transpose Control



Can we use joint torques $\boldsymbol{\tau}_i$ to mimic the **effect** of a force $\boldsymbol{f}$ applied at $\boldsymbol{x}$

Make $\boldsymbol{f}$ and $\boldsymbol{\tau}_i$ done the same power

$$\boldsymbol{\tau} = J^T \boldsymbol{f} \qquad J = \frac{\partial g}{\partial \boldsymbol{\theta}}$$

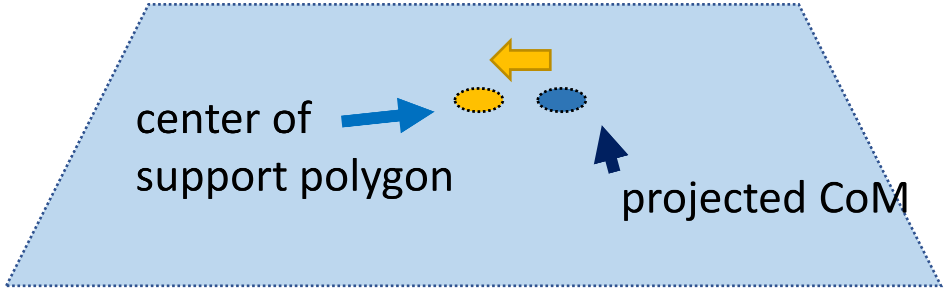
$$\boldsymbol{\tau}_i = (\boldsymbol{x} - \boldsymbol{p}_i) \times \boldsymbol{f}$$

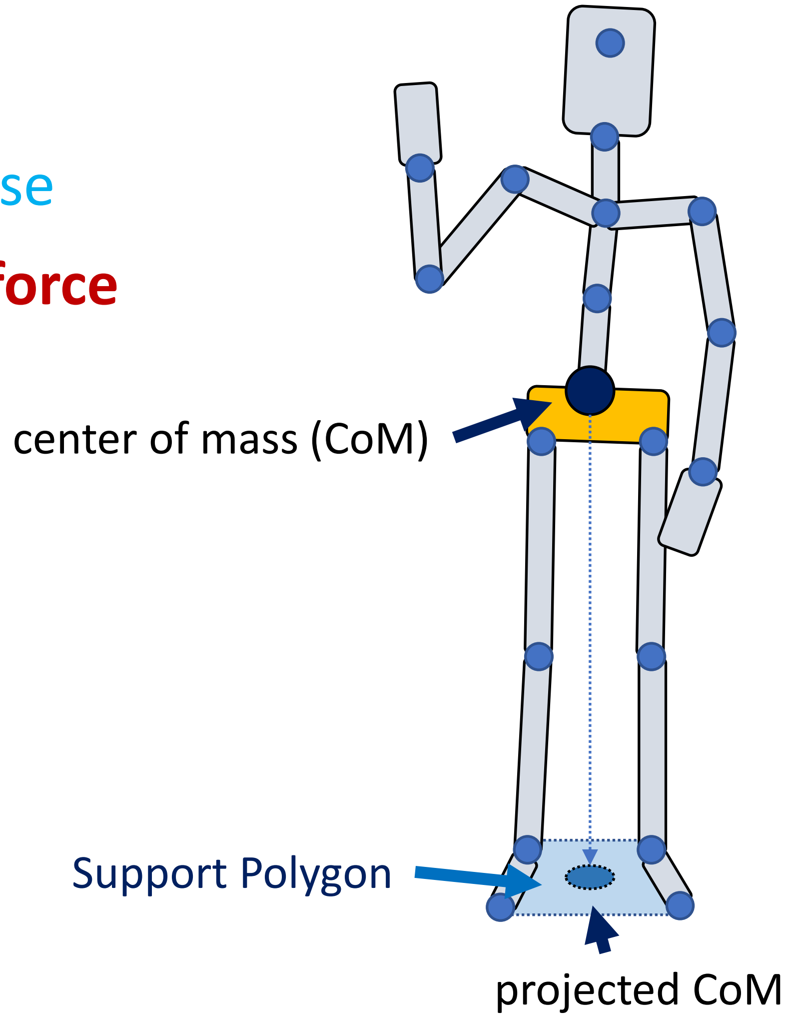
Static Balance

A simple strategy to maintain balance:

- Keep projected CoM close to the center of support polygon while tracking a standing pose
- Use PD control to compute feedback **virtual force**

$$f = k_p(\bar{\boldsymbol{c}} - \boldsymbol{c}) - k_d\dot{\boldsymbol{c}}$$





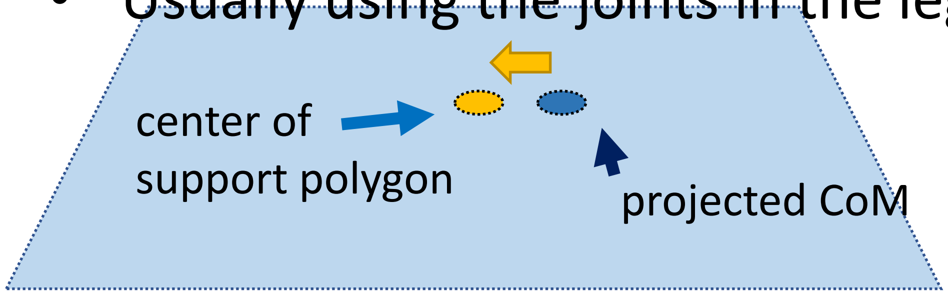
Static Balance

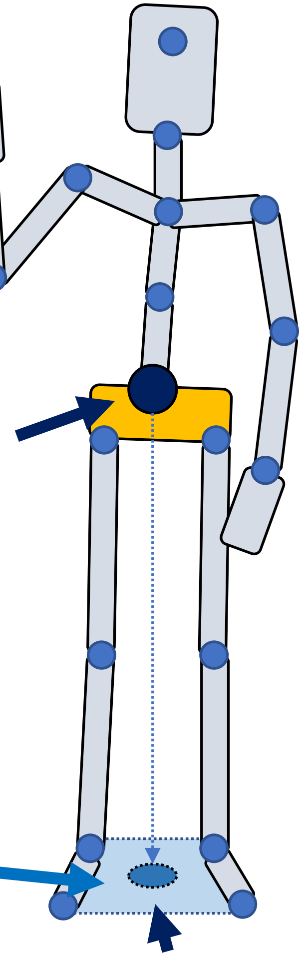
A simple strategy to maintain balance:

- Keep projected CoM close to the center of support polygon while tracking a standing pose
- Use PD control to compute feedback **virtual force**

$$f = k_p(\bar{\boldsymbol{c}} - \boldsymbol{c}) - k_d\dot{\boldsymbol{c}}$$

- Assuming f is applied to the CoM, compute necessary joint torques using Jacobian transpose control to achieve it
  - Usually using the joints in the legs

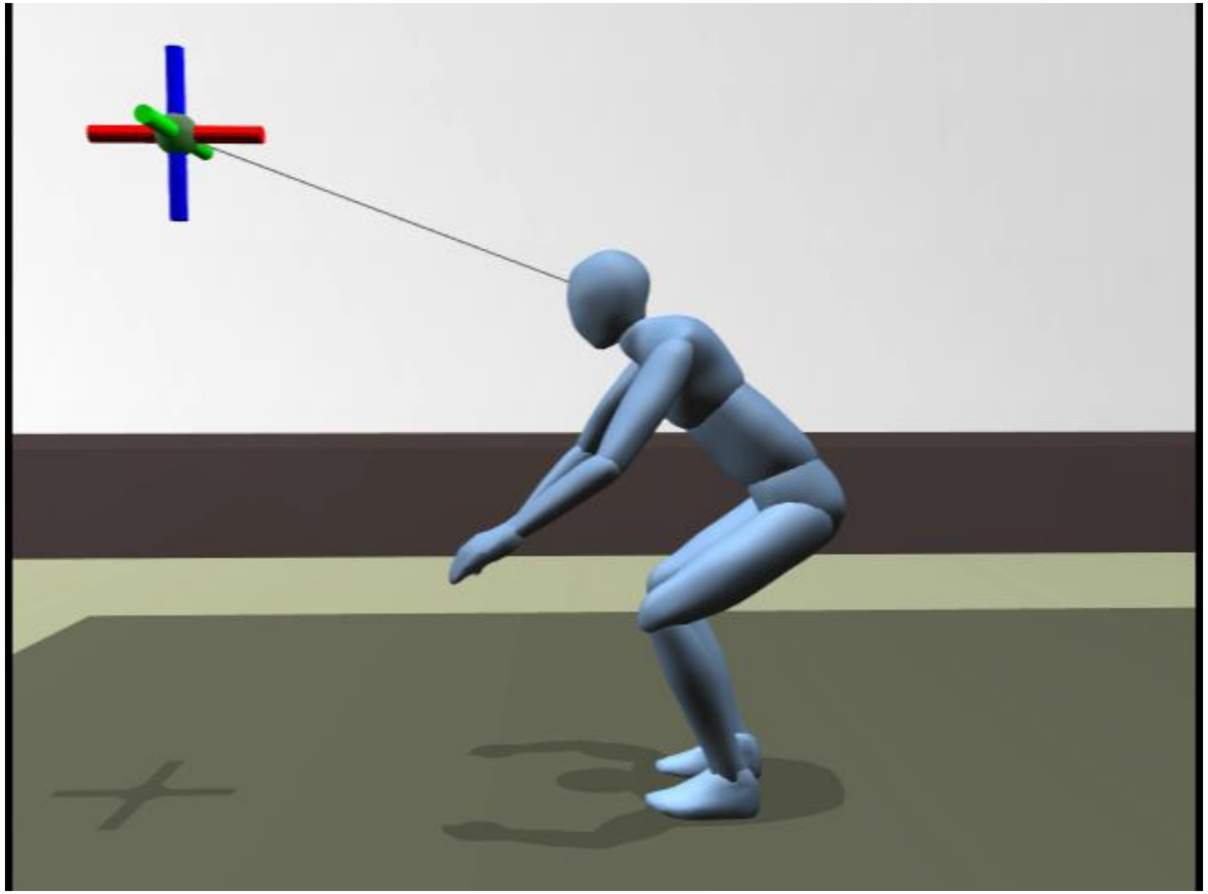




Static Balance

A fancier strategy:

- Mocap tracking as an objective function
- Controlling both the CoM position/momentum and the angular momentum
- Solve a one-step optimization problem to compute joint torques



[Macchietto et al. 2009 - Momentum Control for Balance]

Outline

- More about PD (Proportional-Derivative) control
  - Stable PD control
- Feedforward Motion Control
  - Trajectory optimization
- Feedback Motion Control
  - Static balance
  - Dynamic balance?

Questions?