GAMES 105
Fundamentals of Character Animation

Lecture 09

Actuating Simulated Characters

Libin Liu

School of Intelligence Science and Technology
Peking University

GAMES105 课程交流

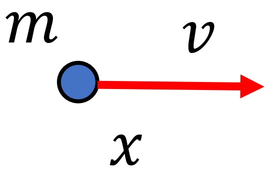
VCL @ PKU

Outline

- Simulating & Actuating Characters
  - Joint torques

- PD (Proportional-Derivative) control

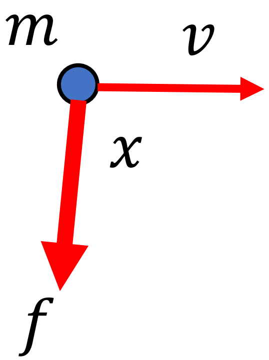
Recap: Dynamics of a Point Mass



x, v

$p = mv$

Recap: Dynamics of a Point Mass

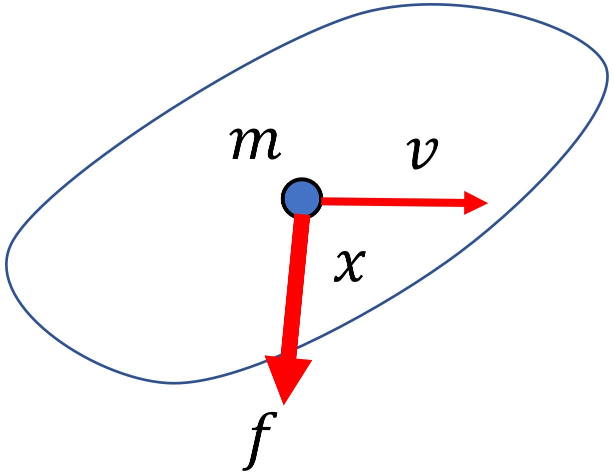


$$x, v$$

$$p = mv$$

Newton's Second Law: $\frac{dp}{dt} = f$ ⇨ $m\dot{v} = f$

Recap: Rigid Body Dynamics



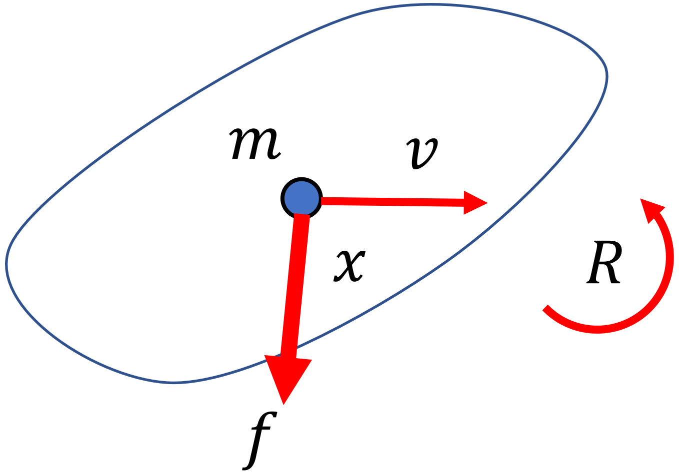
Linear

x, v

$p = mv$

Newton's Second Law: $\frac{dp}{dt} = f$ ➡ $m\dot{v} = f$

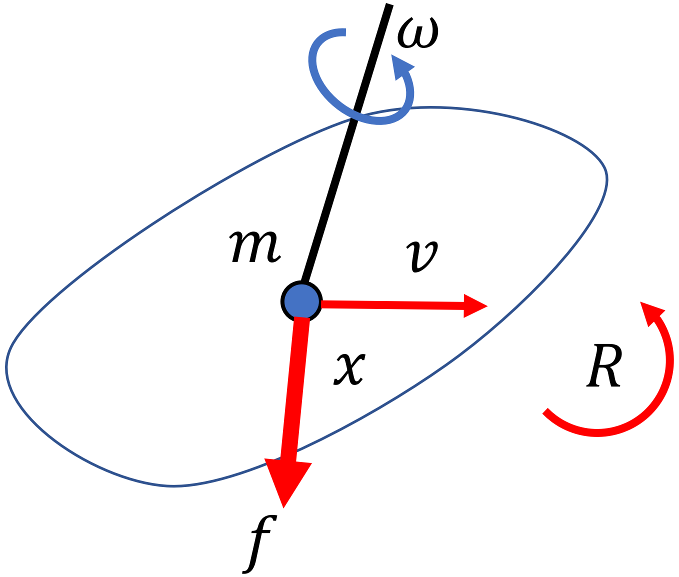
Recap: Rigid Body Dynamics



Linear	Angular
x, v	R
$p = mv$	

Newton's Second Law: $\frac{dp}{dt} = f$ ➡ $m\dot{v} = f$

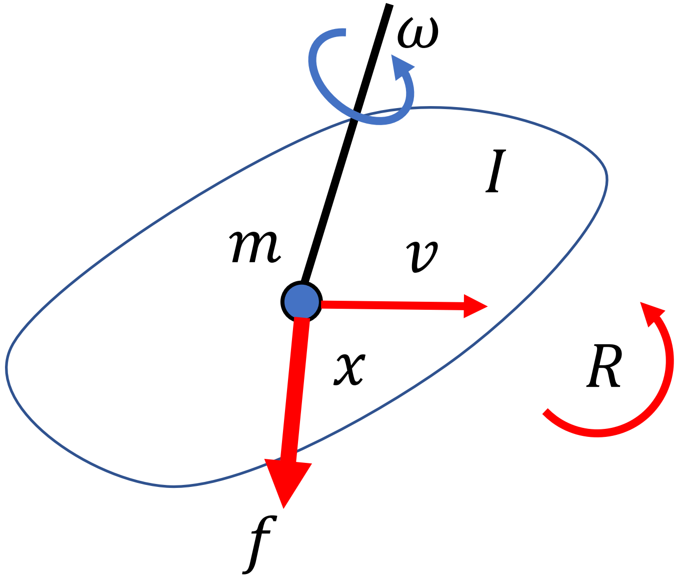
Recap: Rigid Body Dynamics



Linear	Angular
x, v	R, ω
$p = mv$	

Newton's Second Law: $\frac{dp}{dt} = f$ ➡ $m\dot{v} = f$

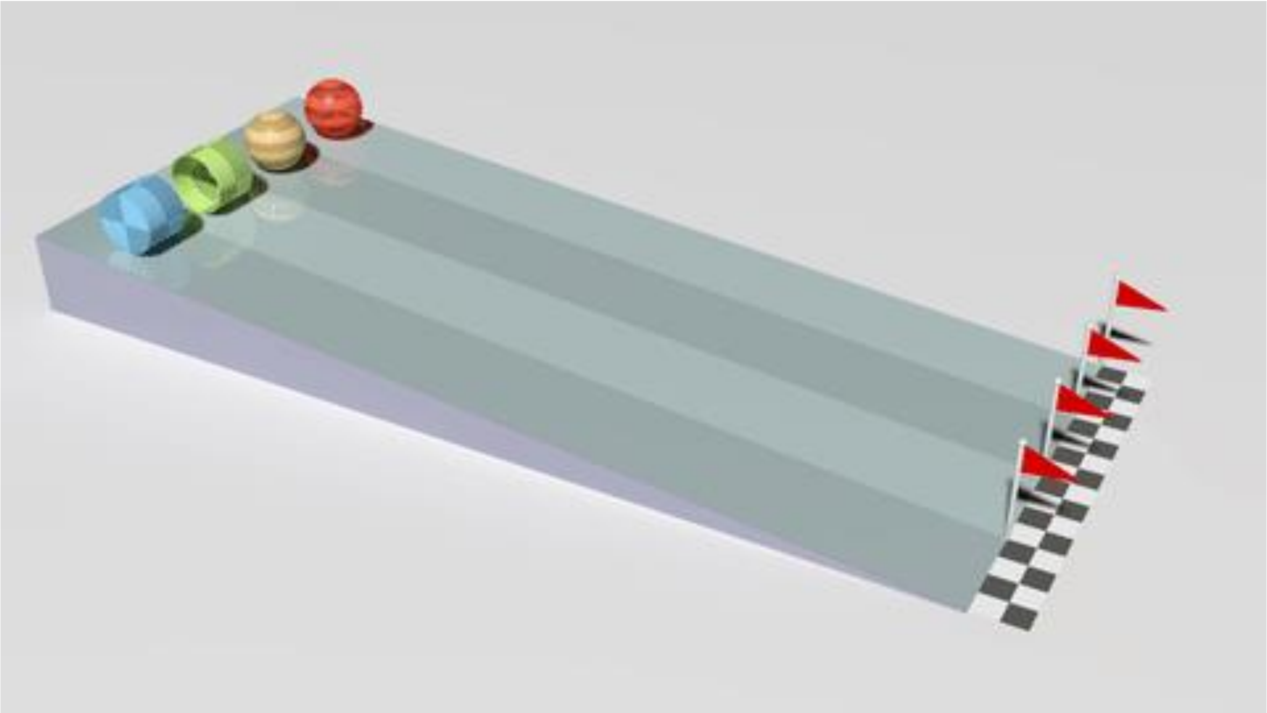
Recap: Rigid Body Dynamics



Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law: $\frac{dp}{dt} = f$ ➡ $m\dot{v} = f$

Recap: Moment of Inertia

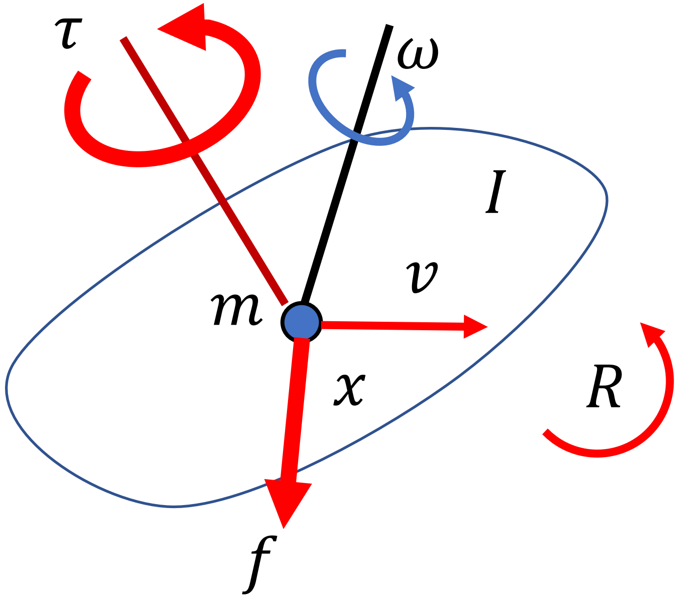


Same mass, different shapes

⬇

Different Moments of Inertia

Recap: Rigid Body Dynamics

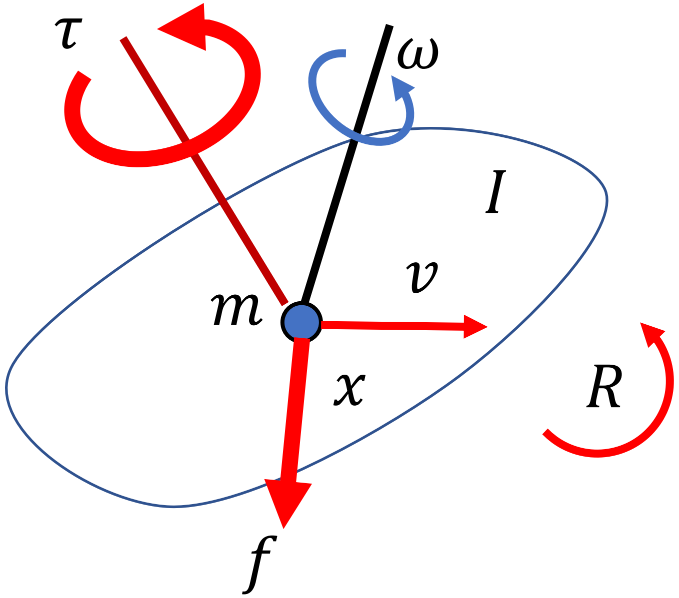


Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law: $\frac{dp}{dt} = f$ ➡ $m\dot{v} = f$

Euler's laws of motion: $\frac{dL}{dt} = \tau$ ➡ $I\dot{\omega} + \omega \times I\omega = \tau$

Recap: Rigid Body Dynamics



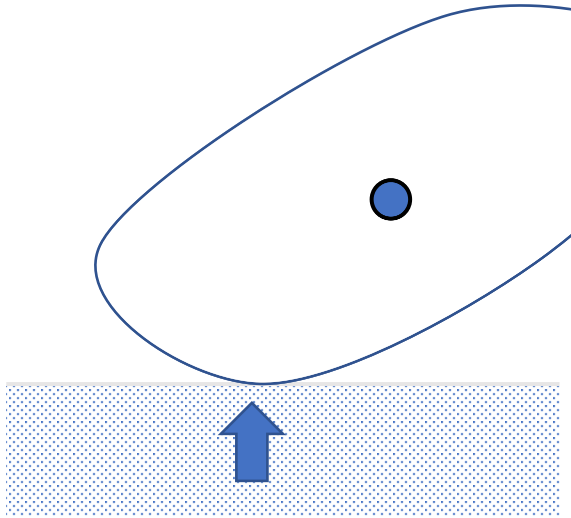
Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law:

Euler's laws of motion:

$$\begin{bmatrix} m\mathrm{I}_3 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} f \\ \tau \end{bmatrix}$$

Defining a Rigid Body



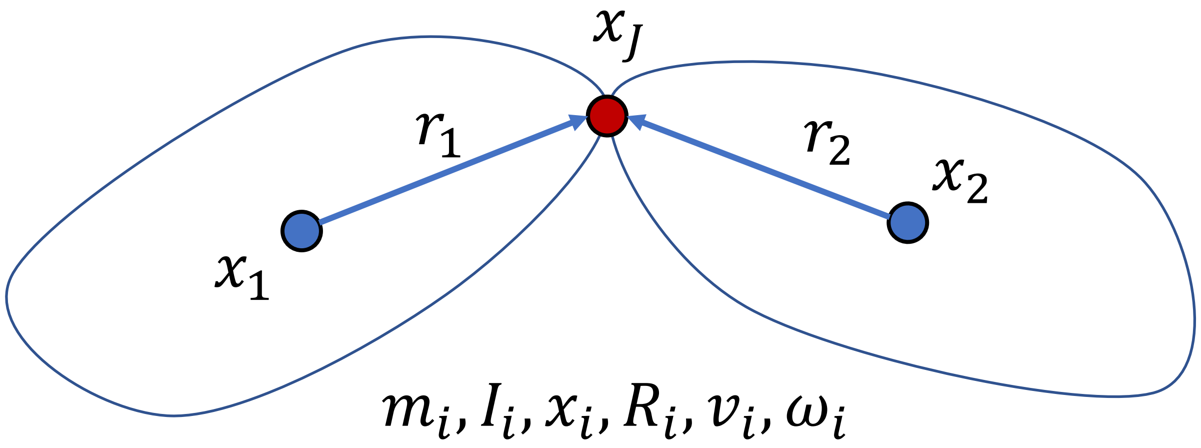
Masses: m, I

Kinematics: $\boldsymbol{x}, \boldsymbol{v}, R, \boldsymbol{\omega}$

Geometry:

- Box, Sphere, Capsule, Mesh, …
- Collision detection
- Compute m, I

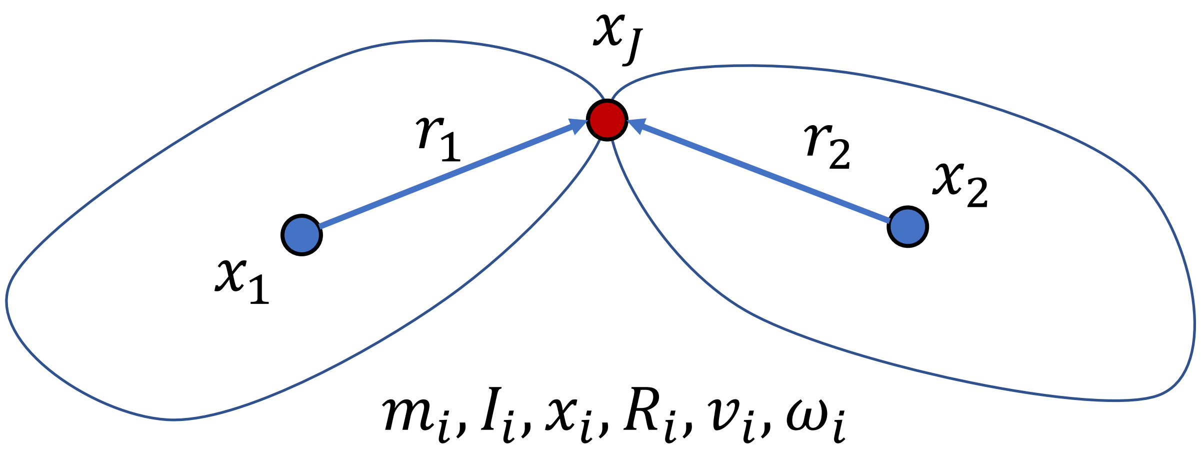
Recap: Dynamics of Articulated Rigid Bodies



$$\boxed{Jv = 0}$$

$$\begin{bmatrix} I_3 & -[r_1]_\times & -I_3 & [r_2]_\times \end{bmatrix} \begin{bmatrix} v_1 \\ w_1 \\ v_2 \\ w_2 \end{bmatrix} = 0$$

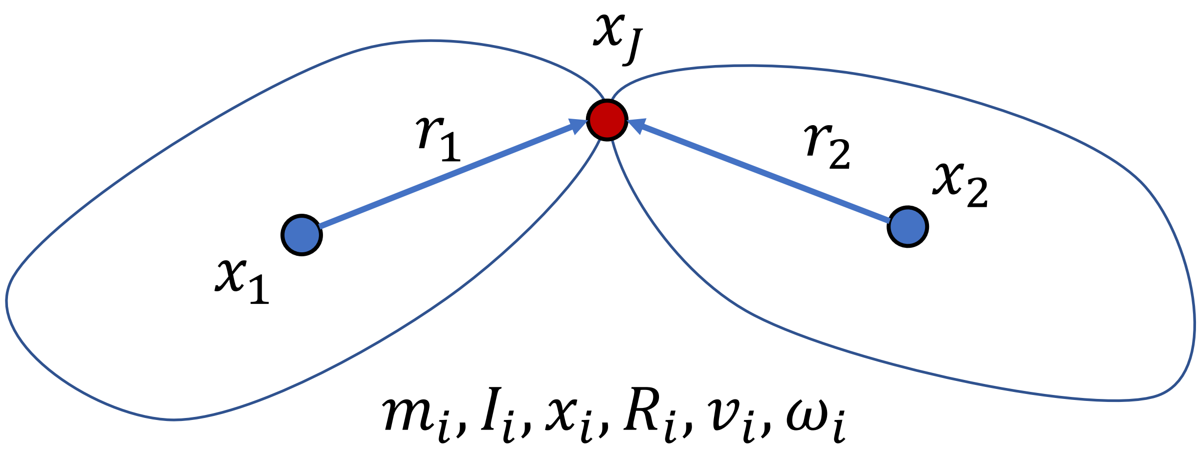
Recap: Dynamics of Articulated Rigid Bodies



$$\begin{bmatrix} m_1\mathrm{I}_3 & & & \\ & I_1 & & \\ & & m_2\mathrm{I}_3 & \\ & & & I_2 \end{bmatrix}\begin{bmatrix} \dot{v}_1 \\ \dot{\omega}_1 \\ \dot{v}_2 \\ \dot{\omega}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_1 \times I_1\omega_1 \\ 0 \\ \omega_2 \times I_2\omega_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ \tau_1 \\ f_2 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} I_3 \\ [r_1]_\times \\ -I_3 \\ -[r_2]_\times \end{bmatrix}\lambda$$

$$Jv = 0$$

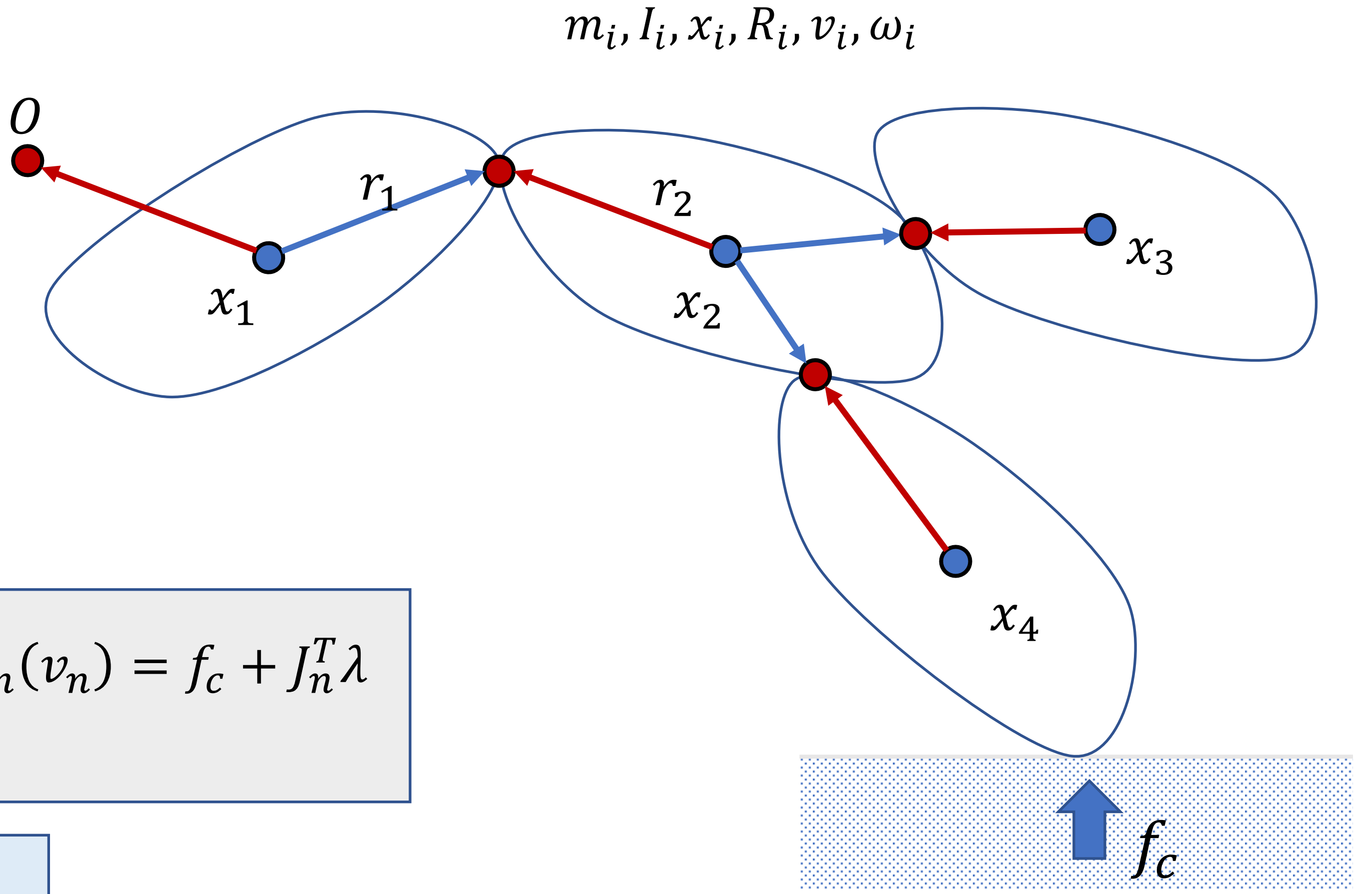
Recap: Dynamics of Articulated Rigid Bodies



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

$$Jv = 0$$

Recap: Simulation of a Rigid Body System



$$I_n = R_n I_0 R_n^T$$

$$M_n(v_{n+1} - v_n)/_h + C_n(v_n) = f_c + J_n^T \lambda$$

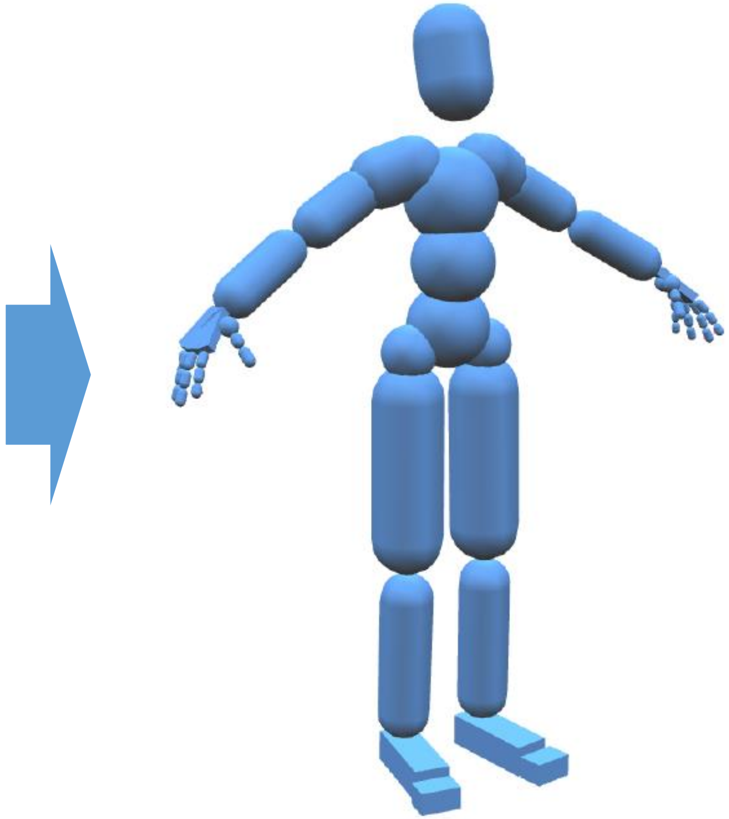
$$J_n v_{n+1} = c_n$$

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

$$q_{n+1} = q_n + \frac{h}{2} \bar{\omega}_{n+1} q$$

Defining a Simulated Character

[Liu et al 2018]



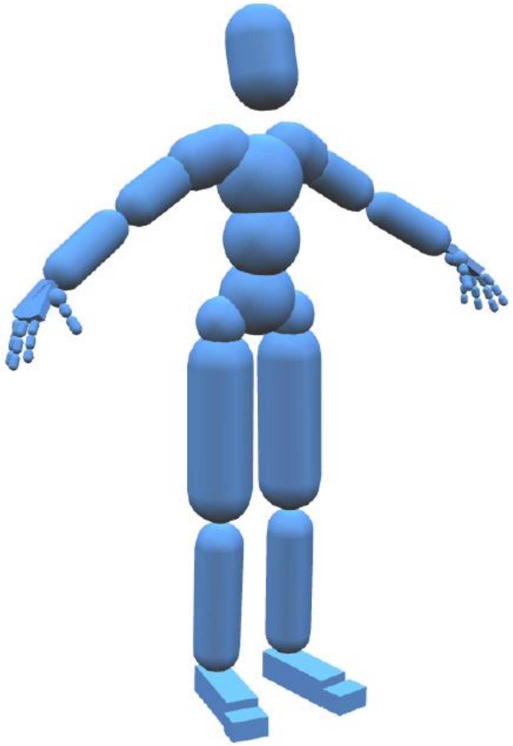
Rigid bodies:

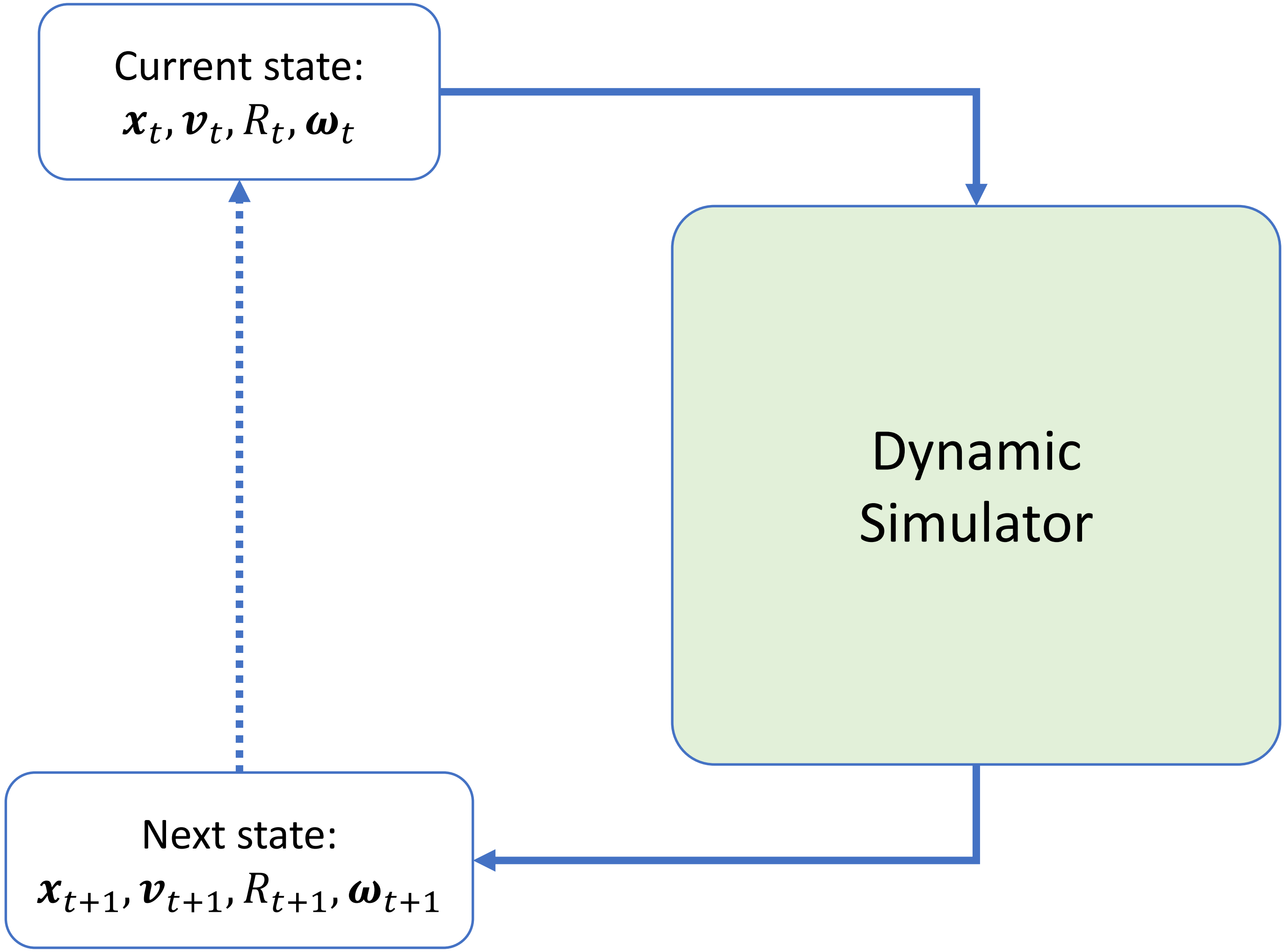
- $m_i, I_i, \boldsymbol{x}_i, R_i$
- Geometries

Joints:

- Position
- Type
- Bodies

Simulating a Character



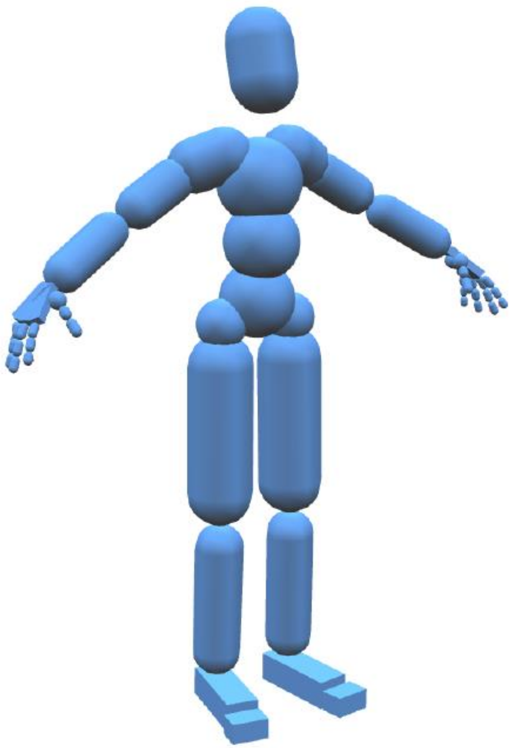


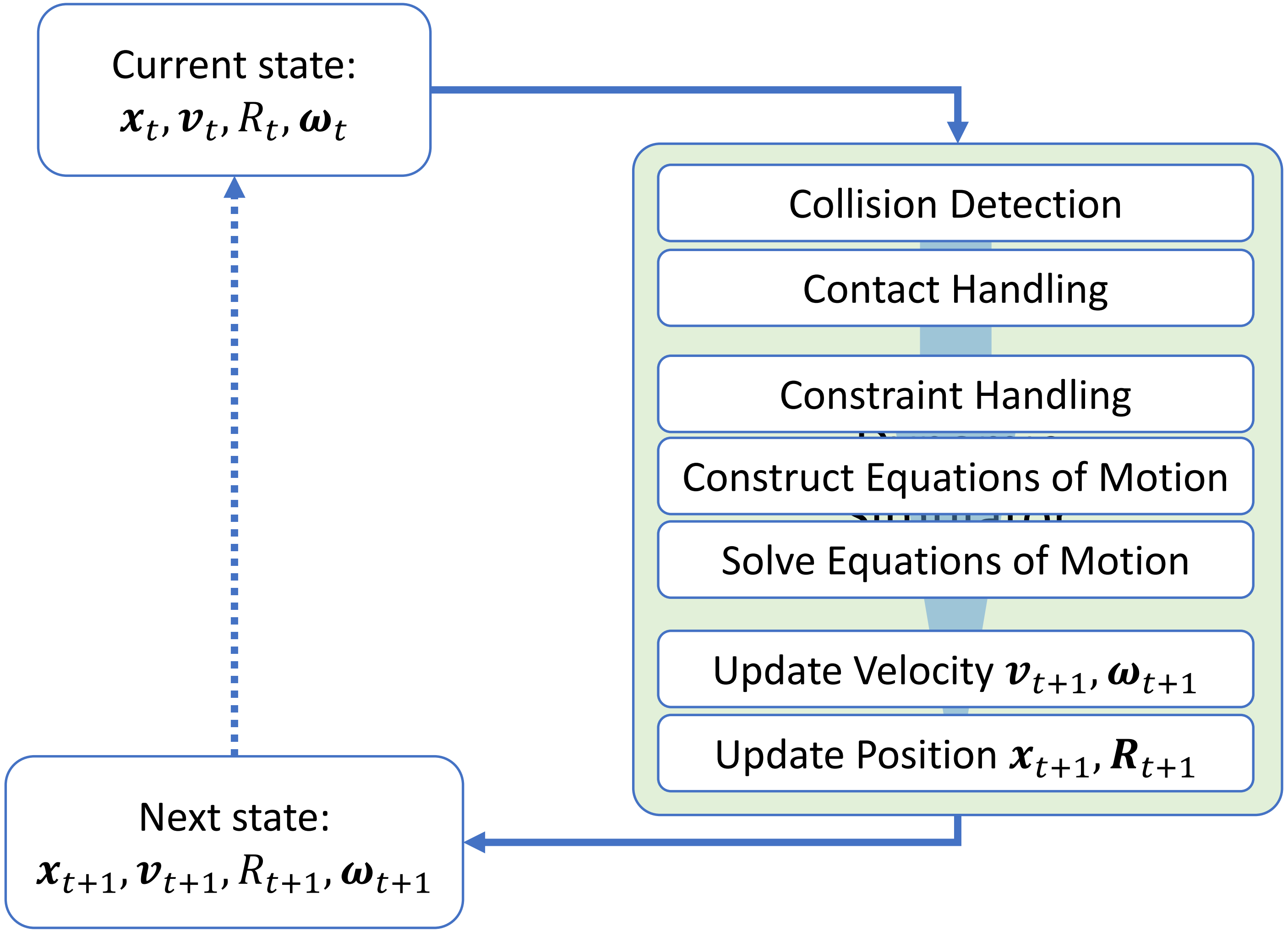
Current state:
$\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t$

Dynamic Simulator

Next state:
$\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1}$

Simulating a Character





Current state:
$\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t$

Collision Detection

Contact Handling

Constraint Handling

Construct Equations of Motion

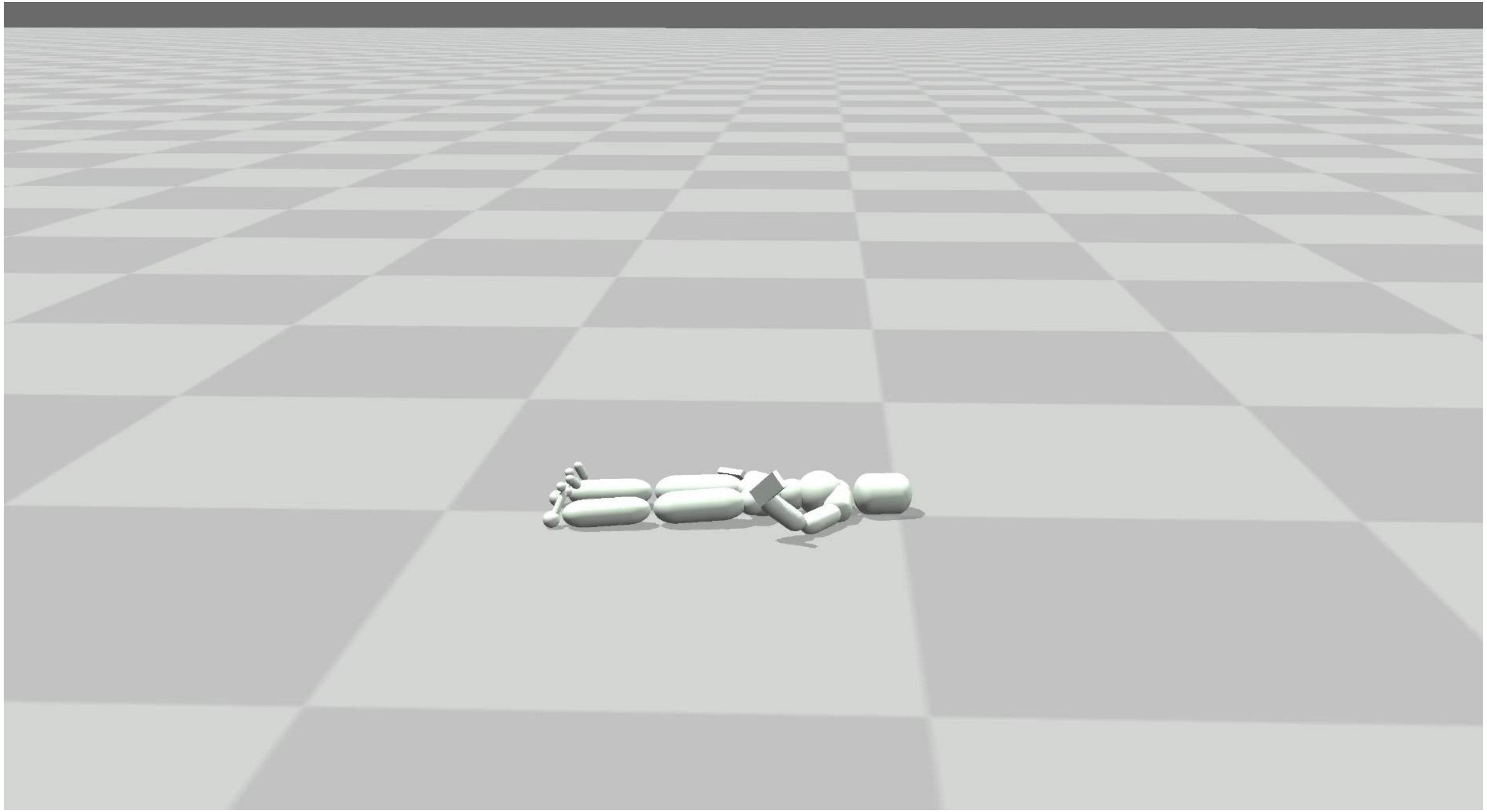
Solve Equations of Motion

Update Velocity $\boldsymbol{v}_{t+1}, \boldsymbol{\omega}_{t+1}$

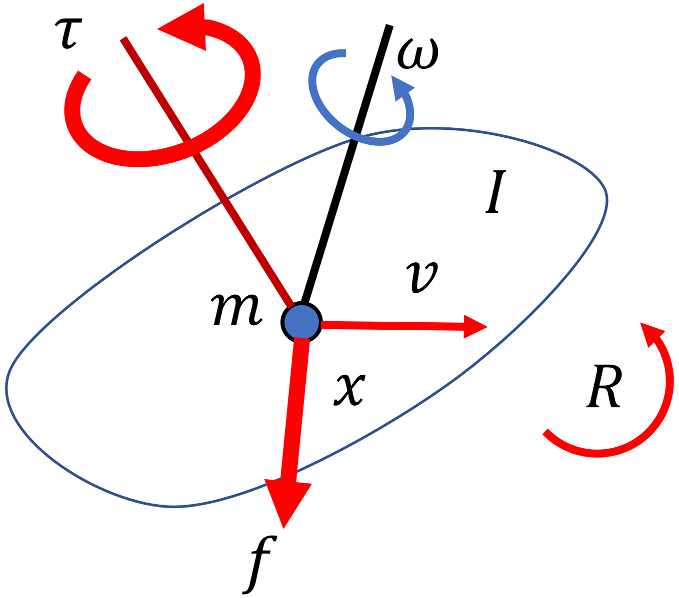
Update Position $\boldsymbol{x}_{t+1}, \boldsymbol{R}_{t+1}$

Next state:
$\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1}$

Ragdoll Simulation



Actuating a Rigid Body



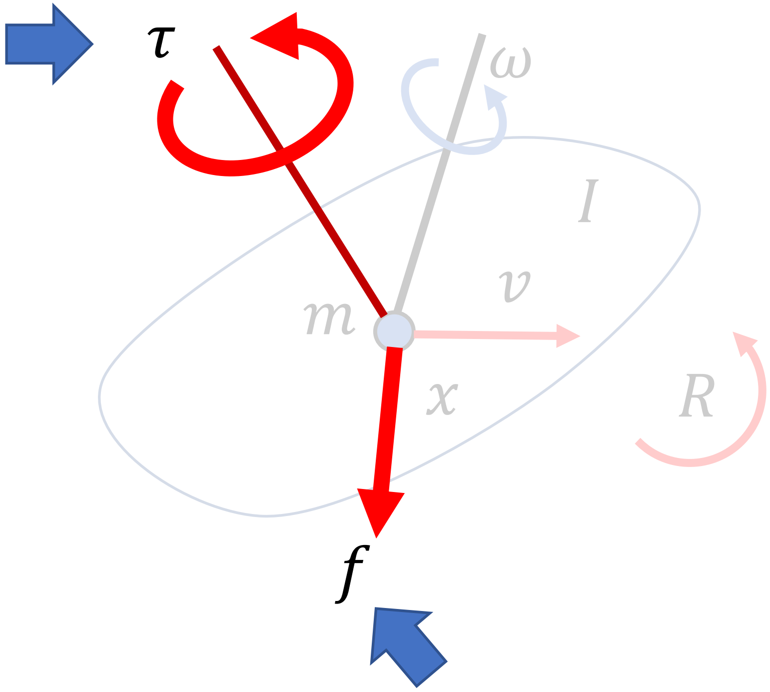
Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law:

Euler's laws of motion:

$$\begin{bmatrix} m\mathrm{I}_3 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix}$$

Actuating a Rigid Body



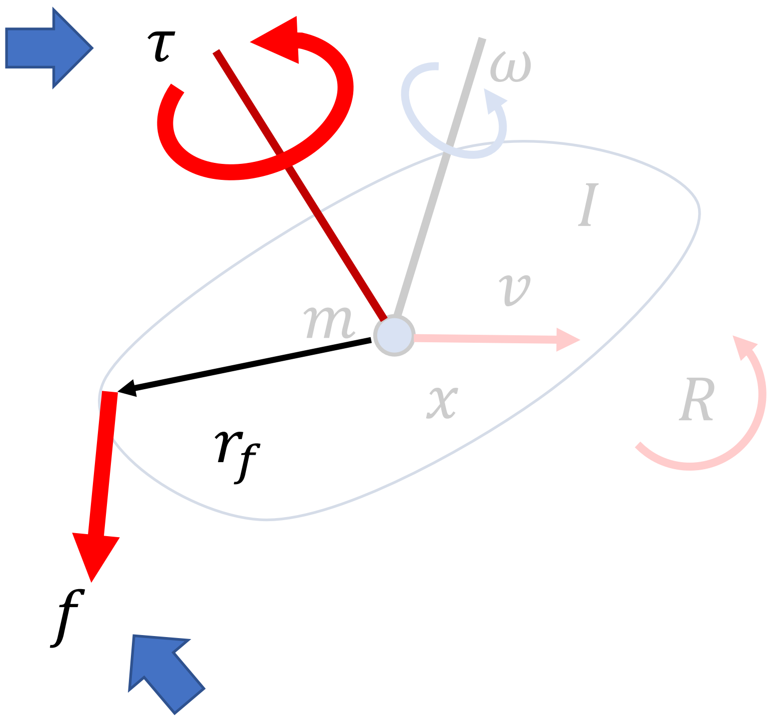
Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law:

Euler's laws of motion:

$$\begin{bmatrix} m\mathrm{I}_3 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix}$$

Actuating a Rigid Body



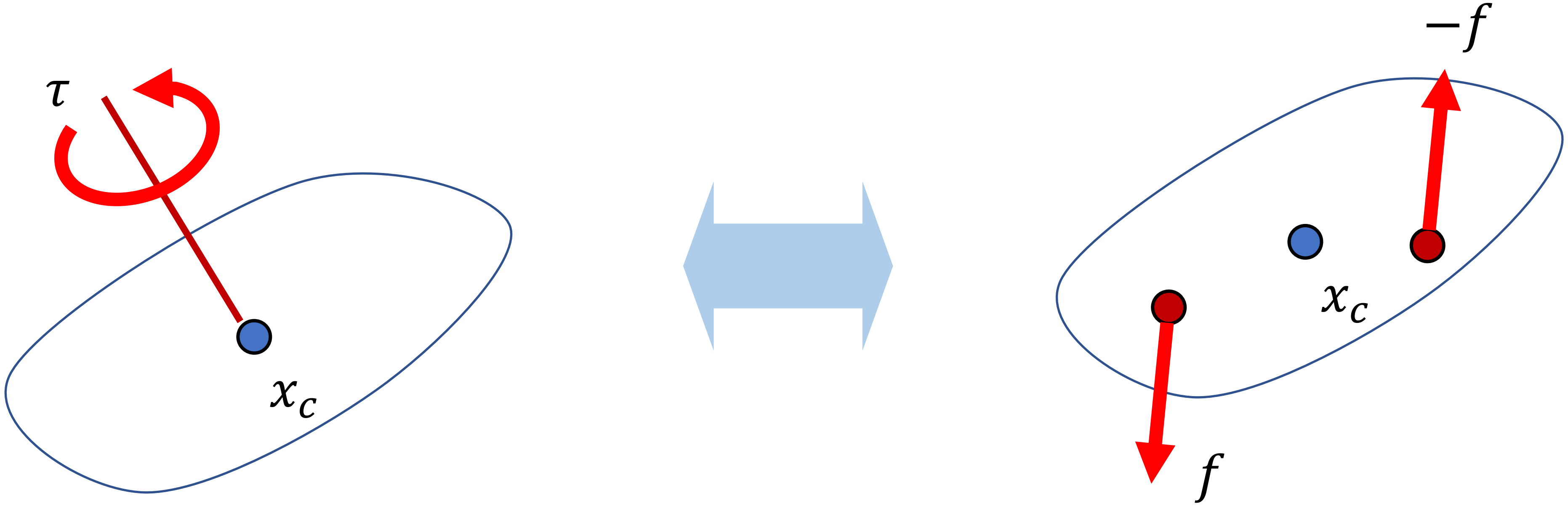
Linear	Angular
x, v	R, ω
$p = mv$	$L = I\omega$

Newton's Second Law:

Euler's laws of motion:

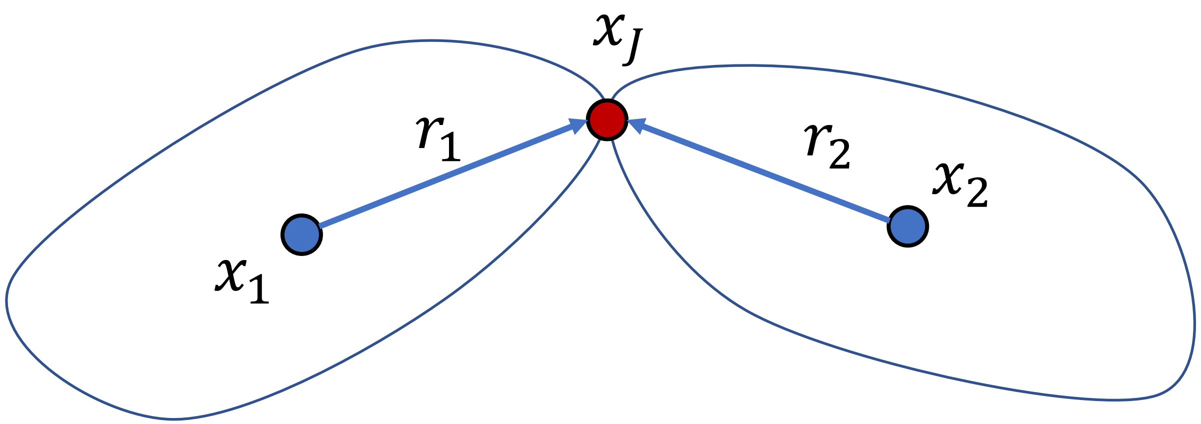
$$\begin{bmatrix} m\mathrm{I}_3 & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix}\dot{v}\\ \dot{\omega}\end{bmatrix} + \begin{bmatrix}0\\ \omega\times I\omega\end{bmatrix} = \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{\tau}+\boldsymbol{r}_{\boldsymbol{f}}\times\boldsymbol{f}\end{bmatrix}$$

Actuating a Rigid Body



Actuating Articulated Rigid Bodies

$m_i, I_i, x_i, R_i, v_i, \omega_i$

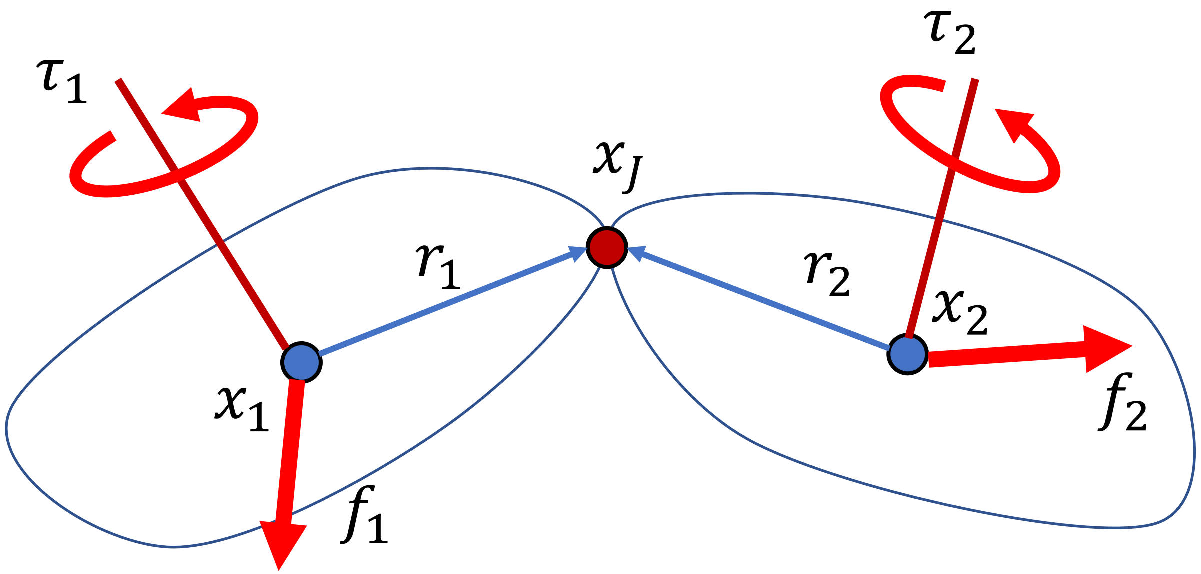


$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + \boldsymbol{J}^T\boldsymbol{\lambda}$$

$$Jv = 0$$

Actuating Articulated Rigid Bodies

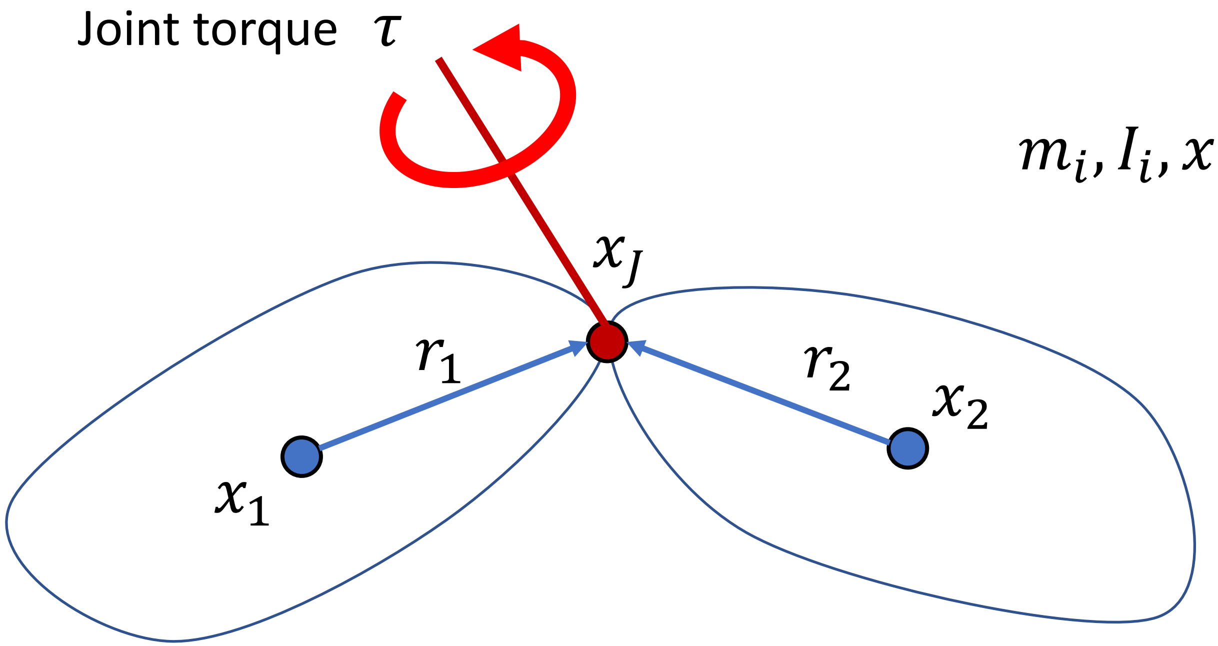
$m_i, I_i, x_i, R_i, v_i, \omega_i$



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

$$Jv = 0$$

Actuating Articulated Rigid Bodies

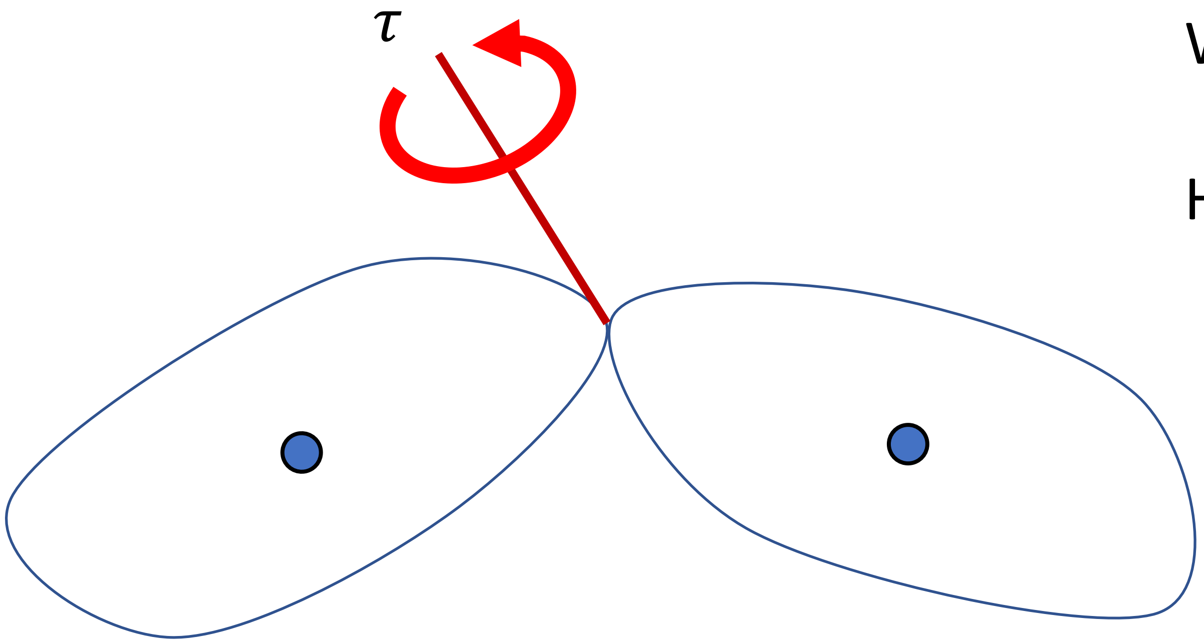


$m_i, I_i, x_i, R_i, v_i, \omega_i$

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

$$Jv = 0$$

Joint Torques



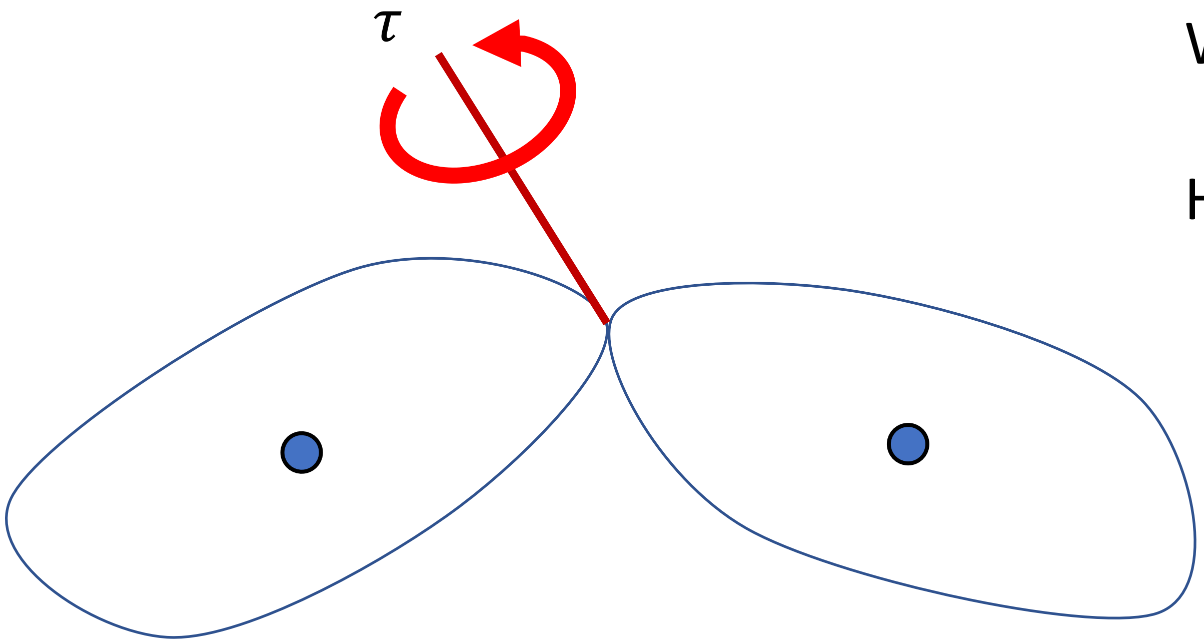
What is a joint torque?

How is a joint torque applied?

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

$$Jv = 0$$

Joint Torques



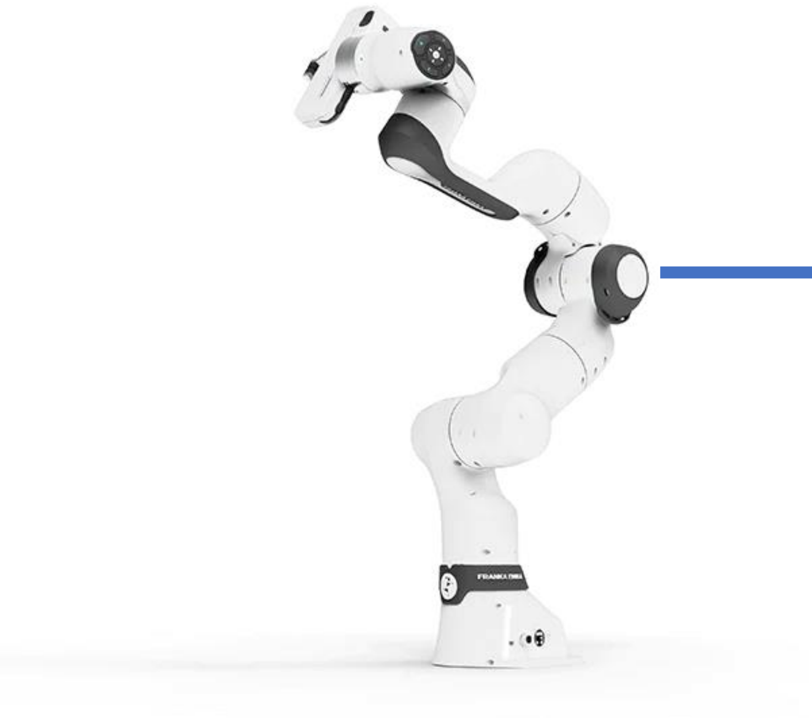
What is a joint torque?

How is a joint torque applied?

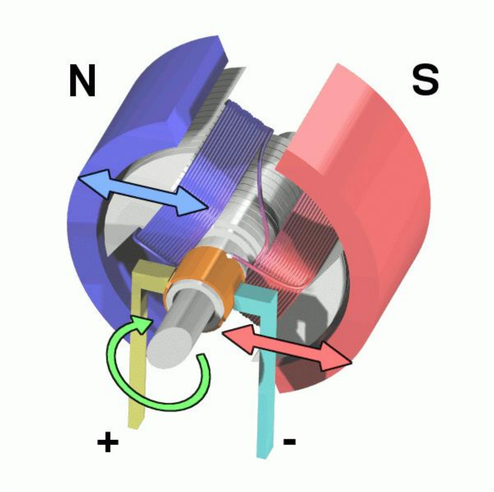
$$\begin{bmatrix} m_1 \mathrm{I}_3 & & & \\ & I_1 & & \\ & & m_2 \mathrm{I}_3 & \\ & & & I_2 \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{\omega}_1 \\ \dot{v}_2 \\ \dot{\omega}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_1 \times I_1 \omega_1 \\ 0 \\ \omega_2 \times I_2 \omega_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ \tau_1 \\ f_2 \\ \tau_2 \end{bmatrix} + J^T \lambda$$

$$Jv = 0$$

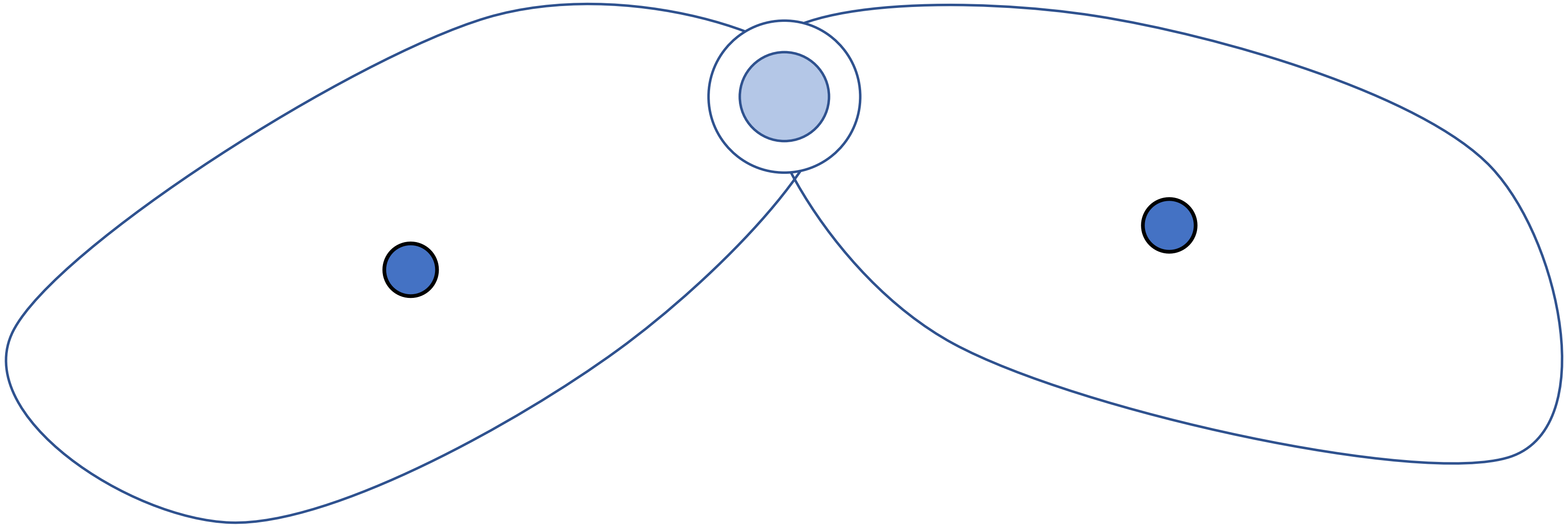
Joint Torques



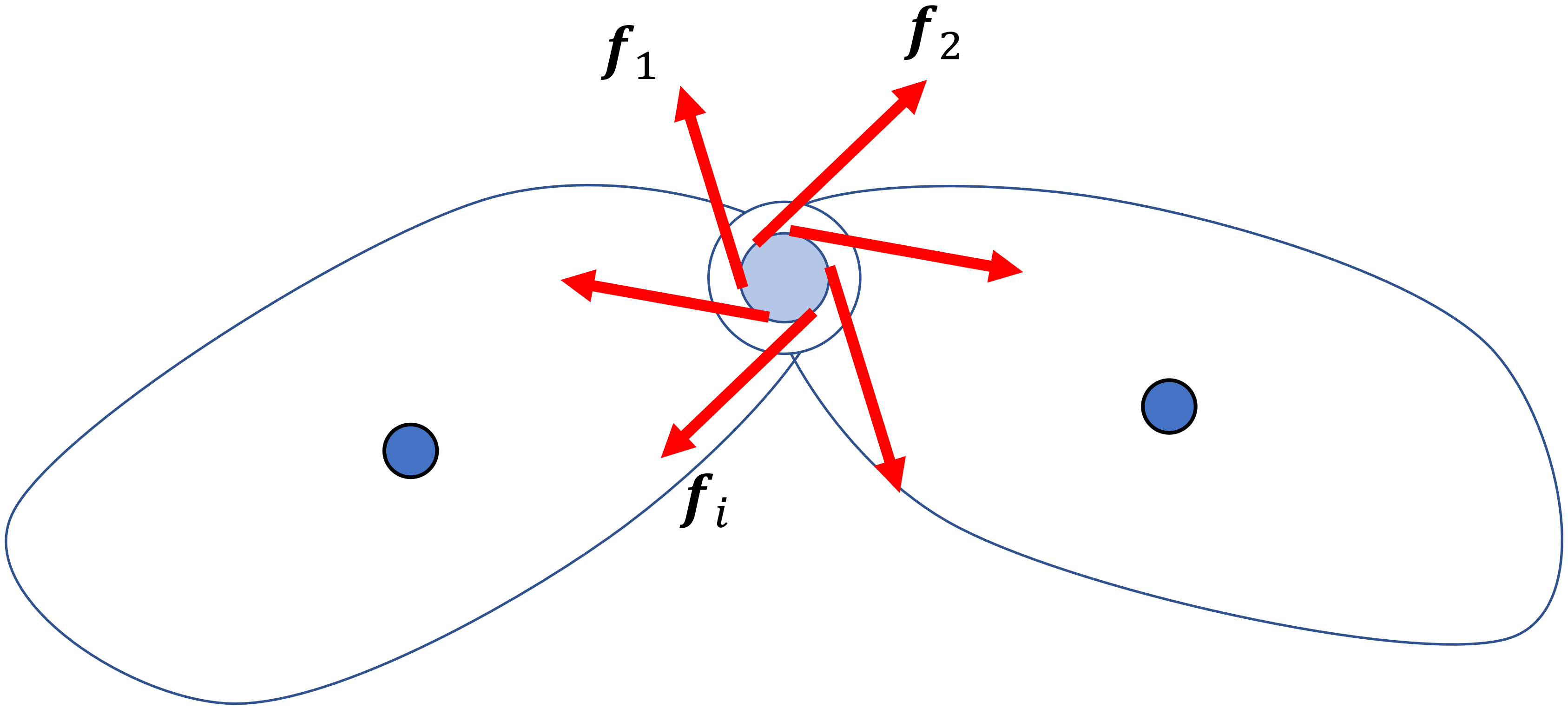
Franka



Joint Torques



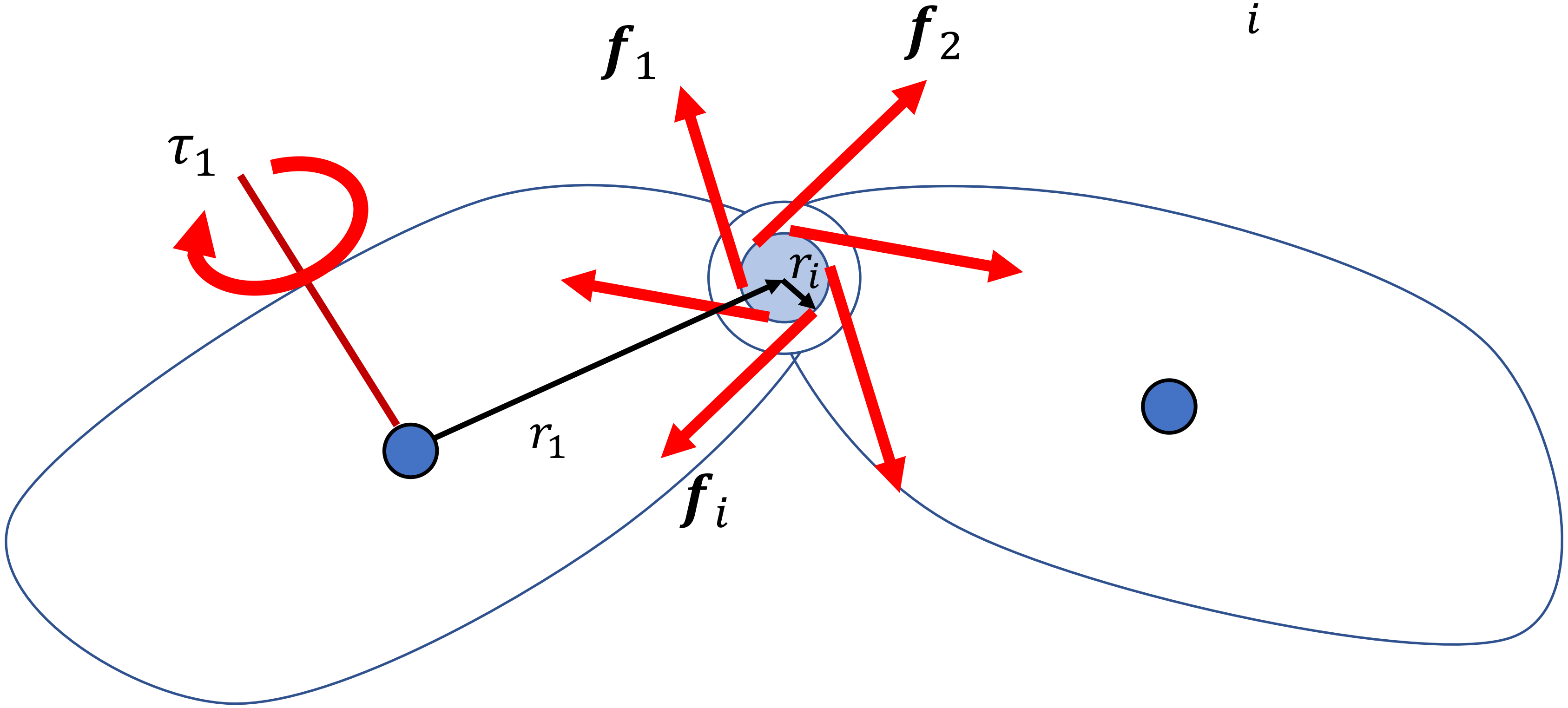
Joint Torques



$$\sum_i f_i = 0$$

Joint Torques

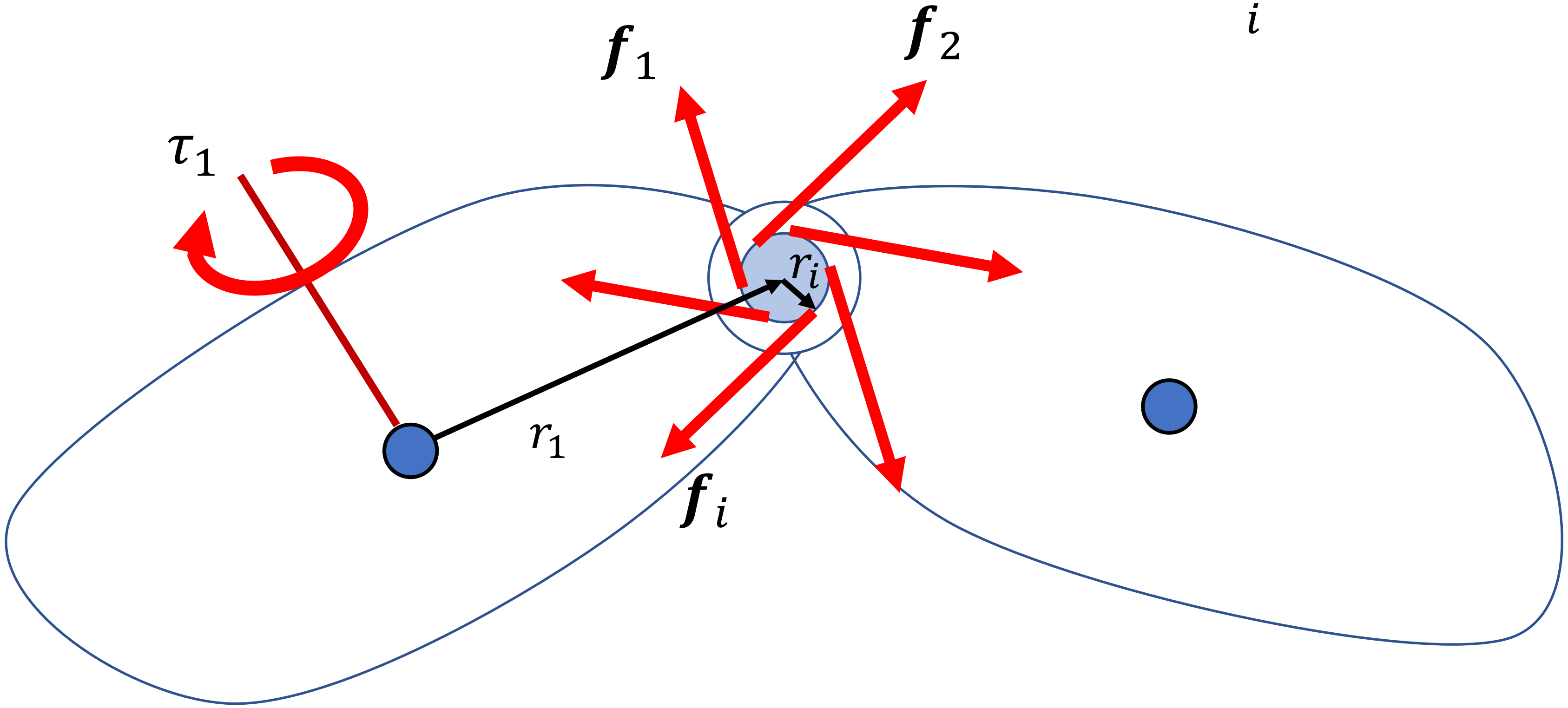
$$\sum_i f_i = 0$$



$$\tau_1 = \sum_i (r_1 + r_i) \times f_i = r_1 \times \sum_i f_i + \sum_i r_i \times f_i$$

Joint Torques

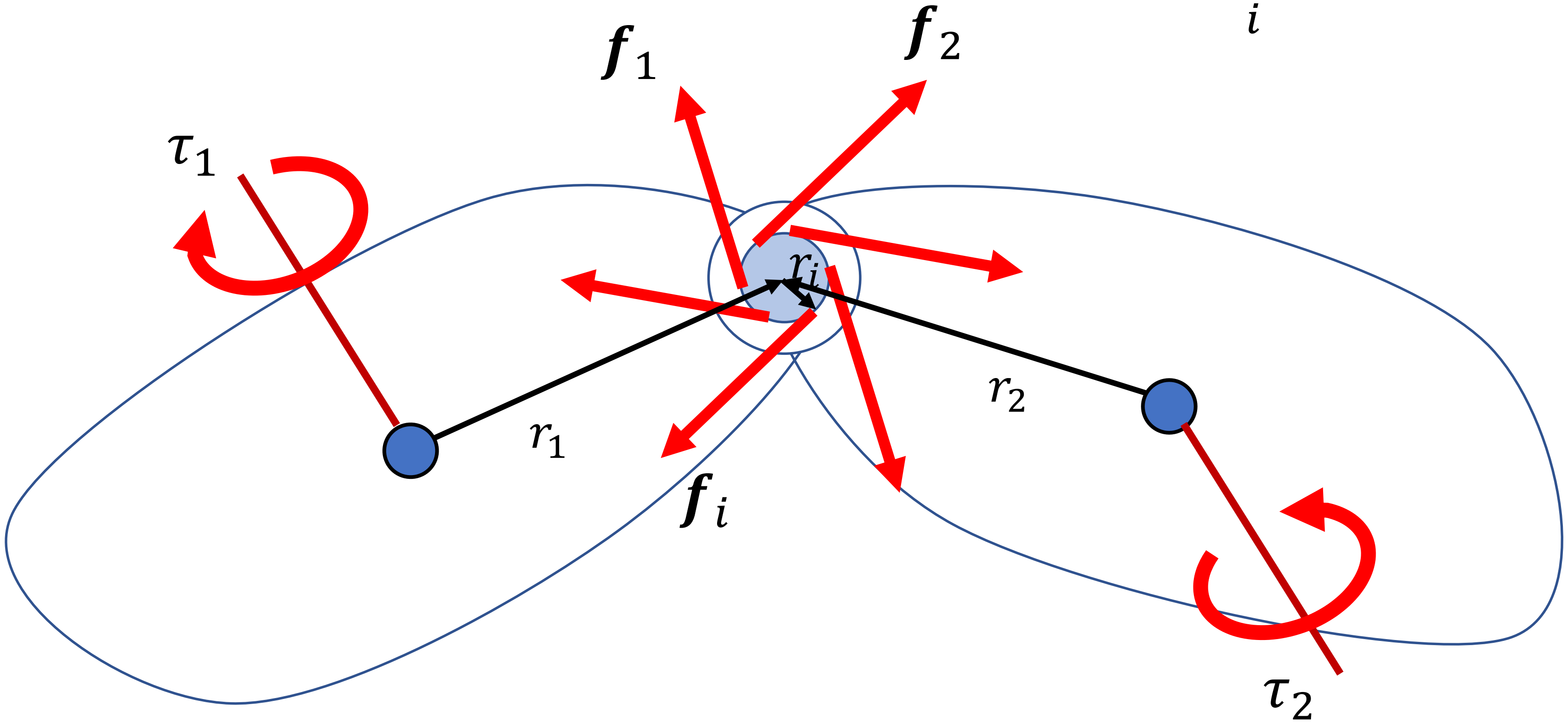
$$\sum_i f_i = 0$$



$$\tau_1 = \sum_i r_i \times f_i$$

Joint Torques

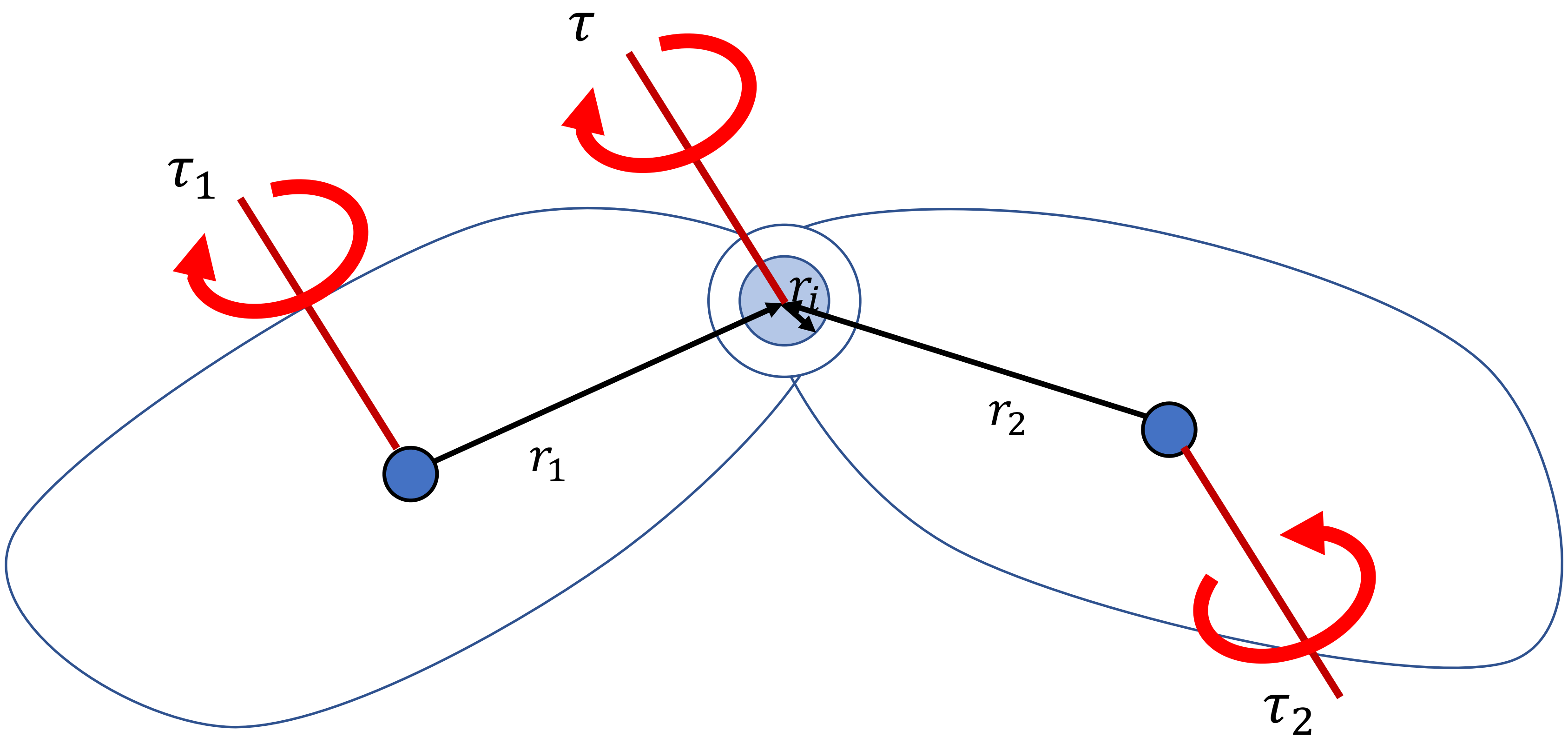
$$\sum_i f_i = 0$$



$$\tau_1 = \sum_i r_i \times f_i$$

$$\tau_2 = -\sum_i r_i \times f_i$$

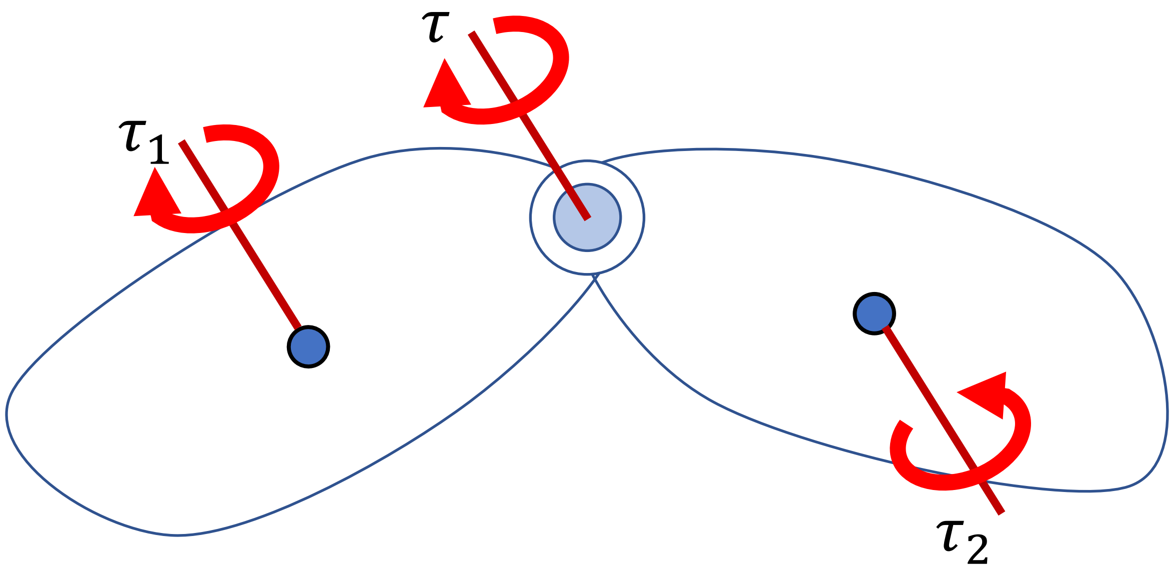
Joint Torques



$\tau_1 = \tau$

$\tau_2 = -\tau$

Joint Torques



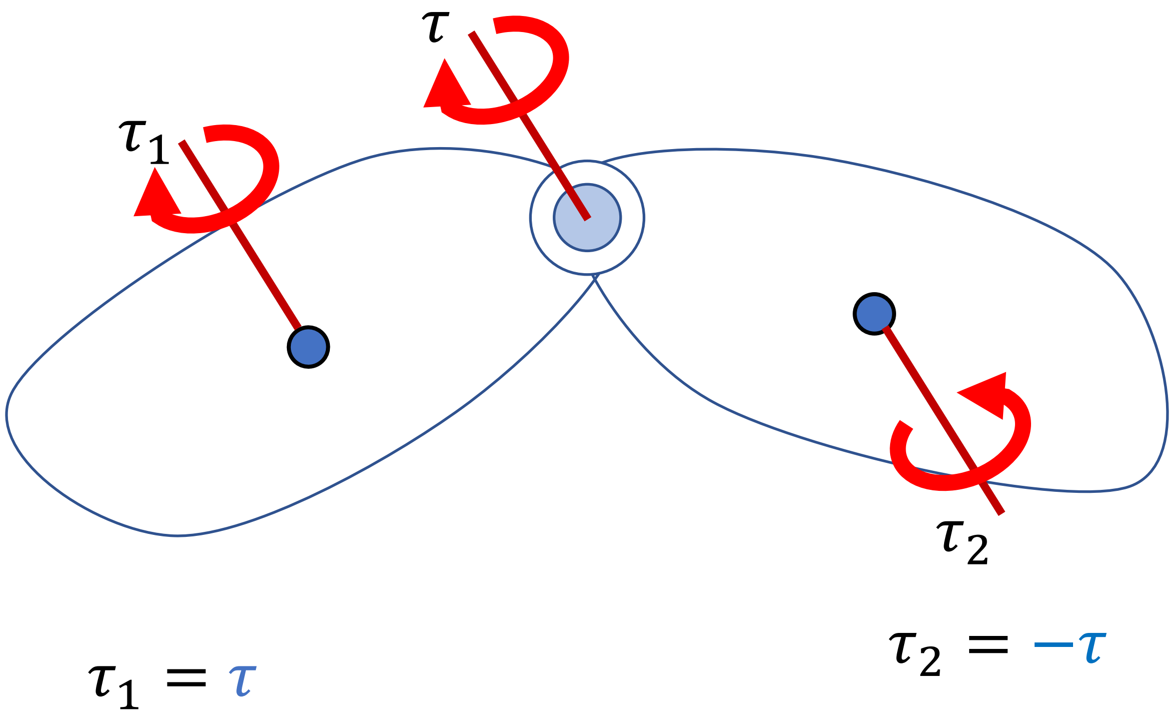
$\tau_1 = \tau$

$\tau_2 = -\tau$

Applying a joint torque τ:

- Add τ to one attached body
- Add $-\tau$ to the other attached body

Joint Torques



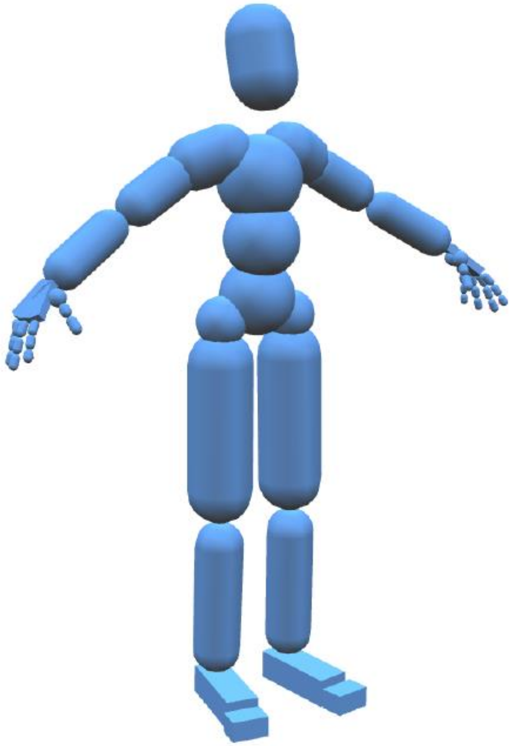
Applying a joint torque τ:

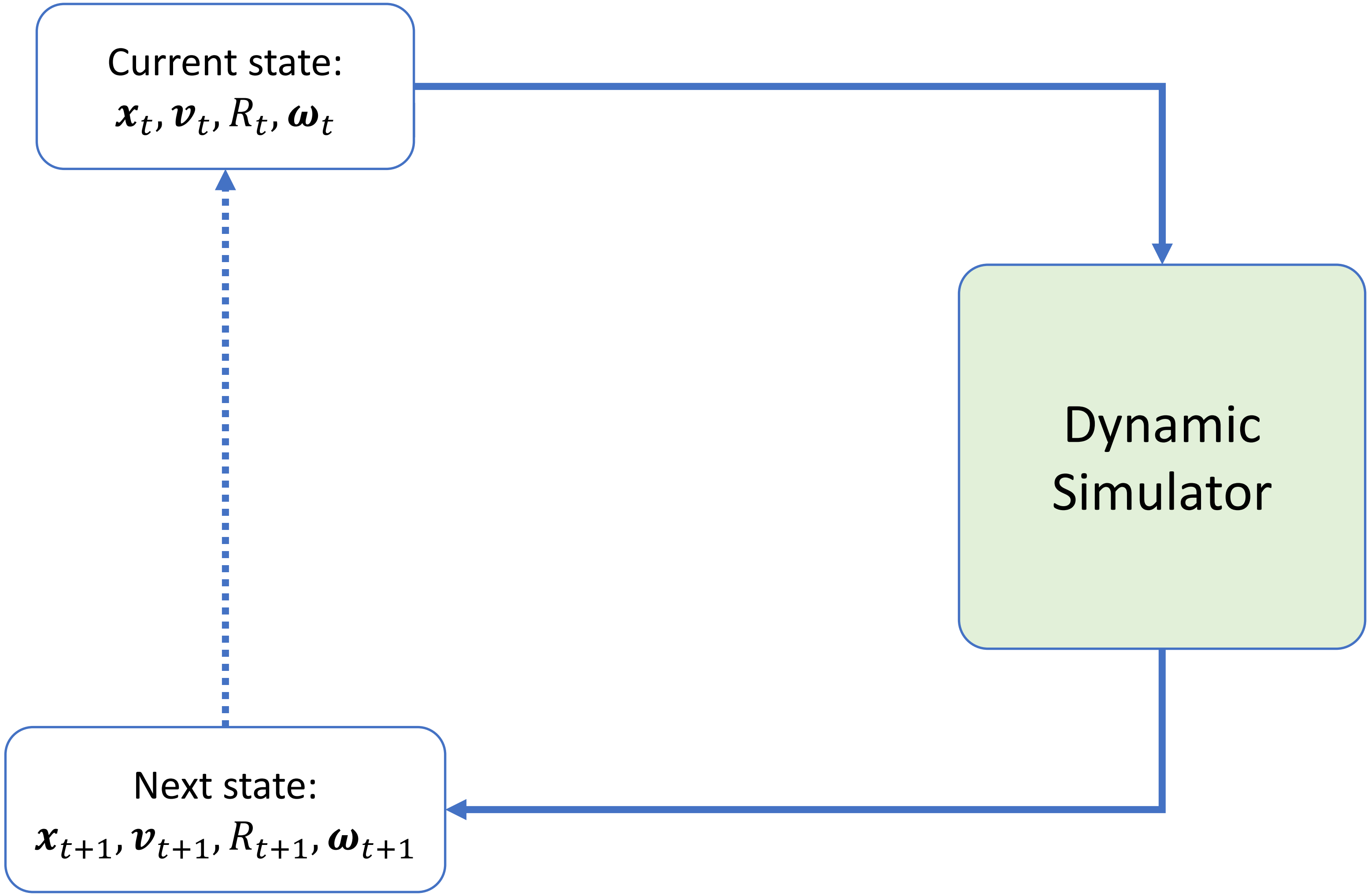
- Add τ to one attached body
- Add $-\tau$ to the other attached body

$$M \begin{bmatrix} \dot{v}_1 \\ \dot{\omega}_1 \\ \dot{v}_2 \\ \dot{\omega}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_1 \times I_1 \omega_1 \\ 0 \\ \omega_2 \times I_2 \omega_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau \\ 0 \\ -\tau \end{bmatrix} + J^T \lambda$$

$$Jv = 0$$

Simulating a Character





Current state:

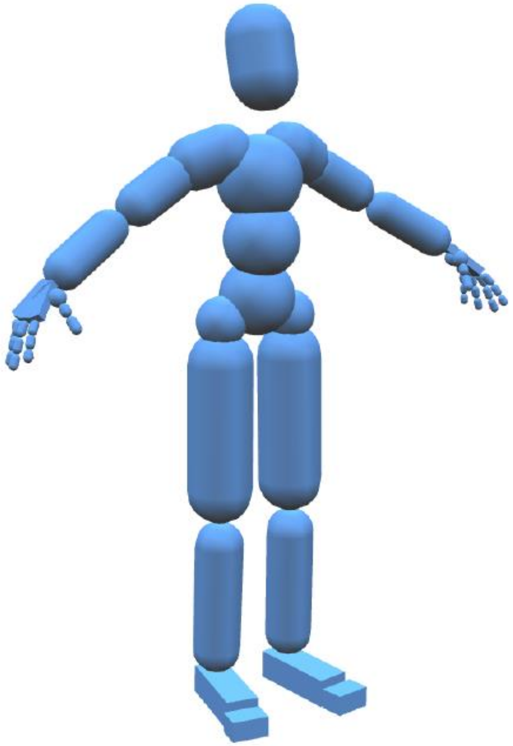
$\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t$

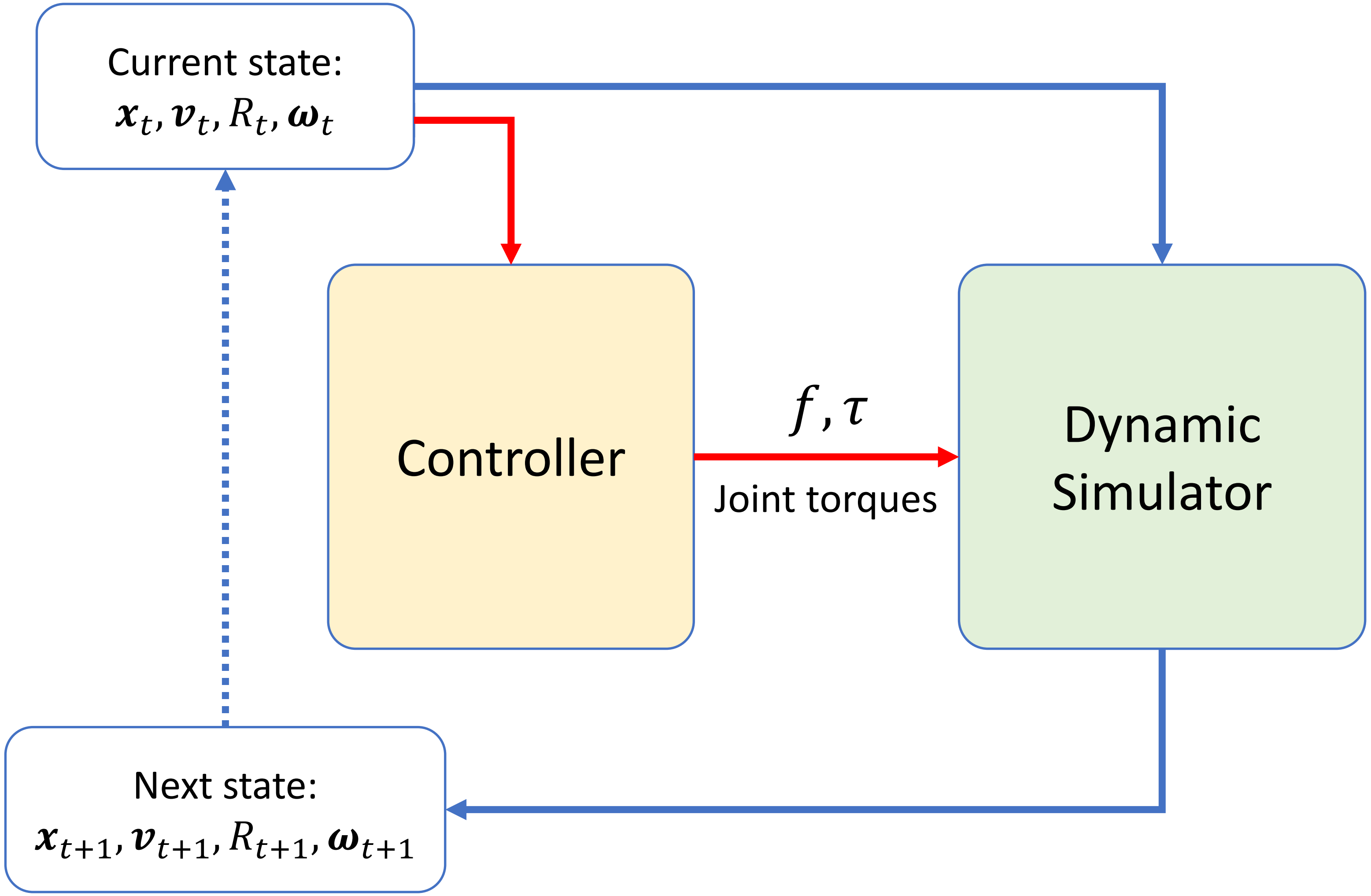
Dynamic Simulator

Next state:

$\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1}$

Simulating + Controlling a Character





Current state:
$\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t$

Controller

f, τ

Joint torques

Dynamic Simulator

Next state:
$\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1}$

Forward Dynamics vs. Inverse Dynamics

Equations of motion of the system:

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda \qquad Jv = 0$$

Forward Dynamics vs. Inverse Dynamics

Equations of motion of the system:

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda \qquad Jv = 0$$

Forward dynamics: $[\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{f}] \mapsto \dot{\boldsymbol{v}}$

given a set of force/torques, compute the motion

Forward Dynamics vs. Inverse Dynamics

Equations of motion of the system:

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda \qquad Jv = 0$$

Forward dynamics: $[\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{f}] \mapsto \dot{\boldsymbol{v}}$

given a set of force/torques, compute the motion

Inverse dynamics: $[\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}}] \mapsto \boldsymbol{f}$

given a motion, compute the forces/torques that give rise to it

Forward Dynamics vs. Inverse Dynamics

Equations of motion of the system:

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda \qquad Jv = 0$$

Forward dynamics: $[\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{f}] \mapsto \dot{\boldsymbol{v}}$

dynamic simulator

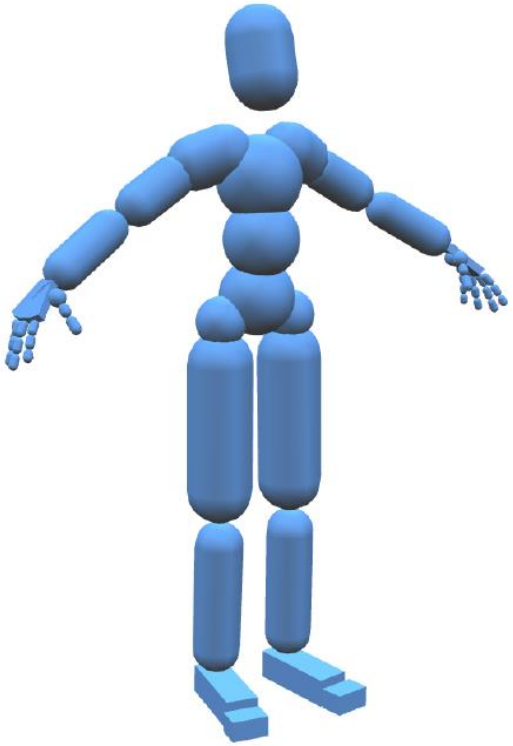
given a set of force/torques, compute the motion

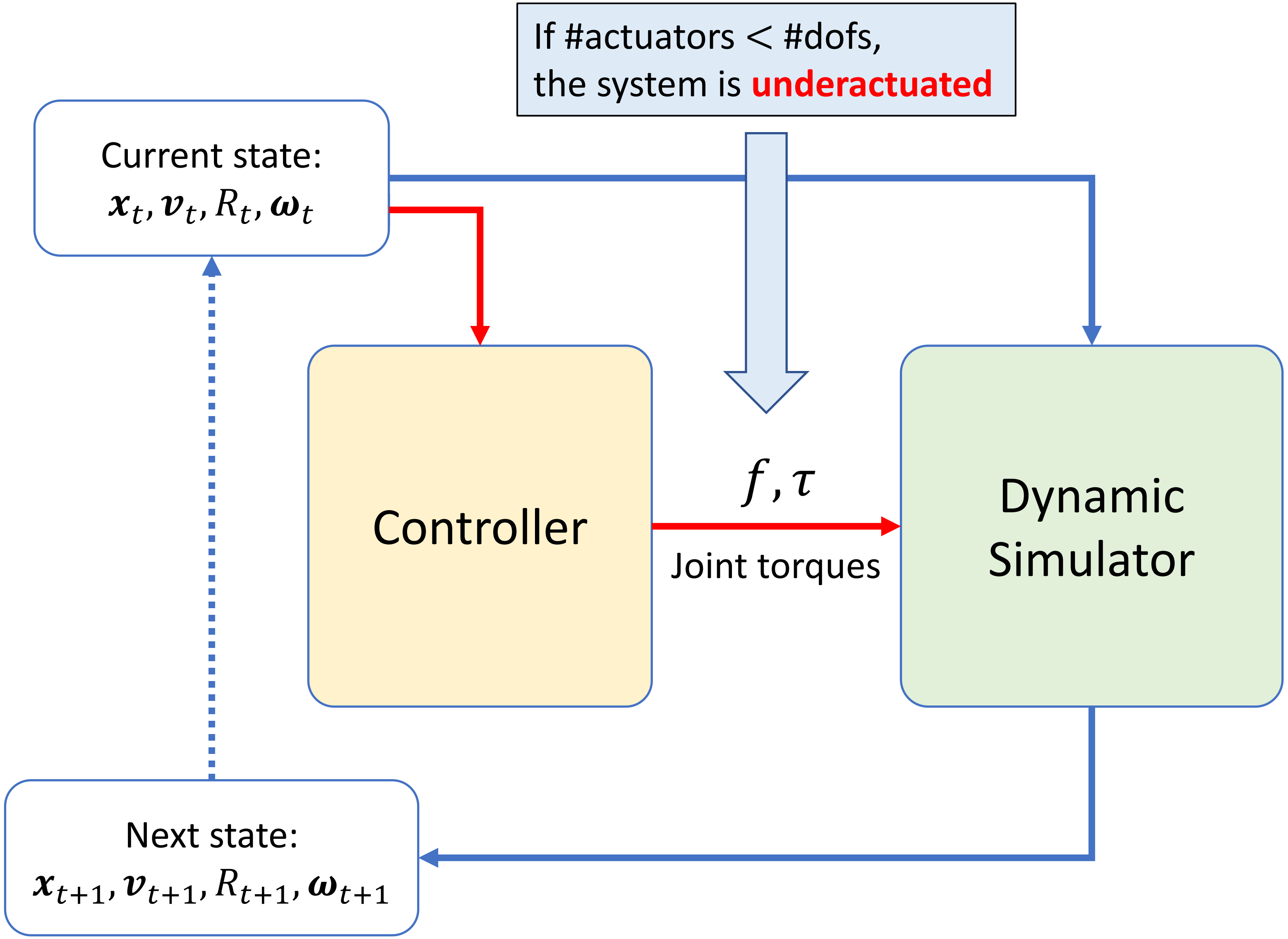
Inverse dynamics: $[\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}}] \mapsto \boldsymbol{f}$

controller

given a motion, compute the forces/torques that give rise to it

Simulating + Control





If #actuators < #dofs,
the system is **underactuated**

Current state:
$\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t$

Controller

f, τ

Joint torques

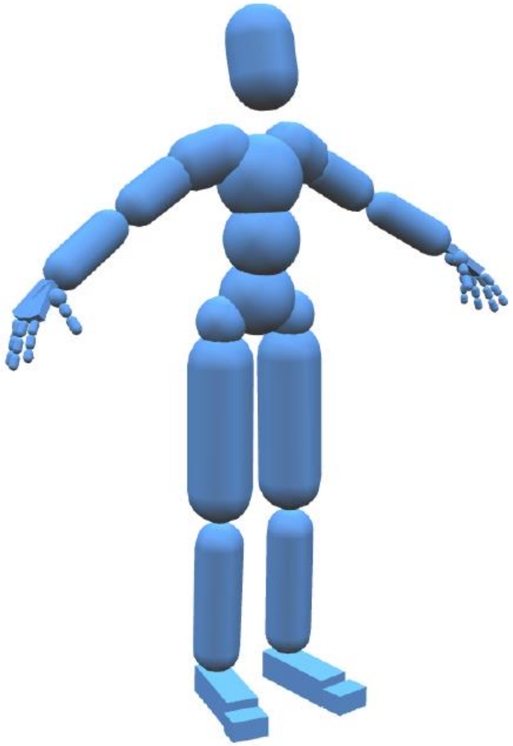
Dynamic Simulator

Next state:
$\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1}$

Fully-Actuated vs. Underactuated

Fully-Actuated

Underactuated



If #actuators $\geq$ #dofs, the system is **fully-actuated**

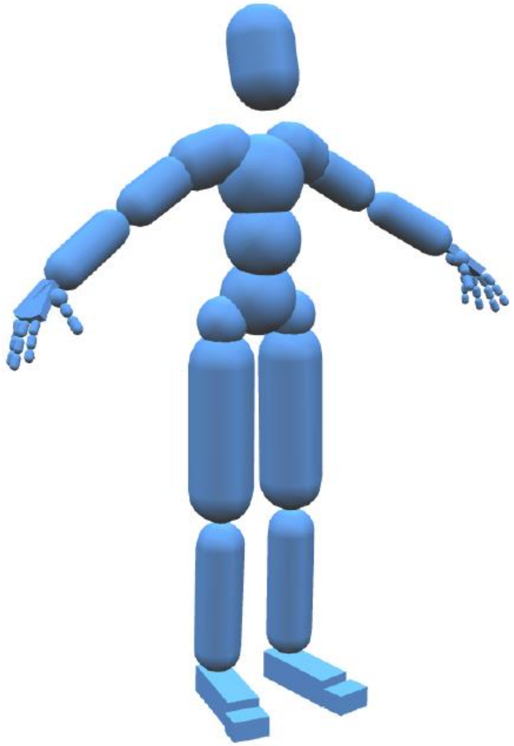
If #actuators $<$ #dofs, the system is **underactuated**

Fully-Actuated vs. Underactuated

Fully-Actuated

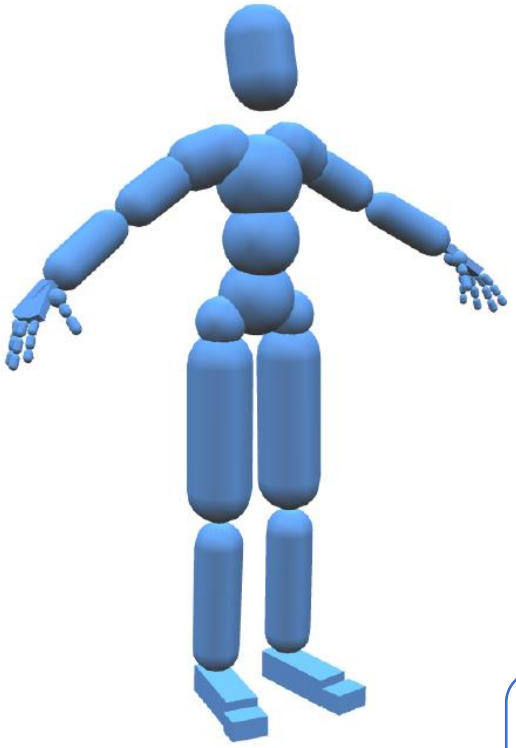
For any $[\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}}]$, there exists an $\boldsymbol{f}$ that produces the motion

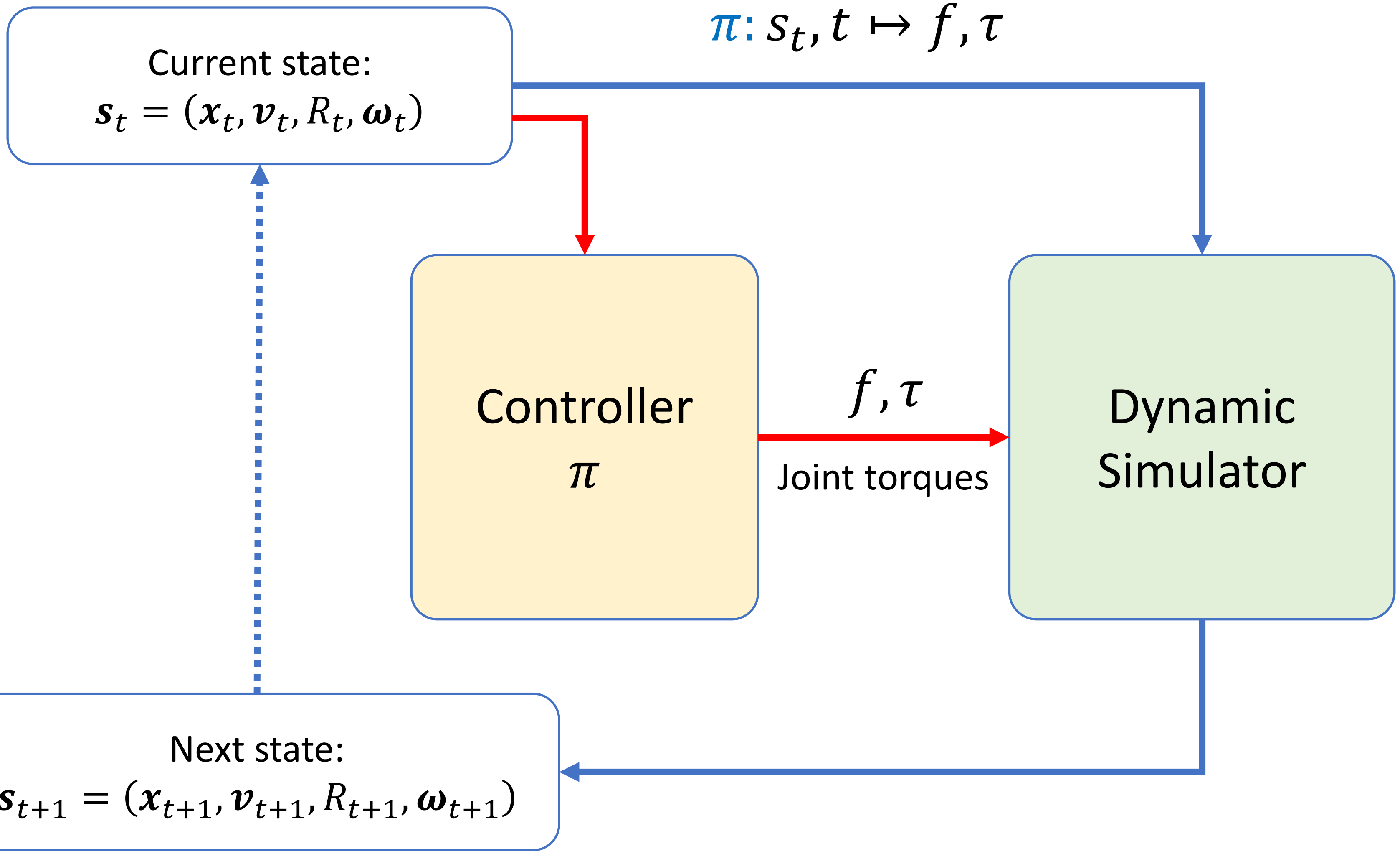
Underactuated



For many $[\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}}]$, there is no such $\boldsymbol{f}$ that produces the motion

Simulating + Control





Current state:

$\boldsymbol{s}_t = (\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t)$

$\pi: s_t, t \mapsto f, \tau$

Controller

π

f, τ

Joint torques

Dynamic Simulator

Next state:

$\boldsymbol{s}_{t+1} = (\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1})$

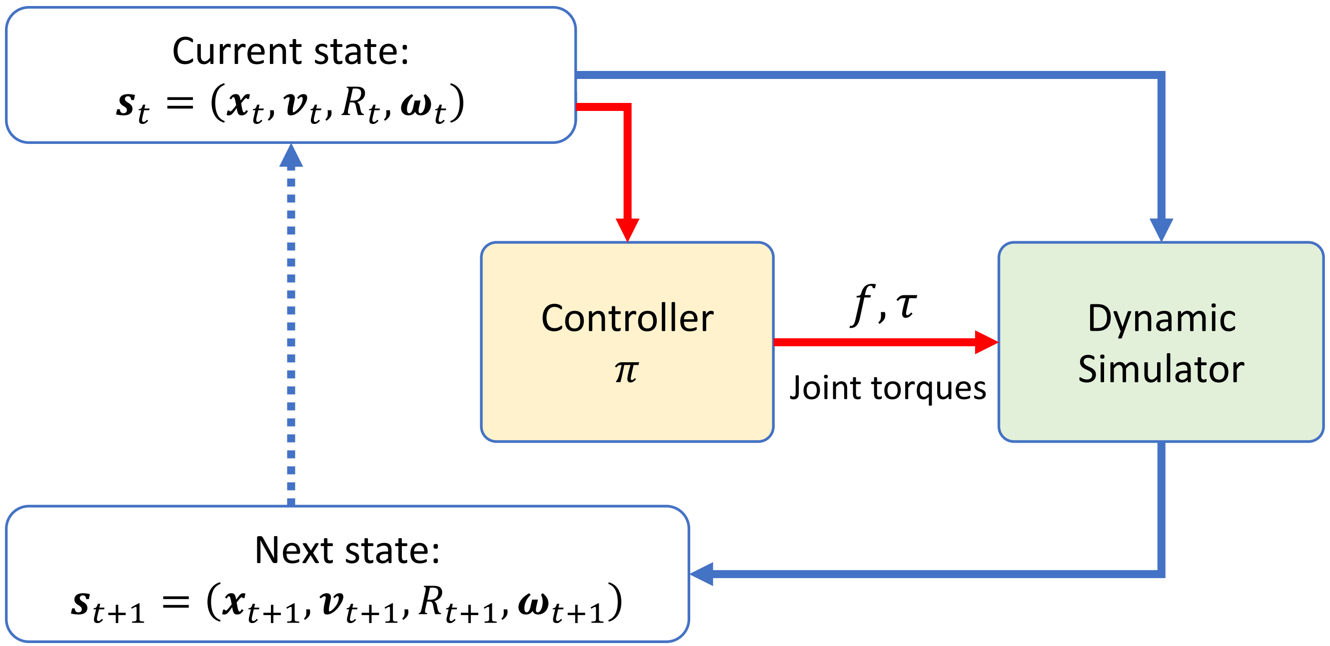
Feedforward vs. Feedback

Feedforward control:

$$f, \tau = \pi(t)$$

- Apply predefined control signals without considering the current state of the system
- Assuming unchanging system. Perturbations may lead to unpredicted results

$$\pi: s_t, t \mapsto f, \tau$$



Current state:
$\boldsymbol{s}_t = (\boldsymbol{x}_t, \boldsymbol{v}_t, R_t, \boldsymbol{\omega}_t)$

Controller
π

f, τ
Joint torques

Dynamic Simulator

Next state:
$\boldsymbol{s}_{t+1} = (\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, R_{t+1}, \boldsymbol{\omega}_{t+1})$

Feedforward vs. Feedback

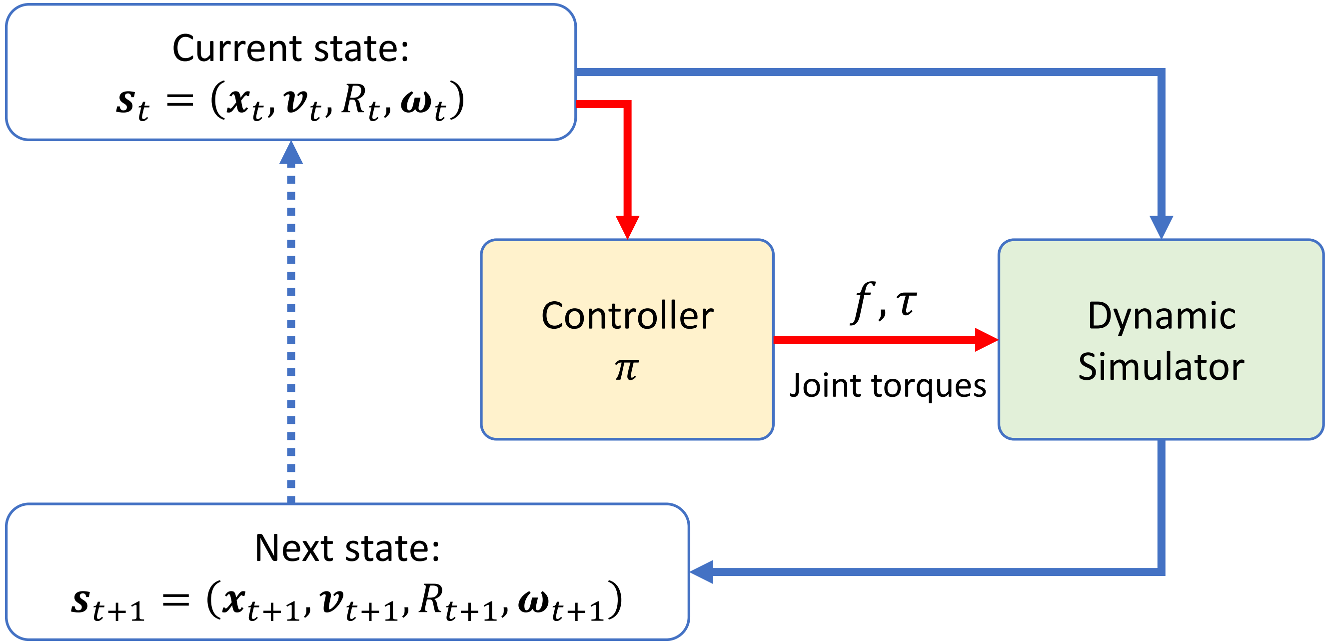
Feedback control:

$$f, \tau = \pi(s_t, t)$$

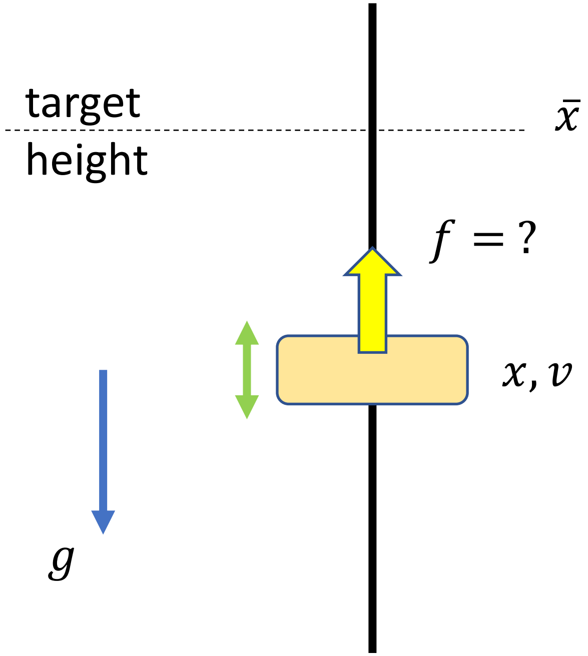
- Adjust control signals based on the current state of the system

- Certain perturbations are expected. The feedback signal will be used to improves the performance at the next state.

$\pi: s_t, t \mapsto f, \tau$

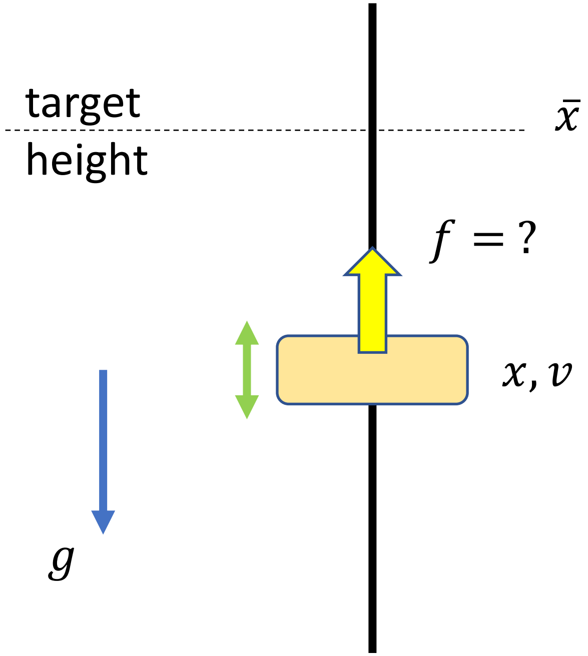


Proportional-Derivative Control



Compute force f to move
the object to the target height

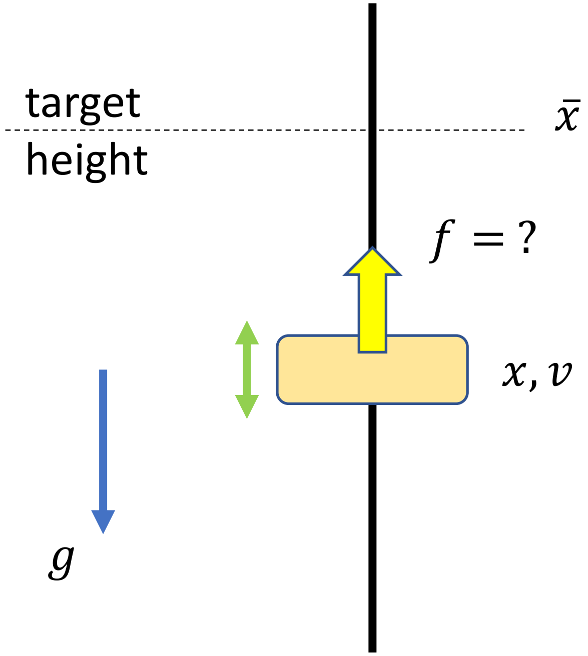
Proportional-Derivative Control



Compute force f to move
the object to the target height

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

Proportional-Derivative Control



Compute force f to move
the object to the target height

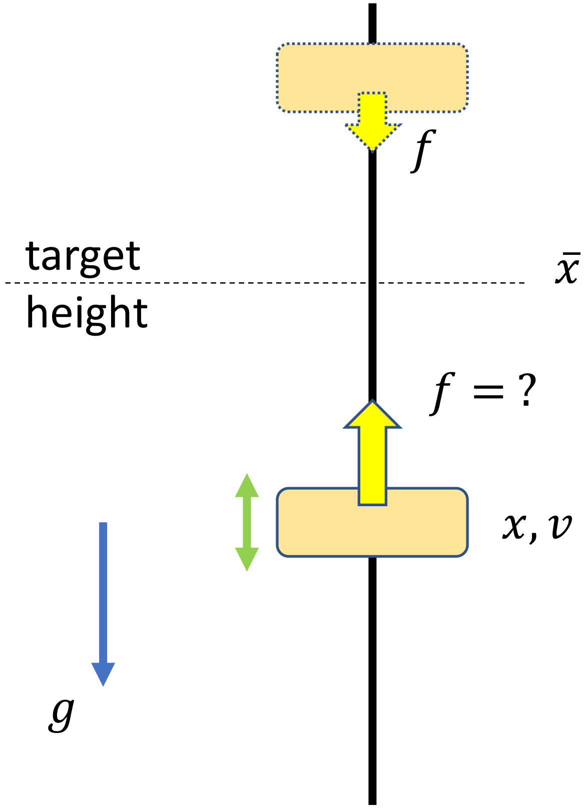
target
state

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

stiffness
(gain)

current
state

Proportional-Derivative Control



Compute force f to move
the object to the target height

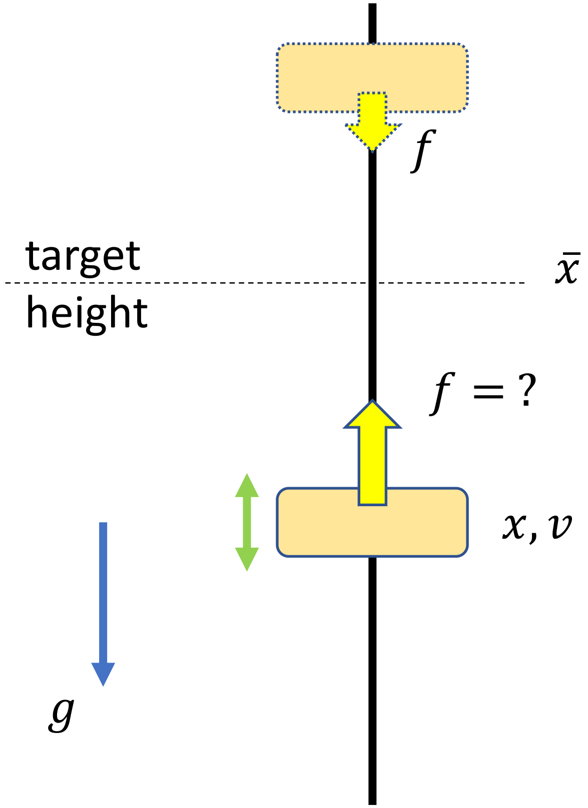
target
state

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

stiffness
(gain)

current
state

Proportional-Derivative Control



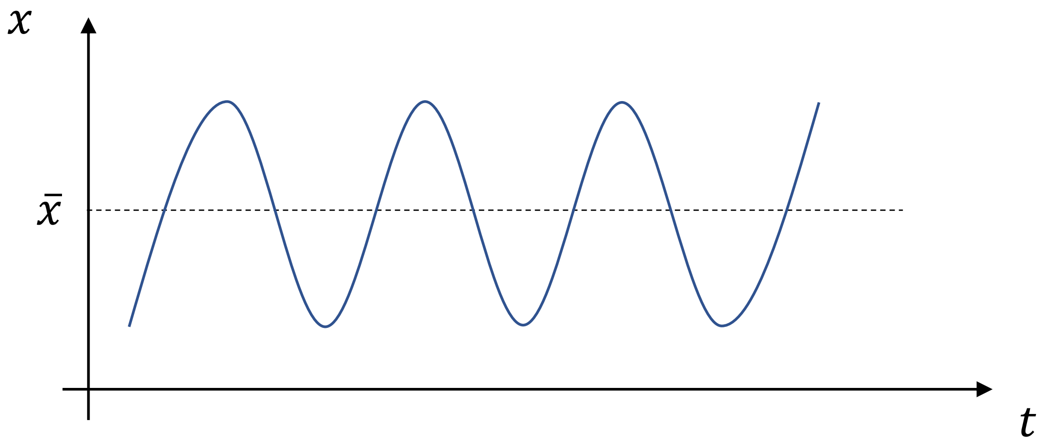
Compute force f to move
the object to the target height

target
state

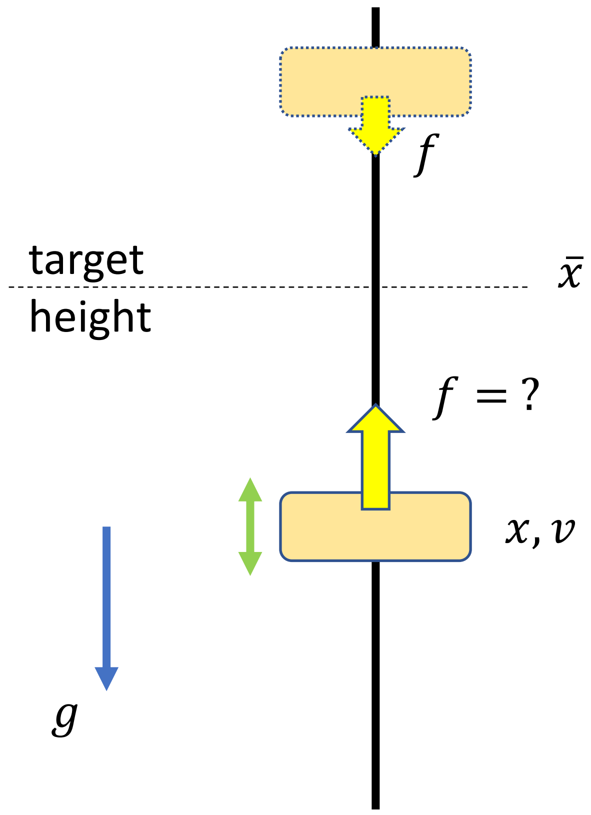
$$f = k_p(\bar{x} - x)$$

stiffness
(gain)

current
state



Proportional-Derivative Control



Compute force f to move the object to the target height

target state

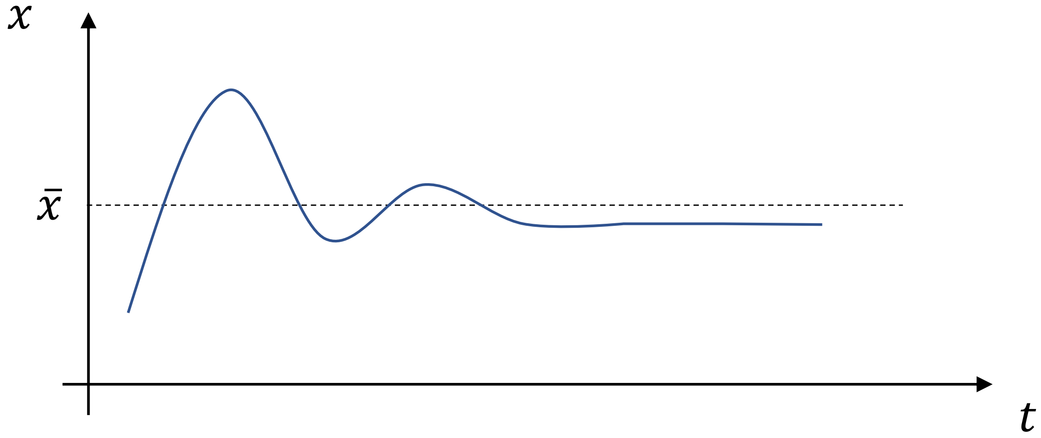
damping

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

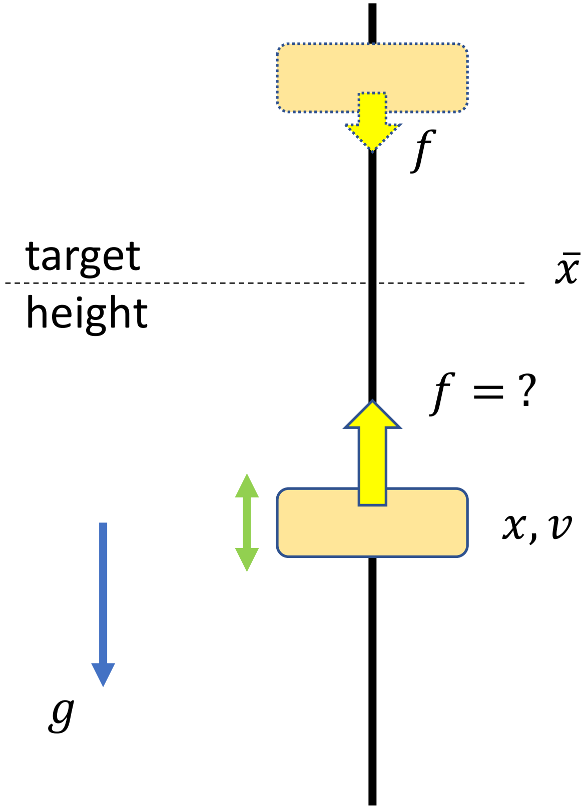
stiffness (gain)

current state

current velocity



Proportional-Derivative Control



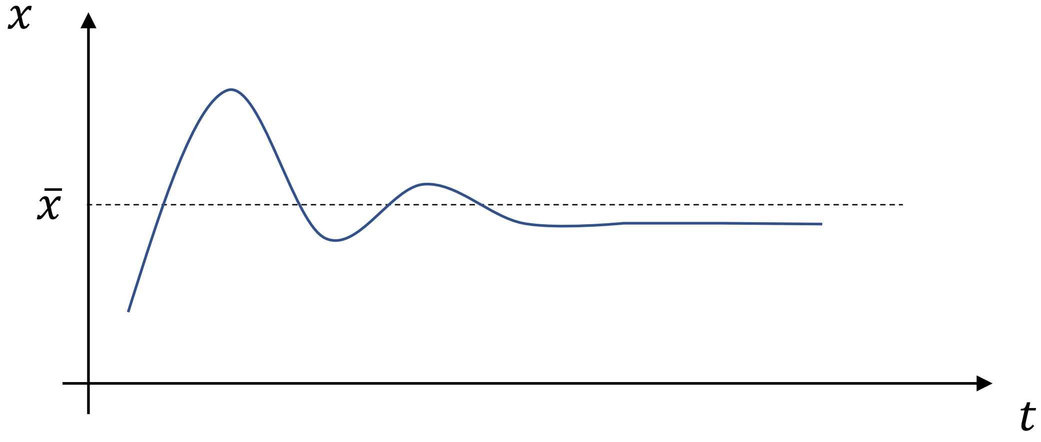
Compute force f to move the object to the target height

Let $e = \bar{x} - x$ be the error

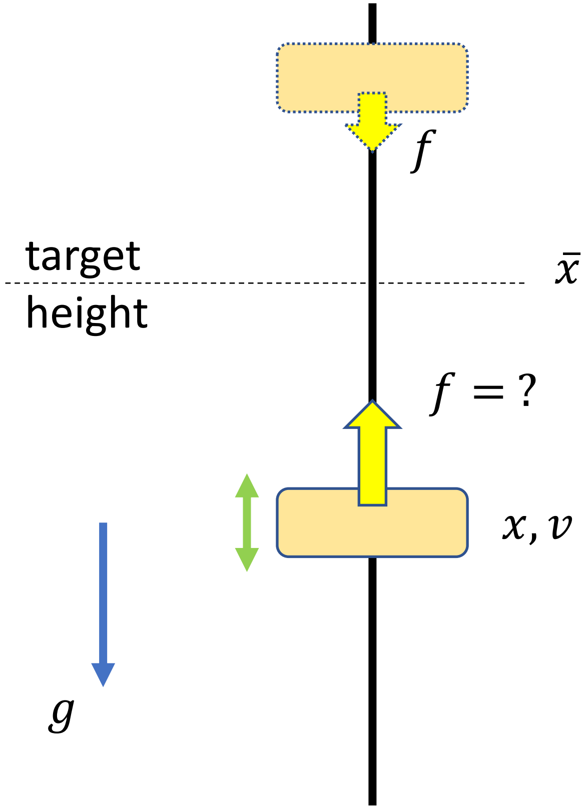
$$f = k_p e + k_d \dot{e}$$

proportional control

derivative control



Proportional-Derivative Control



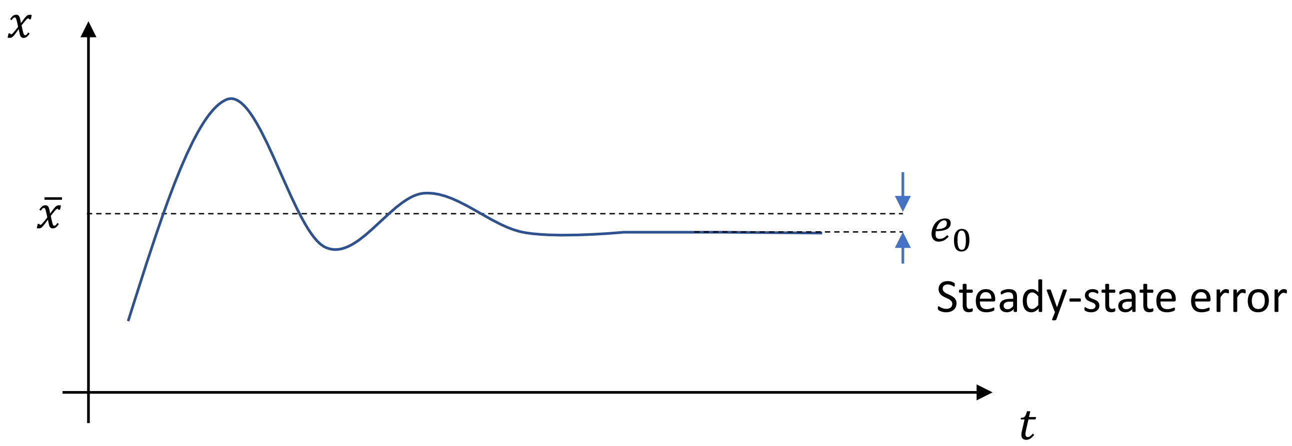
Compute force f to move
the object to the target height

Let $e = \bar{x} - x$ be the error

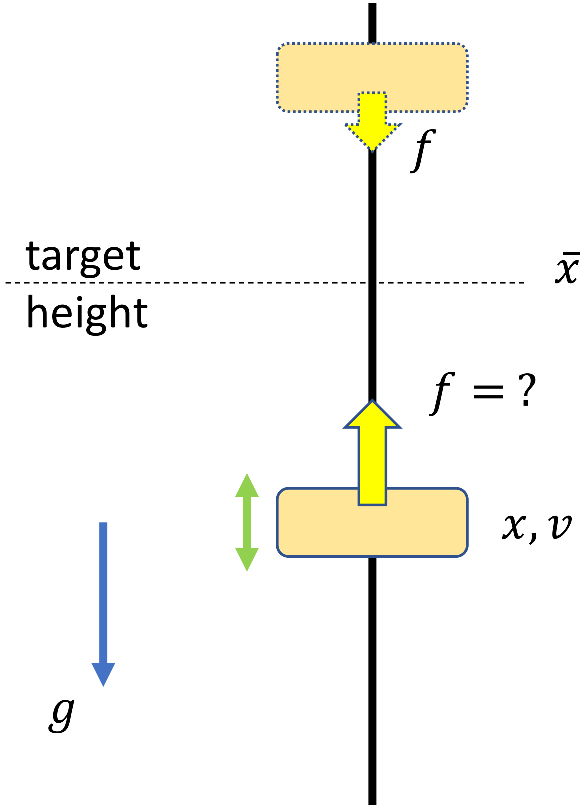
$$f = k_p e + k_d \dot{e}$$

proportional control

derivative control



Proportional-Derivative Control



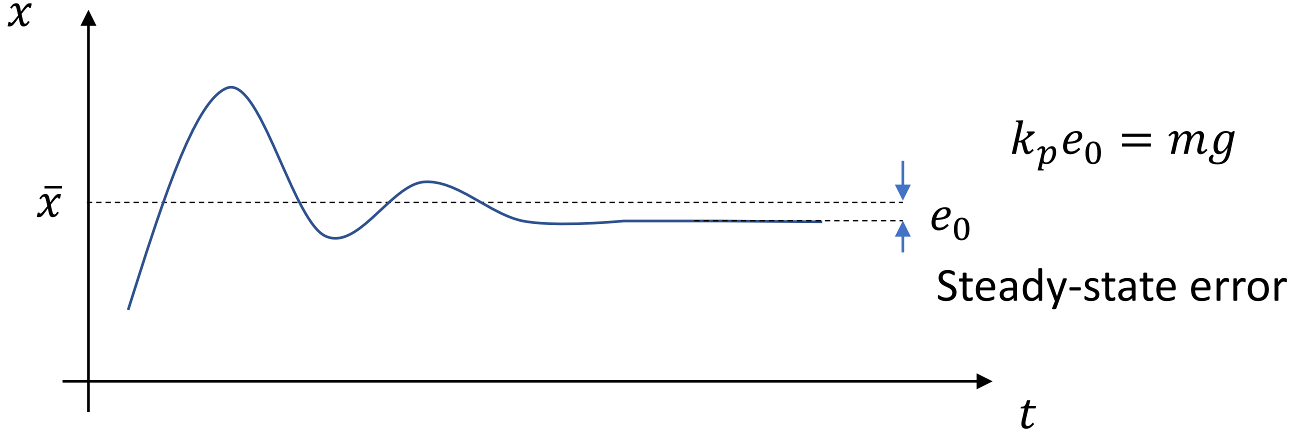
Compute force f to move
the object to the target height

Let $e = \bar{x} - x$ be the error

$$f = k_p e + k_d \dot{e}$$

proportional control

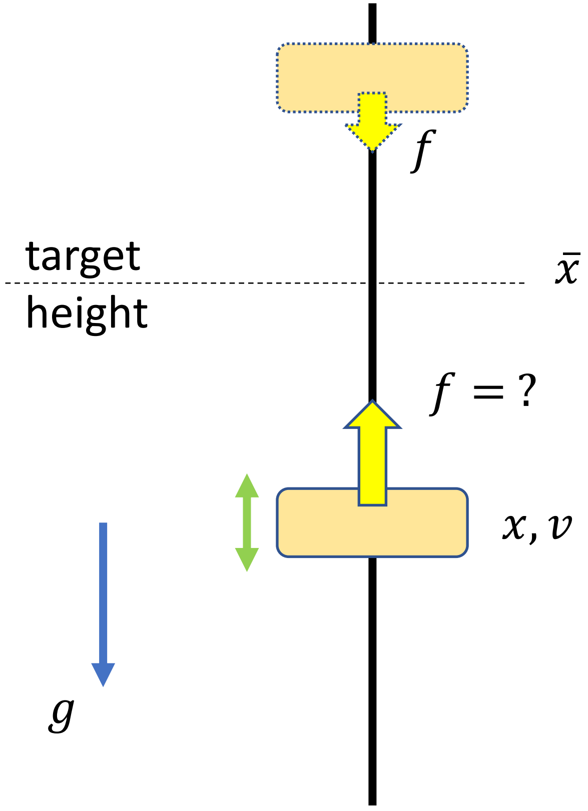
derivative control



$k_p e_0 = mg$

Steady-state error

Proportional-Derivative Control



Compute force f to move the object to the target height

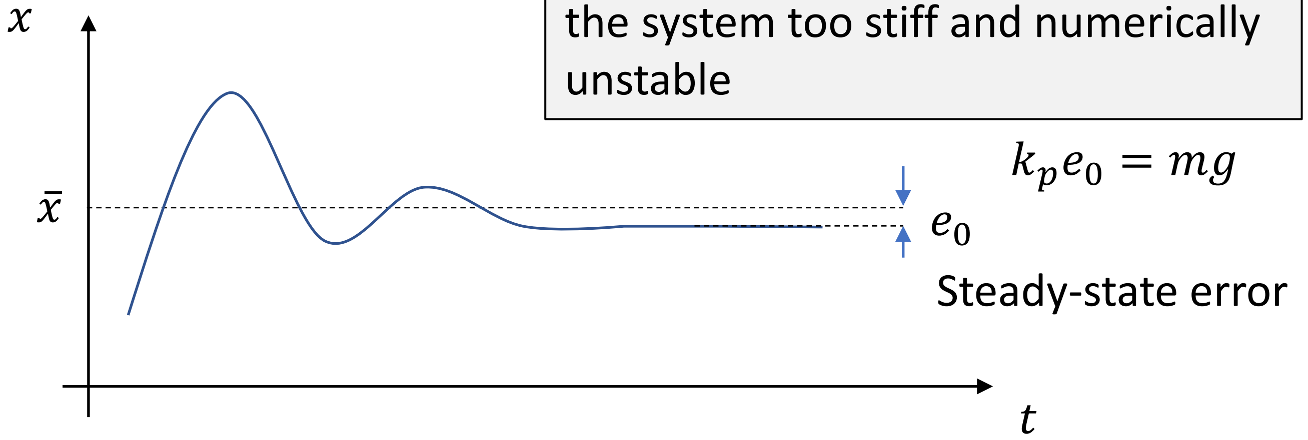
Let $e=\bar{x}-x$ be the error

$$f = k_p e + k_d \dot{e}$$

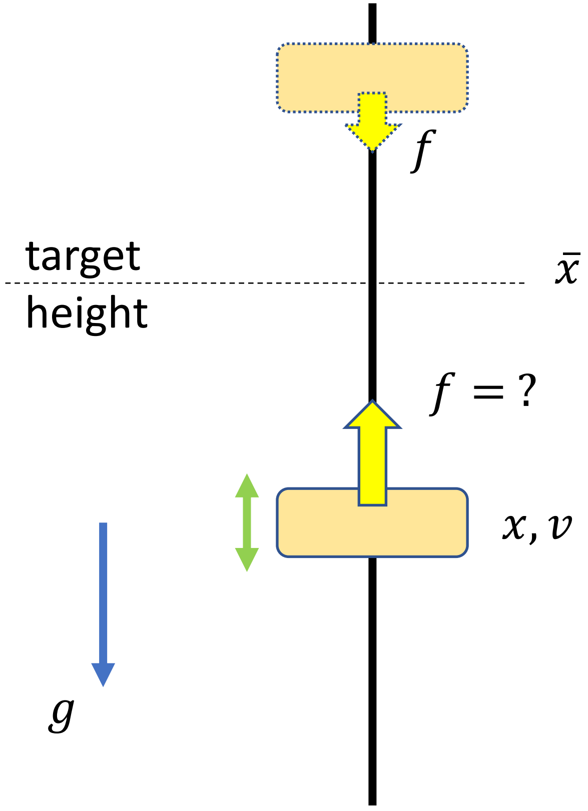
proportional control

derivative control

Increase stiffness k_p reduces the steady-state error, but can make the system too stiff and numerically unstable



Proportional-Integral-Derivative controller



Compute force f to move the object to the target height

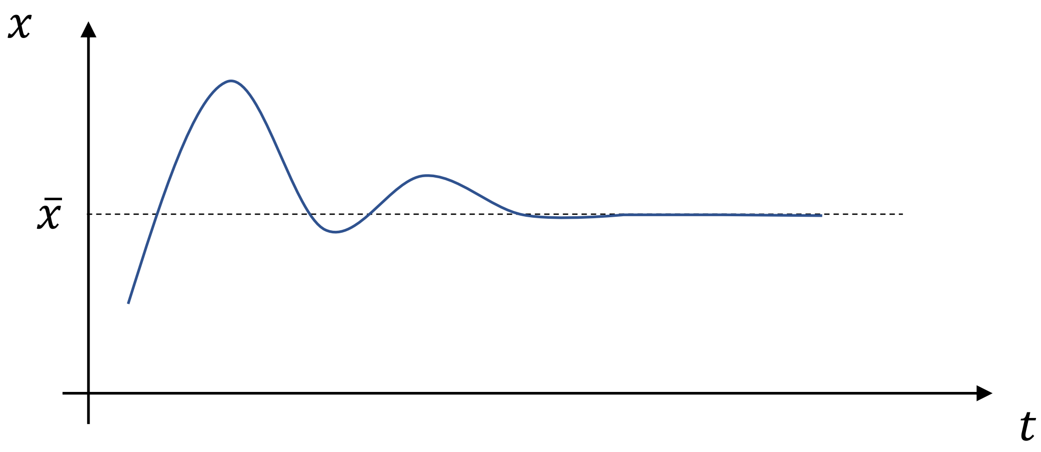
Let $e = \bar{x} - x$ be the error

$$f = k_p e + k_d \dot{e} + k_i \int_t e dt$$

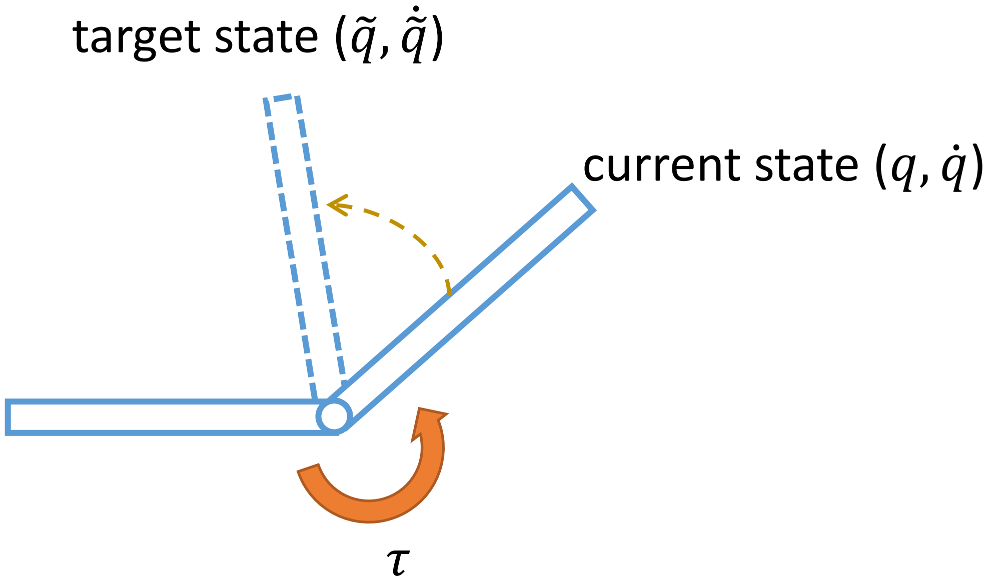
proportional control

derivative control

Integral control (rarely used in animation)



PD Control for Characters



target velocity

often zero

target pose

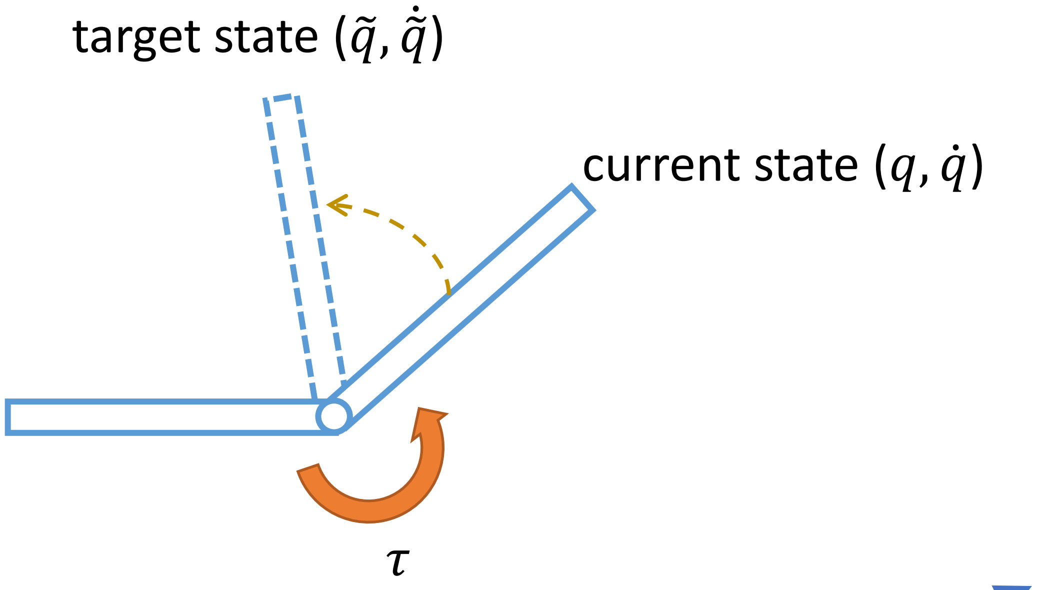
$$\tau = k_p(\bar{q} - q) + k_d(\dot{\bar{q}} - \dot{q})$$

joint torque

proportional control

derivative control

PD Control for Characters



target pose

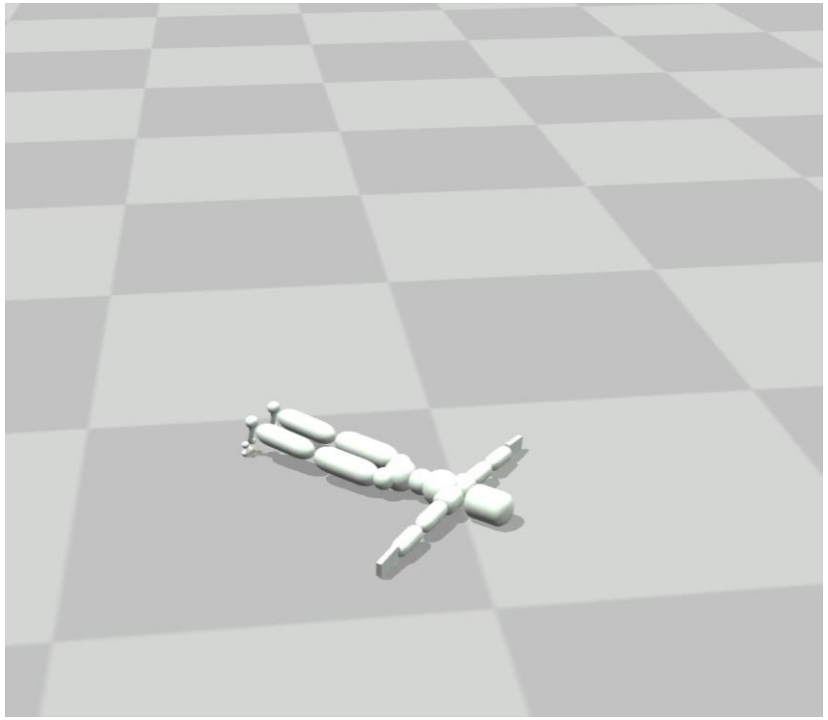
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

joint torque

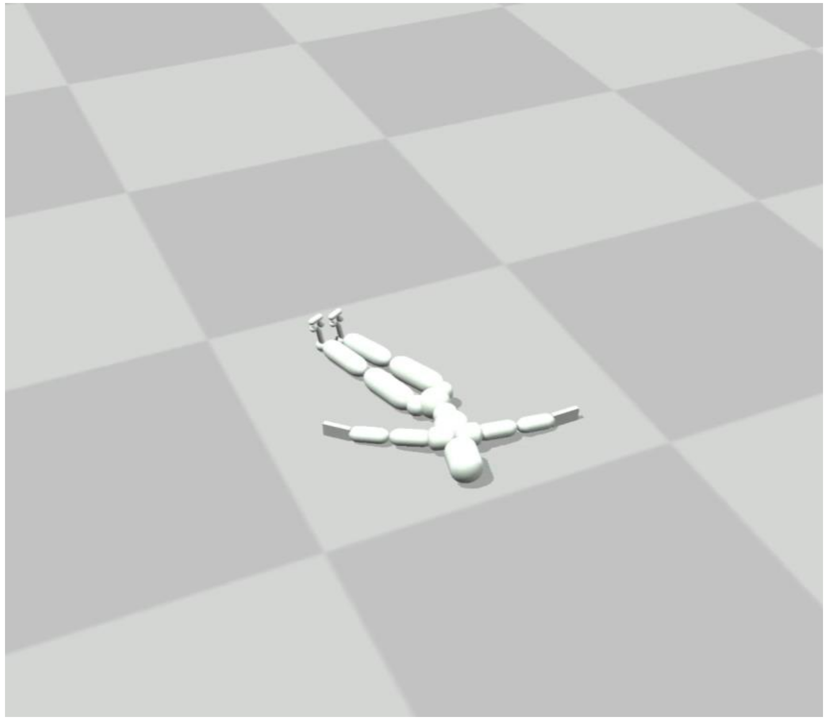
proportional control

derivative control

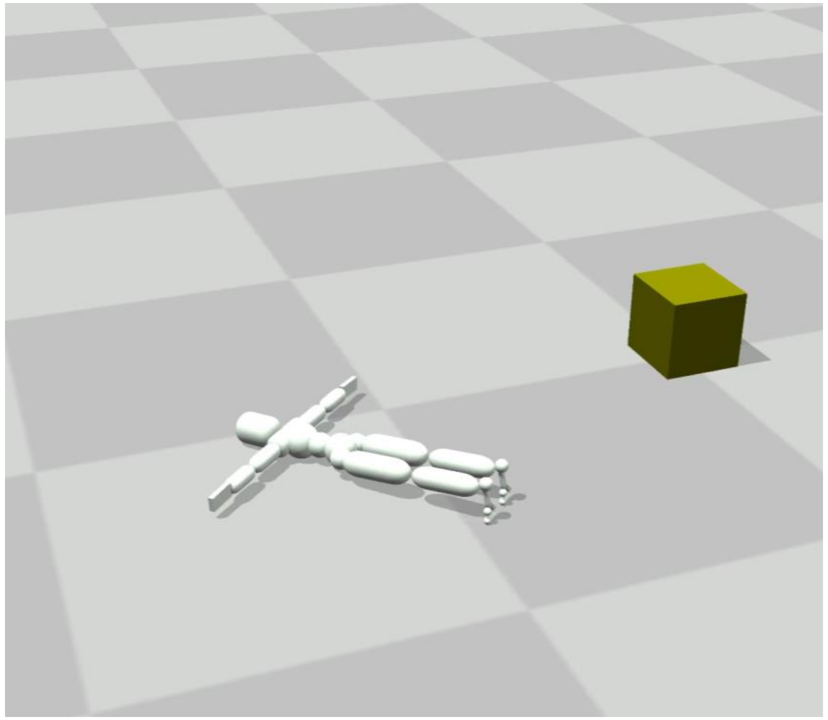
PD Control for Characters



Normal stiffness k_p

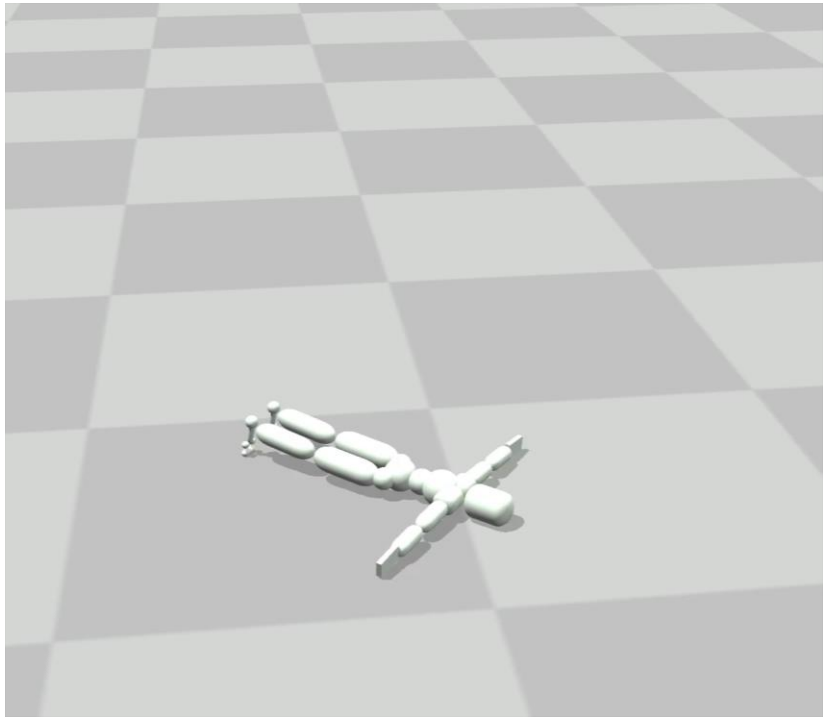


Small stiffness k_p

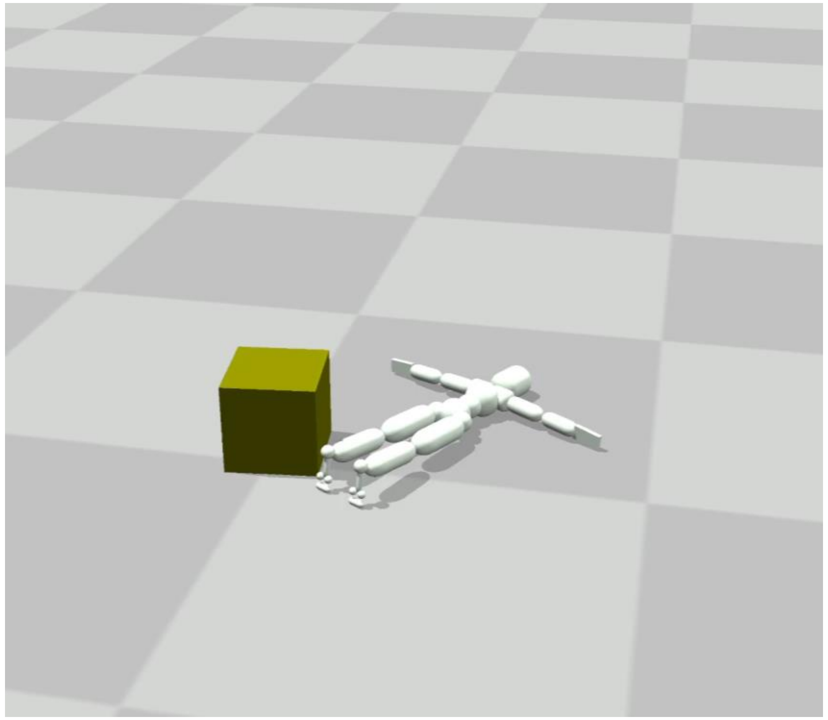


Large stiffness k_p

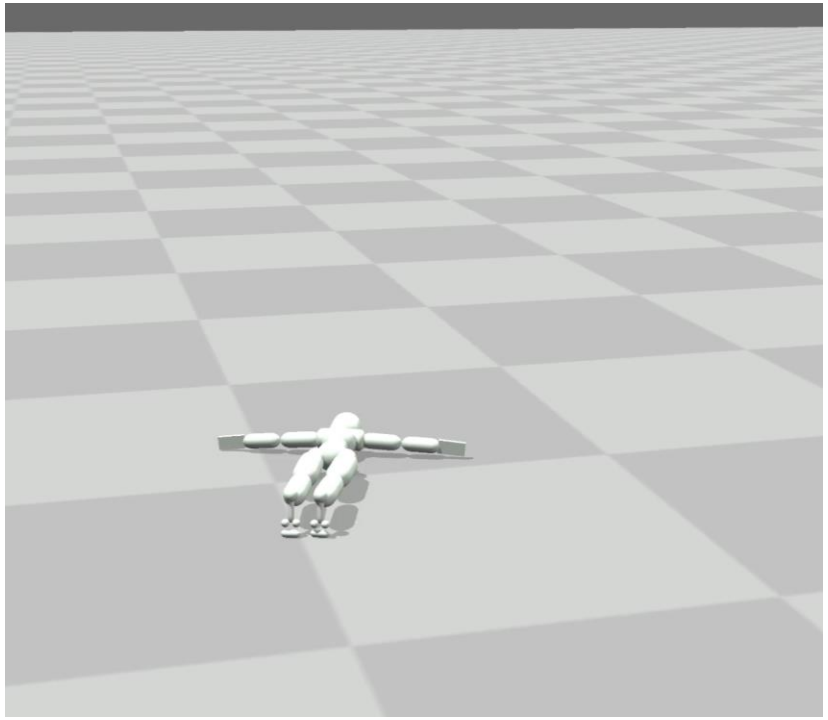
PD Control for Characters



Normal damping k_d

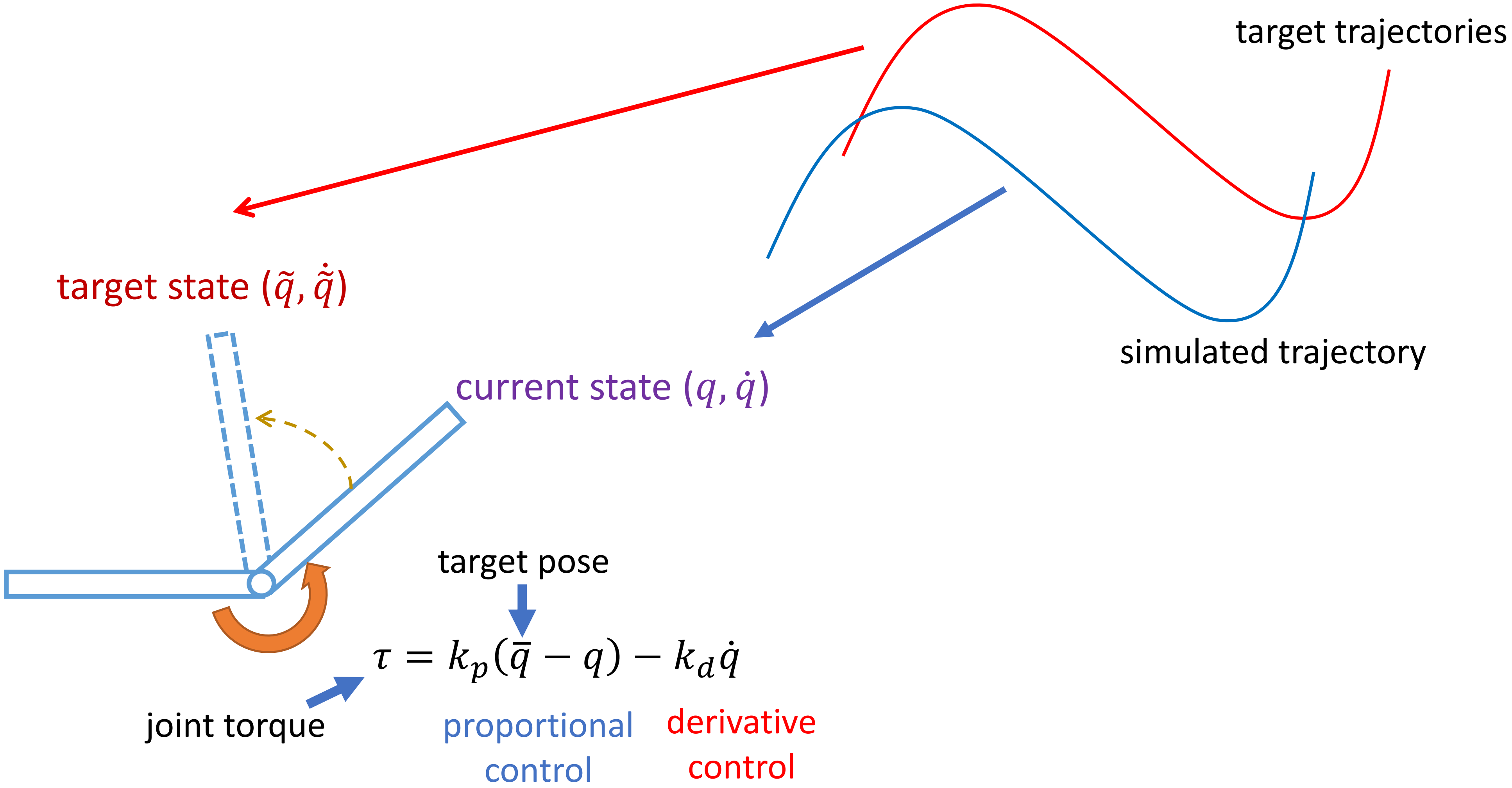


Small damping k_d



Large damping k_d

Tracking Controllers



target trajectories

simulated trajectory

target state $(\tilde{q}, \dot{\tilde{q}})$

current state $(q, \dot{q})$

target pose

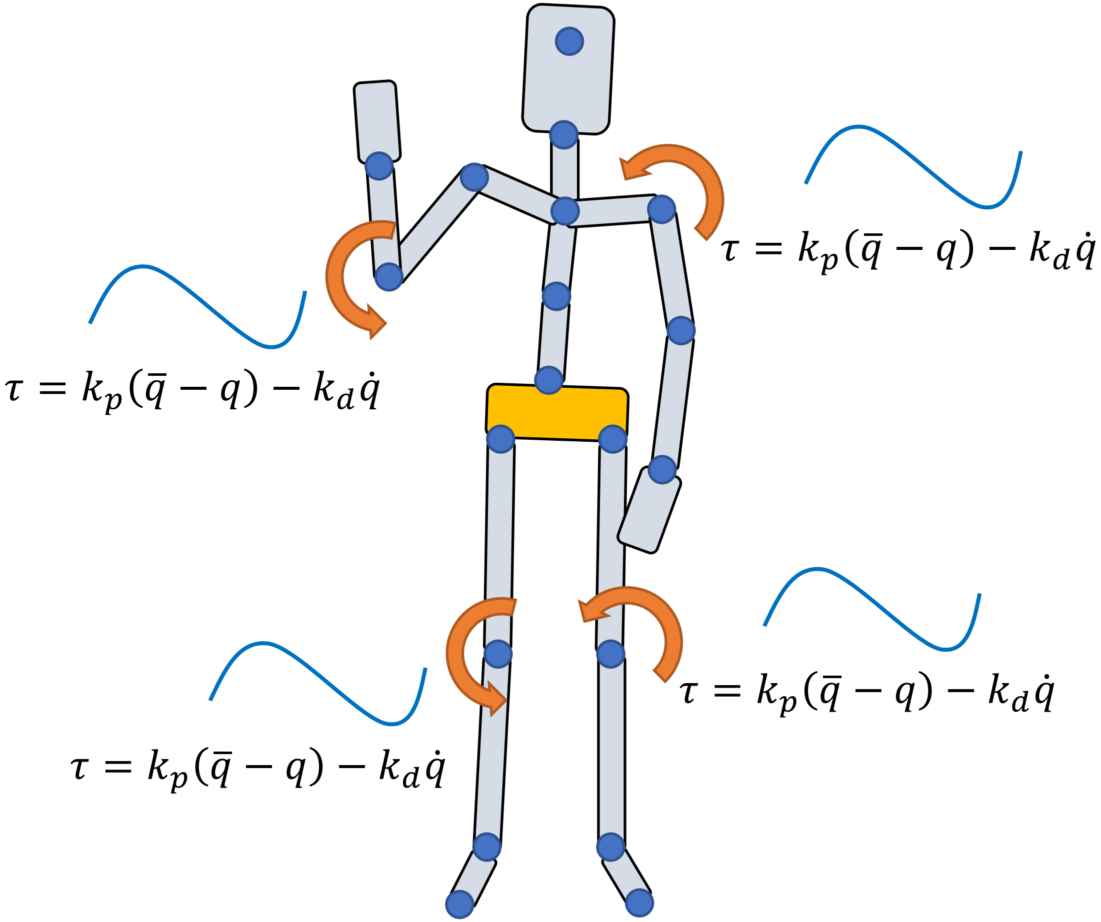
$$\tau = k_p(\bar{q} - q) - k_d\dot{q}$$

joint torque

proportional control

derivative control

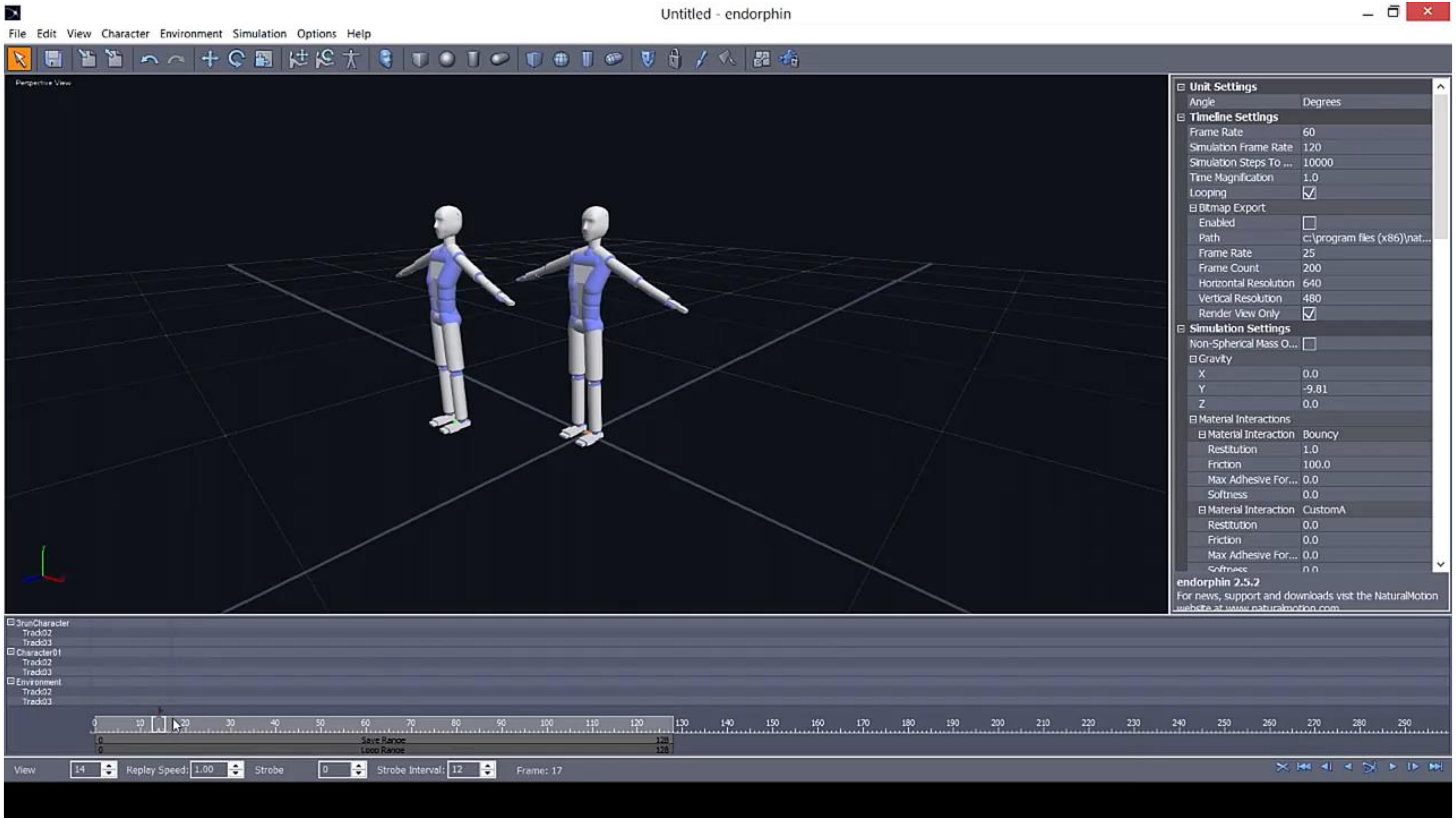
Full-body Tracking Controllers



Tracking a Trajectory

[Hodgins and Wooten 1995,
Animating Human Athletics]

Trajectory Creation



Unit Settings	
Angle	Degrees
Timeline Settings	
Frame Rate	60
Simulation Frame Rate	120
Simulation Steps To ...	10000
Time Magnification	1.0
Looping	☑
Bitmap Export	
Enabled	☐
Path	c:\program files (x86)\nat...
Frame Rate	25
Frame Count	200
Horizontal Resolution	640
Vertical Resolution	480
Render View Only	☑
Simulation Settings	
Non-Spherical Mass O...	☐
Gravity	
X	0.0
Y	-9.81
Z	0.0
Material Interactions	
Material Interaction	Bouncy
Restitution	1.0
Friction	100.0
Max Adhesive For...	0.0
Softness	0.0
Material Interaction	CustomA
Restitution	0.0
Friction	0.0
Max Adhesive For...	0.0
Softness	0.0

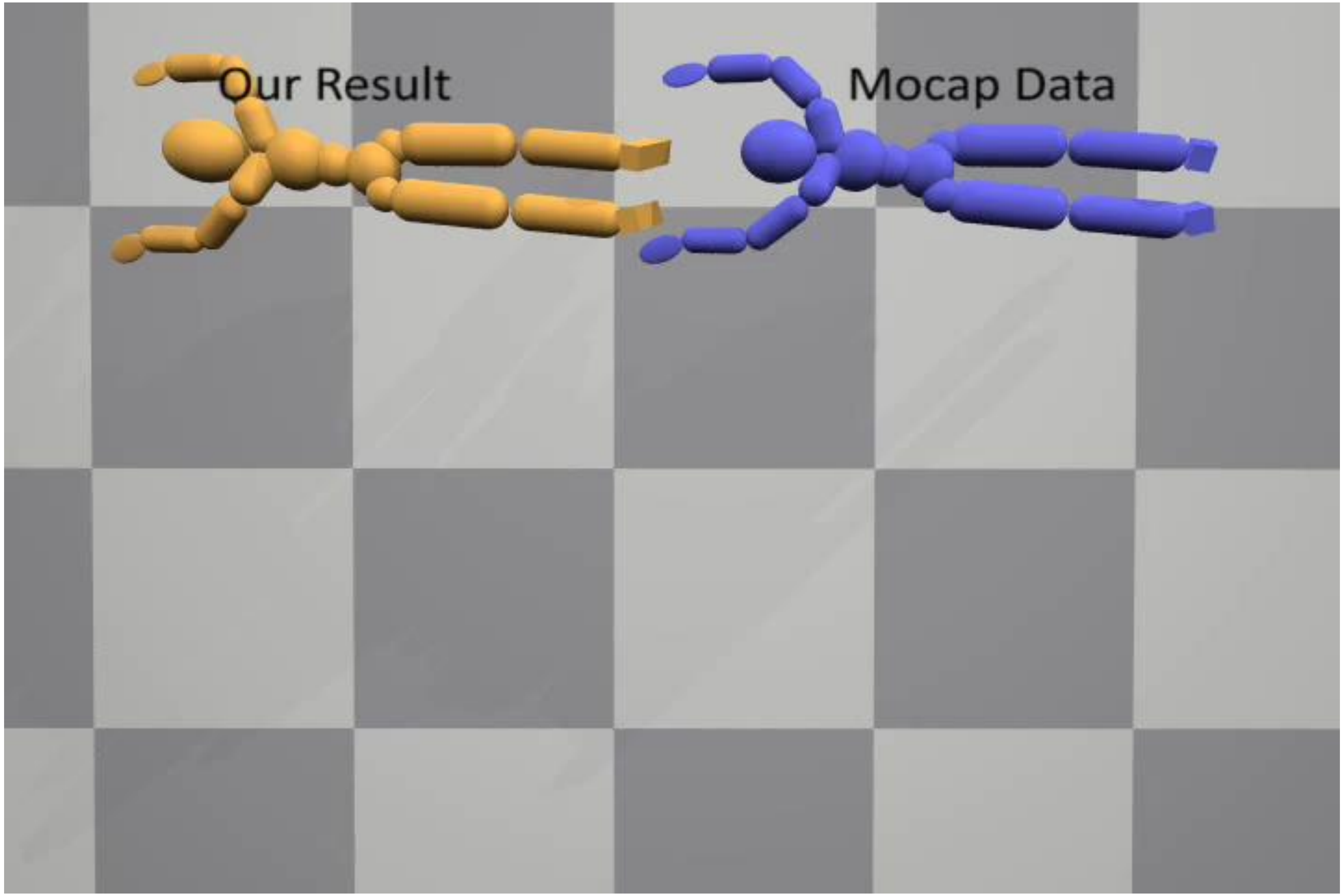
endorphin 2.5.2

For news, support and downloads visit the NaturalMotion website at www.naturalmotion.com

View: 14
Replay Speed: 1.00
Strobe: 0
Strobe Interval: 12
Frame: 17

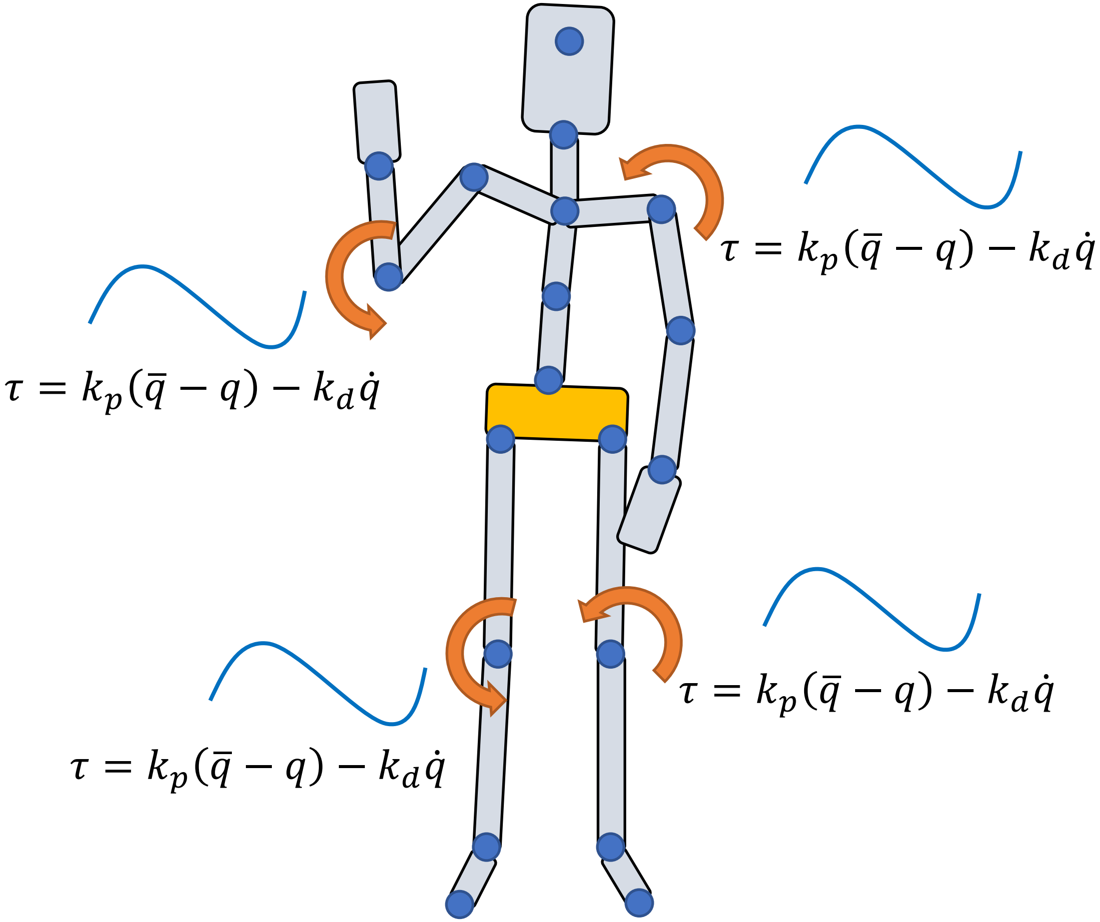
NaturalMotion - Endorphin

Tracking Mocap



[SAMCON – Liu et al 2010]

Full-body Tracking Controllers



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

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

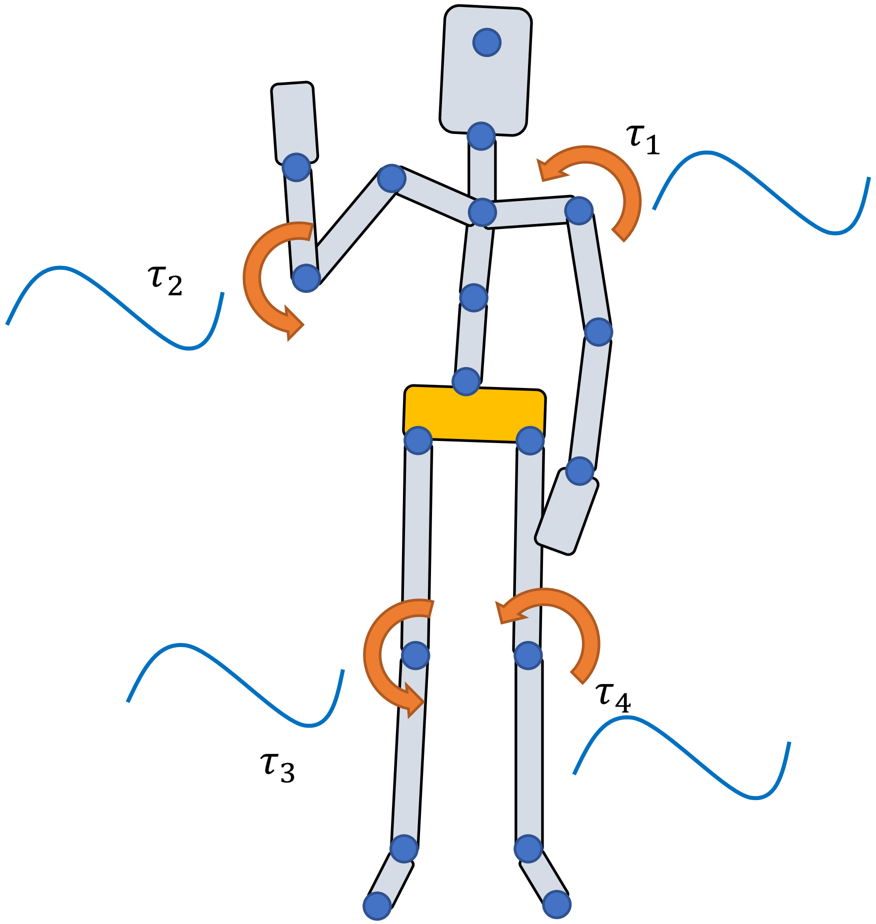
$\tau = k_p(\bar{q} - q) - k_d\dot{q}$

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

Is PD control a feedforward control?

a feedback control?

Tracking Mocap with Joint Torques



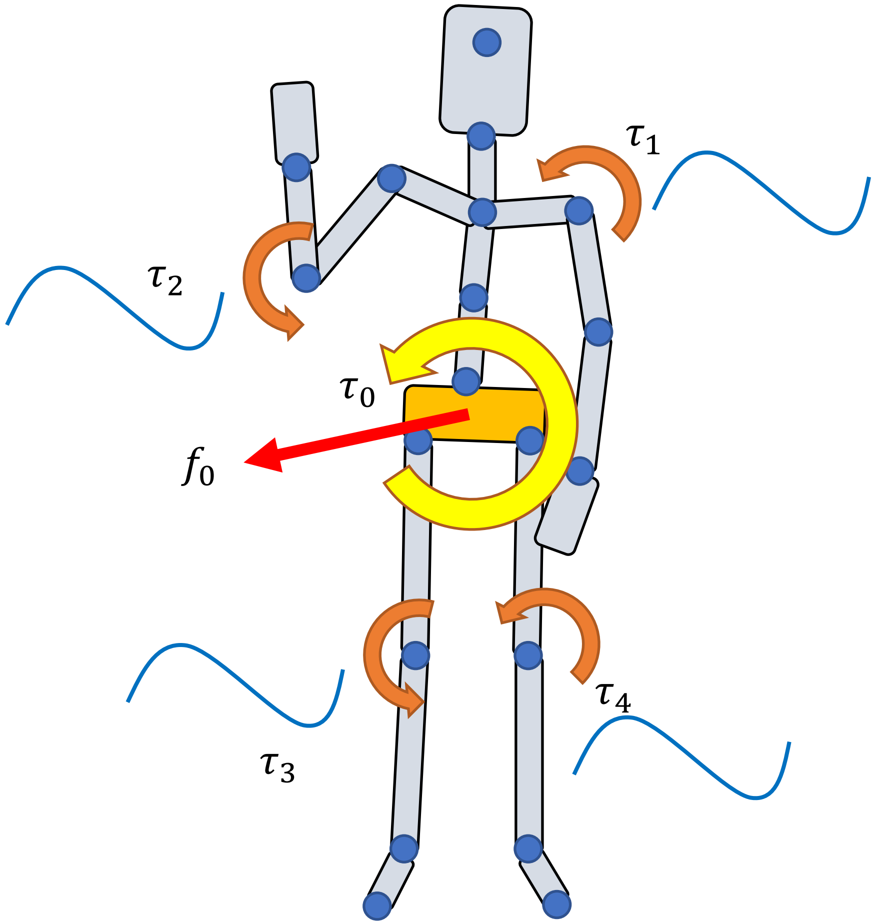
τ_j: joint torques

Apply τ_j to "child" body

Apply $-\tau_j$ to "parent" body

All forces/torques sum up to zero

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!

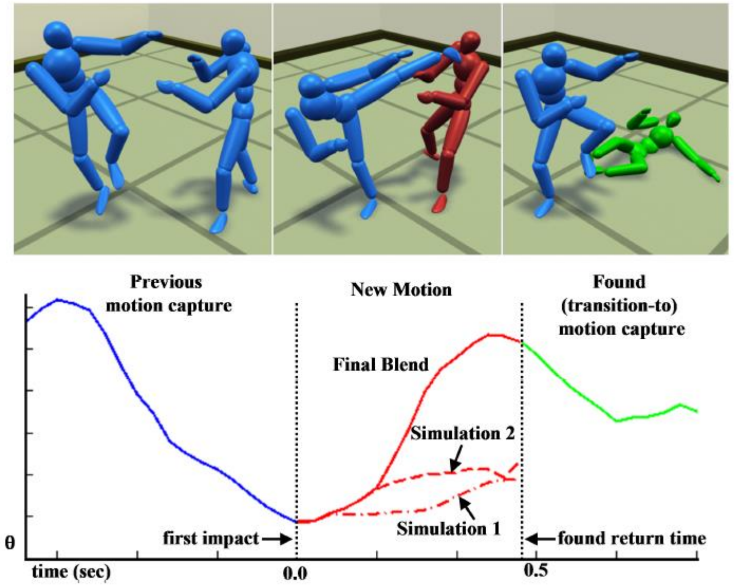
Physically Plausible Animation

Party Animals

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

Mixture Simulation and Mocap

Dynamic Response for Motion Capture Animation



Zordan et al. 2005
Dynamic response for motion capture animation

Questions?