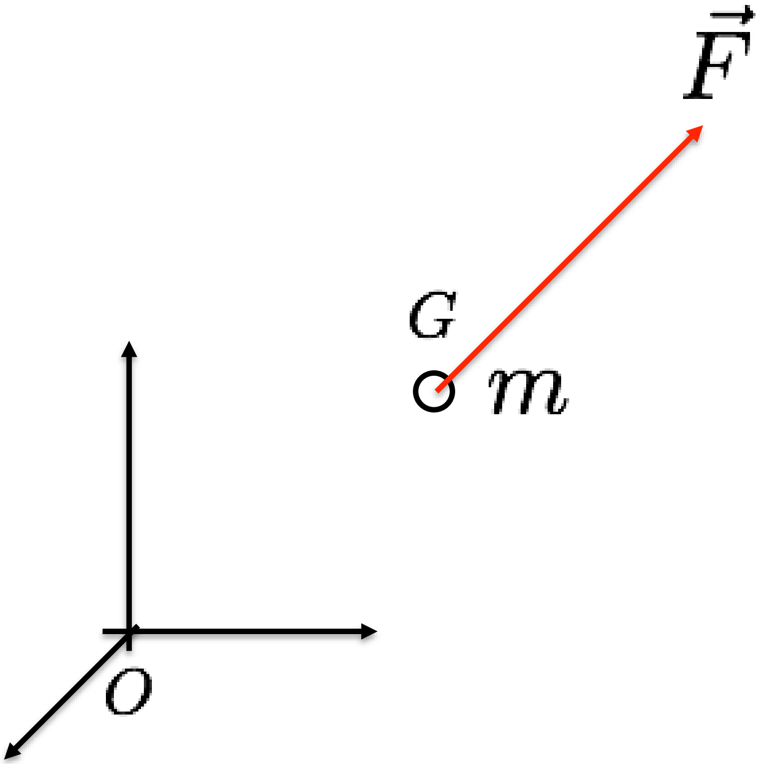
Robot dynamics for whole body control

Gabriele Nava
Stefano Dafarra

iCub Training 2019

Slides taken from VVV18 International Winter School on Humanoid Robot Programming
Credits Daniele Pucci

The point mass equations of motion

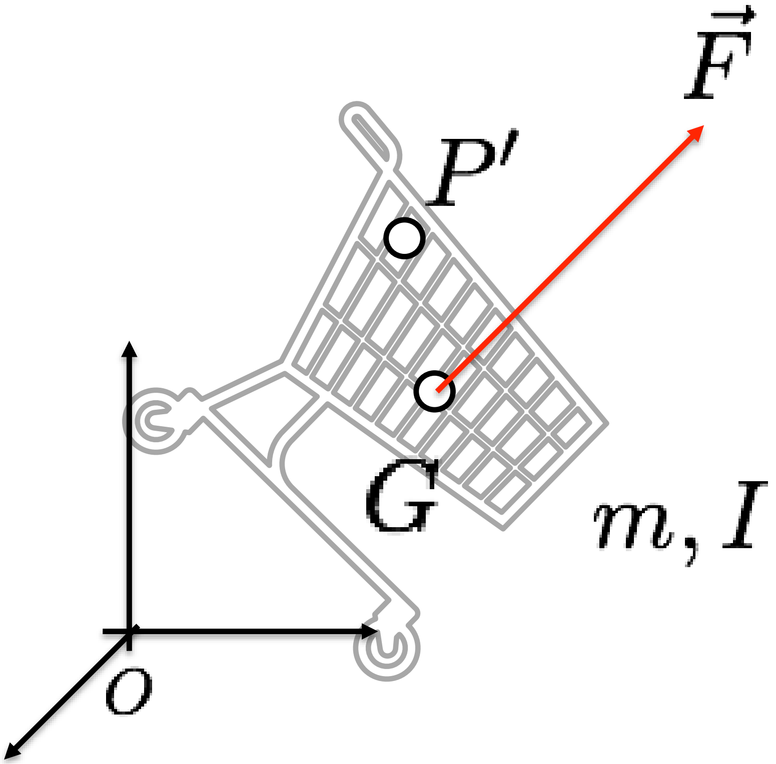


$$m\vec{a} = \vec{F}$$

$$\vec{v} := \frac{d}{dt}\vec{OG}$$

$$\boxed{\frac{d}{dt}(m\vec{v}) = \vec{F}}$$

The rigid body equations of motion 1/2



$$I = -\int_V \rho S(r)^2 dV$$

$$r = P' - G$$

$$\boxed{\frac{d}{dt}\begin{pmatrix} m\vec{v} \\ I\vec{\omega} \end{pmatrix} = \begin{pmatrix} \vec{F} \\ \vec{M} \end{pmatrix}}$$

The rigid body equations of motion 2/2

$$\frac{d}{dt}\begin{pmatrix} m\vec{v} \\ I\vec{\omega} \end{pmatrix} = \begin{pmatrix} \vec{F} \\ \vec{M} \end{pmatrix} \quad \Longrightarrow \quad \frac{d}{dt}\begin{pmatrix} m & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} \vec{v} \\ \vec{\omega} \end{pmatrix} = \begin{pmatrix} \vec{F} \\ \vec{M} \end{pmatrix}$$

$$\frac{d}{dt}\mathbb{M}\nu = \tau \quad \Longrightarrow \quad \mathbb{M}\dot{\nu} + C\nu = \tau$$

$$\boxed{\mathbb{M}\dot{\nu} + C\nu + g = \tau}$$

Forces and torques (tau) do not contain gravity

Robot Dynamics

$\boxed{f = ma \quad \text{for the robot?}}$

Robot Dynamics

$$\boxed{\frac{d}{dt}\frac{\partial}{\partial \dot{q}}\mathcal{L} - \frac{\partial}{\partial q}\mathcal{L} = \tau}$$

- $\mathcal{L} = T - U$: Lagrangian
- $q, \dot{q}, \ddot{q}$: joints' positions, velocities, and accelerations
- τ joint torques

$$\Downarrow$$

$$\boxed{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^\top F_{ext} = \tau}$$

- M, C, g: mass and Coriolis matrices, gravity torques
- F_{ext}, J: **vectorized** external forces and its Jacobian

Robot Dynamics Terminology

$$\boxed{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^\top F_{ext} = \tau}$$

Important Facts

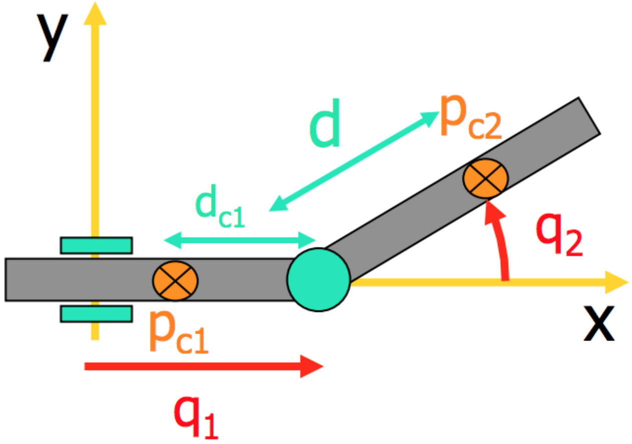
$$\tau = \text{invDyn}(q, \dot{q}, \ddot{q}, F_{ext})$$

Inverse dynamics

$$\ddot{q} = \text{fwDyn}(q, \dot{q}, \tau, F_{ext})$$

Forward dynamics

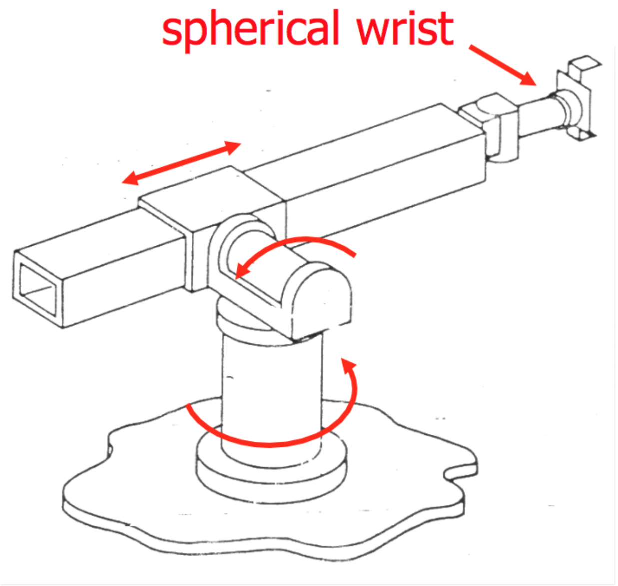
Dynamic Model of PR robot



From http://www.diag.uniroma1.it/~deluca/rob2_en/03_LagrangianDynamics_1.pdf, p23

$$\begin{pmatrix} m_1 + m_2 & -m_2 d \sin(q_2) \\ -m_2 d \sin(q_2) & I_{c_2,zz} + m_2 d^2 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} -m_2 d \cos(q_2) \dot{q}_2^2 \\ 0 \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}$$

Dynamic Model Complexity 1/2



From [http://www.diag.uniroma1.it/~deluca/rob2_en/04_LagrangianDynamics_2.pdf\](http://www.diag.uniroma1.it/~deluca/rob2_en/04_LagrangianDynamics_2.pdf) p2

$$
\begin{aligned}
B_{11} = {} & m_1 k_{122}^2 \\
& + m_2\left[k_{211}^2 s^2\theta_2 + k_{233}^2 c^2\theta_2 + r_2(2\bar{y}_2 + r_2)\right] \\
& + m_3\left[k_{322}^2 s^2\theta_2 + k_{333}^2 c^2\theta_2 + r_3(2\bar{z}_3 + r_3)s^2\theta_2 + r_2^2\right] \\
& + m_4\Big\{\tfrac{1}{2}k_{411}^2\left[s^2\theta_2(2s^2\theta_4 - 1) + s^2\theta_4\right] + \tfrac{1}{2}k_{422}^2(1 + c^2\theta_2 + s^2\theta_4) \\
& \quad + \tfrac{1}{2}k_{433}^2\left[s^2\theta_2(1 - 2s^2\theta_4) - s^2\theta_4\right] + r_3^2 s^2\theta_2 + r_2^2 - 2\bar{y}_4 r_3 s^2\theta_2 + 2\bar{z}_4(r_2 s\theta_4 + r_3 s\theta_2 c\theta_2 c\theta_4)\Big\} \\
& + m_5\Big\{\tfrac{1}{2}(-k_{511}^2 + k_{522}^2 + k_{533}^2)\left[(s\theta_2 s\theta_5 - c\theta_2 s\theta_4 c\theta_5)^2 + c^2\theta_4 c^2\theta_5\right] \\
& \quad + \tfrac{1}{2}(k_{511}^2 - k_{522}^2 + k_{533}^2)(s^2\theta_4 + c^2\theta_2 c^2\theta_4) \\
& \quad + \tfrac{1}{2}(k_{511}^2 + k_{522}^2 - k_{533}^2)\left[(s\theta_2 c\theta_5 + c\theta_2 s\theta_4 s\theta_5)^2 + c^2\theta_4 s^2\theta_5\right] + r_3^2 s^2\theta_2 + r_2^2 \\
& \quad + 2\bar{z}_5\left[r_3(s^2\theta_2 c\theta_5 + s\theta_2 s\theta_4 c\theta_4 s\theta_5) - r_2 c\theta_4 s\theta_5\right]\Big\} \\
& + m_6\Big\{\tfrac{1}{2}(-k_{611}^2 + k_{622}^2 + k_{633}^2)\left[(s\theta_2 s\theta_5 c\theta_6 - c\theta_2 s\theta_4 c\theta_5 c\theta_6 - c\theta_2 c\theta_4 s\theta_6)^2 + (c\theta_4 c\theta_5 c\theta_6 - s\theta_4 s\theta_6)^2\right] \\
& \quad + \tfrac{1}{2}(k_{611}^2 - k_{622}^2 + k_{633}^2)\left[(c\theta_2 s\theta_4 c\theta_5 s\theta_6 - s\theta_2 s\theta_5 s\theta_6 - c\theta_2 c\theta_4 c\theta_6)^2 + (c\theta_4 c\theta_5 s\theta_6 + s\theta_4 c\theta_6)^2\right] \\
& \quad + \tfrac{1}{2}(k_{611}^2 + k_{622}^2 - k_{633}^2)\left[(c\theta_2 s\theta_4 s\theta_5 + s\theta_2 c\theta_5)^2 + c^2\theta_4 s^2\theta_5\right] \\
& \quad + \left[r \,c\theta \,s\theta \,s\theta + (r \,c\theta + r)\,s\theta\right]^2 + (r \,c\theta \,s\theta - r)^2 \ \text{[illegible]}
\end{aligned}
$$

Dynamic Model Complexity 2/2

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

Algorithms to compute dynamic quantities in real time

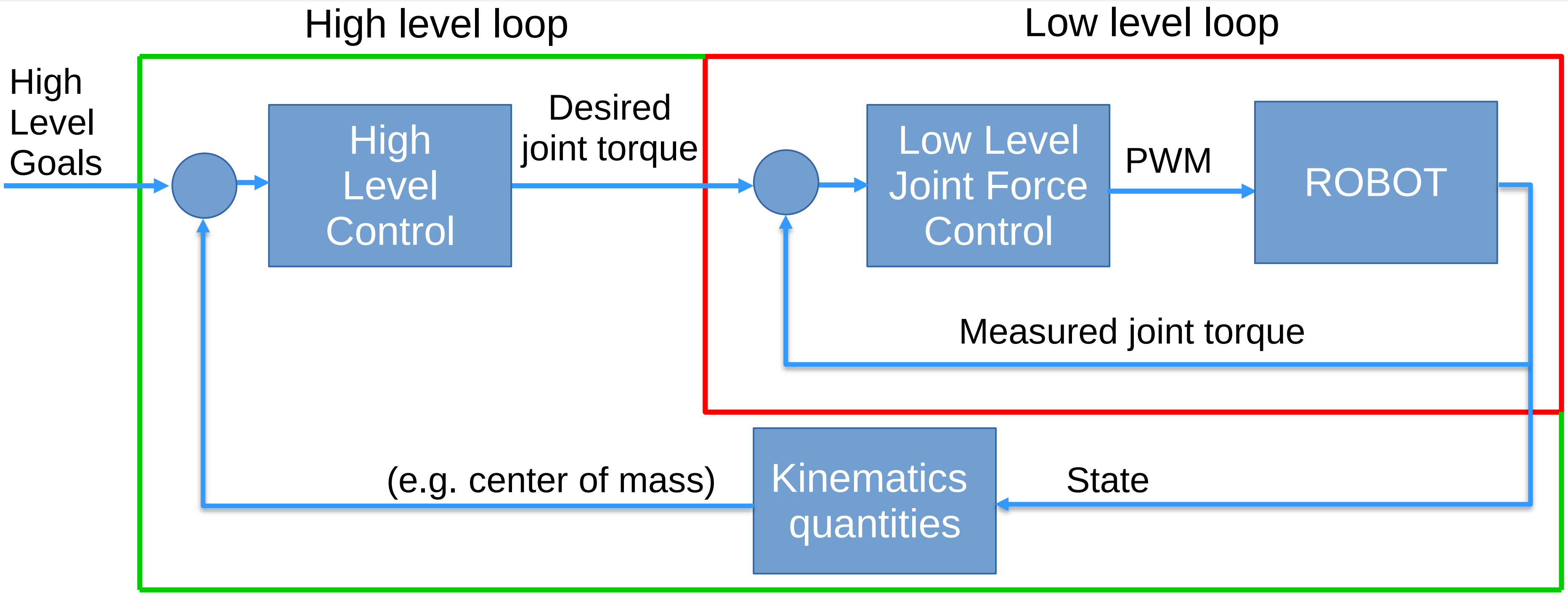
Robot Dynamics

$$\boxed{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau}$$

Joint torques assumed τ as control input

We assume that τ can be chosen at will… in reality

Torque Control Architecture



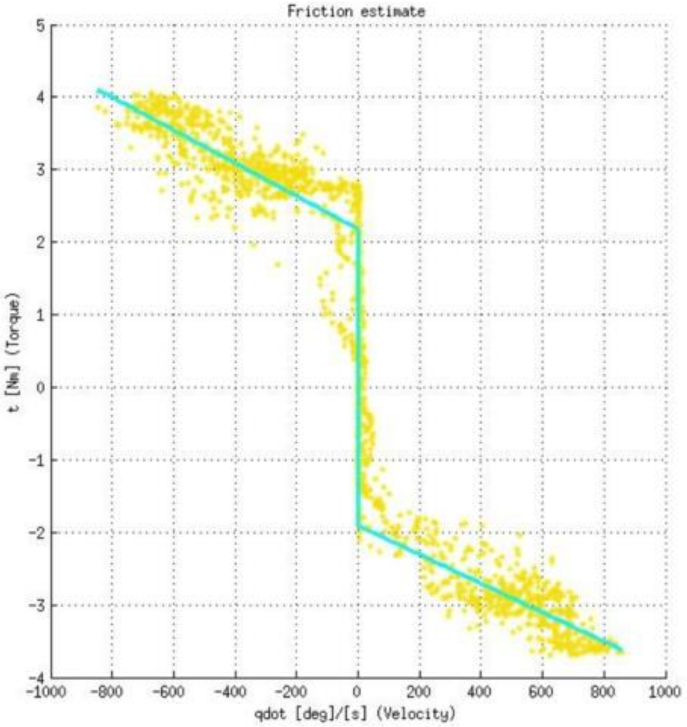
High level loop runs at 100 Hz

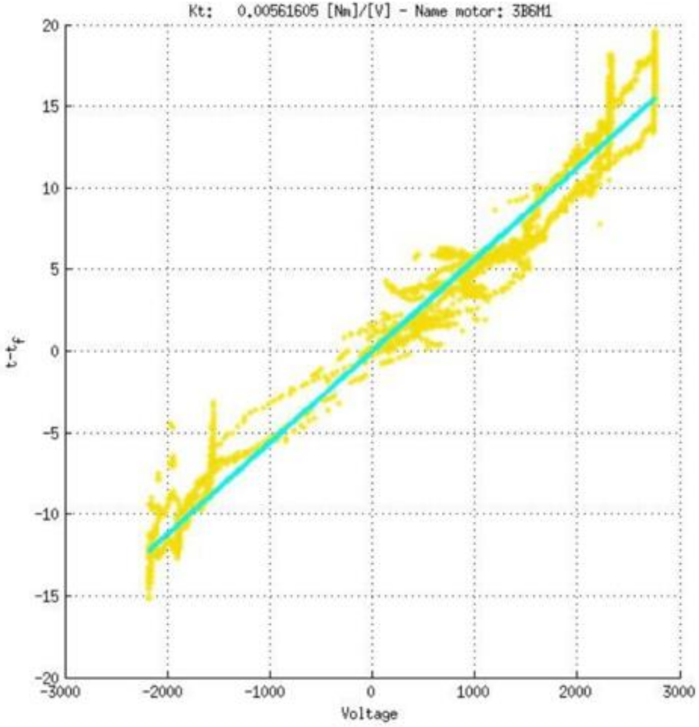
Low level loop runs at 1 kHz

Torque Control Architecture

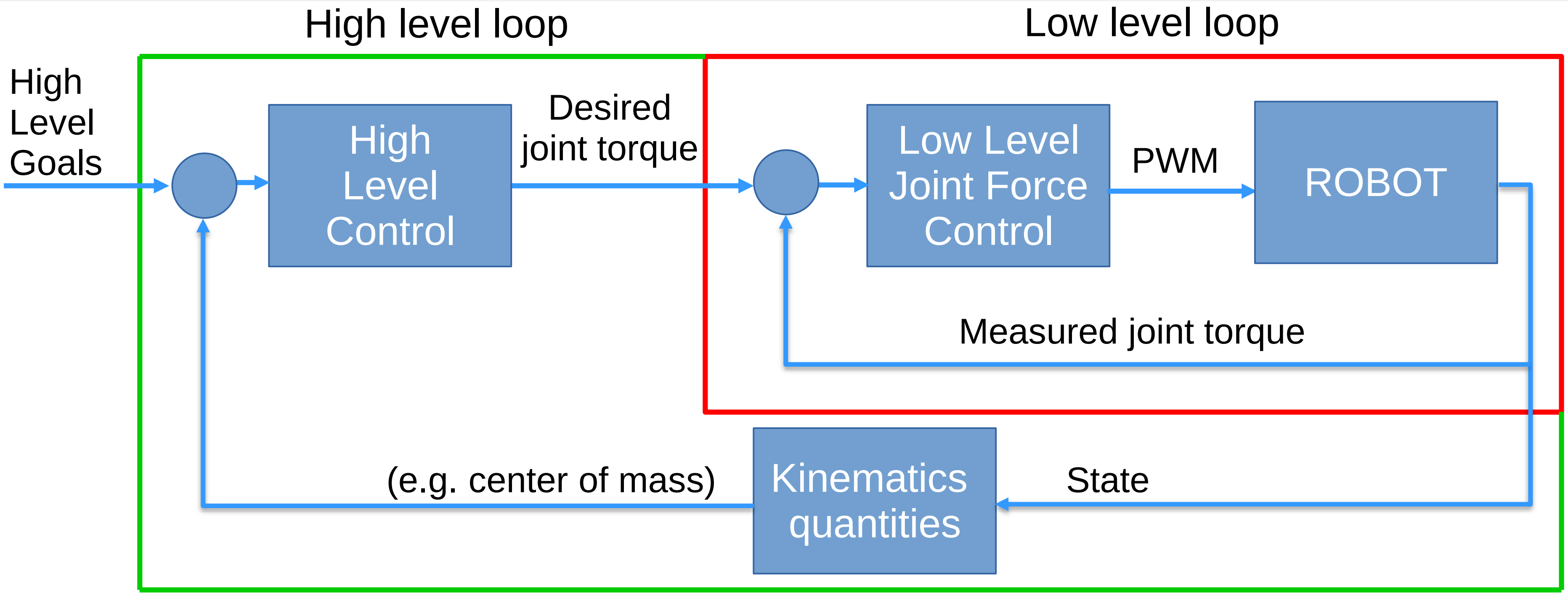
$$\tau = k_\tau PWM - k_v \dot{m} - k_c \text{sign}(\dot{m})$$

$$PWM = \bar{k}_t \tau + \bar{k}_v \dot{m} + \bar{k}_c \text{sign}(\dot{m})$$





Torque Control Architecture



How can we choose the “desired” joint torques?

Control Objective

1) Stabilisation of a desired joint trajectory $q_d(t) \in \mathbb{R}^n$

2) Impose some joint compliance K_p

PD plus gravity compensation control

Joint position error: $q - q_d$

Joint velocity error: $\dot{q} - \dot{q}_d$

Joint torques ensuring stabilisation of joint trajectory:

$$\tau = M(q)\ddot{q}_d - K_p(q - q_d) - K_d(\dot{q} - \dot{q}_d) + C(q, \dot{q})\dot{q}_d + g(q)$$

Joint stiffness: K_p

PD plus gravity compensation control

Fact i):

Stability and convergence of the control law can be proven by using

$$V = \frac{1}{2}(\dot{q} - \dot{q}_d)^\top M(q)(\dot{q} - \dot{q}_d) + \frac{1}{2}(q - q_d)^\top K_p(q - q_d)$$

$$M = M^\top > 0$$
$$\dot{M} - 2C = -(\dot{M} - 2C)^\top$$

$$\dot{V} = -\frac{1}{2}(\dot{q} - \dot{q}_d)^\top K_d(\dot{q} - \dot{q}_d) \leq 0$$

PD plus gravity compensation control

Fact ii):

The control law to stabilise set points, i.e.

$$\dot{q}_d = \ddot{q}_d = 0$$

becomes

$$\tau = g(q) - K_p(q - q_d) - K_d\dot{q}$$

which is simple and does not need Coriolis and mass matrix

Computed torque control law

Joint position error: $q - q_d$

Joint velocity error: $\dot{q} - \dot{q}_d$

Joint torques ensuring stabilisation of joint trajectory:

$$\tau = M(q)\left[\ddot{q}_d - K_p(q - q_d) - K_d(\dot{q} - \dot{q}_d)\right] + C(q, \dot{q})\dot{q} + g(q)$$

Joint stiffness: $M(q)K_p$

Computed torque control law

Fact i):

Stability and convergence of the control law can be proven by using

$$V = \frac{1}{2}(\dot{q} - \dot{q}_d)^\top(\dot{q} - \dot{q}_d) + \frac{1}{2}(q - q_d)^\top K_p (q - q_d)$$

$$\Rightarrow$$

$$\dot{V} = -\frac{1}{2}(\dot{q} - \dot{q}_d)^\top K_d (\dot{q} - \dot{q}_d) \leq 0$$

Computed torque control law

Fact ii):

Given the dynamics $$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

with

$$\tau = M(q)\left[\ddot{q}_d - K_p(q - q_d) - K_d(\dot{q} - \dot{q}_d)\right] + C(q,\dot{q})\dot{q} + g(q)$$

gives

$$\ddot{q} = \ddot{q}_d - K_p(q - q_d) - K_d(\dot{q} - \dot{q}_d)$$

Decoupling

Computed torque control law

<u>Fact iii):</u>

The control law to stabilise set points, i.e.

$$\dot{q}_d = \ddot{q}_d = 0$$

becomes

$$\tau = C(q,\dot{q})\dot{q} + g(q) - M(q)K_p(q - q_d) - M(q)K_d\dot{q}$$

which needs Coriolis and mass matrix

Comparisons of control laws for set points

PD plus gravity compensation:

$$\tau = g(q) - K_p(q - q_d) - K_d\dot{q}$$

Prons

- needs only gravity
- ensures a constant stiffness
- robust

Cons

- does not ensure decoupling

Computed torque

$$\tau = C(q,\dot{q})\dot{q} + g(q) - M(q)K_p(q - q_d) - M(q)K_d\dot{q}$$

Prons

- ensures decoupling

Cons

- does not ensure constant stiffness
- Requires mass matrix and coriolis terms
- less robust

What is the difference?

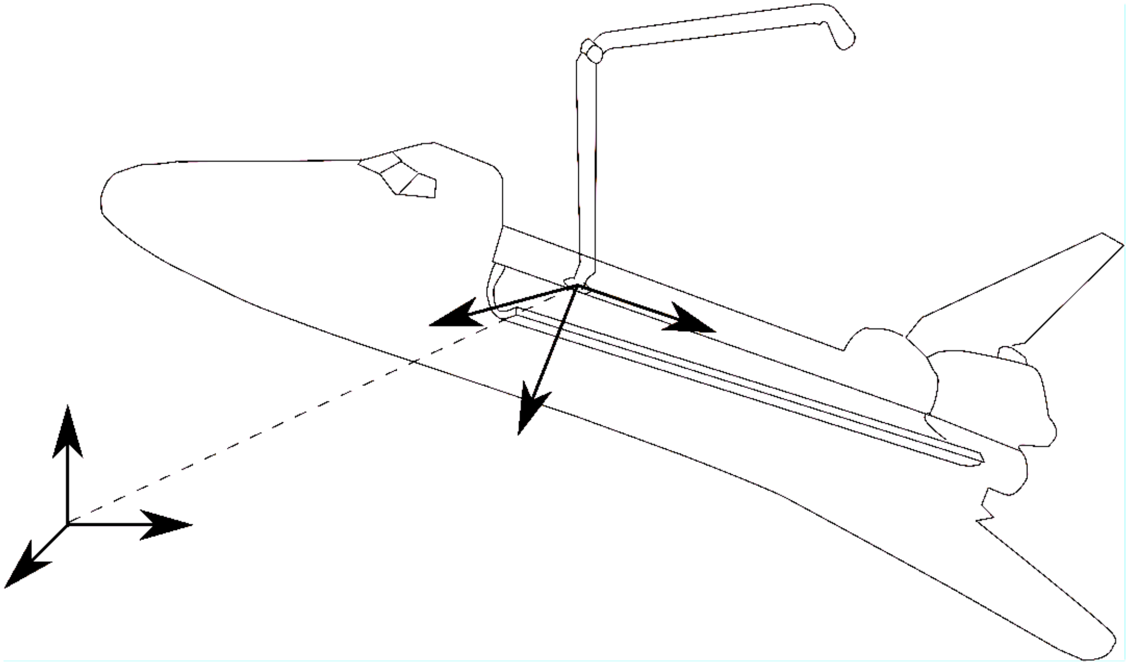
What is the difference?

The system is called **fixed-base** if one of the bodies composing it has a constant pose with respect to the inertial frame

The system is called **floating-base** if none of the bodies composing it has a constant pose with respect to the inertial frame

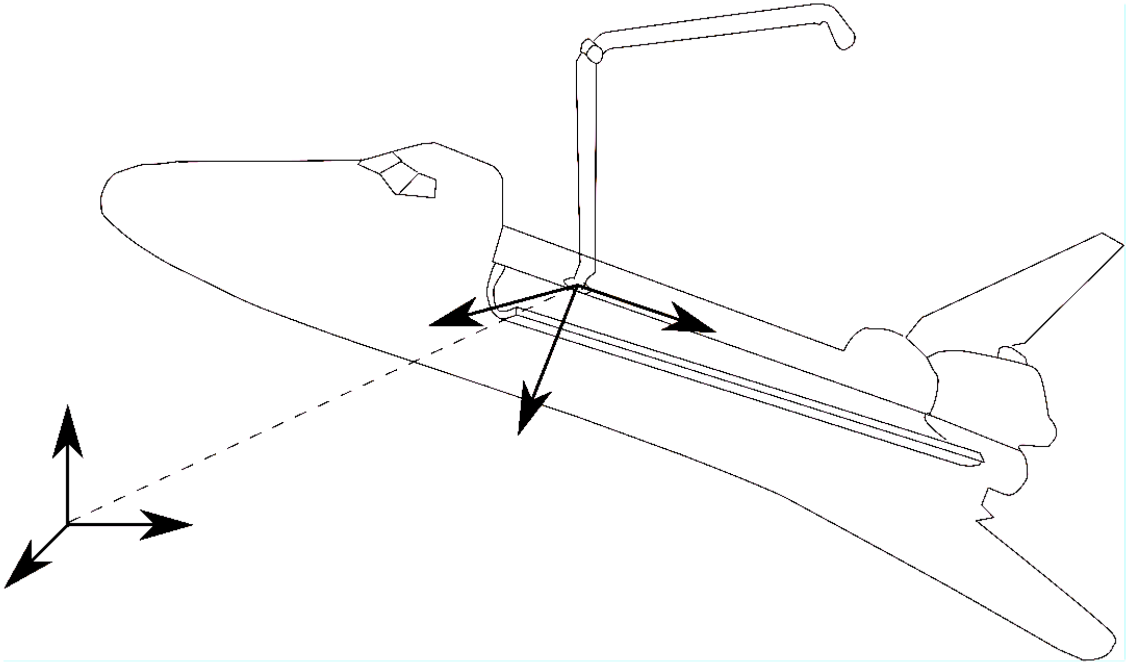
Extension to floating base systems

Pose of the base frame with respect to the inertial frame must be characterised



Extension to floating base systems

Additional six degrees of freedom modelled as additional six fictitious joints



Extension to floating base systems

$$\boxed{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^T F_{ext} = \begin{pmatrix} 0_6 \\ \tau \end{pmatrix}}$$

- $q = \begin{pmatrix} q_b \\ q_j \end{pmatrix} \qquad q_b \in \mathbb{R}^6,\ q_j \in \mathbb{R}^{DOF}$
- q_b: base's position and orientation
- q_j: joint positions
- $\tau \in \mathbb{R}^{DOF}$
- J: Jacobian, F_{ext}: vectorized external forces

Extension to floating base systems

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^T F_{ext} = \begin{pmatrix} 0_6 \\ \tau \end{pmatrix}$$

Problem: $q_b \in \mathbb{R}^6$ is the orientation and position of the base frame

$q_b \in \mathbb{R}^6$ is a local representation of $SE(3) \Rightarrow$

(Forwards) dynamics is not globally defined!

Extension to floating base systems

Configuration space: $\mathbb{Q} = \mathbb{R}^n$

Euler-Lagrange for equations of motion

Configuration space: $\mathbb{Q} = SE(3) \times \mathbb{R}^n$

Euler-Poincaré for equations of motion

Extension to floating base systems

$$\boxed{M(q)\dot{\nu} + C(q,\nu)\nu + g(q) - J^T F_{ext} = \begin{pmatrix} 0_6 \\ \tau \end{pmatrix}}$$

- $q \in SE(3) \times \mathbb{R}^n$, e.g. $q = ({}^wT_b, q_j)$
- $v \in se(3) \times \mathbb{R}^n$, e.g. $v = (v_b, \dot{q}_j)$
- ${}^wT_b = (p_b, {}^w R_b)$: position and rotation matrix of the base
- q_j: joint positions
- $v_b = (\dot{x}_b, \omega_b)$: linear and angular velocity of base frame