CZECH TECHNICAL UNIVERSITY IN PRAGUE

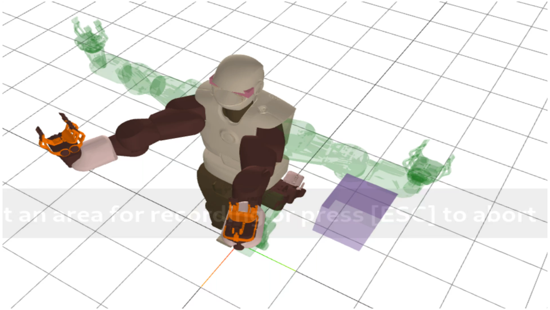
Robotics: Dynamics of open chain

Vladimír Petrík

vladimir.petrik@cvut.cz

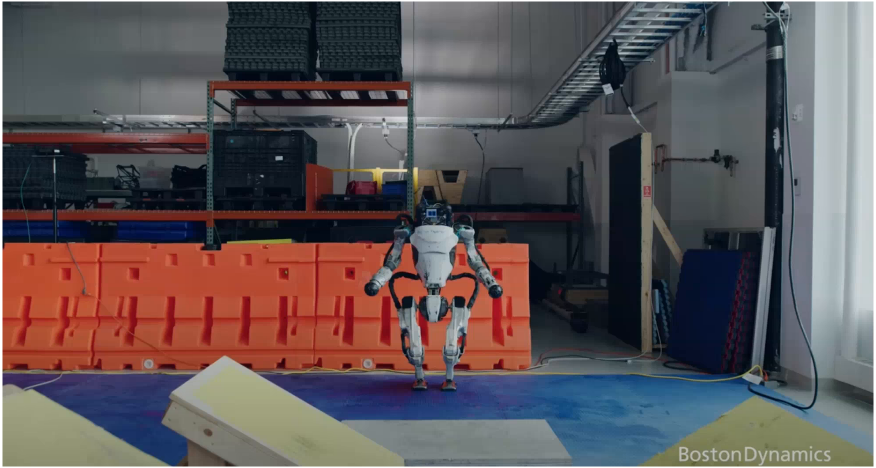
25.11.2024

Motivation



- We studied kinematics of open chains
  - Forward kinematics
  - Inverse kinematics
  - Planning of paths/trajectories
- Dynamics of open chains
  - Motion of the robot taking into account forces, torques, and gravity
  - Motion described by the equation of motion
  - Can be used to compute control of the robot
  - It can answer the question when humanoid robot falls down

Motivation



Equation of motion

- Describes the motion of the robot
- Differential equation of the second order
- For robotics, equation of motion has the form $\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + h(\boldsymbol{q}, \dot{\boldsymbol{q}})$
  - $\boldsymbol{\tau}$ - vector of joint forces/torques
  - M - mass matrix
  - h - vector of Coriolis, gravity and friction terms
  - h is often in the form $h = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q})$
    - C - Coriolis matrix
    - g - effect of gravity

Dynamics tasks

- Forward dynamics
  - Given $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\boldsymbol{\tau}$ compute $\ddot{\boldsymbol{q}}$
  - Why we need it?
  - Used for simulation
  - How the robot moves for given forces/torques
  - $\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q})(\boldsymbol{\tau} - h(\boldsymbol{q}, \dot{\boldsymbol{q}}))$
- Inverse dynamics
  - Given $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ compute $\boldsymbol{\tau}$
  - Why we need it?
  - Used for control
  - What forces/torques are needed to move the robot in desired way
  - $\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + h(\boldsymbol{q}, \dot{\boldsymbol{q}})$

Forward dynamics integration - simulation

- Explicit Euler Integration
- $\dot{\boldsymbol{q}}_{t+1} = \dot{\boldsymbol{q}}_t + \ddot{\boldsymbol{q}}_t \Delta t$
  - $\ddot{\boldsymbol{q}}_{\boldsymbol{t}} = M^{-1}(\boldsymbol{q}_t)(\boldsymbol{\tau}_t - h(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_{\boldsymbol{t}}))$
  - Δt - time step, e.g. 0.001 s (unstable for large time steps)
- $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \dot{\boldsymbol{q}}_{t+1} \Delta t$

$\boldsymbol{\tau} = \begin{pmatrix} 0 & 0 \end{pmatrix}^\top$

$\boldsymbol{\tau} = \begin{pmatrix} 1 & 1 \end{pmatrix}^\top$

Equation of motion derivation

- Lagrangian formulation
  - Kinetic energy
  - Potential energy
  - Elegant for simple structures
- Newton-Euler formulation
  - Dynamic equation of rigid body
  - Efficient recursive formulation for forward/inverse dynamics
- Both formulations lead to the same equation of motion

Lagrangian formulation

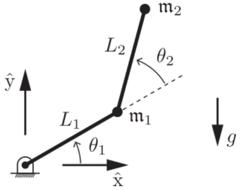
- Generalized coordinates $\boldsymbol{q}$
- Generalized forces $\boldsymbol{\tau}$
- Lagrangian
  $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{P}(\boldsymbol{q})$
  - Kinetic energy $\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}})$
  - Potential energy $\mathcal{P}(\boldsymbol{q})$
- Equation of motion
  $\boldsymbol{\tau} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}$
- Also called Euler-Lagrange equation with external forces
- Examples:
  - Particle of mass moving vertically in gravitation field
  - Planar robot arm

Simulation of PP

$$\boldsymbol{\tau} = \begin{pmatrix} 0 & 0 \end{pmatrix}^\top$$

$$\boldsymbol{\tau} = \begin{pmatrix} 0 & -100 y_G \end{pmatrix}^\top$$

Equation of Motion - RR



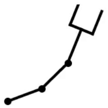
$$
\begin{aligned}
\tau_1 &= \left(\mathfrak{m}_1 L_1^2 + \mathfrak{m}_2(L_1^2 + 2L_1L_2\cos\theta_2 + L_2^2)\right)\ddot{\theta}_1 \\
&\quad + \mathfrak{m}_2(L_1L_2\cos\theta_2 + L_2^2)\ddot{\theta}_2 - \mathfrak{m}_2 L_1L_2\sin\theta_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \\
&\quad + (\mathfrak{m}_1 + \mathfrak{m}_2)L_1 g\cos\theta_1 + \mathfrak{m}_2 g L_2\cos(\theta_1 + \theta_2), \\
\tau_2 &= \mathfrak{m}_2(L_1L_2\cos\theta_2 + L_2^2)\ddot{\theta}_1 + \mathfrak{m}_2 L_2^2\ddot{\theta}_2 + \mathfrak{m}_2 L_1L_2\dot{\theta}_1^2\sin\theta_2 \\
&\quad + \mathfrak{m}_2 g L_2\cos(\theta_1 + \theta_2).
\end{aligned}
$$

$$
M(\theta) = \begin{bmatrix} \mathfrak{m}_1 L_1^2 + \mathfrak{m}_2(L_1^2 + 2L_1L_2\cos\theta_2 + L_2^2) & \mathfrak{m}_2(L_1L_2\cos\theta_2 + L_2^2) \\ \mathfrak{m}_2(L_1L_2\cos\theta_2 + L_2^2) & \mathfrak{m}_2 L_2^2 \end{bmatrix},
$$

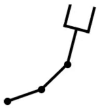
$$
c(\theta, \dot{\theta}) = \begin{bmatrix} -\mathfrak{m}_2 L_1L_2\sin\theta_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \\ \mathfrak{m}_2 L_1L_2\dot{\theta}_1^2\sin\theta_2 \end{bmatrix},
$$

$$
g(\theta) = \begin{bmatrix} (\mathfrak{m}_1 + \mathfrak{m}_2)L_1 g\cos\theta_1 + \mathfrak{m}_2 g L_2\cos(\theta_1 + \theta_2) \\ \mathfrak{m}_2 g L_2\cos(\theta_1 + \theta_2) \end{bmatrix},
$$

Simulation of RRR

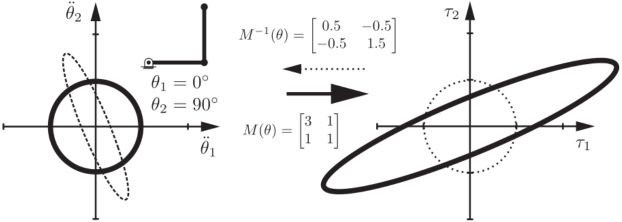


$$\boldsymbol{\tau} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^\top$$



$$\boldsymbol{\tau} = \begin{pmatrix} 10 & 10 & 10 \end{pmatrix}^\top$$

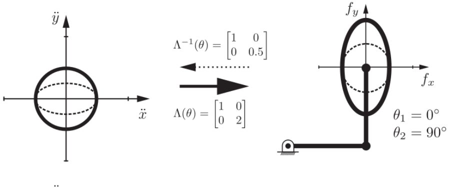
Understanding mass matrix

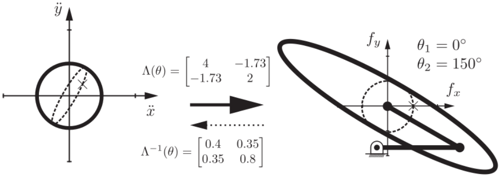


- Kinetic energy
  - Point mass $\frac{1}{2}m\dot{x}^2$
  - Robot $\frac{1}{2}\dot{\boldsymbol{q}}^\top M(\boldsymbol{q})\dot{\boldsymbol{q}}$
- Mass
  - Point mass m is positive
  - $M(\boldsymbol{q})$ is symmetric positive definite matrix
- Point mass in Cartesian coordinates
  - Independent of direction of acceleration
  - Acceleration is scalar multiplication of force
- Mass matrix in generalized coordinates
  - Effective mass depends on the acceleration direction
  - Unit acceleration mapping to torques
  - The same magnitude of acceleration can be achieved by different torques (depending on the direction)

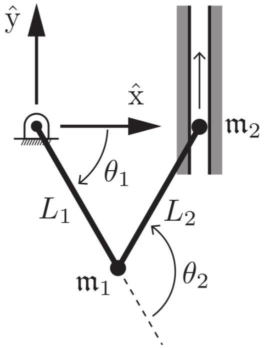
End-effector effective mass

- How massy would end-effector feel if we move it by hand? Depends on the direction of force.
  - Kinetic energy must be constant: $\frac{1}{2}V^\top \Lambda(\boldsymbol{q})V = \frac{1}{2}\dot{\boldsymbol{q}}^\top M(\boldsymbol{q})\dot{\boldsymbol{q}}$
    - $\Lambda(\boldsymbol{q})$ effective mass of end-effector
    - $V = (\dot{x}, \dot{y})^\top$ velocity of end-effector
  - Jacobian $V = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
  - $V^\top \Lambda(\boldsymbol{q})V = (J^{-1}V)^\top M(\boldsymbol{q})(J^{-1}V)^\top = V^\top(J^{-\top}M(\boldsymbol{q})J^{-1})V$
  - End-effector mass matrix: $\Lambda(\boldsymbol{q}) = J^{-\top}(\boldsymbol{q})M(\boldsymbol{q})J^{-1}(\boldsymbol{q})$



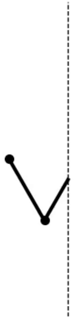


Constrained dynamics



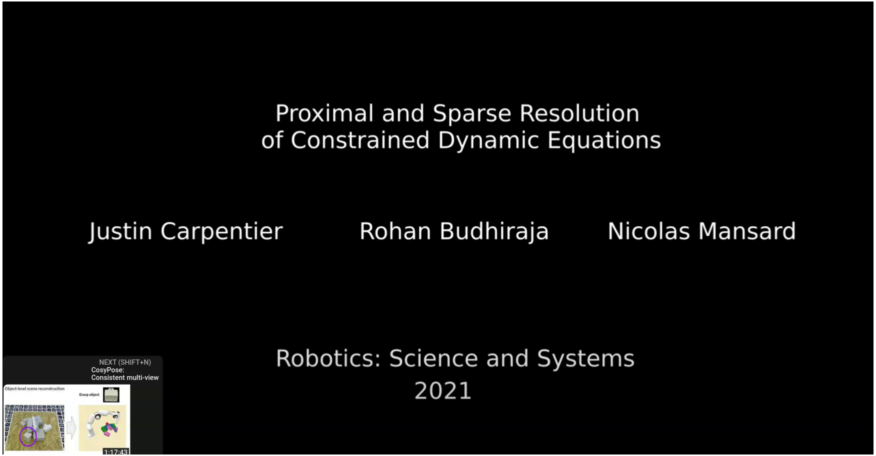
- Robot subject to a set of k velocity constraints
  - e.g. closed kinematics chain
  - writing with a pen (constant height)
  - $A(\boldsymbol{q})\dot{\boldsymbol{q}} = 0, A \in \mathbb{R}^{k\times n}$
- Equation of motion
  - $\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + h(\boldsymbol{q}, \dot{\boldsymbol{q}}) + A^\top(\boldsymbol{q})\boldsymbol{\lambda}, \quad \text{s.t. } A(\boldsymbol{q})\dot{\boldsymbol{q}} = 0$
  - $\boldsymbol{\lambda}$ - vector of Lagrange multipliers
  - $A^\top(\boldsymbol{q})\boldsymbol{\lambda}$ - force applied against constraints expressed as joint forces/torques
  - Lambda can be computed analytically:
    $\boldsymbol{\lambda} = (AM^{-1}A^\top)^{-1}(AM^{-1}(\boldsymbol{\tau} - h) + \dot{A}\dot{\boldsymbol{q}})$

Constrained dynamics tasks



- Forward dynamics
  - first compute $\boldsymbol{\lambda}$
  - compute $\ddot{\boldsymbol{q}}$
- Inverse dynamics
  - compute $\boldsymbol{\tau}$ from given $\boldsymbol{\lambda}$ and $\ddot{\boldsymbol{q}}$
  - $\boldsymbol{\lambda}$ defines force against constraints
    - if constraint is in the end-effector space: $J^\top \boldsymbol{f} = A^\top \boldsymbol{\lambda}$
    - e.g. how much pushing against the table with $\boldsymbol{f}_d$
    - $\boldsymbol{\lambda} = (J^{-\top} A^\top)^\dagger \boldsymbol{f}_d$

Use of constrained dynamics



Summary

- Dynamics of open chains
- Equation of motion
  - Lagrangian formulation
  - Newton-Euler formulation
- Forward dynamics
- Inverse dynamics
- Constrained dynamics