MEE5114 Advanced Control for Robotics

Lecture 9: Dynamics of Open Chains

Prof. Wei Zhang

CLEAR Lab
Department of Mechanical and Energy Engineering
Southern University of Science and Technology, Shenzhen, China
https://www.wzhanglab.site/

Outline

- Introduction
- Inverse Dynamics: Recursive Newton-Euler Algorithm (RNEA)
- Analytical Form of the Dynamics Model
- Forward Dynamics Algorithms

From Single Rigid Body to Open Chains

- Recall Newton-Euler Equation for a single rigid body:

$$\mathcal{F} = \frac{d}{dt} h = \mathcal{I}\mathcal{A} + \mathcal{V} \times^* \mathcal{I}\mathcal{V}$$

- Open chains consist of multiple rigid links connected through joints

- Dynamics of adjacent links are coupled.

- This lecture: model multi-body dynamics subject to joint constraints.

Preview of Open-Chain Dynamics

- Equations of Motion are a set of 2nd-order differential equations:

$$\tau = M(\theta)\ddot{\theta} + \tilde{c}(\theta, \dot{\theta})$$

  - $\theta \in \mathbb{R}^n$: vector of joint variables; $\tau \in \mathbb{R}^n$: vector of joint forces/torques
  - $M(\theta) \in \mathbb{R}^{n \times n}$: mass matrix
  - $\tilde{c}(\theta, \dot{\theta}) \in \mathbb{R}^n$: forces that lump together centripetal, Coriolis, gravity, friction terms, and torques induced by external forces. These terms depend on θ and/or $\dot{\theta}$

- **Forward dynamics:** Determine acceleration $\ddot{\theta}$ given the state $(\theta, \dot{\theta})$ and the joint forces/torques:

$$\ddot{\theta} \leftarrow \mathsf{FD}(\tau, \theta, \dot{\theta}, \mathcal{F}_{ext})$$

- **Inverse dynamics:** Finding torques/forces given state $(\theta, \dot{\theta})$ and desired acceleration $\ddot{\theta}$

$$\tau \leftarrow \mathsf{ID}(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext})$$

Lagrangian vs. Newton-Euler Methods

- There are typically two ways to derive the equation of motion for an open-chain robot: Lagrangian method and Newton-Euler method

Lagrangian Formulation

- Energy-based method
- Dynamic equations in closed form
- Often used for study of dynamic properties and analysis of control methods

Newton-Euler Formulation

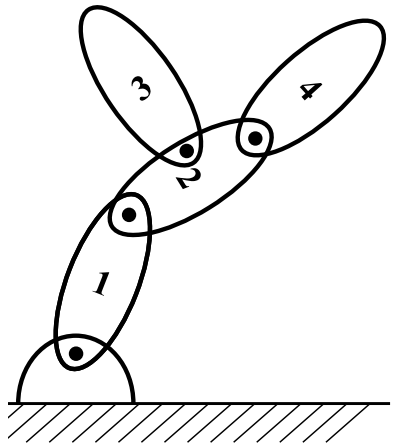
- Balance of forces/torques
- Dynamic equations in numeric/recursive form
- Often used for numerical solution of forward/inverse dynamics

- We focus on Newton-Euler Formulation

Outline

- Introduction
- Inverse Dynamics: Recursive Newton-Euler Algorithm (RNEA)
- Analytical Form of the Dynamics Model
- Forward Dynamics Algorithms

RNEA: Notations

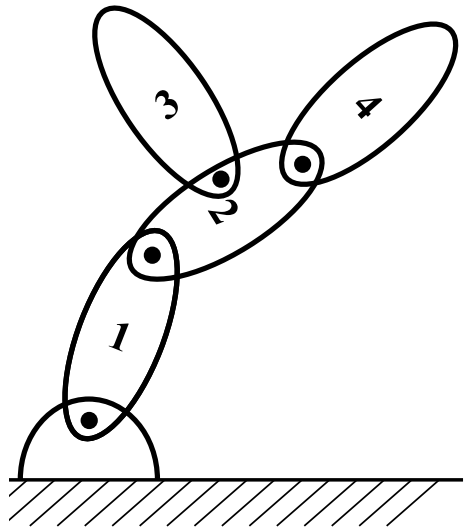


- Number bodies: 1 to N
  - Parent: $p(i)$
  - Children: $c(i)$
- Joint i connects $p(i)$ to i
- Frame $\{\text{i}\}$ attached to body i
- $\mathcal{S}_i$: Spatial velocity (screw axis) of joint i
- $\mathcal{V}_i$ and $\mathcal{A}_i$: spatial velocity and acceleration of body i
- $\mathcal{F}_i$: force (wrench) onto body i from body $p(i)$
- Note: By default, all vectors $(\mathcal{S}_i, \mathcal{V}_i, \mathcal{F}_i)$ are expressed in local frame $\{\text{i}\}$

RNEA: Velocity and Accel. Propagation (Forward Pass)

Goal: Given joint velocity $\dot{\theta}$ and acceleration $\ddot{\theta}$, compute the body spatial velocity $\mathcal{V}_i$ and spatial acceleration $\mathcal{A}_i$

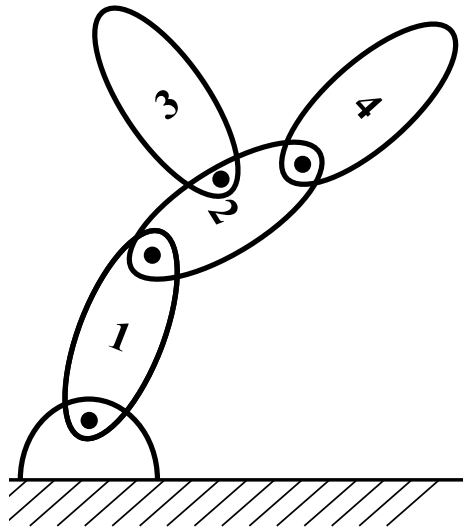
$$\begin{cases} \text{Velocity Propagation:} & {}^{i}\mathcal{V}_i = \left({}^{i}X_{p(i)}\right) {}^{p(i)}\mathcal{V}_{p(i)} + {}^{i}\mathcal{S}_i \, \dot{\theta}_i \\ \text{Accel Propagation:} & {}^{i}\mathcal{A}_i = \left({}^{i}X_{p(i)}\right) {}^{p(i)}\mathcal{A}_{p(i)} + {}^{i}\mathcal{V}_i \times {}^{i}\mathcal{S}_i \dot{\theta}_i + {}^{i}\mathcal{S}_i \ddot{\theta}_i \end{cases}$$



RNEA: Force Propagation (Backward Pass)

Goal: Given body spatial velocity $\mathcal{V}_i$ and spatial acceleration $\mathcal{A}_i$, compute the joint wrench $\mathcal{F}_i$ and the corresponding torque $\tau_i = \mathcal{S}_i^T \mathcal{F}_i$

$$\begin{cases} \mathcal{F}_i & = \mathcal{I}_i \mathcal{A}_i + \mathcal{V}_i \times^* \mathcal{I}_i \mathcal{V}_i + \sum_{j \in c(i)} \mathcal{F}_j \\ \tau_i & = \mathcal{S}_i^T \mathcal{F}_i \end{cases}$$



Recursive Newton-Euler Algorithm

$$\tau \leftarrow \mathsf{RNEA}(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext}; \mathsf{Model})$$

- Forward pass:

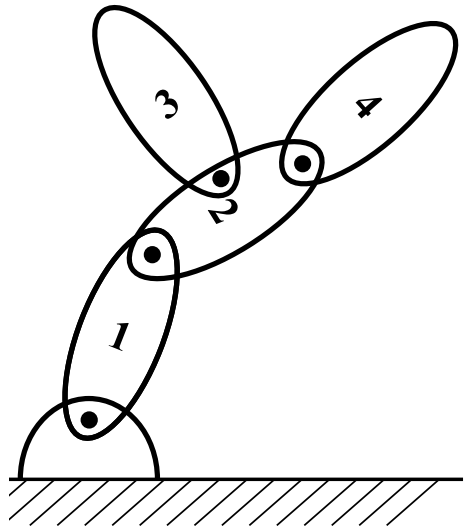
- Backward pass:

Outline

- Introduction
- Inverse Dynamics: Recursive Newton-Euler Algorithm (RNEA)
- Analytical Form of the Dynamics Model
- Forward Dynamics Algorithms

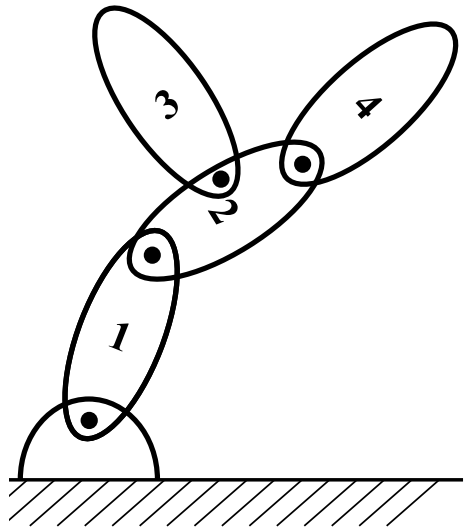
Structures in Dynamic Equation (1/3)

- Jacobian of each link (body): $J_1, \ldots, J_4$



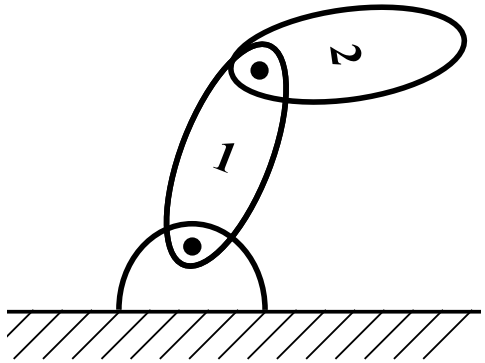
Structures in Dynamic Equation (2/3)

- Torque required to generate a "force" $\mathcal{F}_4$ to body 4



Structures in Dynamic Equation (3/3)

- Overall torque expression:



Derivation of Overall Dynamics Equation

-

$$\tau = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta})\dot{\theta} + \tau_g + J^T(\theta)\mathcal{F}_{ext} \tag{1}$$

Properties of Dynamics Model of Multi-body Systems

-

Outline

- Introduction
- Inverse Dynamics: Recursive Newton-Euler Algorithm (RNEA)
- Analytical Form of the Dynamics Model
- Forward Dynamics Algorithms

Forward Dynamics Problem

$$\tau = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta})\dot{\theta} + \tau_g + J^T(\theta)\mathcal{F}_{ext} \tag{2}$$

- Inverse dynamics: $\tau \leftarrow \text{RNEA}(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext})$ $O(N)$ complexity
  - RNEA can work directly with a given URDF model (kinematic tree + joint model + dynamic parameters). It does not require explicit formula for $M(\theta), \tilde{c}(\theta, \dot{\theta})$

- **Forward dynamics:** Given $(\theta, \dot{\theta})$, τ, $\mathcal{F}_{ext}$, find $\ddot{\theta}$

1. Calculate $\tilde{c}(\theta, \dot{\theta})$
2. Calculate mass matrix $M(\theta)$
3. Solve $M\ddot{\theta} = \tau - \tilde{c}$

Calculations of $\tilde{c}$ and M

- Denote our inverse dynamics algorithm: $\tau = \text{RNEA}(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext})$

- **Calculation of** $\tilde{c}$: obviously, $\tau = \tilde{c}(\theta, \dot{\theta})$ if $\ddot{\theta} = 0$. Therefore, $\tilde{c}$ can be computed via:

$$\tilde{c}(\theta, \dot{\theta}) = \text{RNEA}(\theta, \dot{\theta}, 0, \mathcal{F}_{ext})$$

- **Calculation of** M**:** Note that $\tilde{c}(\theta, \dot{\theta}) = c(\theta, \dot{\theta})\dot{\theta} - \tau_g - J^T(\theta)\mathcal{F}_{ext}$.
  - Set $\mathrm{g} = 0$, $\mathcal{F}_{ext} = 0$, and $\dot{\theta} = 0$, then $\tilde{c}(\theta, \dot{\theta}) = 0 \Rightarrow \tau = M(\theta)\ddot{\theta}$

  - We can compute the jth column of $M(\theta)$ by calling the inverse algorithm

$$M_{:,j}(\theta) = \text{RNEA}(\theta, 0, \ddot{\theta}_j^0, 0)$$

    where $\ddot{\theta}_j^0$ is a vector with all zeros except for a 1 at the jth entry.

- A more efficient algorithm for computing M is the *Composite-Rigid-Body Algorithm (CRBA)*. Details can be found in Featherstone's book.

Forward Dynamics Algorithm

- Now assume we have $\theta, \dot{\theta}, \tau, M(\theta), \tilde{c}(\theta, \dot{\theta})$, then we can immediately compute $\ddot{\theta}$ as $\ddot{\theta} = M^{-1}(\theta)\left[\tau - \tilde{c}(\theta, \dot{\theta})\right]$

- This provides a 2nd-order differential equation in $\mathbb{R}^n$, we can easily simulate the joint trajectory over any time period (under given ICs θ^o and $\dot{\theta}^o$)

- Computational Complexity:
  - RNEA: $O(N)$
  - $\tilde{c} = RNEA(\theta, \dot{\theta}, 0, \mathcal{F}_{ext})$: $O(N)$
  - $M(\theta)$: $O(N^2)$
  - $M^{-1}(\theta)$: $O(N^3)$
  - Most efficient forward dynamics algorithm: Articulated-Body Algorithm (ABA): $O(N)$

More Discussions

-

More Discussions

-