Computational Robotics

L3: Kinematic and Dynamic model of a manipulator

Roberto Marino, PhD

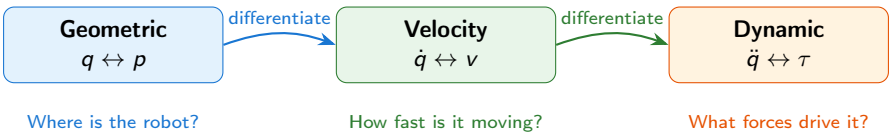
Computational Modeling for Robotics and Human Computer Interaction
University of Messina - FCRLab
roberto.marino@unime.it

April 14, 2026



UniMe
1548

Three Levels of Robot Modeling



Direct problem (joint space $\rightarrow$ Cartesian)

Always unique: just evaluate a formula. *Fast and simple.*

Inverse problem (Cartesian $\rightarrow$ joint space)

Generally harder: multiple solutions, singularities.

Direct Velocity Kinematics — The Jacobian

Differentiating $p = f(q)$ with respect to time:

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = J(q)\,\dot{q} \qquad J(q) \in \mathbb{R}^{6\times n}$$

Geometric Jacobian – column i (MDH Khalil convention)

Revolute joint:

$$J_i = \begin{pmatrix} z_i \times (p_{EE} - p_i) \\ z_i \end{pmatrix}$$

Prismatic joint:

$$J_i = \begin{pmatrix} z_i \\ 0 \end{pmatrix}$$

$z_i = T_0^i[:3,2]$ (axis), $p_i = T_0^i[:3,3]$ (origin). **Khalil:** z_i is the axis of the *current* frame i, not of the preceding frame (as in Craig).

$J(q)$ depends only on q, not on $\dot{q}$. The direct problem is a single matrix–vector product — always fast.

Jacobian Structure and Manipulability

Natural block split:

$$J(q) = \begin{pmatrix} J_v(q) \\ J_\omega(q) \end{pmatrix} \quad \begin{matrix} \leftarrow \text{linear velocity } (3\times n) \\ \leftarrow \text{angular velocity } (3\times n) \end{matrix}$$

	$J_{v,i}$	$J_{\omega,i}$
Revolute	$z_i \times (p_{EE} - p_i)$	z_i
Prismatic	z_i	0

Yoshikawa Manipulability

$$w(q) = \sqrt{\det(J_v J_v^T)} = \prod_i \sigma_i$$

- $w = 0$ at a singularity
- Large w: isotropic motion
- Maximise w to avoid singular configurations

Inverse Velocity Kinematics — Problem Statement

Given $v \in \mathbb{R}^6$, find $\dot{q} \in \mathbb{R}^n$ such that $J(q)\,\dot{q} = v$.

Square: $n = 6$

$$\dot{q} = J^{-1} v$$

Unique if $\det(J) \neq 0$.

Fails at singularities.

Redundant: $n > 6$

$$\dot{q} = J^{\dagger} v + N\dot{q}_0$$

Infinite solutions.
$J^{\dagger}$: min. norm.
N: null-space term.

Deficient: $n < 6$

$$\dot{q} = J^{\dagger} v$$

No exact solution.

Least-squares
best approx.

Panda robot ($n = 7$, $m = 6$) – redundant

At every non-singular configuration there is a 1-dimensional null-space: one direction of joint motion that produces *zero* EE velocity (self-motion).

The Moore-Penrose Pseudo-inverse

For $J \in \mathbb{R}^{m \times n}$, compute via SVD:

$$J = U\Sigma V^T \implies J^\dagger = V\Sigma^\dagger U^T \qquad \Sigma^\dagger_{ii} = \begin{cases} 1/\sigma_i & \sigma_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Fundamental property (redundant case):

$\dot{q}^\star = J^\dagger v$ is the **minimum-norm** solution:

$$\dot{q}^\star = \arg\min_{\dot{q}} \|\dot{q}\|_2 \text{ s.t. } J\dot{q} = v$$

Smallest joint motion among all solutions.

Near singularities: DLS

$\sigma_{min} \to 0 \;\Rightarrow\; \|J^\dagger v\| \to \infty$

Damped Least Squares:

$$J^\dagger_\lambda = J^T(JJ^T + \lambda^2 I)^{-1}$$

$$(\Sigma^\dagger_\lambda)_{ii} = \frac{\sigma_i}{\sigma_i^2 + \lambda^2}$$

λ large: stable, imprecise
λ small: precise, unstable

Null-Space and Self-Motion

Null-space of J

$$\mathcal{N}(J) = \{\dot{q} \in \mathbb{R}^n : J\dot{q} = 0\} \qquad \dim \mathcal{N}(J) = n - \operatorname{rank}(J)$$

Any motion in $\mathcal{N}(J)$ moves the joints **without moving the EE**.

General solution (redundant, full rank):

$$\dot{q} = \underbrace{J^{\dagger} v}_{\text{particular solution}} + \underbrace{(I - J^{\dagger} J)\, \dot{q}_0}_{\text{null-space term}}$$

Choose $\dot{q}_0 = \nabla H(q)$ to optimise a secondary objective *without moving the EE*:

Maximise manipulability:

$$H(q) = w(q) = \sqrt{\det(J_v J_v^T)}$$

Avoid joint limits:

$$H(q) = -\sum_i \frac{(q_i - \bar{q}_i)^2}{(q_{max,i} - q_{min,i})^2}$$

Kinematic Singularities

A **singularity** at q^*: $J(q^*)$ loses rank. The robot cannot move the EE in some direction.

Three types:

1. **Boundary** — robot fully extended or retracted; EE on workspace boundary.
2. **Interior** — two or more joint axes become aligned inside the workspace. *(Panda wrist: $q_5 = 0$)*
3. **Representation** — gimbal lock with Euler angles; not a physical property (cured by quaternions).

Physical meaning

At q^* the robot loses at least one Cartesian DOF.

Detected by: $\sigma_{min}(J) \approx 0$

Quantified by: $w(q) = \prod_i \sigma_i \approx 0$

Connection to numerical IGM:

$$q_{k+1} = q_k + \alpha\, J_\lambda^\dagger(q_k)\, e_k$$

Each iteration solves one inverse velocity step for error e_k.

The Equation of Motion

The complete dynamics of an n-joint serial robot:

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

M(q) – Inertia

$M \in \mathbb{R}^{n\times n}$

Resistance to acceleration.

Symmetric, pos. definite.

Depends on: q
Scales with: $\ddot{q}$

C(q,dq) – Coriolis

$C \in \mathbb{R}^{n\times n}$

Coriolis and centrifugal.

Quadratic in $\dot{q}$.

Depends on: $q, \dot{q}$
Scales with: $\dot{q}^2$

G(q) – Gravity

$G \in \mathbb{R}^{n}$

Torque to hold still vs gravity.

Zero in zero-gravity.

Depends on: q
Independent of $\dot{q}$

Physical interpretation

The Inertia Matrix $M(q)$ in Detail

Lagrangian derivation (Khalil — via link Jacobians):

$$T(q,\dot{q}) = \tfrac{1}{2}\dot{q}^T M(q)\dot{q} \implies M(q) = \sum_{i=1}^{n} \left[m_i J_{v,c_i}^T J_{v,c_i} + J_{\omega,c_i}^T {}^0I_i \, J_{\omega,c_i} \right]$$

J_{v,c_i}, J_{ω,c_i}: CoM Jacobians of link i; $\quad {}^0I_i = R_i I_i R_i^T$: inertia tensor in base frame.

Entry-level interpretation:

- M_{ii}: torque at joint i to accelerate joint i alone
- M_{ij} $(i \neq j)$: acceleration of joint j induces reaction at joint i (coupling)

Key properties:

- $M = M^T$ (symmetric)
- $\dot{q}^T M \dot{q} = 2T > 0$ (pos. def.)
- Always invertible
- Varies with posture: robot is effectively "heavier" in some configurations

Coriolis Matrix C and Gravity Vector G

Christoffel symbols (2nd kind):

$$c_{ijk} = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right) \qquad C_{ij} = \sum_k c_{ijk}\dot{q}_k$$

- **Coriolis**: perpendicular force when a body simultaneously rotates and translates.
- **Centrifugal**: outward force during rotation; $\propto \dot{q}^2$.

Passivity

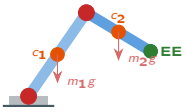
$\dot{M}(q) - 2C(q,\dot{q})$ is **skew-symmetric**.

The robot cannot generate energy.
Any passive controller is
guaranteed stable (Lyapunov).

Gravity vector:

$$G(q) = \frac{\partial U}{\partial q} = -\sum_{i=1}^{n} m_i J_{v,c_i}^T g_0$$

$g_0 = [0, 0, -9.81]^T$, depends only on q.



Direct (Forward) Dynamics

Direct Dynamics

Given: torques τ and state $(q, \dot{q})$ **Find:** accelerations $\ddot{q}$

$$\ddot{q} = M(q)^{-1}\left[\tau - C(q,\dot{q})\dot{q} - G(q)\right]$$

Main use: simulation.

Given control law $\tau(t, q, \dot{q})$, integrate the ODE:

$$\frac{d}{dt}\begin{pmatrix} q \\ \dot{q} \end{pmatrix} = \begin{pmatrix} \dot{q} \\ M^{-1}[\tau - C\dot{q} - G] \end{pmatrix}$$

Numerical integration via **RK4**:

$$\begin{pmatrix} q \\ \dot{q} \end{pmatrix}_{k+1} = \mathrm{RK4}\left(\begin{pmatrix} q \\ \dot{q} \end{pmatrix}_k, \Delta t\right)$$

Notes:

- Invert M: cost $O(n^3)$
- M always invertible (pos. def.)
- RK4: 4 evaluations per step
- Smaller Δt: more accurate

Inverse Dynamics

Inverse Dynamics

Given: desired trajectory $(q, \dot{q}, \ddot{q})$ **Find:** required torques τ

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

No matrix inversion. Just evaluate three terms — always fast.

Main use: computed-torque control.

$$\tau = M(q)\underbrace{[\ddot{q}_d + K_d\dot{e} + K_p e]}_{v} + C\dot{q} + G$$

where $e = q_d - q$. Cancels the nonlinear dynamics exactly, leaving a linear error system:

$$\ddot{e} + K_d\dot{e} + K_p e = 0$$

Critical damping rule

Choose $K_d = 2\sqrt{K_p}$

Guarantees critically damped second-order error dynamics: fast convergence, no overshoot.

Other uses:

Lagrangian vs. Newton-Euler: Two Paths, Same Answer

Lagrangian approach

- Global, matrix-based formulation
- Builds M, C, G explicitly
- Complexity $O(n^3)$
- Natural for control design
- Insight into energy structure

Recursive Newton-Euler

- Local, link-by-link formulation
- Computes τ directly (no M)
- Complexity O(n) — linear!
- Standard in industrial controllers
- Preferred for real-time simulation

Newton-Euler: two passes

Forward pass (base to EE)
Propagate velocities ω_i, $\dot{p}_i$
and accelerations $\dot{\omega}_i$, $\ddot{p}_{c_i}$

Backward pass (EE to base)
Propagate forces f_i and moments m_i

Revolute: $\tau_i = m_i \cdot z_i$
Prismatic: $\tau_i = f_i \cdot z_i$

Key point

Both formulations give the same τ.
Use Lagrangian for analysis;
Newton-Euler for simulation.

All Four Models at a Glance

Model	**Input**	**Output**	**Main use**	**Cost**
Direct kinematics	$\dot{q}$	v, ω	Workspace monitoring	$O(n)$
Inverse kinematics	v, ω	$\dot{q}$	Velocity control	$O(n^2)$
Inverse dynamics	$q, \dot{q}, \ddot{q}$	τ	Feedforward control	$O(n)$–$O(n^3)$
Direct dynamics	τ	$\ddot{q}$	Simulation	$O(n^3)$

The complete control loop:

