



Kinematics and Dynamics Essentials

Robotics in depth 01 | Frames and rotations, homogeneous transforms, forward and inverse kinematics of a 2-link arm worked by hand, the Jacobian and singularities, the manipulator equation, gravity torque, RNEA/ABA/CRBA and Pinocchio / MuJoCo code

#Robotics #Kinematics #Dynamics #Jacobian #Pinocchio

Why kinematics and dynamics first

Kinematics and dynamics



Kinematics answers 'where is the hand if the joints are at these angles?'; dynamics answers 'how much motor torque does it take to move it that way?'.

Every planner, controller and learned policy talks in either joint space q or task space x ; FK, IK and the Jacobian convert between them, the manipulator equation turns motion into torques.

- URDF/MJCF describe the robot
- Pinocchio, MuJoCo, Drake compute the rest



Poses live on the Lie group $SE(3)$, not in a vector space; interpolate with the exponential map, regularise IK near singularities (damped least squares), exploit $O(n)$ recursive algorithms for whole-body control at 1 kHz.

PICTURE IT

Your arm: knowing your shoulder and elbow angles tells you where your hand is (FK); reaching a cup means choosing those angles (IK); holding a full bucket at arm's length tells you about dynamics.

Before Kinematics and dynamics, can you explain...

Not yet Partly Yes

MUST KNOW Vectors, dot products, matrix product

Learn: maths 02-03

☐ ☐ ☐

MUST KNOW Sine, cosine, radians

Learn: maths 01

☐ ☐ ☐

MUST KNOW Derivatives and the chain rule

Learn: maths 05-06

☐ ☐ ☐

HELPFUL Newton's second law $F = m a$

Learn: school physics

☐ ☐ ☐

HELPFUL NumPy arrays

Learn: programming 02

☐ ☐ ☐

ADVANCED Lie groups $SO(3)/SE(3)$

Learn: Lynch and Park, Modern Robotics, ch. 3

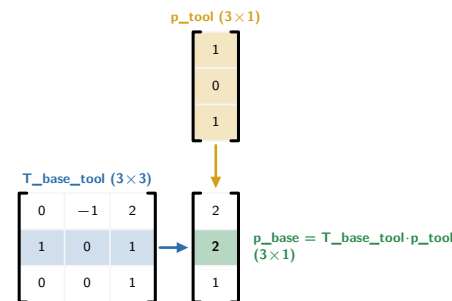
☐ ☐ ☐

Recall (perception-and-generation 08): a robot runs a perceive $\rightarrow$ plan $\rightarrow$ control loop. This sheet is the geometry and physics that every block of that loop uses; control (sheet 02), estimation (03) and planning (04) build on it.

Frames, rotations and transforms

Representation	Numbers	Good for	Trap
Rotation matrix R	9 (3 free)	composing frames ($R_{ac} = R_{ab} \cdot R_{bc}$), rotating vectors	drift off orthogonality after many products: re-orthonormalise
Euler / roll-pitch-yaw	3	human input, URDF rpy attribute	gimbal lock at pitch $\pm 90^\circ$; 12 axis conventions
Axis-angle / rotation vector	3	small rotations, angular velocity, learning targets	angle wraps at π
Unit quaternion	4	interpolation (slerp), storage, filters	q and $-q$ are equal; component order differs by library

Four ways to write the same 3-D rotation; pick by job, convert at the boundaries



Shapes: T_base_tool is 3×3 , p_tool is $3 \times 1 \rightarrow p_base = T_base_tool \cdot p_tool$ is 3×1 (the inner sizes 3 = 3 must match)

$p_base[2,1] = \sum T_base_tool[2,k] \cdot p_tool[k,1]$ for $k = 1 \dots 3 = 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 = 2$

Cost: $m \cdot n \cdot p = 3 \cdot 3 \cdot 1 = 9$ multiplications for the whole of p_base

Where: T_base_tool : $m \times n$ matrix ($m = 3$ rows, $n = 3$ columns); p_tool : $n \times p$ matrix ($p = 1$); $p_base[i,j]$: entry in row i , column j of p_base = dot product of row i of T_base_tool with column j of p_tool ; k : index over the n shared columns of T_base_tool / rows of p_tool ; Σ : sum

A homogeneous transform in 2-D: tool frame rotated 90° and shifted by $(2, 1)$ m in the base frame. The tool-frame point $(1, 0)$ lands at $(2, 2)$ in the base frame; the last row $[0 \ 0 \ 1]$ keeps the trick working

Quaternion order bites every robotics project

ROS 2 messages (`geometry_msgs/Quaternion`) and SciPy `Rotation.as_quat()` use (x, y, z, w) ; MuJoCo (`body_quat`, `qpos`) and Eigen's constructor use (w, x, y, z) . Swapping them gives a valid but wrong rotation, so tests pass and the robot turns the wrong way. Name the order in every variable (`quat_xyzw`).

Quantity	Formula
Homogeneous transform	$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$
Chaining frames	$T_{ac} = T_{ab} T_{bc}$
Inverse transform	$T^{-1} = \begin{bmatrix} R^\top & -R^\top p \\ 0 & 1 \end{bmatrix}$
Rotation about z by θ	$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

where $R = 3 \times 3$ rotation matrix of the child frame in the parent frame (-); p = position of the child origin in the parent frame (m); T_{ab} = pose of frame b expressed in frame a (-); θ = rotation angle (rad)

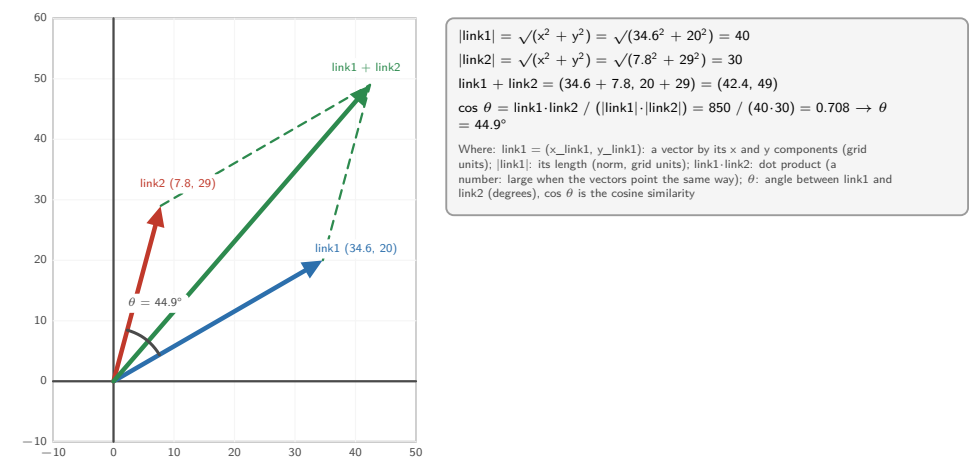
Transforms you will write every week

● Forward and inverse kinematics

Quantity	Formula
FK, x	$x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$
FK, y	$y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$
IK, elbow	$\cos \theta_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}$ ($\pm$ gives elbow up / down)
IK, shoulder	$\theta_1 = \text{atan2}(y, x) - \text{atan2}(l_2 \sin \theta_2, l_1 + l_2 \cos \theta_2)$
Velocity	$\dot{x} = J(q) \dot{q}, \quad \det J = l_1 l_2 \sin \theta_2$
Static force	$\tau = J(q)^\top F$

where l_1, l_2 = upper-arm and forearm lengths (m); θ_1, θ_2 = shoulder angle from the x axis, elbow angle relative to link 1 (rad); x, y = tool position in the base frame (m); $q, \dot{q}$ = joint vector (θ_1, θ_2) and joint velocities (rad, rad/s); $J = 2 \times 2$ Jacobian $\partial(x, y) / \partial(\theta_1, \theta_2)$ (m/rad); F, τ = force at the tool, torques at the joints (N, N·m)

Planar 2-link arm (shoulder and elbow, both revolute)



The tool position is the vector sum of the two links (in cm): link 1 = 40 cm at 30° , link 2 = 30 cm at $30^\circ + 45^\circ = 75^\circ$.
The sum is the FK answer; the angle between the vectors is the elbow angle θ_2

PROBLEM $l_1 = 0.40$ m, $l_2 = 0.30$ m, $\theta_1 = 30^\circ$, $\theta_2 = 45^\circ$, joint velocities $\dot{q} = (0.1, 0.2)$ rad/s. Where is the tool, how fast does it move, and how far is the arm from a singularity?

GIVEN

- $\cos 30^\circ = 0.8660$, $\sin 30^\circ = 0.5$, $\cos 75^\circ = 0.2588$, $\sin 75^\circ = 0.9659$ ($\theta_1 + \theta_2 = 75^\circ$)
- x, y = tool position in m; $\dot{x}, \dot{y}$ = tool velocity in m/s; J = Jacobian in m/rad

1 **FK, x**

$$0.40 \times 0.8660 + 0.30 \times 0.2588 = 0.3464 + 0.0776 = 0.4241 \text{ m}$$

Why: sum of the link projections on x

2 **FK, y**

$$0.40 \times 0.5 + 0.30 \times 0.9659 = 0.2000 + 0.2898 = 0.4898 \text{ m}$$

Why: same on y

3 **IK check of the elbow**

$$(0.4241^2 + 0.4898^2 - 0.16 - 0.09) / (2 \times 0.12) = 0.1697 / 0.24 = 0.7071 \rightarrow \theta_2 = 45^\circ$$

Why: law of cosines recovers the elbow; -45° is the mirror (elbow-down) solution

4 **Jacobian**

$$J = \begin{bmatrix} -0.4898 & -0.2898 \\ 0.4241 & 0.0776 \end{bmatrix}$$

Why: row 1 = $\partial x / \partial \theta_1$, $\partial x / \partial \theta_2 = -y$, $-l_2 \sin 75^\circ$; row 2 = x , $l_2 \cos 75^\circ$

5 **Tool velocity**

$$\dot{x} = -0.4898 \times 0.1 - 0.2898 \times 0.2 = -0.107 \text{ m/s}; \dot{y} = 0.4241 \times 0.1 + 0.0776 \times 0.2 = 0.058 \text{ m/s}$$

Why: $\dot{x} = J \dot{q}$

6 **Distance to singularity**

$$\det J = 0.40 \times 0.30 \times \sin 45^\circ = 0.0849 \text{ m}^2$$

Why: zero when $\theta_2 = 0^\circ$ (arm stretched) or 180° (folded)

ANSWER Tool at (0.424, 0.490) m, moving at (−0.107, 0.058) m/s, i.e. 0.122 m/s; manipulability $|\det J| = 0.085$, about 71 % of its maximum 0.12 reached at $\theta_2 = 90^\circ$

CHECK ☒ Stretch the arm ($\theta_2 = 0$): $\det J = 0$, the tool cannot move along the arm's own direction whatever the joint speeds

FK, IK check and Jacobian of a 2-link arm, by hand

Method	How	Use when	Tools (as of 2026)
Closed form (analytic)	geometry like above; 6-DoF arms with a spherical wrist decompose into position + orientation	industrial 6-axis arms, real-time	IKFast (OpenRAVE), vendor solvers
Damped least squares	$\Delta q = J^T (J J^T + \lambda^2 I)^{-1} \Delta x$, iterate	any arm; λ keeps steps finite near singularities	Pinocchio, KDL, MoveIt KDL plugin
Optimisation (QP / NLP)	minimise pose error + joint-limit, collision, posture costs	redundant arms (7-DoF), humanoids, whole body	TRAC-IK, Pink, Drake, cuRobo (GPU)
Learned IK	network maps pose $\rightarrow$ q	rarely alone; seeds an optimiser	research

IK in practice: how real stacks solve it

⚡ Dynamics: from motion to torque

Quantity	Formula
----------	---------

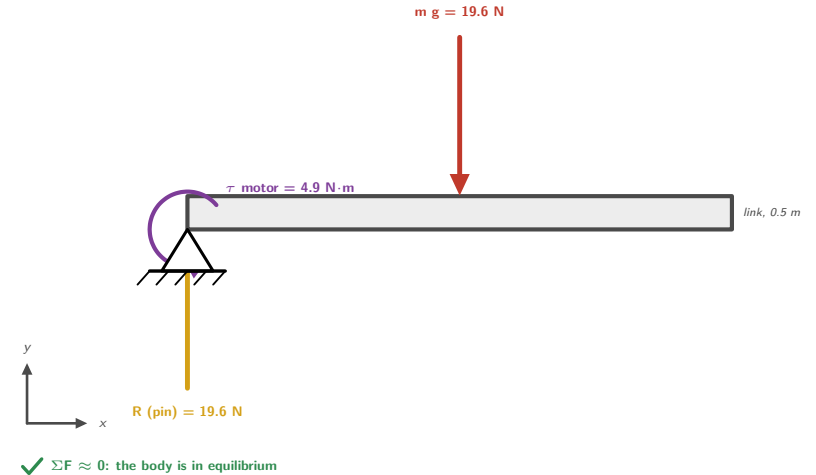
Equation of motion	$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau + J^T F_{ext}$
--------------------	---

Gravity torque, one link $\tau_g = m g l_c \cos \theta$

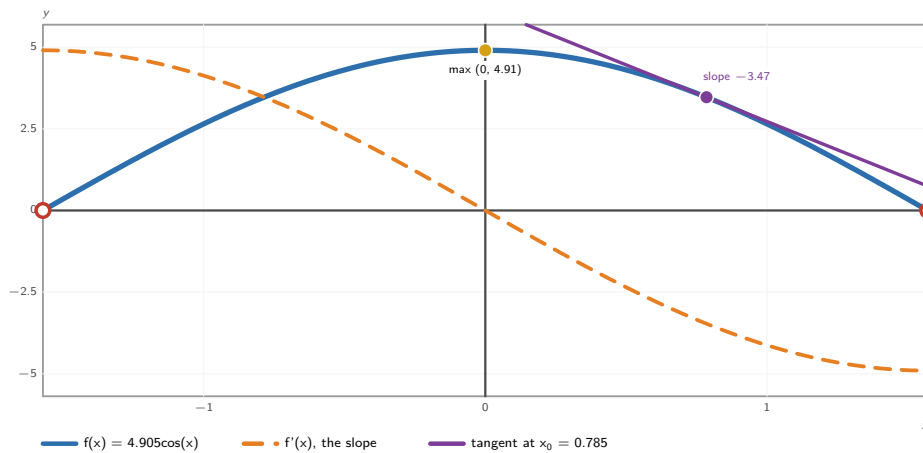
Gear reduction $\tau_{joint} = N \eta \tau_{motor}$, $I_{reflected} = N^2 I_{rotor}$

where $M(q) = n \times n$ mass (inertia) matrix ($\text{kg} \cdot \text{m}^2$); $C(q, \dot{q}) \dot{q}$ = Coriolis and centrifugal torques ($\text{N} \cdot \text{m}$); $g(q)$ = gravity torques ($\text{N} \cdot \text{m}$); τ , F_{ext} = motor torques; external force at the contact point ($\text{N} \cdot \text{m}$, N); m , l_c , g = link mass, distance joint $\rightarrow$ centre of mass, 9.81 m/s^2 (kg , m , m/s^2); θ = link angle above the horizontal (rad); N , η = gear ratio, gearbox efficiency (0-1) (-); I_{rotor} = motor rotor inertia ($\text{kg} \cdot \text{m}^2$)

The manipulator equation: every rigid arm, legged robot or humanoid obeys it



One horizontal link held still: weight $m g = 2 \text{ kg} \times 9.81 = 19.6 \text{ N}$ at its centre of mass, 0.25 m from the joint. The joint pin pushes up 19.6 N and the motor must supply $\tau = 19.6 \times 0.25 = 4.9 \text{ N} \cdot \text{m}$ counter-clockwise



$f(x) = 4.905\cos(x)$

Roots ($f(x) = 0$): $x \approx -1.57, 1.57$ (sign change on the grid, refined by bisection)

Local extrema ($f'(x)$ changes sign): maximum $f(0) = 4.91$

$f'(x) \approx [f(x+h) - f(x-h)] / (2h)$, $h = 3.14 \times 10^{-5}$ (central difference)

Tangent at $x_0 = 0.785$: $y = f(x_0) + f'(x_0) \cdot (x - x_0) = 3.468 + (-3.468) \cdot (x - 0.785)$

Where: x : input value; $f(x)$: output; $f'(x)$: derivative = slope, change of f per unit of x ; h : small step of the difference quotient; x_0 : point of tangency

Gravity torque $\tau_g(\theta) = 4.905 \cos \theta$ N·m for the same link (θ in rad, horizontal = 0): maximum when horizontal, zero when vertical; the slope is what a gravity-compensating controller must track

Algorithm	Computes	Cost	Used by
RNEA (recursive Newton-Euler)	inverse dynamics $\tau = \text{ID}(q, \dot{q}, \ddot{q})$; also $g(q)$ with $\dot{q} = \ddot{q} = 0$	$O(n)$	feed-forward torque, gravity compensation
ABA (articulated-body)	forward dynamics $\ddot{q} = \text{FD}(q, \dot{q}, \tau)$	$O(n)$	simulators
CRBA (composite rigid-body)	mass matrix $M(q)$	$O(n^2)$	operational-space and whole-body control, MPC

The three recursive algorithms (Featherstone) that make dynamics real-time

Pinocchio (Python): FK, Jacobian, gravity and inverse dynamics from a URDF

```
import numpy as np
import pinocchio as pin

model = pin.buildModelFromUrdf("arm.urdf")
data = model.createData()
q = pin.neutral(model) # joint vector
v = np.zeros(model.nv); a = np.zeros(model.nv)

pin.forwardKinematics(model, data, q)
pin.updateFramePlacements(model, data)
```

```
tool = model.getFrameId("tool0")
print(data.oMf[tool].translation) # FK: tool position
```

```
J = pin.computeFrameJacobian(
    model, data, q, tool, pin.LOCAL_WORLD_ALIGNED) # 6 x nv
g = pin.computeGeneralizedGravity(model, data, q)
tau = pin.rnea(model, data, q, v, a) # = g when v = a = 0
M = pin.crba(model, data, q) # mass matrix
```

INFO

MuJoCo equivalent: `mujoco.mj_forward(m, d)` fills `d.xpos` (body positions), `d.qfrc_bias` ($C \dot{q} + g$) and, with `mj_fullM`, the dense `M`; `mujoco.mj_jacSite(m, d, jaccp, jacr, site_id)` gives the Jacobian. Drake (MultibodyPlant) and MuJoCo also read URDF; MuJoCo prefers MJCF (mjcf) with contact parameters.

Confusions, myths and recap

Forward kinematics

joints $\rightarrow$ tool pose, unique

vs

Inverse kinematics

tool pose $\rightarrow$ joints, $0.. \infty$ answers

Tell: Which side is given?

Kinematics

geometry of motion, no forces

vs

Dynamics

forces and torques that cause motion

Tell: Do masses appear?

Joint space q

one number per joint

vs

Task space x

tool pose, 6 numbers in 3-D

Tell: Is it measured at the motors or at the tool?

Inverse dynamics

motion $\rightarrow$ torques (RNEA)

vs

Forward dynamics

torques $\rightarrow$ motion (ABA, simulators)

Tell: Are torques the output or the input?

Pairs that get mixed up in robotics interviews

YOU MAY HEAR
"IK always has one solution"

BUSTED

✓ **ACTUALLY**
It can have none, several or infinitely many
 A 6-axis arm has up to 8 (some up to 16) closed-form solutions; a 7-DoF arm has a continuous family (redundancy).

YOU MAY HEAR
"Euler angles are fine for control"

BUSTED

✓ **ACTUALLY**
They are singular at gimbal lock
 Use quaternions or rotation matrices inside; Euler only for display and human input.

YOU MAY HEAR
"A learned policy does not need kinematics"

BUSTED

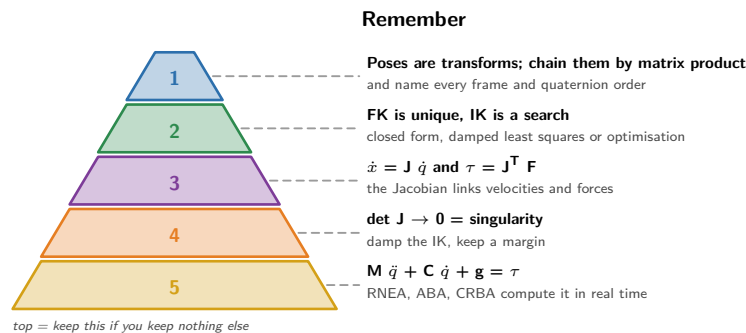
✓ **ACTUALLY**
Its actions are still expressed in a frame
 End-effector deltas go through IK; joint targets go through limits; wrong frames are a top cause of failed transfers.

YOU MAY HEAR
"Gravity is a small disturbance"

BUSTED

✓ **ACTUALLY**
It often dominates the torque at low speed
 Gravity compensation $g(q)$ from RNEA is the first feed-forward term every arm controller adds.

Beliefs that cost debugging days



Q1 FK of a planar 2-link arm?

A $x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$, $y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$.

Q2 Why can IK have two solutions for a planar 2-link arm?

A $\cos \theta_2$ fixes θ_2 up to its sign: elbow-up and elbow-down reach the same point.

Q3 What does the Jacobian map?

A Joint velocities to tool velocity ($\dot{x} = J \dot{q}$), and by transpose tool forces to joint torques ($\tau = J^T F$).

Q4 When is a 2-link arm singular?

A When $\det J = l_1 l_2 \sin \theta_2 = 0$: arm fully stretched ($\theta_2 = 0$) or folded ($\theta_2 = 180^\circ$).

Q5 Name the four terms of the manipulator equation.

A Inertia $M(q)\ddot{q}$, Coriolis/centrifugal $C(q,\dot{q})\dot{q}$, gravity $g(q)$, and the applied torque τ (plus $J^T F_{\text{ext}}$ for contacts).

Q6 What do RNEA and ABA compute?

A RNEA: inverse dynamics (torques from motion), $O(n)$. ABA: forward dynamics (accelerations from torques), $O(n)$.

Q7 Quaternion order in ROS 2 vs MuJoCo?

A ROS 2 and SciPy: (x, y, z, w). MuJoCo and Eigen's constructor: (w, x, y, z).

Q8 Why does a gearbox help and hurt?

A Torque $\times N$ (minus losses) but reflected rotor inertia $\times N^2$ and friction, making the joint stiff and hard to back-drive.

Q9 Gravity torque of a 2 kg link with its centre 0.25 m from the joint, horizontal?

A $\tau = 2 \times 9.81 \times 0.25 \times \cos 0 = 4.9 \text{ N}\cdot\text{m}$.