

The essentials: the geometry and kinematics a robot-learning paper assumes

What this reading supports. Sessions 2.2–2.4 (Decks 02 and 03) teach frames, poses, forward kinematics, the Jacobian and differential inverse kinematics. This reading is their written companion, and it is the kinematic input to Optimization I (Deck 04). The last calculation of its core, one joint-position update proposed from a local model, is the step that Deck 04 turns into an iterative solver. A robot-learning paper that uses a real arm writes $\mathbf{q} \mapsto \mathbf{X}(\mathbf{q})$, $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$ or $\dot{\mathbf{q}} = \mathbf{J}^+(\mathbf{q})\dot{\mathbf{x}}$ without explanation, and says that a policy “outputs end-effector deltas” or that a controller “tracks in joint space”. The core of this reading is what those sentences assume. The long reading (`notes.md`) gives the broader theory and adds what a robotics specialist would want; this document is the part of that chapter without which the learning half of the course cannot be followed.

What you should be able to do afterwards. Each item names the section that teaches it and the calculation you should be able to reproduce.

1. Read a frame label, convert a point’s coordinates between two frames, and compose and invert poses (§§1–3; the change of frame of Figure 1, its inverse check, and self-check questions 1 and 2).
2. Compute the two-link arm’s tip pose from its joint angles by multiplying four factors, and apply the same construction to one edge of the FR3 chain (§4).
3. Build a desired tool pose from an object pose and a grasp choice, and state the inverse problem with its possible outcomes (§5).
4. Predict the tip’s small displacement from a small joint change and its velocity from joint velocities, and compute the gripper’s angular velocity from the joint axes (§6).
5. Choose joint velocities for a requested tip velocity, explain why the stretched arm loses a direction, choose among several valid answers, and explain null-space motion relative to the stated task (§7).
6. Turn a joint velocity into a position command over one control period, and propose one joint-position update from the local model, check it by forward kinematics and read its remaining error (§7).
7. Say what a policy’s action format asks the robot-side stack to do (§9).

Reading map. The core is §§1–7, §9 and §10, read in order. Section 8 and the appendices are optional reference material. Section 8 measures singularities with singular values and the manipulability ellipse. Appendix A lists the notation papers use. Appendix B collects pose-error and orientation-loss variants. Appendix C derives the null-space projector and the damped step and shows the repeated local step converging. Appendix D holds the FR3 constants, the checkpoint positions along its

chain and the viewer's controls. No required item depends on an optional part; the recognition items of §10 may point to one. Where the core points to an optional part, the pointer says what it adds.

How to read it. Sections 1–8 each carry a three-line status, and §8's status says that the section is optional. *Must understand* items are required: the reader can explain them and reproduce the calculation. *Recognize* items are recognition-level: the reader knows what they are when a paper names them and need not reproduce them; a longer derivation, where this document has one, sits in the optional parts. *Deferred* items are treated in the long reading, in a deck's notes or in later sessions, and a status line may also point ahead to a later section of this document. The checklist in §10 collects the main items of the first two categories. Read the core in two passes: follow the geometric examples first, then return to the equations and the status boxes. Allow extra time to reproduce the calculations. The running example is a planar arm with two links, $l_1 = 0.4 \text{ m}$ and $l_2 = 0.3 \text{ m}$, at joint angles $\mathbf{q} = (30^\circ, 45^\circ)$. The main worked numbers are reproduced by `code/kinematics_note.py`. The expanded FK chains use `../02-D2-transforms-fk-jacobian/code/fk_chain_walkthrough.py` and `../../drake/01c-fr3-grasp-chain/fr3_grasp_chain.py`; the NumPy example after §10 reproduces the round-trip of §3, the Jacobian and its finite-difference check, the velocity and integration calculations and the joint-position update of §7. Angles in equations and code are in radians unless a degree sign is shown. A dot means a time derivative: $\dot{q}$ is a joint rate, not another position.

Notation. This course follows `notation.md`: vectors bold lowercase, matrices bold uppercase, frames plain capital letters. Papers spell the same objects differently, and a change of name can also signal a change of convention. Appendix A lists the common spellings next to ours.

1. Frames

Problem. A perception module reports the mug at $(0.15, 0.05, 0.05)$. Three numbers measured from where, along whose axes? A number without its frame is read against the wrong origin and no error is raised. Explicit frame labels let us detect such mismatches before calculating; bare numerical arrays cannot do that bookkeeping for us. Datasets, scene files and policy inputs all need a frame convention.

Geometry. A **frame** is an origin (a point) with three perpendicular unit axes attached to it, right-handed: curling the right hand from the positive x axis toward the positive y axis points the thumb along positive z . A frame is a measuring reference we attach to a body; it is not an extra physical part. We use frames for the world W , each link of the arm, the end-effector E (the gripper), the camera C , each object O . A position is an arrow from one frame's origin to another's; its **coordinates** are the three numbers obtained by projecting the arrow onto some frame's axes. The arrow is a geometric

object; the numbers depend on the frame chosen to write them. In the plane, the arrow with coordinates $(1, 0)$ in one frame has coordinates $(0, -1)$ in axes turned counter-clockwise by 90° : it now points along the new negative y axis. The arrow has not moved; the measuring axes have.

Notation. The course uses the monogram notation (Drake, Tedrake): three slots, three facts.

${}^A\mathbf{p}_C^B$: the position of B relative to A , written in the axes of C .

The left superscript is the reference (where the arrow starts), the right superscript the target (where it ends), the subscript the frame whose axes supply the numbers; the subscript is omitted when it equals the reference. The rotation ${}^A\mathbf{R}^B$ and the pose ${}^A\mathbf{X}^B$ of §§2–3 carry the first two slots only. Read a symbol aloud before using it: “ ${}^T\mathbf{p}^O$, the position of the object relative to the tray, in the tray’s axes”, where T names the tray. Changing only the subscript changes the coordinates of the same arrow; changing the left superscript changes where that arrow starts. In code the three slots are `p_T0_C`; if the expression frame is the tray, the abbreviation is `p_T0`. Deck 02 uses M for the **mug** in its tray example; here C denotes the **camera** and O the object, so read the local frame legend.

Why it appears in robot learning. A policy’s observation vector, a demonstration’s end-effector labels, a camera’s detections and a goal specification may use different frames. A frame mismatch can produce a plausible but wrong vector without any software error; keeping the labels is cheap insurance.

Must understand: what a frame is; the three slots of ${}^A\mathbf{p}_C^B$; that the same arrow has different coordinates in different frames.

Recognize: the frame names W, E, C, O ; `p_AB` in code.

Deferred: point-versus-vector algebra; how frames are stored in URDF or `.g` files (long reading §2, §5.2).

2. Rotations

Problem. “Which way is the mug turned?” needs a mathematical object, and the object must be something that can be applied to coordinates: turn the axes, and the numbers change.

Geometry. Rotating the plane counter-clockwise by θ sends the point (p_x, p_y) to $(p_x \cos \theta - p_y \sin \theta, p_x \sin \theta + p_y \cos \theta)$, which is a matrix product:

$$\mathbf{p}' = \mathbf{R}(\theta) \mathbf{p}, \quad \mathbf{R}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}. \quad (1)$$

Applying (1) to $(1, 0)$ and $(0, 1)$ gives the two columns: **the columns of a rotation matrix are the rotated axes**. Read as a change of frame, the columns of ${}^A\mathbf{R}^B$ are the axes of B written in the coordinates of A , which is why a rotation needs no “expressed in” slot. To see the conversion direction, a vector with coordinates $(1, 0)$ in B points along B ’s first axis. Multiplying by ${}^A\mathbf{R}^B$ selects its first column, giving that axis in A ’s coordinates. Thus ${}^A\mathbf{R}^B$ converts **from B to A** ; converting from A to B uses its inverse.

An **active rotation** turns the arrow while keeping the measuring axes fixed. A **passive change of coordinates** keeps the arrow fixed and changes which axes describe it. For B turned 90° counter-clockwise from A , actively rotating $(1, 0)$ gives $(0, 1)$ in A . Expressing the original, unmoved arrow in B instead gives $(0, -1)$ and uses $\mathbf{R}(90^\circ)^\top$. Always state the input and output frames; “rotate the frame” alone does not tell us which way to multiply.

Because the columns are unit vectors and mutually perpendicular,

$$\mathbf{R}^\top \mathbf{R} = \mathbf{I}, \quad \det \mathbf{R} = +1, \quad \text{hence } \mathbf{R}^{-1} = \mathbf{R}^\top. \quad (2)$$

The set of 3×3 matrices satisfying (2) is written $\text{SO}(3)$ ($\text{SO}(2)$ in the plane); the determinant condition excludes mirrors. **A rotation is undone by transposing it**; no matrix inversion is ever needed. A 3×3 rotation has nine entries subject to six independent constraints, leaving three degrees of freedom. More precisely, there are three unit-length constraints and three pairwise-perpendicularity constraints; they do not fix six particular entries. A degree of freedom is an independently adjustable quantity, not simply a stored number.

The other spellings. A rotation matrix is convenient for transforming vectors, but a file or network can encode the same orientation differently.

- A **rotation vector** $\theta \hat{\mathbf{u}}$ specifies a turn by angle θ about the unit axis $\hat{\mathbf{u}}$, using the right-hand rule. Its length is the angle.
- A **unit quaternion** $r = (w, x, y, z)$ stores four numbers whose squares sum to one. The encodings r and $-r$ represent the same orientation, so a coordinate-wise loss can penalize two predictions that mean exactly the same thing. Drake and RAI use scalar-first ordering; SciPy’s default and ROS messages use scalar-last ordering. Check the API or dataset.
- **Euler angles** specify three successive rotations about stated axes in a stated order. At certain orientations, two of these rotation axes align: changing two angles then has the same instantaneous effect. This is **gimbal lock**, a failure of the coordinates, not a physical loss of the body’s ability to turn.

- A **6D rotation representation** encodes two axes; after normalizing them and making them perpendicular, their cross product supplies the third. Six stored numbers do not mean six rotational freedoms.

These encodings can be converted to $\mathbf{R}$. Ordinary averaging may leave the set of valid rotations or ignore equivalent encodings; orientation losses and averaging therefore need to respect what the numbers represent.

Why it appears in robot learning. Orientation contributes three of a pose's six degrees of freedom. A network's output representation and its loss must agree: ordering, quaternion sign equivalence, and angle wrapping can all matter. Section 3 constructs an orientation error from the relative rotation, so its meaning does not depend on the encoding.

Must understand: $\mathbf{R}$ as a matrix whose columns are the rotated axes; $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ and $\mathbf{R}^{-1} = \mathbf{R}^\top$; three degrees of freedom.

Recognize: $\text{SO}(3)$; quaternions (and their two orderings), rotation vectors, Euler angles, and gimbal lock as a defect of Euler-angle coordinates.

Deferred: the derivation of (1), the three elementary rotations, active versus passive rotation in 3D (long reading §3.2) and Rodrigues' formula (Deck 02 notes §4.6). Angular velocity is constructed in §6 of this document.

3. Poses and rigid transformations

Problem. Where is the mug relative to the gripper, given where the camera saw it and where the gripper is? Positions from different frames must be combined, and combining them requires both the offset and the rotation between the frames.

Geometry. If frame B sits at ${}^A\mathbf{p}^B$ with orientation ${}^A\mathbf{R}^B$, a point known in B as ${}^B\mathbf{p}^P$ has coordinates in A

$${}^A\mathbf{p}^P = {}^A\mathbf{p}^B + {}^A\mathbf{R}^B {}^B\mathbf{p}^P : \quad (3)$$

arrows add head to tail, and the second arrow must first be expressed in A 's axes (Figure 1). For the figure's planar example, $\mathbf{R}(30^\circ)(0.15, 0.05) = (0.1049, 0.1183)$ m. Adding the origin offset $(0.40, 0.20)$ gives $(0.5049, 0.3183)$ m. These are two separate steps: change the axes used to describe the offset, then add where the new origin lies. Appending a 1 to every position, $\tilde{\mathbf{p}} = (\mathbf{p}, 1)$, turns (3) into one matrix product with the **homogeneous transform**, the course's **pose**:

$${}^A\mathbf{X}^B = \begin{bmatrix} {}^A\mathbf{R}^B & {}^A\mathbf{p}^B \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \text{SE}(3), \quad {}^A\tilde{\mathbf{p}}^P = {}^A\mathbf{X}^B {}^B\tilde{\mathbf{p}}^P. \quad (4)$$

The extra coordinate is bookkeeping: block multiplication gives

$$\begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{b} + \mathbf{p} \\ 1 \end{bmatrix}.$$

Here $\mathbf{b}$ is the point's position in B . A direction uses a final 0 instead, so it rotates without receiving an origin offset. The notation $\text{SE}(3)$ names the set of these rotation-plus-translation matrices in 3D. The 4×4 matrix (3×3 in the plane) holds a rotation and a position; it is *the pose of B as seen from A* and, at the same time, *the operator that converts B -coordinates into A -coordinates*. Two rules govern all its uses:

$$\begin{aligned} {}^A\mathbf{X}^C &= {}^A\mathbf{X}^B {}^B\mathbf{X}^C \quad (\text{inner letters cancel}), \\ ({}^A\mathbf{X}^B)^{-1} &= {}^B\mathbf{X}^A = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \end{aligned} \quad (5)$$

Composition is a matrix product whose inner frame letters must match; if they do not, the product is not a transform between the frames intended, and no number reveals it. To invert, first subtract the offset and then rotate back: $\mathbf{b} = \mathbf{R}^\top(\mathbf{a} - \mathbf{p})$, where $\mathbf{a}$ is the point's position in A . This explains both blocks $\mathbf{R}^\top$ and $-\mathbf{R}^\top \mathbf{p}$ in (5); transposing the whole pose matrix would be wrong. The mug relative to the gripper is therefore ${}^G\mathbf{X}^O = ({}^W\mathbf{X}^G)^{-1} {}^W\mathbf{X}^C {}^C\mathbf{X}^O$. Read its action from right to left: object coordinates become camera coordinates, then world coordinates, then gripper coordinates. Matching frame labels make this chain checkable.

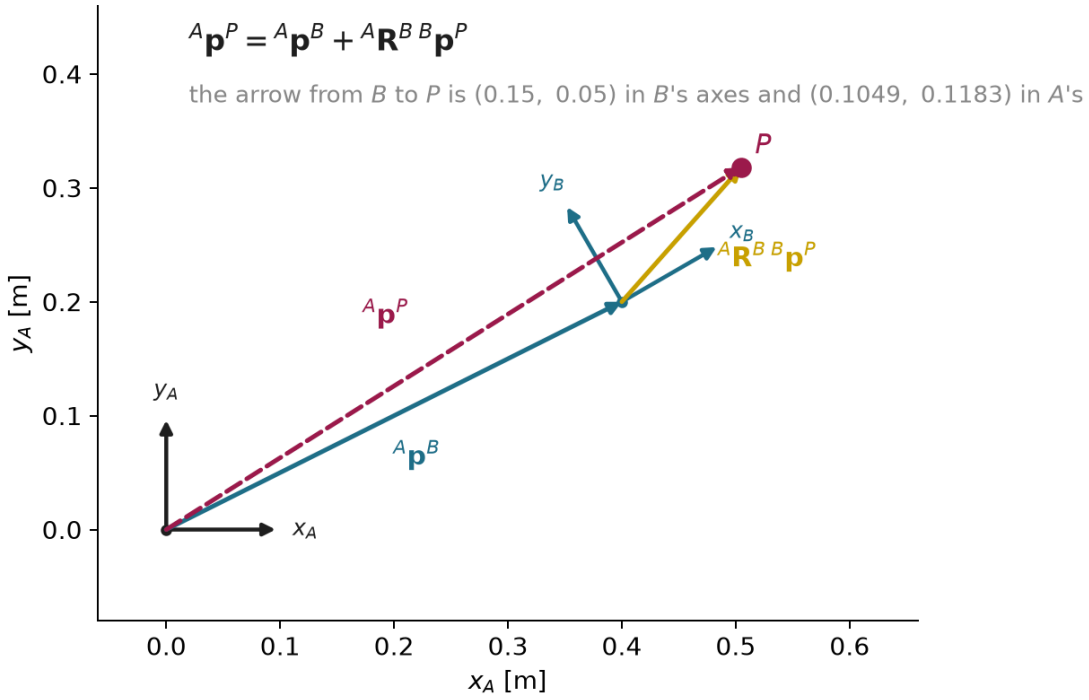


Figure 1. Equation (3) as arrows: frame B at $A\mathbf{p}^B = (0.40, 0.20)$ turned by 30° , a point P at $(0.15, 0.05)$ in B 's axes. The gold arrow is that vector rotated into A 's axes; the dashed arrow is the sum $A\mathbf{p}^P$. The same arrow from B to P reads $(0.15, 0.05)$ in one frame and $(0.1049, 0.1183)$ in the other.

Reverse the example to check (5). The point P of Figure 1 has A -coordinates $A\mathbf{p}^P \approx (0.5049, 0.3183)$ m. To recover its B -coordinates, first subtract the origin offset and then rotate back:

$$\mathbf{R}(30^\circ)^\top \left(\begin{bmatrix} 0.5049 \\ 0.3183 \end{bmatrix} - \begin{bmatrix} 0.40 \\ 0.20 \end{bmatrix} \right) \approx \begin{bmatrix} 0.8660 & 0.5000 \\ -0.5000 & 0.8660 \end{bmatrix} \begin{bmatrix} 0.1049 \\ 0.1183 \end{bmatrix} \approx \begin{bmatrix} 0.15 \\ 0.05 \end{bmatrix}$$

The same two operations are the blocks of (5). The inverse pose has rotation block $\mathbf{R}^\top$ and position block $-\mathbf{R}^\top \mathbf{p} \approx (-0.4464, 0.0268)$; applied to the homogeneous column $(0.5049, 0.3183, 1)$, it returns $(0.15, 0.05, 1)$. The displayed entries are rounded. The unrounded calculation returns $(0.15, 0.05)$ exactly, and the product of the pose with this inverse is the identity. Transposing the whole 3×3 pose matrix instead would give approximately $(0.5964, 0.0232, 1.2656)$. That does not recover $(0.15, 0.05)$, and the transposed matrix's last row is $(0.40, 0.20, 1)$ rather than the $(0, 0, 1)$ that a rigid transform must have, so the transposed matrix is not the inverse pose.

Pose changes and pose errors. Two successive rigid motions combine by multiplication. Adding their matrices does not produce the combined motion: even $\mathbf{I} + \mathbf{I}$ is not a rotation. To compare the current pose $A\mathbf{X}^B$ with the desired pose $A\mathbf{X}^{B_d}$, form the relative transform

$$\mathbf{X}_{\text{err}} = ({}^A\mathbf{X}^B)^{-1} {}^A\mathbf{X}^{B_d}, \quad {}^A\mathbf{X}^B \mathbf{X}_{\text{err}} = {}^A\mathbf{X}^{B_d}.$$

Its position block expresses the desired-origin offset in the **current body's axes**. Its rotation block describes the remaining turn in those axes. If current and desired poses agree, the relative transform is the identity: zero translation and no rotation. It is a transform, not yet a six-vector to feed into a Jacobian.

A scalar training loss is a different object. For predicted position and orientation $\mathbf{p}, \mathbf{R}$ and targets $\mathbf{p}^*, \mathbf{R}^*$ in the same frame, one choice is

$$\mathcal{L} = \|\mathbf{p} - \mathbf{p}^*\|^2 + w\theta^2, \quad w > 0.$$

The first term is the sum of squared position errors. The second penalizes the angle θ of the relative rotation $\mathbf{R}^{*\top} \mathbf{R}$, irrespective of its axis; an identical orientation has $\theta = 0$. Appendix B.1 shows how θ is computed from that matrix and the numerical care the computation needs. The weight w specifies how much rotational error matters relative to position error; meters and radians do not supply that preference for us.

Differences of matrix entries are also valid inside a **loss**, where they only score a mismatch, even though a difference of matrices is not a rigid motion. Appendix B.1 gives that variant and its relation to the angle θ . The essential distinction is between composing a motion (a matrix product), representing a motion error (a relative transform) and assigning that error a scalar cost (a loss).

Why it appears in robot learning. Poses are the currency between perception, goals and policies: a grasp is a pose, a camera calibration is a pose, an end-effector action is often a pose or a change of pose. The composition rule is how a policy trained on camera-frame observations is applied to a robot in the world frame, and the no-addition rule is why pose losses and pose averaging need care.

Must understand: (3); the pose ${}^A\mathbf{X}^B$ as one matrix that both locates B and converts coordinates; composition with matching inner letters; inversion by rotating back and reversing the translated offset; that poses compose and do not add.

Recognize: $\text{SE}(3)$; the pose error as a relative transform; scalar pose losses, including losses on matrix entries (Appendix B); the seven-number (position plus quaternion) storage of a pose; the paper spellings of a pose, such as Lynch & Park's T_{ab} (Table A.1).

Deferred: the block-multiplication proofs (long reading §4.1–§4.3). Worked 4×4 examples follow in §4.2 and §5 of this document.

4. Configuration and forward kinematics

Problem. An arm's sensors report its joint angles. Where is the gripper?

Geometry. For the fixed-base arms used here, each joint is a one-degree-of-freedom hinge or slider. The **configuration** $\mathbf{q} \in \mathbb{R}^n$ collects one number per joint, an angle for a **revolute** joint (a hinge) and a distance for a **prismatic** joint (a slider); n is the number of degrees of freedom (two for the planar arm, seven for a Franka Panda). Its time derivative $\dot{\mathbf{q}}$, the vector of joint rates, is a different kind of object with different units and is needed from §6 on. The robot is a **kinematic tree**: the links are nodes, the joints are edges, and each edge carries a transform of which exactly one factor depends on one joint value. The pose of any body is the product of the transforms along the unique path from the world to it, and **forward kinematics** is that product for the end-effector:

$$\varphi : \mathbf{q} \mapsto {}^W\mathbf{X}^E(\mathbf{q}) = \mathbf{X}_0 \mathbf{X}_1(q_1) \mathbf{X}_2(q_2) \cdots \mathbf{X}_n(q_n) \mathbf{X}_{n+1}, \quad (6)$$

with constant factors from the robot's geometry and one factor per joint that turns or slides by q_i . The output of (6) is a pose, but a task rarely constrains all six of its freedoms. The **task map** picks out the coordinates the task cares about: for a positioning task it is the translation block of the pose, ${}^W\mathbf{p}^E(\mathbf{q})$, three numbers (two in the plane); for a full pose it is all six freedoms; for pushing a button it may be a position and one direction. The number of coordinates the task constrains is m , the dimension of the **task space**, and the same letter φ names whichever map is in use. For a position task, $\varphi(\mathbf{q}) \in \mathbb{R}^m$. A full pose instead lies in $\text{SE}(3)$: six degrees of freedom do not make its matrix six freely additive coordinates. Section 6 will construct six **instantaneous motion components** for it. Configuration space belongs to the robot; task space belongs to the task.

4.1 Follow one numerical chain from the base to the tip

First fix the question and the input. We know the two joint angles and link lengths. We want the tip's position and orientation in the world. The arm is held at this one configuration:

$$\mathbf{q} = (30^\circ, 45^\circ), \quad l_1 = 0.4 \text{ m}, \quad l_2 = 0.3 \text{ m}.$$

Joint 1 is at the world origin and turns link 1. Joint 2 is at the elbow and turns link 2 **relative to link 1**. Both axes point out of the plane along positive world z . The four steps below walk along that fixed arm. They are a calculation, not a command to turn and translate the physical robot in four successive motions.

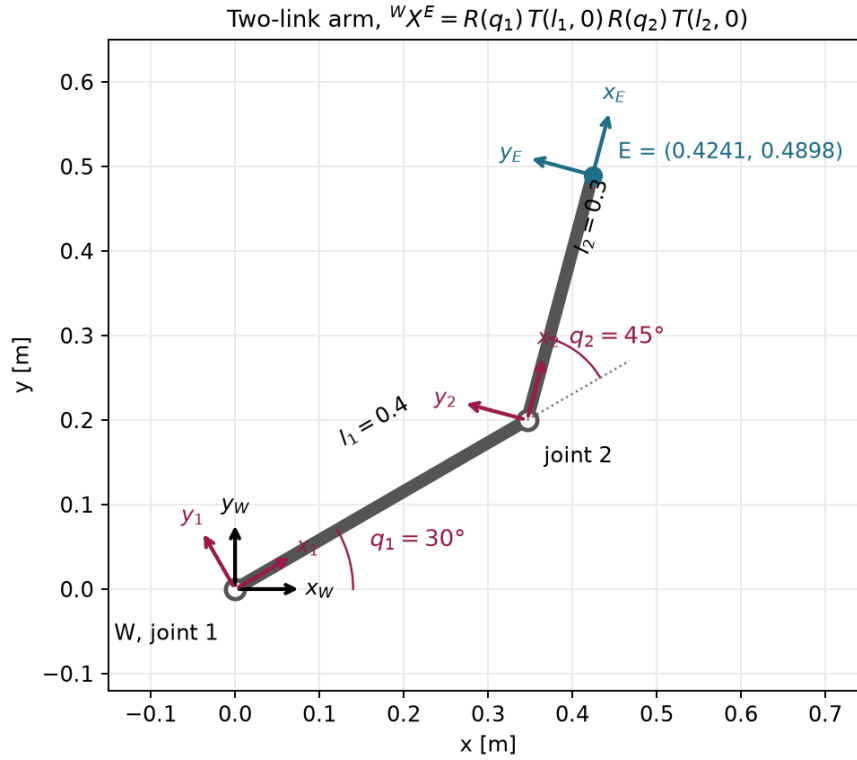


Figure 2. The two-link arm at the worked configuration. Follow the frames from the base, along link 1, around the elbow, and along link 2 to the tip.

Write the small operations before multiplying them. In the plane, a point has two coordinates, so its homogeneous column has three entries. The final entry is one for a point. For this planar example only, $\mathbf{R}(\theta)$ denotes a **homogeneous rotation**, and $\mathbf{T}(a, b)$ denotes a homogeneous translation:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}(a, b) = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}.$$

The rotation block inside these planar poses is two by two. These three-by-three matrices must not be confused with the three-by-three **rotation block of a spatial pose** in §3. We will restore the spatial four-by-four form after completing the planar calculation.

From the base, turn into link 1's direction, advance along link 1, turn into link 2's direction, and advance along link 2. That geometry determines the order:

$${}^W\mathbf{X}^E = \mathbf{R}(q_1) \mathbf{T}(l_1, 0) \mathbf{R}(q_2) \mathbf{T}(l_2, 0).$$

Here are all four numerical factors. Translations are in meters. Printed decimals are rounded; calculate with the unrounded sines and cosines before rounding each result.

$$\mathbf{R}(q_1) \approx \begin{bmatrix} 0.8660 & -0.5000 & 0 \\ 0.5000 & 0.8660 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}(l_1, 0) = \begin{bmatrix} 1 & 0 & 0.4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$\mathbf{R}(q_2) \approx \begin{bmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}(l_2, 0) = \begin{bmatrix} 1 & 0 & 0.3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

One rule explains every update. Suppose the accumulated pose has rotation block $\mathbf{R}_{\text{old}}$ and origin $\mathbf{p}_{\text{old}}$. The next local factor has rotation block $\mathbf{R}_{\text{next}}$ and translation $\mathbf{t}$. Block multiplication gives

$$\begin{bmatrix} \mathbf{R}_{\text{old}} & \mathbf{p}_{\text{old}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\text{next}} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\text{old}}\mathbf{R}_{\text{next}} & \mathbf{p}_{\text{old}} + \mathbf{R}_{\text{old}}\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

The next translation is expressed in the current local axes. Rotate it into world axes before adding it to the current world origin. A pure rotation has zero translation, so it changes heading without moving that origin. A pure translation has identity rotation, so it changes the origin without changing heading.

We write the accumulated result after step s as $\mathbf{X}^{(s)}$, starting with identity. The superscript counts **construction steps**, not joints or physical frames.

Step 1 turns the frame at the base. We append the shoulder rotation because link 1 points 30° from world x . No link length has been traversed yet:

$$\mathbf{X}^{(1)} = \mathbf{R}(q_1) \approx \begin{bmatrix} 0.8660 & -0.5000 & 0 \\ 0.5000 & 0.8660 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The origin is still the base, and the heading is 30° . The first column is the direction along which the next translation will advance.

Step 2 advances to the elbow. The first link is 0.4 m long along the current local x axis. Its contribution in world coordinates is

$$\begin{bmatrix} 0.8660 & -0.5000 \\ 0.5000 & 0.8660 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0.3464 \\ 0.2000 \end{bmatrix} \text{ m}.$$

Therefore the accumulated pose is

$$\mathbf{X}^{(2)} = \mathbf{X}^{(1)}\mathbf{T}(l_1, 0) \approx \begin{bmatrix} 0.8660 & -0.5000 & 0.3464 \\ 0.5000 & 0.8660 & 0.2000 \\ 0 & 0 & 1 \end{bmatrix}.$$

This frame is at the elbow, still aligned with link 1. Adding 0.4 directly to world x would place the elbow incorrectly because the link is turned.

Step 3 turns into link 2's direction. The elbow joint contributes another 45° relative to the incoming link. Multiply its rotation on the right:

$$\mathbf{X}^{(3)} = \mathbf{X}^{(2)} \mathbf{R}(q_2) \approx \begin{bmatrix} 0.2588 & -0.9659 & 0.3464 \\ 0.9659 & 0.2588 & 0.2000 \\ 0 & 0 & 1 \end{bmatrix}.$$

For example, the upper-left entry comes from row 1 times column 1:

$$\cos 30^\circ \cos 45^\circ - \sin 30^\circ \sin 45^\circ = \cos 75^\circ \approx 0.2588.$$

The lower-left entry similarly gives the sine of 75° . The last column of the right factor is $(0, 0, 1)^\top$, so the existing translation column is unchanged. The origin stays at the elbow; the frame now points along link 2 at 75° in the world.

Step 4 advances to the tip. Link 2 is 0.3 m along this new local x axis. Add its world-expressed displacement to the elbow position:

$$\mathbf{p}_{\text{tip}} = \begin{bmatrix} 0.4 \cos 30^\circ \\ 0.4 \sin 30^\circ \end{bmatrix} + \begin{bmatrix} 0.3 \cos 75^\circ \\ 0.3 \sin 75^\circ \end{bmatrix} \approx \begin{bmatrix} 0.4241 \\ 0.4898 \end{bmatrix} \text{ m}.$$

The complete planar pose is

$${}^W\mathbf{X}^E = \mathbf{X}^{(4)} = \mathbf{X}^{(3)} \mathbf{T}(l_2, 0) \approx \begin{bmatrix} 0.2588 & -0.9659 & 0.4241 \\ 0.9659 & 0.2588 & 0.4898 \\ 0 & 0 & 1 \end{bmatrix}.$$

Its last column locates the tip. Its first two columns give the tip frame's axes. The tip's heading is 75° . The last turn did not move the elbow; the final shift did not change heading. That separation is what lets us check each stage of the calculation.

Collect all four factors in one multiplication. The same result can be checked without writing the intermediate poses again:

$$\begin{aligned} {}^W\mathbf{X}^E &= \mathbf{R}(30^\circ) \mathbf{T}(0.4, 0) \mathbf{R}(45^\circ) \mathbf{T}(0.3, 0) \\ &\approx \begin{bmatrix} .8660 & -.5000 & 0 \\ .5000 & .8660 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & .4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} .7071 & -.7071 & 0 \\ .7071 & .7071 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & .3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &\approx \begin{bmatrix} .2588 & -.9659 & .4241 \\ .9659 & .2588 & .4898 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Matrix multiplication is associative, so grouping the factors into successive stages gives this same product. The factor order remains fixed. The decimals shown are rounded; the reported final pose uses the unrounded sines and cosines. Each factor is a planar pose matrix with a two-by-two rotation block and a translation column.

With arbitrary joint angles, the same steps produce the familiar closed form:

$$\begin{aligned} \varphi(\mathbf{q}) &= (l_1 \cos q_1 + l_2 \cos(q_1 + q_2), l_1 \sin q_1 + l_2 \sin(q_1 + q_2)), \\ \text{heading} &= q_1 + q_2. \end{aligned} \quad (7)$$

The frame letters give the same chain. Let frame 1 be at the shoulder after its turn, frame 2 at the elbow after its turn, and E at the tip. Grouping the factors gives

$$\begin{aligned} {}^W\mathbf{X}^1 &= \mathbf{R}(q_1), & {}^1\mathbf{X}^2 &= \mathbf{T}(l_1, 0)\mathbf{R}(q_2), & {}^2\mathbf{X}^E &= \mathbf{T}(l_2, 0). \\ {}^W\mathbf{X}^E &= {}^W\mathbf{X}^1 {}^1\mathbf{X}^2 {}^2\mathbf{X}^E. \end{aligned}$$

The unturned intermediate elbow frame after step 2 helps the arithmetic; it is not an additional actuated joint. This grouping also explains equation (6): its initial constant factor is identity here, and its two joint-dependent factors can be the first two grouped transforms above.

Building the product and applying it are different readings of the same equation. We built the chain from base to tip by appending factors on the right. To convert the tip's own origin to world coordinates, the rightmost matrix acts on its coordinate column first:

$$\begin{aligned} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} &\xrightarrow{\mathbf{T}(l_2, 0)} \begin{bmatrix} 0.3 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{\mathbf{R}(q_2)} \begin{bmatrix} 0.2121 \\ 0.2121 \\ 1 \end{bmatrix}. \\ \begin{bmatrix} 0.2121 \\ 0.2121 \\ 1 \end{bmatrix} &\xrightarrow{\mathbf{T}(l_1, 0)} \begin{bmatrix} 0.6121 \\ 0.2121 \\ 1 \end{bmatrix} \xrightarrow{\mathbf{R}(q_1)} \begin{bmatrix} 0.4241 \\ 0.4898 \\ 1 \end{bmatrix}. \end{aligned}$$

The displayed coordinates are rounded. Both readings give the same tip. We have neither reversed the physical chain nor changed the convention for matrix multiplication.

Restore the spatial pose before moving to a real arm. The planar arm lies at world height zero, and its z axis is unchanged. Insert that unchanged coordinate to obtain

$${}^W\mathbf{X}_{3D}^E \approx \begin{bmatrix} 0.2588 & -0.9659 & 0 & 0.4241 \\ 0.9659 & 0.2588 & 0 & 0.4898 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A spatial arm uses this four-by-four format at every edge, including joints whose axes point in different directions. The composition rule is unchanged. The robot description supplies the constant offsets and axes; the measured joint angles supply the variable rotations.

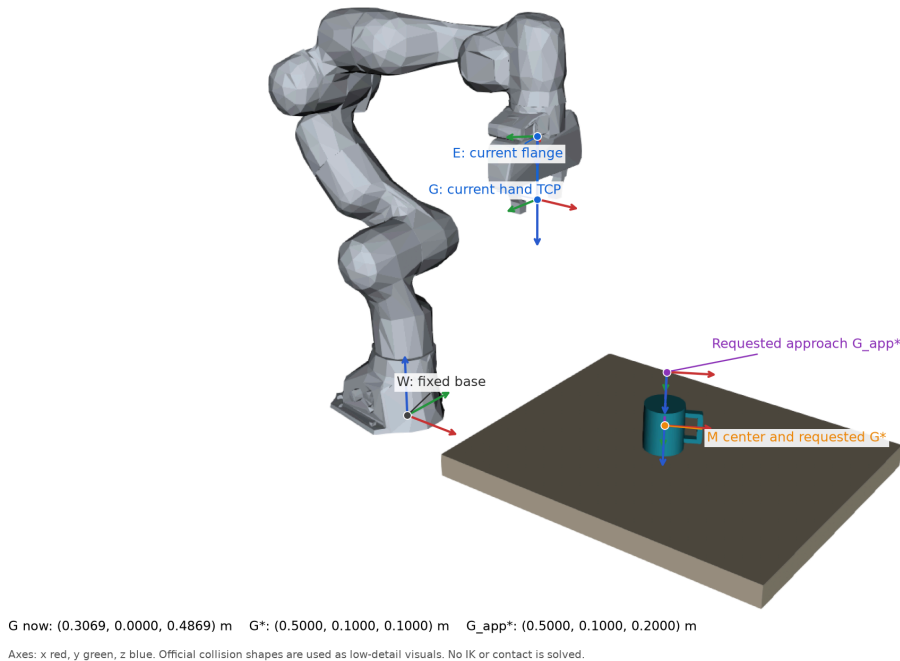
Run `lectures/02-D2-transforms-fk-jacobian/code/fk_chain_walkthrough.py` to print the factors, intermediate products and coordinate checks without rounding during multiplication.

4.2 Use the same construction on the FR3

Keep the question the same. Seven measured joint angles now replace the planar arm's two. We still multiply relative transforms to find the pose of a named frame. This example uses the official **Franka Research 3 (FR3)** description, release 2.9.0, pinned in `drake/01c-fr3-grasp-chain/`. The older Panda examples elsewhere in the course keep their own model and numbers. This is a simulated nominal FR3, not a measurement of the lab robot.

FR3 model at the exact ready configuration

Current pose comes from the joints. The mug pose and grasp choice supply the targets.



The FR3 at the ready configuration. The current flange and tool center are blue; the supplied grasp and approach targets are orange and purple. The targets will be constructed in §5.

The world frame W coincides with the fixed robot base. The model's flange frame is E . The tool-center frame between the fingers is G , so G is the point we eventually want to place for a grasp. The fingers remain open in this lesson; they do not change the flange-to-tool transform. The input arm configuration is

$$\mathbf{q} = \left(0, -\frac{\pi}{4}, 0, -\frac{3\pi}{4}, 0, \frac{\pi}{2}, \frac{\pi}{4} \right),$$

or, for reading the diagram, zero, -45° , zero, -135° , zero, 90° and 45° . The executable example uses the exact radian values, not the three-decimal approximation of this pose.

Name the links and joints before using them. Link 0 is the fixed base. Joint i turns link i relative to link $i - 1$, and frame i is attached to link i at the axis of joint i . So joint 2 joins links 1 and 2 and has angle q_2 , and frame 2 moves with link 2. The flange frame E is fixed to link 7, and the tool-center frame G is fixed to the flange.

Read one edge before reading seven. In this model each arm joint turns about its own local z axis. That does not mean all seven axes point along world z : fixed rotations between the link frames change their world directions. For joint i , the model first supplies a constant pose and then the joint contributes a rotation:

$${}^{i-1}\mathbf{X}^i(q_i) = \begin{bmatrix} \mathbf{R}_x(\alpha_i) & \mathbf{t}_i \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_z(q_i) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

Here the numbered frames are the FR3 model's link frames. The constant translation $\mathbf{t}_i$ is expressed in the parent link's axes. The constant angle α_i orients the next joint frame; it is part of the robot's construction, not a commanded angle. All nonzero fixed arm rotations in this particular description are about parent x , which is why this formula suffices. A different robot may need a general constant rotation block.

The pinned model supplies such constants for every joint, and Appendix D lists all seven rows. The worked edge below names each constant where it is used; the numbers supply the same small operations we wrote by hand for the planar arm and are not dimensions to memorize. After link 7, a fixed shift of 0.107 m along its local z reaches the flange E .

The complete flange chain is therefore

$${}^W\mathbf{X}^E(\mathbf{q}) = {}^W\mathbf{X}^{00}\mathbf{X}^1(q_1){}^1\mathbf{X}^2(q_2)\cdots{}^6\mathbf{X}^7(q_7){}^7\mathbf{X}^E,$$

where the world-to-base factor is identity and the last factor is that fixed flange shift. The script prints each factor and every cumulative matrix. Work through joint 2 first. Joint 1 has zero angle and places its origin 0.333 m above the base, with identity rotation. Joint 2 has zero fixed translation. Its two rotations therefore give the following factor, where c is exactly the square root of two divided by two:

$${}^1\mathbf{X}^2(-\pi/4) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{fixed } \mathbf{R}_x(-\pi/2)} \underbrace{\begin{bmatrix} c & c & 0 & 0 \\ -c & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{joint } \mathbf{R}_z(-\pi/4)} = \begin{bmatrix} c & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c & -c & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The first factor's rows select the second factor's first row, third row, and negative second row, respectively. This accounts for each resulting rotation entry. Prepending joint 1's pose preserves this rotation and adds the base height. The full pose-matrix multiplication is

$$\begin{aligned}
{}^W\mathbf{X}^2 &= {}^W\mathbf{X}^{1\ 1}\mathbf{X}^2 \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.333 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c & -c & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c & -c & 0 & 0.333 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad c = \frac{\sqrt{2}}{2}.
\end{aligned}$$

The first matrix's top-left identity block copies the child rotation. Its third row adds the base height to the child's zero translation. For example, entry (3, 4) is

$$0 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0.333 \cdot 1 = 0.333.$$

After joint 2 the cumulative rotation and position are

$${}^W\mathbf{R}^2 \approx \begin{bmatrix} 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \\ 0.7071 & -0.7071 & 0 \end{bmatrix}, \quad {}^W\mathbf{p}^2 = \begin{bmatrix} 0 \\ 0 \\ 0.333 \end{bmatrix} \text{ m.}$$

The next joint's origin has parent-frame offset $(0, -0.316, 0)$. Apply exactly the update used at the planar elbow:

$${}^W\mathbf{p}^3 = {}^W\mathbf{p}^2 + {}^W\mathbf{R}^2 \begin{bmatrix} 0 \\ -0.316 \\ 0 \end{bmatrix} \approx \begin{bmatrix} -0.2234 \\ 0 \\ 0.5564 \end{bmatrix} \text{ m.}$$

The constant rotation of the next frame changes its axes but does not add another shift to this origin. Continuing the same multiplication through joints 4 to 7 and the flange shift reaches the flange. Appendix D lists every intermediate origin as a checkpoint for your own multiplication. The complete flange matrix contains both the final position and orientation:

$${}^W\mathbf{X}^E \approx \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0.3069 \\ -0.7071 & -0.7071 & 0 & 0 \\ 0 & 0 & -1 & 0.5903 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Take the last step to the point the task uses. The flange is the mounting frame. The fingers' tool center is displaced from it and has a fixed orientation relative to it. The manufacturer's hand and tool-center frames give

$${}^E\mathbf{X}^G \approx \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0.1034 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This combines a -45° turn about flange z and an offset of 0.1034 m along that axis. Compose it with the current flange pose:

$${}^W\mathbf{X}^G(\mathbf{q}) = {}^W\mathbf{X}^E(\mathbf{q}) {}^E\mathbf{X}^G \approx \begin{bmatrix} 1 & 0 & 0 & 0.3069 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.4869 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In particular, the current flange z axis points down in the world. Thus a positive local 0.1034 m offset moves the tool center **down**, giving the world height

$$0.5902820523 - 0.1034 = 0.4868820523 \text{ m}.$$

Adding 0.1034 directly to world height would put the task point on the wrong side of the flange. This is the same local-versus-world translation issue as the two-link example.

Run `drake/01c-fr3-grasp-chain/fr3_grasp_chain.py` to reproduce every factor and cumulative matrix; add `--meshcat` to change the joints and follow the chain. Appendix D records how the script is checked against Drake and what its viewer controls show.

What kind of function φ is. Deterministic and cheap: one matrix product per joint. Smooth: sums of products of sines and cosines, differentiable to every order. And a *program*: a chain of small differentiable operations composed in a fixed order, which is exactly a computational graph, so it can be placed inside a network's forward pass and differentiated through.

Why it appears in robot learning. Demonstrations are recorded as joint angles; the end-effector positions a loss or a plot needs are recomputed from them by φ . “Differentiable kinematics” or a “kinematic loss” in a paper means φ used as a layer. And a policy that emits joint positions does not require an inverse-kinematics conversion of that command. Its controller or safety checks may still use kinematics. A policy that emits end-effector poses usually relies on the robot-side software to solve the inverse problem (§5, §7).

Must understand: $\mathbf{q}$ and n ; task space and m ; forward kinematics as the product of one transform per joint, $\mathbf{q} \mapsto {}^W\mathbf{X}^E(\mathbf{q})$; the two-link closed form (7), obtained by checking each intermediate pose; the difference between building the chain and applying it to coordinates; the fixed flange-to-tool transform when the task point is between the fingers; that φ is smooth, cheap and differentiable.

Recognize: the kinematic tree; revolute versus prismatic joints; URDF and `.g` files as the tree written down; how the model supplies constant offsets and joint axes for the FR3 chain (Appendix D lists them).

Deferred: the general camera branch and tree recursion, the torus structure of configuration space (long reading §5, §6).

5. Inverse kinematics: the problem

5.1 The question, and a concrete target on the FR3

Problem. Inverse kinematics (IK) starts with a target, a position $\mathbf{y}^*$ or a full pose. Which $\mathbf{q}$ puts the end-effector there? Forward kinematics gives one pose for each configuration. For one target, the inverse problem can have no answer, one answer, several answers or a continuum of answers, so in general it is not a function. A solver's behavior follows from these outcomes, so we state the problem before the solver. We first construct a concrete target on the FR3, then use the planar arm to see the outcomes.

Construct a target before asking for joints. Continue the FR3 example with an upright mug. Its frame M is at the mug's center, with its x axis pointing toward the handle. Suppose the mug center is at $(0.50, 0.10, 0.10)$ m in the world and its heading is 30° . Those are supplied scene data. In a later perception lesson, an estimate from images will take their place.

$${}^W\mathbf{X}^M \approx \begin{bmatrix} 0.8660 & -0.5000 & 0 & 0.50 \\ 0.5000 & 0.8660 & 0 & 0.10 \\ 0 & 0 & 1 & 0.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We also supply a grasp choice. Place the tool center at the mug center, point its z axis down, and align its x axis with the handle direction. Call this desired tool frame G^* . Their origins and positive x -axes coincide, while their y - and z -axes point in opposite directions. A half-turn about mug x expresses the grasp choice:

$${}^M\mathbf{X}^{G^*} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The first transform places the mug; the second places the desired tool relative to that mug. The adjoining M labels match, so composition gives the desired world pose:

$${}^W\mathbf{X}^{G^*} = {}^W\mathbf{X}^M {}^M\mathbf{X}^{G^*} \approx \begin{bmatrix} 0.8660 & 0.5000 & 0 & 0.50 \\ 0.5000 & -0.8660 & 0 & 0.10 \\ 0 & 0 & -1 & 0.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This calculation specifies a pose. It has not shown that the fingers can grasp the mug, that the arm can reach it, or that the motion avoids collisions. Those are separate tests.

1. Choose the approach offset. Choose a pose 0.10 m above the grasp with the same orientation. Because desired tool z points down, the offset expressed in its own axes is negative along z :

$${}^{G^*}\mathbf{X}^{G_{\text{app}}^*} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2. Compose the poses. Append that local translation to the world grasp pose. The inner frame labels match, just as they did in the forward-kinematics chain:

$${}^W\mathbf{X}^{G_{\text{app}}^*} = {}^W\mathbf{X}^{G^*} {}^{G^*}\mathbf{X}^{G_{\text{app}}^*}.$$

$$\begin{aligned}
& \underbrace{\begin{bmatrix} 0.8660 & 0.5000 & 0 & 0.50 \\ 0.5000 & -0.8660 & 0 & 0.10 \\ 0 & 0 & -1 & 0.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{W \mathbf{X}^{G^*}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{G^* \mathbf{X}^{G_{\text{app}}^*}} \\
& \approx \underbrace{\begin{bmatrix} 0.8660 & 0.5000 & 0 & 0.50 \\ 0.5000 & -0.8660 & 0 & 0.10 \\ 0 & 0 & -1 & 0.20 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{W \mathbf{X}^{G_{\text{app}}^*}}.
\end{aligned}$$

The third-row, fourth-column entry comes from row 3 of the left factor and column 4 of the right factor:

$$0 \cdot 0 + 0 \cdot 0 + (-1)(-0.10) + 0.10 \cdot 1 = 0.20.$$

So the approach height is 0.20 m. The rotation stays unchanged because the appended rotation is the identity. The matrices show rounded entries; the code uses the exact rotations. Choosing these two target poses still does not produce a path between them.

3. Ask for the joints. In §4 we knew the joints and computed the current tool pose. For the approach, we now know the desired pose and seek joints satisfying

$$W \mathbf{X}^G(\mathbf{q}) = W \mathbf{X}^{G_{\text{app}}^*}.$$

Reaching the grasp itself is a separate request:

$$W \mathbf{X}^G(\mathbf{q}) = W \mathbf{X}^{G^*}.$$

At the ready configuration, the current tool center is approximately (0.3069, 0, 0.4869) m. Its remaining world displacement to the grasp center is

$$\mathbf{e}_{\text{pos}} = {}^W \mathbf{p}^{G^*} - {}^W \mathbf{p}^G(\mathbf{q}) \approx \begin{bmatrix} 0.1931 \\ 0.1000 \\ -0.3869 \end{bmatrix} \text{ m.}$$

The orientations differ too, so matching that position alone would not solve the full-pose task. Compare the same physical frame on both sides: current G with desired G^* . If a solver instead reports flange E , convert the target with the known tool transform:

$$W \mathbf{X}^{E^*} = W \mathbf{X}^{G^*} ({}^E \mathbf{X}^G)^{-1}.$$

We have specified the problem, not a solution algorithm. The planar geometry below explains why solutions can be absent or nonunique. The Jacobian in §6 will then describe how a small joint change affects the current pose, and Optimization I will

derive an iterative step.

5.2 How many answers: the planar geometry

Geometry. For the two-link arm the set of reachable tip positions, the **workspace**, is the ring between radii $l_1 - l_2 = 0.1$ m and $l_1 + l_2 = 0.7$ m (Figure 3). This ideal workspace ignores joint limits and obstacles and identifies angles differing by a full turn. A target outside the ring, including its central hole, has **no** solution. A target on its outer circle has exactly **one**, the arm stretched straight. A target strictly between the two circles has **two**, the elbow bent one way or the other: for $\mathbf{y}^* = (0.5, 0.2)$ they are $\mathbf{q} = (-11.52^\circ, +80.41^\circ)$ and $(55.12^\circ, -80.41^\circ)$. For a reachable target away from singularities and active joint limits, an arm with more joints than independent task constraints ($n > m$) can have a **continuum** of solutions: a Panda ($n = 7$) holding its gripper fixed ($m = 6$) can still swing its elbow along a one-dimensional curve of configurations. This is **redundancy relative to a task**. The same Panda with only its fingertip position fixed has three constraints and, at full task rank, four local freedoms remaining. The two-link arm's two-position task has $n = m = 2$: its two discrete solutions are not redundancy. (The Drake demo `drake/01b-redundancy-self-motion` shows it: one slider moves a Panda along that curve while its gripper stays on a mug.)

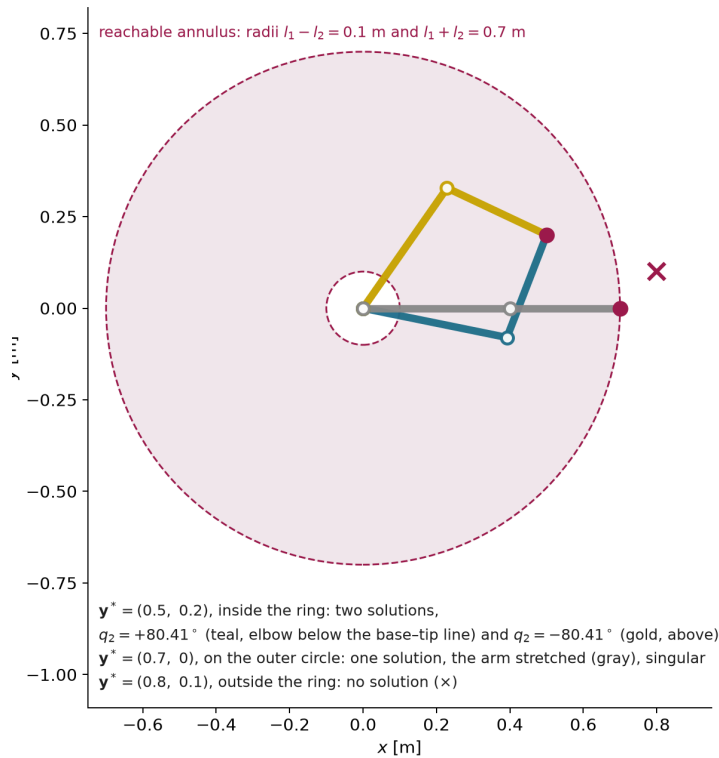


Figure 3. The reachable ring of the two-link arm and three targets: inside, two solutions (teal and gold, distinguished by the sign of q_2); on the outer circle, one solution, the stretched arm; outside, none.

5.3 How we will solve it, and the objective

IK names a problem, not a particular algorithm. Some robot geometries admit analytic formulas. The two-link arm is one example. The long reading derives its joint angles in §7.2, as equations (22)–(23), from the law of cosines stated in its §7.1. Such formulas are useful, but depend on that geometry and do not by themselves check a collision-free path.

This course develops **numerical IK** because a common formulation can use different arms' FK maps and accommodate posture preferences, joint limits and clearance constraints. We still derive the mathematics, but through local models and optimization. First, the Jacobian predicts what a small joint change will do. Optimization I derives how to choose a change that reduces target error. Optimization II adds constraints. The differential-IK calculation in §7 instead selects joint velocities for a requested instantaneous tip velocity; repeated Jacobian-based updates can also improve a joint-position guess for a fixed target. These uses are related, and neither guarantees convergence from every starting guess.

Statement as an optimization. For a position target, one useful numerical formulation minimizes the squared target error:

$$\min_{\mathbf{q}} \|\varphi(\mathbf{q}) - \mathbf{y}^*\|^2, \quad (8)$$

“the configuration whose end-effector is as close as possible to the target”. Here φ is the position map, and the squared norm adds the squared coordinate errors. For the planar arm this is $(\varphi_x(\mathbf{q}) - y_x^*)^2 + (\varphi_y(\mathbf{q}) - y_y^*)^2$. For a full-pose task, two different objects appear. A scalar objective scores the mismatch, for example §3's squared position error plus a weighted squared rotation angle, or a loss on matrix entries; any such scalar can be minimized. A local update with the geometric Jacobian of §6 instead needs a geometric error whose components match the Jacobian's frame and stacking order, the angular part first and the position part second, so that the Jacobian relates it to a small joint change to first order. A method that linearizes an orientation-error residual, as Optimization I's Newton or Gauss–Newton step would when applied to a full pose, needs the derivative of that residual with respect to the joint angles. Near zero error that derivative is approximated by the angular rows of the geometric Jacobian with matching sign and frame; a feedback command built directly from the geometric Jacobian needs no such derivative. Appendix B.2 gives the conventions.

A best approximation exists when the allowed configurations form a nonempty closed, bounded set and the objective is continuous. Finding that global best point is a separate problem: a local solver can stop at a worse one, and different starting guesses can find different solutions. If joint-limit and obstacle constraints leave no feasible configuration, there is no feasible answer at all.

A preference for a posture $\mathbf{q}_0$ can be added as a weighted term $\|\mathbf{q} - \mathbf{q}_0\|^2$, but it trades target accuracy against that preference. Here $\mathbf{q}_0$ is a fixed preferred posture. It differs from the starting guess of an iteration, which Deck 04 writes $\mathbf{q}^{(0)}$. To choose a posture **without sacrificing an exactly reachable target**, keep the target as an equality constraint or use a task-preserving secondary motion (§7). The optimization block develops these choices; §7 first introduces the local step they use.

Why it appears in robot learning. A policy that outputs a gripper pose delegates a kinematic conversion to the robot. A policy that outputs joint positions must instead produce useful joint commands for its training embodiment; this does not prove it has learned a general IK solver. The non-uniqueness matters for learning: two demonstrations of the same gripper motion may use different elbows, and a regression onto joint angles can average two valid answers into a configuration that misses the target. The average of the two angle pairs above is approximately $(21.80^\circ, 0^\circ)$: the arm is straight and reaches radius 0.7 m, beyond the target's radius of about 0.539 m. Learned IK samplers and learned initial guesses for (8) appear later in the course.

Must understand: constructing a desired tool pose by composing the object pose with a supplied grasp, and the approach offset written in the tool's own axes; the inverse problem and its three outcomes (none, one, many); redundancy $n > m$ and the local continuum of solutions at full task rank; (8) as a position-error objective.

Recognize: the workspace; that some arms admit closed-form IK but numerical methods are general; “elbow up/down” as the two-solution case.

Deferred: the law-of-cosines solution (22)–(23) of the long reading §7.2, and the count of degrees of freedom (long reading §7).

6. The Jacobian

6.1 The columns of the position Jacobian

From position to local motion. Forward kinematics answers “where is the hand for this $\mathbf{q}$?” Now imagine holding the arm at that configuration and nudging each joint in turn, leaving the others fixed. Each nudge moves the tip along a particular direction. A table of these effects tells us how to combine joint motions to move the hand where we want it to go. This is the **Jacobian**. The picture is local: after the arm moves, the directions generally change.

Recall the scalar derivative: a small input change produces $\Delta y \approx (dy/dx)\Delta x$. For a vector task $\mathbf{x} = \varphi(\mathbf{q})$, the same idea gives

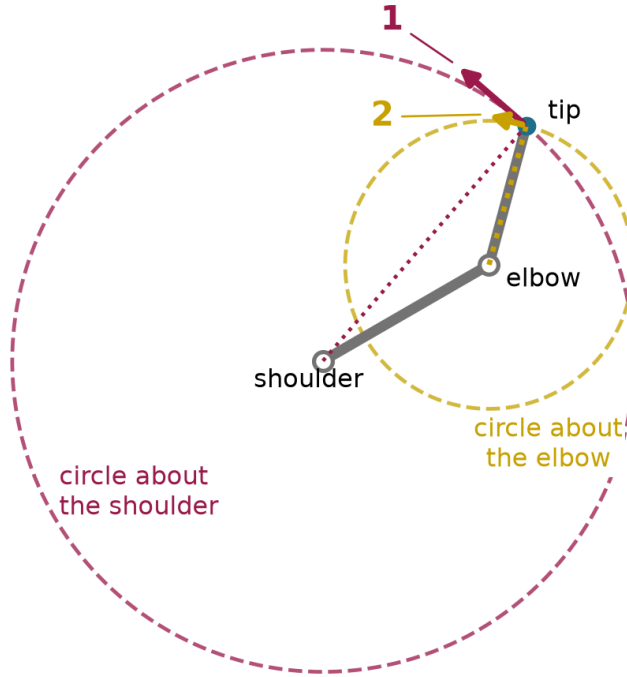
$$\Delta \mathbf{x} := \varphi(\mathbf{q} + \Delta \mathbf{q}) - \varphi(\mathbf{q}) \approx \mathbf{J}(\mathbf{q})\Delta \mathbf{q}.$$

Here $J_{ij} = \partial\varphi_i/\partial q_j$ records how sensitive task component i is to joint j . The matrix has m rows for m task components and n columns for n joints. First we will construct it for the two-link arm's position; then we will add orientation.

One joint at a time: the columns. At $(30^\circ, 45^\circ)$, the two-link tip is at $\mathbf{p} = (0.4241, 0.4898)$ m. Turning **joint 1** alone rotates the whole arm rigidly about the base, so the tip follows a circle of radius $\|\mathbf{p}\| = 0.648$ m. For a small positive turn, the displacement is tangent to that circle, perpendicular to the base–tip line. Per radian of joint motion, its first-order displacement is the arrow $(-0.4898, 0.4241)$ m/rad. This tangent is constructed by rotating the **shoulder-to-tip radius vector** counterclockwise by 90° ; the tip itself does not move through that angle. Turning **joint 2** alone leaves link 1 fixed and rotates link 2 about the elbow. The circle now has radius $l_2 = 0.3$ m, and its tangent is perpendicular to link 2, which points at 75° . The corresponding arrow is $(-0.2898, 0.0776)$ m/rad.

Each column's magnitude equals the radius of its circle. A larger radius gives more tip motion for the same small joint turn. Multiply a column by that joint's small angle change to predict a displacement, or by its angular rate to obtain the instantaneous velocity. The other joint is held fixed in each case.

Both tangents start at the current tip



When only joint 1 turns, the tip follows the maroon circle about the shoulder. When only joint 2 turns, it follows the gold circle about the elbow. Arrows 1 and 2 show the positive tangents at the current tip, the two Jacobian columns scaled by 0.3 radians for display. They are not exact finite displacements along the circles.

Putting these arrows side by side gives the **position Jacobian**, measured in meters per radian:

$$\mathbf{J}_{\text{pos}}(\mathbf{q}) = \begin{bmatrix} \partial x / \partial q_1 & \partial x / \partial q_2 \\ \partial y / \partial q_1 & \partial y / \partial q_2 \end{bmatrix}.$$

Differentiate (7) before substituting numbers. The position map (7) is

$$x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2), \quad y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2).$$

For the first column, hold q_2 fixed and differentiate with respect to q_1 . Both terms depend on q_1 . The second does so through the sum $q_1 + q_2$, and the derivative of that sum with respect to q_1 is 1, so the chain rule contributes a factor of one and nothing else:

$$\frac{\partial x}{\partial q_1} = -l_1 \sin q_1 - l_2 \sin(q_1 + q_2), \quad \frac{\partial y}{\partial q_1} = l_1 \cos q_1 + l_2 \cos(q_1 + q_2).$$

For the second column, hold q_1 fixed and differentiate with respect to q_2 . Only the l_2 terms depend on q_2 , again through the sum, whose derivative with respect to q_2 is also 1:

$$\frac{\partial x}{\partial q_2} = -l_2 \sin(q_1 + q_2), \quad \frac{\partial y}{\partial q_2} = l_2 \cos(q_1 + q_2).$$

Together,

$$\mathbf{J}_{\text{pos}}(\mathbf{q}) = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix}.$$

The first column is the tip position (x, y) turned by 90° , the tangent to the circle about the base; the second is link 2's vector turned by 90° , the tangent to the circle about the elbow. The symbolic matrix says what the arrows said.

To evaluate it at the worked pose, substitute the link lengths and joint angles into these derivatives. The link lengths are 0.4 m and 0.3 m; the first joint is at 30° , and the two joint angles add to 75° :

$$\begin{aligned} \mathbf{J}_{\text{pos}} &= \begin{bmatrix} -0.4 \sin 30^\circ - 0.3 \sin 75^\circ & -0.3 \sin 75^\circ \\ 0.4 \cos 30^\circ + 0.3 \cos 75^\circ & 0.3 \cos 75^\circ \end{bmatrix} \\ &\approx \begin{bmatrix} -0.4898 & -0.2898 \\ 0.4241 & 0.0776 \end{bmatrix}. \end{aligned} \tag{9}$$

The degree labels identify the configuration and nothing else. The derivatives hold for angles in radians, because the derivative of the sine is the cosine only when the angle is in radians, so the units of the entries stay meters per radian.

Reading the rows. Column 1 describes both position components when joint 1 moves; row 1 collects both joints' contributions to the x displacement. In full,

$$\begin{aligned} \Delta x &\approx -0.4898 \Delta q_1 - 0.2898 \Delta q_2, \\ \Delta y &\approx 0.4241 \Delta q_1 + 0.0776 \Delta q_2. \end{aligned}$$

For example, $\Delta \mathbf{q} = (0.01, 0)$ rad predicts $\Delta \mathbf{p} \approx (-0.004898, 0.004241)$ m: about 4.90 mm left and 4.24 mm up. The two equations also show how simultaneous joint motions combine: multiply each column by its joint's change, then add the resulting arrows.

6.2 From displacement to velocity

Over a short interval Δt , divide the displacement relation by Δt . As $\Delta t \rightarrow 0$, the ratios become velocities, and the approximation becomes the exact instantaneous relation

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad \mathbf{J}(\mathbf{q}) = \frac{\partial \varphi}{\partial \mathbf{q}} \in \mathbb{R}^{m \times n}. \quad (10)$$

This velocity calculation is **differential forward kinematics**: the current joint positions and joint velocities are given, and the task velocity is the output. It uses matrix multiplication. This is the chain rule for $\mathbf{x}(t) = \varphi(\mathbf{q}(t))$. For the position task, $\mathbf{J} = \mathbf{J}_{\text{pos}}$: each column is also the tip velocity produced by moving that joint at 1 rad/s while holding the others still. The same numbers in (9) therefore map joint rates in rad/s to tip velocity in m/s. Differentiating the position map (7) gives (9).

For example, let joint 1 turn at 0.01 rad/s while joint 2 stays still. At the worked configuration, multiply the same position Jacobian by those angular velocities:

$$\dot{\mathbf{p}} = \mathbf{J}_{\text{pos}}(\mathbf{q})\dot{\mathbf{q}} \approx \begin{bmatrix} -0.4898 & -0.2898 \\ 0.4241 & 0.0776 \end{bmatrix} \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.004898 \\ 0.004241 \end{bmatrix} \text{ m/s}.$$

The tip moves left at about 4.90 mm/s and up at about 4.24 mm/s at that instant. This uses the same coefficients as the displacement prediction above, with velocity units. The instantaneous velocity relation is exact; the displayed decimals are rounded. A finite change in joint positions still gives only an approximate displacement prediction.

6.3 Adding orientation in the plane

Imagine a gripper fixed to the end of link 2. Moving its origin to the right place does not necessarily point its fingers the right way. We therefore describe its planar pose with three task coordinates:

$$\begin{bmatrix} \theta \\ x \\ y \end{bmatrix}, \quad \theta = q_1 + q_2.$$

The first entry is its heading; the other two are its position coordinates. This is not a point in 3D. It describes one angle and a position in the plane.

Differentiate the heading with respect to each joint, holding the other joint fixed:

$$\frac{\partial \theta}{\partial q_1} = 1, \quad \frac{\partial \theta}{\partial q_2} = 1.$$

The dotted joint coordinates below are **joint angular velocities**, measured in rad/s. “Joint rates” is a shorter way to refer to these time derivatives:

$$\dot{q}_i = \frac{dq_i}{dt}.$$

Now apply the chain rule to obtain the heading’s rate:

$$\begin{aligned}\dot{\theta} &= \frac{\partial \theta}{\partial q_1} \dot{q}_1 + \frac{\partial \theta}{\partial q_2} \dot{q}_2 \\ &= \dot{q}_1 + \dot{q}_2.\end{aligned}$$

Stack the heading derivatives above the position derivatives already computed in (9):

$$\mathbf{J}_{\text{full}} = \begin{bmatrix} \partial \theta / \partial q_1 & \partial \theta / \partial q_2 \\ \partial x / \partial q_1 & \partial x / \partial q_2 \\ \partial y / \partial q_1 & \partial y / \partial q_2 \end{bmatrix}.$$

Multiplying each row by the same two joint rates gives the three motion components. At the worked pose,

$$\begin{bmatrix} \dot{\theta} \\ \dot{x} \\ \dot{y} \end{bmatrix} \approx \underbrace{\begin{bmatrix} 1 & 1 \\ -0.4898 & -0.2898 \\ 0.4241 & 0.0776 \end{bmatrix}}_{\mathbf{J}_{\text{full}} \text{ at } (30^\circ, 45^\circ)} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}.$$

The first row reports angular velocity in rad/s; the other two report linear velocity in m/s. Thus planar rigid-body motion needs **one angular plus two linear components**. The robot still has two joints, so the matrix still has two columns. The same two joint rates set all three outputs. At this pose, the position Jacobian is invertible: choosing the tip velocity fixes both joint rates, which also fixes their sum, the heading rate. There is no extra joint rate to adjust the heading independently.

We can see why using the two tip-motion arrows in the circle diagram. At this pose they are not parallel, so their weighted sum can vanish only if both weights are zero. The weights are the joint angular velocities. A still tip therefore requires both joint angular velocities to be zero, and their sum gives zero gripper angular velocity. This argument is specific to regular position configurations; at a singularity the arrows can become parallel.

For example, rates of 1 rad/s and -1 rad/s keep the heading fixed, but move the tip:

$$\begin{bmatrix} \dot{\theta} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \mathbf{J}_{\text{full}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \approx \begin{bmatrix} 0 \\ -0.2000 \\ 0.3464 \end{bmatrix}.$$

This evaluation uses unrounded coefficients. More generally, holding heading fixed requires the joint rates to cancel. Substituting that condition into the position rows gives

$$\dot{q}_2 = -\dot{q}_1, \quad \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = l_1 \begin{bmatrix} -\sin q_1 \\ \cos q_1 \end{bmatrix} \dot{q}_1.$$

This velocity vanishes only when both joint rates vanish, so the tip and heading can both stay fixed only if the arm stays still. At the worked pose, keeping the tip still also prevents turning the gripper in place. The arm cannot produce every independently requested triple of heading and position rates; it is **kinematically deficient for this task**.

6.4 Spatial velocity and the angular block

From planar motion to motion in 3D. Picture carrying a mug: you can move its center while keeping it upright, or turn it while keeping its center in place. Describing a rigid gripper's motion similarly requires both the velocity of its origin and how it is turning. A particular robot may couple these motions, just as the two-link arm does; the description itself still needs both parts.

Reaching the right point is not enough for a grasp. The fingers must also face the intended grasp direction. Angular velocity describes how we turn them toward that orientation.

The origin's linear velocity is $\mathbf{v} = (v_x, v_y, v_z)$. Instantaneous rotation is described by the **angular-velocity vector** $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$: when nonzero, its direction gives the instantaneous rotation axis by the right-hand rule, and its length gives angular speed in rad/s. Its three components describe one instantaneous rotation, not three successive finite rotations. Together with $\mathbf{v}$, it determines the velocity of every point fixed in the rigid gripper.

Point your right thumb along the vector; your curled fingers indicate the rotation direction. The vector describes rotation at the current instant, rather than the gripper's orientation:

$$\text{angular speed} = \|\boldsymbol{\omega}\|.$$

Its direction does not locate an axis line in space. If the vector is zero, there is no instantaneous rotation and no unique axis direction. For the planar arm, all turning is about the world z-axis, so the scalar heading rate fits into the 3D vector as

$$\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}.$$

Positive heading rate points along positive z ; negative heading rate reverses that direction.

A rotation matrix $\mathbf{R}$ has nine entries, but its columns must remain orthonormal, so those entries cannot change independently. Its derivative $\dot{\mathbf{R}}$ exists, but its nine entries are not nine independent rotational velocities. The vector $\boldsymbol{\omega}$ describes those changes with three numbers. It is generally **not** the vector of derivatives of three Euler angles. The single heading derivative $\dot{\theta}$ works in the plane because all rotations there share one fixed axis; that shortcut does not extend to arbitrary 3D rotations.

We can now stack the **three angular plus three linear components**. In this course the result is called the **spatial velocity**, with angular velocity first:

$${}^W\mathbf{V}^E = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \in \mathbb{R}^6.$$

Here both vectors describe the end-effector E relative to the fixed world W , expressed in world axes, and $\mathbf{v}$ is specifically the velocity of E 's **origin**. Papers also use “twist”, “spatial twist”, and “body twist”; check their frame, reference point, and stacking order before comparing numbers. These names do not all imply the convention used here.

The six-row geometric Jacobian. We know the current joint positions and the joint velocities. We want to predict how the gripper will turn and how its origin will move. Each Jacobian column gives those six velocity components when its joint moves at unit rate and the other joints stay still. Multiply each column by its joint velocity and add the contributions. This is **differential forward kinematics**. For the turning joints in our arm examples, the inputs are joint angular velocities; a sliding joint would instead have a linear velocity.

At the current configuration, the two blocks determine both parts linearly:

$$\boldsymbol{\omega} = \mathbf{J}_{\text{ang}}(\mathbf{q})\dot{\mathbf{q}}, \quad \mathbf{v} = \mathbf{J}_{\text{pos}}(\mathbf{q})\dot{\mathbf{q}}.$$

Constructing the angular block from the joint axes. At the current joint angles, express each hinge's positive rotation-axis direction in world coordinates. The existing forward-kinematics chain supplies the required rotation matrices. This converts the coordinates of a direction; it does not compute a point on the axis line.

In our FR3 model, each hinge's local unit axis is $\mathbf{z}$. Its child-frame rotation therefore gives its world-axis direction by

$${}^W\hat{\mathbf{z}}_i = {}^W\mathbf{R}^i(\mathbf{q}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

This is the third column of that rotation matrix. The hat marks a unit vector. The subscript identifies the joint; it does not mean the joint points along world $\mathbf{z}$. For a model with a different declared local axis, rotate that unit vector instead.

Freeze all other joints and turn hinge i at one radian per second. The end-effector turns about that same axis at one radian per second. Thus its angular-Jacobian column is the axis direction. For a seven-hinge arm,

$$\mathbf{J}_{\text{ang}}(:, i) = {}^W\hat{\mathbf{z}}_i, \quad \boldsymbol{\omega} = \sum_{i=1}^7 {}^W\hat{\mathbf{z}}_i \dot{q}_i.$$

All contributions use world coordinates before we add them. A sliding joint changes no orientation, so its angular column is zero. Where the hinge axis passes in space affects linear velocity, but not its angular contribution.

A numerical continuation of the FR3 chain. Use the same ready configuration as before:

$$\mathbf{q} = (0, -\pi/4, 0, -3\pi/4, 0, \pi/2, \pi/4)^\top.$$

The rotation block of the previously computed second-link world pose is

$${}^W\mathbf{R}^2 = \begin{bmatrix} c & c & 0 \\ 0 & 0 & 1 \\ c & -c & 0 \end{bmatrix}, \quad c = \frac{\sqrt{2}}{2}.$$

Rotate the model's local axis into world coordinates:

$${}^W\hat{\mathbf{z}}_2 = \begin{bmatrix} c & c & 0 \\ 0 & 0 & 1 \\ c & -c & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Joint 1's fixed placement translates by 0.333 m without rotating. Its joint angle is zero at the ready pose, so frame 1 has the identity rotation relative to the world:

$${}^W\mathbf{R}^1 = \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Apply the same axis conversion to joint 1:

$${}^W \hat{\mathbf{z}}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Thus its positive axis is world z. We have now computed both directions, so the first two angular columns are

$$\begin{bmatrix} {}^W \hat{\mathbf{z}}_1 & {}^W \hat{\mathbf{z}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Move these joints at 0.1 and 0.2 rad/s while holding the other five still:

$$\dot{\mathbf{q}} = (0.1, 0.2, 0, 0, 0, 0)^\top \text{ rad/s}.$$

Multiply the selected columns by those angular velocities:

$$\boldsymbol{\omega} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.2 \\ 0.1 \end{bmatrix} \text{ rad/s}.$$

The other five Jacobian columns still exist, but contribute nothing because their joint velocities are zero. This result is one instantaneous angular-velocity vector. It is not a sequence of finite rotations. For the planar arm both axes point along world z, so the same construction gives

$$\mathbf{J}_{\text{ang}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{bmatrix}.$$

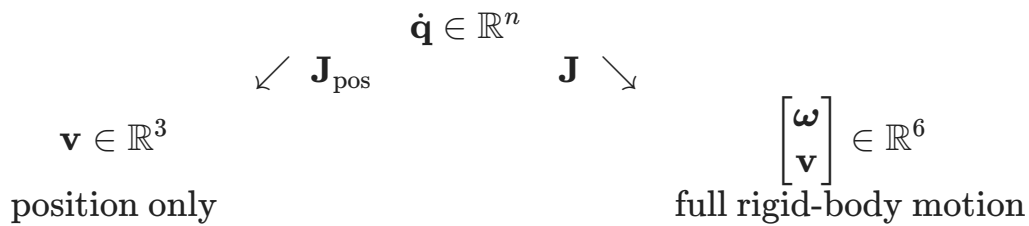
Run `drake/01c-fr3-grasp-chain/fr3_grasp_chain.py` to reproduce the FR3 angular columns and check them against Drake at the ready pose and a changed configuration.

Both blocks are $3 \times n$ in 3D. Stacking them gives the **geometric Jacobian**:

$${}^W \mathbf{V}^E = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{J}_{\text{ang}} \\ \mathbf{J}_{\text{pos}} \end{bmatrix} \in \mathbb{R}^{6 \times n}.$$

Rows 1–3 describe angular velocity; rows 4–6 describe the velocity of the origin. Each column still says what one joint moving at unit rate does. This matrix is not the entry-wise derivative of the 4×4 pose matrix: it maps joint rates to the six motion components just constructed. Equation (10)’s ordinary derivative applies directly when φ outputs vector task coordinates, such as position; for a full pose we use this geometric velocity relation.

The choice of task determines which motion components we keep:



Columns correspond to joints; rows correspond to task-motion components.

The Jacobian tells us, at the current configuration, how each infinitesimal joint motion changes each component of the task.

For the seven-joint Panda, the full end-effector geometric Jacobian is 6×7 , whereas its position Jacobian is 3×7 . A scalar distance task would have a 1×7 Jacobian. There is no mandatory size for “the Jacobian”. More joints than task components ($n > m$) gives **redundancy**; more task components than joints ($m > n$) prevents arbitrary task velocities, as in the planar position-and-heading example.

6.5 Locality, rank, and the Jacobian in learning code

Locality and rank. Move the arm and its column arrows change. The velocity equation is exact at each instant, but a finite-displacement prediction using one fixed Jacobian is only accurate for small steps: turning joint 1 by 5° gives a 2.5 mm position-prediction error, and 30° gives 8.8 cm.

Even the directions available at an instant depend on the configuration. At the worked pose, the two position-column arrows are not parallel, so their combinations can produce any planar tip velocity. They provide two independent directions. The number of independent directions is the **rank**. This count concerns the chosen position task.

When the rank is less than m , some task-velocity directions are unavailable; §7 shows this on the stretched arm. Section 8, which is optional reference material, measures the available directions with singular values and the manipulability ellipse.

Computing it. For the two-link position map, differentiate (7) by hand. Robotics libraries can also construct columns from joint geometry; finite differences provide a numerical check. For a differentiable program returning vector task coordinates, `torch.autograd.functional.jacobian(phi, q)` computes the ordinary task Jacobian, whether `phi` is analytic kinematics or a neural network. If `phi` returns a rotation or pose matrix instead, autodiff differentiates those entries; converting their derivatives to angular velocity is a separate step. The long reading §8.4 derives that conversion.

The same matrix in robot learning. Suppose a scalar loss $\mathcal{L}$ depends on vector task coordinates $\mathbf{x} = \varphi(\mathbf{q})$, such as the tip position:

$$\mathbf{q} \xrightarrow{\varphi} \mathbf{x} \xrightarrow{\mathcal{L}} \mathcal{L}.$$

For each joint, the chain rule adds its contributions through all task coordinates:

$$\frac{\partial \mathcal{L}}{\partial q_j} = \sum_{i=1}^m \frac{\partial \mathcal{L}}{\partial x_i} \frac{\partial x_i}{\partial q_j}. \quad \text{Thus} \quad \nabla_{\mathbf{q}} \mathcal{L} = \mathbf{J}^\top \nabla_{\mathbf{x}} \mathcal{L}.$$

The gradients here are column vectors. Motion propagates forward through $\mathbf{J}$; loss sensitivity propagates back through $\mathbf{J}^\top$. This transpose is not an inverse and does not solve for the joint motion needed to achieve a desired hand motion; that is §7's problem.

Four views of the same local derivative:

- Sensitivity: $J_{ij} = \partial x_i / \partial q_j$.
- Small-displacement prediction: $\Delta \mathbf{x} \approx \mathbf{J} \Delta \mathbf{q}$.
- Instantaneous motion: $\dot{\mathbf{x}} = \mathbf{J} \dot{\mathbf{q}}$.
- Backpropagation: $\nabla_{\mathbf{q}} \mathcal{L} = \mathbf{J}^\top \nabla_{\mathbf{x}} \mathcal{L}$.

Must understand: columns as joint effects and rows as chosen task-motion components; the small-displacement and velocity relations; why adding heading adds a row; why full 3D rigid-body motion needs three angular and three linear components; the angular column of a hinge as its world axis direction, and the FR3 two-joint angular-velocity calculation; locality and rank.

Recognize: the angular-first stacking order of the $6 \times n$ geometric Jacobian; finite differences, autodiff, and the transpose in backpropagation. Singular values and the position manipulability ellipse are optional reference material (§8).

Deferred: the position columns of a general spatial arm, the rotation-matrix derivative and twist-convention conversions (long reading §8), and forward versus reverse mode (long reading §8.5 and §14.1).

We have supplied joint velocities and predicted the hand's motion. Next, keep the current joint positions known, request a hand velocity, and solve for the joint velocities that produce it.

7. Differential inverse kinematics

To lift a mug, we can specify that the hand should move straight upward at 0.1 m/s. The Jacobian has already told us how joint velocities produce hand velocity. We now reverse the question: which joint velocities produce the motion we requested? For the two-link position task, the upward request is

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \text{ m/s.}$$

This calculation chooses motion at the current instant, and it is linear in the unknown joint velocities. The same linear model returns at the end of this section, where it proposes a joint-position update toward a static target.

7.1 The question, and the square case

Problem. At the current configuration, we know what each joint does to the hand. Now choose joint rates whose combined effect matches a desired hand velocity. Write this as $\mathbf{J}\dot{\mathbf{q}} = \dot{\mathbf{x}}$, with $\dot{\mathbf{x}}$ the **requested** velocity; the achieved velocity will be $\mathbf{J}\dot{\mathbf{q}}$. Unlike the full inverse-kinematics problem, this is linear in the unknown joint rates because $\mathbf{J}$ is evaluated at a fixed $\mathbf{q}$. It may still have no exact solution or more than one.

The four questions now have distinct inputs and outputs:

Question	Given	Compute or seek
Forward kinematics	Joint positions	Tip position or pose
Inverse kinematics	Desired tip position or pose	Joint positions that achieve it
Differential forward kinematics	Current joint positions and joint velocities	Tip linear velocity, or full spatial velocity
Differential inverse kinematics	Current joint positions and requested tip velocity	Joint velocities that produce that motion

For both velocity questions, the current joint positions determine the Jacobian. A position-only task uses linear velocity; full spatial velocity also includes angular velocity and names the frame origin whose motion is measured. An inverse request can be unreachable or have several answers, as the examples below show.

The square case. If $\mathbf{J}$ is square and invertible,

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}. \quad (11)$$

Here the task specifies only the two components of tip position. Use the 2×2 position Jacobian in (9), evaluated at the known current joint positions. The gripper's angular velocity is not an independent request in this calculation.

Moving the two-link tip straight up at 0.1 m/s from $(30^\circ, 45^\circ)$ requires $\dot{\mathbf{q}} = (0.3415, -0.5772)$ rad/s: shoulder forward, elbow back. Multiplication by (9) checks the answer: the horizontal contributions cancel, while the vertical contributions add to 0.1 m/s. The negative elbow rate is part of producing a straight upward tip velocity, not a command to move the whole hand backward.

$$\mathbf{J}_{\text{pos}} \dot{\mathbf{q}} \approx \begin{bmatrix} -0.4898 & -0.2898 \\ 0.4241 & 0.0776 \end{bmatrix} \begin{bmatrix} 0.3415 \\ -0.5772 \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \text{ m/s.}$$

7.2 A lost direction, and damping

The inverse can lose a motion direction. Now change the configuration and keep the task. Stretch the same arm along world x , so that $\mathbf{q} = (0^\circ, 0^\circ)$. Its position Jacobian becomes

$$\mathbf{J}_{\text{pos}}(0, 0) = \begin{bmatrix} 0 & 0 \\ 0.7 & 0.3 \end{bmatrix} \text{ m/rad.}$$

Both columns now point along world y . Their combinations can move the tip across the arm, but cannot give it instantaneous velocity along the arm. The count of independent directions has dropped from two to one. This loss is a **singularity for the position task**, and the ordinary inverse no longer exists. Away from the stretched and folded configurations the position Jacobian has rank 2, so a singularity is a configuration where the rank falls below the value that the task's Jacobian usually has. The position-and-heading Jacobian of §6 never reaches rank 3 anywhere; that is a property of the task and not a singularity. The arm can still move sideways, and bending it can bring the tip inward through a finite motion. The missing direction concerns the current instant.

Near the stretched configuration, the arrows are almost parallel. Producing a requested velocity with a component along the arm can require large joint velocities because the stronger sideways effects must almost cancel. A modest tip-velocity request can therefore be difficult to execute even before the rank drops. The optional §8 measures this with singular values.

Damping is the usual response to that cost. When the inverse would demand very large joint rates, a damped step, called **damped least squares**, adds a penalty on joint speed to the velocity-matching problem. It then tracks the request only approximately and keeps the joint rates bounded as the arm approaches a singularity. The larger the penalty weight ρ , the less joint motion the step asks for and the more tracking error it accepts; near a singularity it can give up a large part of the request rather than demand joint rates far above any joint's speed limit. The tradeoff is soft. Hard speed

limits and joint limits need explicit constraints. Appendix C.2 states the damped step and derives it from its cost, and Optimization I applies the same penalty to the joint-position update at the end of this section.

7.3 Several answers, and the minimum-norm choice

Several joint motions can satisfy one request. Return to the bent configuration $(30^\circ, 45^\circ)$ of the upward example; the stretched arm served only to show a lost direction. Keep that configuration and change what the task requires. Request only a horizontal tip velocity of 0.1 m/s and leave the vertical velocity unspecified. Leaving it unspecified differs from requesting zero vertical velocity. A zero vertical velocity would be a second requirement, and the square case showed that two requirements fix both joint rates at this pose. The task now uses only the first row of the position Jacobian (9):

$$\mathbf{J}_x \dot{\mathbf{q}} = 0.1, \quad \mathbf{J}_x \approx \begin{bmatrix} -0.4898 & -0.2898 \end{bmatrix} \text{ m/rad}.$$

Moving joint 1 alone or joint 2 alone can satisfy that request:

$$\dot{\mathbf{q}}_A \approx \begin{bmatrix} -0.2042 \\ 0 \end{bmatrix}, \quad \dot{\mathbf{q}}_B \approx \begin{bmatrix} 0 \\ -0.3451 \end{bmatrix} \text{ rad/s}.$$

Multiplying either choice by the full position Jacobian gives a horizontal velocity of approximately 0.1 m/s. Their vertical velocities differ, approximately -0.0866 and -0.0268 m/s. Both answers are valid because the task leaves vertical motion free. Requesting both velocity components, as in the square example, would remove this freedom at this pose.

Draw the possible joint velocities on a graph with $\dot{q}_1$ on the horizontal axis and $\dot{q}_2$ on the vertical axis. Every point represents a joint-velocity command. The valid answers form a line. The origin means both joints are stationary, so it cannot satisfy this nonzero request. The closest point on the line to the origin has the smallest combined joint angular speed:

$$\dot{\mathbf{q}}_{\min} \approx \begin{bmatrix} -0.1512 \\ -0.0895 \end{bmatrix} \text{ rad/s}, \quad \|\dot{\mathbf{q}}_{\min}\| \approx 0.1757 \text{ rad/s}.$$

Its vertical tip velocity is approximately -0.0711 m/s, which the task also permits. The matrix that selects this answer is the **pseudo-inverse** $\mathbf{J}_x^+$.

For this nonzero row, the direction perpendicular to the solution line is the row written as a column. The closest point to the origin lies in that direction. Divide the column by the sum of the row's squared entries so that multiplying it by the row gives one:

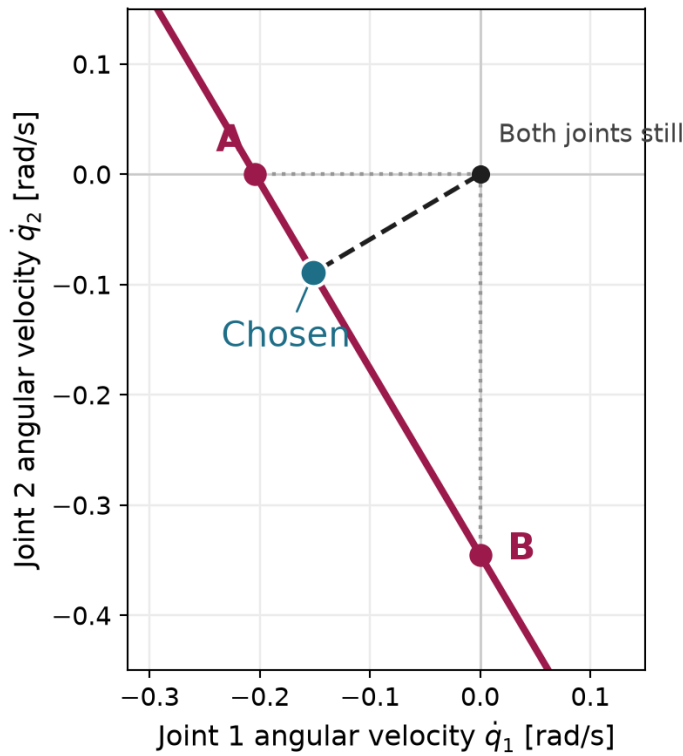
$$(-0.4898)^2 + (-0.2898)^2 = 0.32388808.$$

$$\mathbf{J}_x^+ = \frac{\mathbf{J}_x^\top}{\mathbf{J}_x \mathbf{J}_x^\top} \approx \frac{\begin{bmatrix} -0.4898 \\ -0.2898 \end{bmatrix}}{0.32388808} \approx \begin{bmatrix} -1.5123 \\ -0.8948 \end{bmatrix} \text{ rad/m.}$$

The denominator is a scalar because the task Jacobian has one row. Multiplication by the requested horizontal speed now gives the selected joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}_x^+(0.1) \approx \begin{bmatrix} -0.1512 \\ -0.0895 \end{bmatrix} \text{ rad/s.}$$

The displayed calculation uses the rounded coefficients. The code uses the unrounded Jacobian and checks this construction against `np.linalg.pinv`. This simple transpose-and-scalar-division formula applies to a nonzero one-row Jacobian.



The axes show the two joint angular velocities. Every point on the burgundy line satisfies the horizontal tip-velocity request. A and B are the single-joint choices. The chosen point is closest to the origin, so it has the smallest joint-velocity norm.

The general choice first matches the velocity, then minimizes joint motion. A Panda's full geometric Jacobian is 6×7 and has no ordinary inverse. Other tasks can have several joint-velocity answers, or no exact answer. The same selection rule extends to these cases:

$$\dot{\mathbf{q}} = \mathbf{J}^+(\mathbf{q})\dot{\mathbf{x}}. \quad (12)$$

For an invertible square Jacobian, $\mathbf{J}^+$ equals $\mathbf{J}^{-1}$, so (12) contains (11). It makes two decisions in order. First, minimize the squared velocity error $\|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|^2$. This is **least squares**: add the squared mismatch in each task component. When no joint motion

can produce the requested velocity exactly, as for an along-arm request at the stretched configuration, this first decision keeps the part the columns can produce and drops the rest. At full extension, the runnable example at the end of this reading meets an across-arm request exactly and achieves zero velocity for an along-arm request; §8 works through the numbers. If several joint-rate vectors attain that same minimum error, choose the one with smallest $\|\dot{\mathbf{q}}\|^2$, the sum of squared joint rates. This second choice is the **minimum-norm** answer. It is not generally the minimum-energy motion, and it does not include joint limits or collisions.

7.4 Null-space motion

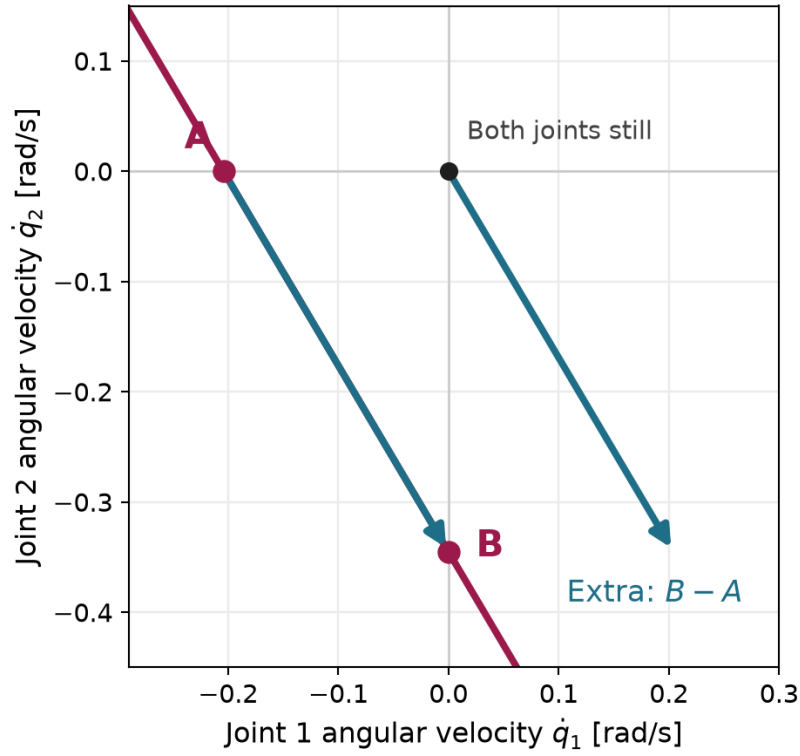
Change the joint motion while preserving the task. Return to the two valid answers above. Their difference is an extra coordinated joint velocity:

$$\dot{\mathbf{q}}_{\text{null}} = \dot{\mathbf{q}}_B - \dot{\mathbf{q}}_A \approx \begin{bmatrix} 0.2042 \\ -0.3451 \end{bmatrix} \text{ rad/s.}$$

Because both answers produce the same horizontal tip velocity, their difference produces zero horizontal velocity:

$$\mathbf{J}_x \dot{\mathbf{q}}_{\text{null}} = \mathbf{J}_x \dot{\mathbf{q}}_B - \mathbf{J}_x \dot{\mathbf{q}}_A = 0.1 - 0.1 = 0.$$

The equality uses the unrounded answers. Adding this coordinated motion to answer A gives answer B without changing the requested horizontal velocity. Its vertical tip velocity is approximately 0.0598 m/s. So preserving this task does not mean keeping the entire tip still. The joint velocities that produce zero motion in the chosen task form its Jacobian's **null space**. On the graph they form a line through the origin, parallel to the line of valid answers.



The blue arrow from A to B is their joint-velocity difference. The same arrow is drawn at the origin. Its joint effects cancel in horizontal tip velocity. Adding this arrow to A gives B while preserving the horizontal request. Vertical velocity can change.

Keep a gripper pose while changing the arm posture. Now change robots and tasks. On the seven-joint Panda, hold the gripper's full pose. Moving one joint alone generally moves the gripper. Several joints can move together so their effects cancel at the gripper. The elbow can then move while the gripper keeps its position and orientation. Each Jacobian column is one joint's effect; null-space motion combines those effects to give zero task velocity:

$$\mathbf{J}(\mathbf{q})\dot{\mathbf{q}}_{\text{null}} = \mathbf{0}.$$

The number of independent coordinated-motion directions is

$$n - \text{rank } \mathbf{J}.$$

At a regular configuration, the Panda's full-pose Jacobian has rank 6. Its seven joint velocities therefore leave one independent pattern of coordinated motion. This does not mean one particular joint can move arbitrarily. The first redundancy demo holds the flange's full pose. The second holds only the tool-center point between the fingertips, 10.3 cm beyond the flange, and lets the gripper turn. At position-Jacobian rank 3, four independent coordinated patterns remain. Both the task and the held reference point have changed.

Appendix C shows how a preferred joint motion is projected into this null space. The demos can be followed from the behavior described above, and the construction is optional.

7.5 From rates to commands

A velocity is a rate, whereas a position-command interface expects a target angle. Over a short control period Δt , integrate approximately:

$$\mathbf{q}_{\text{next}} = \mathbf{q} + \Delta t \dot{\mathbf{q}}.$$

For the two-link upward-motion example at $(30^\circ, 45^\circ)$, $\Delta t = 0.01$ s gives $\Delta \mathbf{q} = (0.003415, -0.005772)$ rad, predicting a 0.001 m upward tip displacement. The controller recomputes the Jacobian after the robot moves. Other interfaces may accept velocity or torque commands; this integration step specifically describes a position interface.

A desired **position delta** over the same control period becomes a velocity request when it is divided by Δt . For a full pose, build the request in the Jacobian's frame: express the position displacement there, convert the relative rotation of §3 to a rotation-vector error in that same frame, divide both by Δt , and stack the angular part above the position part. That stacked six-vector is the spatial-velocity request for (12). A difference of the sixteen pose-matrix entries can score a loss (§3), but it is not this six-component motion. Appendix B.2 gives the frame conventions, including the world-expressed and body-expressed choices. One integrated joint step need not attain a finite pose target; the next example treats a target without any time step.

7.6 Use a local model to propose a joint-position update

Return to the two-link inverse problem of §5 and to its target $\mathbf{y}^* = (0.5, 0.2)$ m, which lies inside the reachable ring. The two-link arm is at the worked configuration $\mathbf{q} = (30^\circ, 45^\circ)$, and the task is the tip position. Unlike the two previous paragraphs, nothing here is a rate and there is no control period. The unknown is a change of the configuration guess, in radians, and the right-hand side is a displacement, in meters.

1. The current tip follows from (7): $\varphi(\mathbf{q}) \approx (0.424056, 0.489778)$ m.
2. The remaining displacement is the target minus the current tip:

$$\mathbf{e} = \mathbf{y}^* - \varphi(\mathbf{q}) \approx \begin{bmatrix} 0.075944 \\ -0.289778 \end{bmatrix} \text{ m}, \quad \|\mathbf{e}\| \approx 0.2996 \text{ m}.$$

3. The local equation comes from §6's model of a small joint change. Asking the predicted displacement to equal the remaining one gives a linear equation for the unknown update:

$$\mathbf{J}(\mathbf{q}) \Delta \mathbf{q} = \mathbf{e}, \quad \det \mathbf{J} = l_1 l_2 \sin q_2 \approx 0.0849.$$

Here $\mathbf{J}$ is the 2×2 matrix (9). Its determinant is nonzero, so it is invertible and the inverse of (11) applies with a displacement in place of a velocity.

4. The proposed update solves that equation:

$$\Delta \mathbf{q} = \mathbf{J}^{-1}(\mathbf{q}) \mathbf{e} \approx \begin{bmatrix} -0.920116 \\ 1.293088 \end{bmatrix} \text{ rad} = \begin{bmatrix} -52.72^\circ \\ 74.09^\circ \end{bmatrix},$$

$$\mathbf{q} + \Delta \mathbf{q} \approx (-22.72^\circ, 119.09^\circ).$$

5. Forward kinematics at the updated configuration, (7) again, gives

$$\varphi(\mathbf{q} + \Delta \mathbf{q}) \approx \begin{bmatrix} 0.335682 \\ 0.143665 \end{bmatrix} \text{ m}.$$

6. The remaining displacement at the updated configuration, with the same target-minus-tip convention as step 2, is

$$\mathbf{e}_{\text{new}} = \mathbf{y}^* - \varphi(\mathbf{q} + \Delta \mathbf{q}) \approx \begin{bmatrix} 0.164318 \\ 0.056335 \end{bmatrix} \text{ m},$$

$$\|\mathbf{e}_{\text{new}}\| \approx 0.1737 \text{ m}.$$

The linear model predicted that the tip would land on the target, because $\mathbf{J}\Delta \mathbf{q} = \mathbf{e}$ holds exactly. The arm lands 0.17 m away. The target is reachable and the equation was solved correctly, but the linear model is accurate only for small joint changes, as §6 showed, and this update turns the elbow by 74° . For comparison, the integration step above moved the joints by at most 0.33° , and forward kinematics after it differs from the predicted 1 mm rise by about $3 \mu\text{m}$; the runnable example at the end of this reading prints that actual tip increment. The update did reduce the error, from 0.30 m to 0.17 m, but nothing in the calculation guaranteed that. A proposed update is checked by evaluating the forward kinematics, as step 5 did. Figure 4 shows the start, the target and the arm after the update. All values are computed from unrounded inputs; this reading's `code/kinematics_note.py`, the NumPy example after §10 and Deck 03's `code/two_link.py` (part (k)) print them.

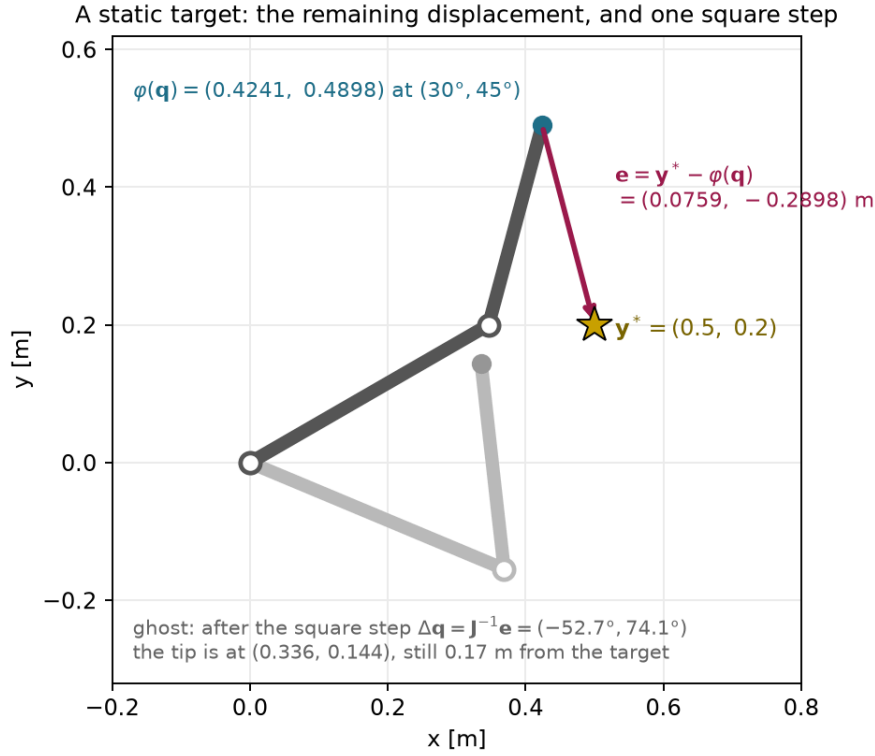


Figure 4. One joint-position update from the local model. The dark arm is at $(30^\circ, 45^\circ)$ with its tip at $(0.4241, 0.4898)$ m; the star is the target $(0.5, 0.2)$ m and the arrow is the remaining displacement $\mathbf{e}$. The gray arm is the configuration after the update $\Delta \mathbf{q} = \mathbf{J}^{-1} \mathbf{e}$; its tip is at $(0.336, 0.144)$ m, still 0.17 m from the target. Deck 03 calls this update the square step and draws the result as a ghost arm. (Deck 03, `fig/two_link_bridge.png`.)

Optimization I (Deck 04) starts from this calculation and answers the questions it leaves open: how far to trust one update, how to repeat it with a fresh Jacobian, how to measure the remaining error and when to stop. It writes the same problem with two conventions that are worth knowing now:

$$\mathbf{r} = \varphi(\mathbf{q}) - \mathbf{y}^* = -\mathbf{e}, \quad \mathbf{J} \boldsymbol{\delta} = -\mathbf{r}, \quad f(\mathbf{q}) = \frac{1}{2} \|\mathbf{r}\|^2, \quad \nabla f = \mathbf{J}^\top \mathbf{r}.$$

The residual $\mathbf{r}$ is the current tip minus the target, the opposite sign of $\mathbf{e}$, and the step is written $\boldsymbol{\delta}$, so the local equation $\mathbf{J} \Delta \mathbf{q} = \mathbf{e}$ of step 3 is the same equation with the sign moved to the other side. The objective f is (8) with a factor one half in front. The half leaves the minimizer unchanged and makes the gradient $\mathbf{J}^\top \mathbf{r}$ free of a factor of two. Appendix C.3 shows what repeating the update does on this example, as an illustration only.

Why it appears in robot learning. The calculations of §§7.1–7.6, from the velocity request to the joint-position update, appear in a learning system as follows. An end-effector velocity action becomes a joint command through this robot-specific calculation. The policy’s training data may or may not cover configurations where it behaves poorly. The same task-space command can be passed to different arms, each

using its own Jacobian, but their joint rates, reachable motions and limits differ. A common action format helps define a shared interface; it does not by itself guarantee transfer.

Must understand: requested versus achieved task velocity; the square case; the loss of an instantaneous direction at the stretched arm and the large joint rates a request can demand near it; several valid answers to one request; the pseudo-inverse's two decisions (least squares, then minimum norm), including that an unreachable component is dropped; null-space motion as the difference of two valid answers, defined relative to the stated task; integration as rate times time; one joint-position update proposed from the local model, checked by forward kinematics, and why its remaining error can be large.

Recognize: damping as a tradeoff between tracking and joint speed (Appendix C.2 has the step and its derivation); the null-space projector; geometric pose errors expressed in the Jacobian's frame; the singular value decomposition (SVD) as the construction behind $\mathbf{J}^+$; Optimization I's residual sign and half factor. Appendices B and C hold the derivations, and §8 describes the SVD.

Deferred: the SVD derivation of the pseudo-inverse (long reading §9.2); the control and convergence of the repeated step (Optimization I; Appendix C only illustrates it); general orientation-residual derivatives (Appendix B.2 states only the rule near agreement); constrained differential IK (the optimization block).

8. Singularities

This section is optional reference material. The required ideas are in §7: a singular configuration loses an instantaneous motion direction for the chosen task, and near it a request can demand large joint rates. Read this section when a paper uses singular values, manipulability ellipses or a condition number, or when you want the numbers behind §7's warning.

8.1 The ellipse and singular values at the worked pose

At the worked pose, the unit circle of joint rates maps to an ellipse. Figure 8.1 draws that image for the two-link **position** task. Its semi-axis lengths are the **singular values** $\sigma_1 \geq \sigma_2$: the tip's largest and smallest speeds for unit joint-rate norm. At $(30^\circ, 45^\circ)$, they are $(0.7037, 0.1206)$. Their ratio, the largest to the smallest, is 5.84; it is called the **condition number** of $\mathbf{J}_{\text{pos}}$ at this pose. The **rank** is the number of independent column directions, equivalently the number of nonzero singular values. If the rank is less than m , some task-velocity directions are unavailable. Figure 8.2 below shows the ellipse collapsing as the arm straightens. These ellipse numbers

concern position only; angular and linear rows have different units, so a combined pose task needs a choice of scaling before interpreting its singular values as motion gains.

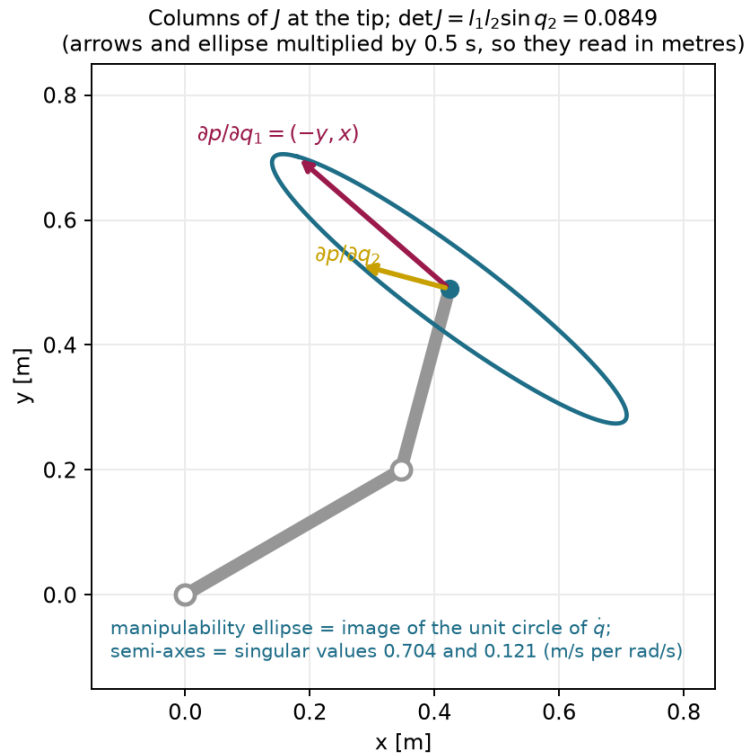


Figure 8.1. The two columns of (9) as arrows at the tip, one per joint, and the ellipse of tip velocities produced by joint-rate vectors of unit norm. Its semi-axes are $\sigma_1 = 0.7037$ and $\sigma_2 = 0.1206$.

(Deck 03, `fig/two_link_jacobian.png`.)

8.2 The stretched arm: rank loss and finite motion

Geometry. Stretch the two-link arm straight. Both column arrows now point across the arm, so they cannot combine into an instantaneous velocity along it. The ellipse becomes a segment (Figure 8.2). This is a loss of a previously available task direction. A configuration is **singular for a chosen task** when its Jacobian has lower rank than the maximum rank that same task Jacobian attains on the robot. For the planar position task the rank drops from 2 to 1 at $q_2 = 0$ (stretched) or $q_2 = 180^\circ$ (folded), modulo a full turn.

This definition separates singularity from task deficiency. The planar position-and-heading Jacobian is 3×2 and never has rank 3, but it need not lose its usual rank 2 at those configurations: its heading row can still distinguish the joints' effects. Always say which task Jacobian you are examining. When a Jacobian normally has full row or column rank, a loss of rank is equivalent to its smallest singular value becoming zero; for a square matrix it is also equivalent to its determinant becoming zero.

“No first-order motion” does not mean “no finite motion”. Starting with the arm straight along x , a small elbow turn δ changes the tip’s x coordinate by $l_2(\cos \delta - 1) \approx -l_2\delta^2/2$. The inward change exists, but it is second order: it vanishes after division by δ as $\delta \rightarrow 0$. The Jacobian therefore predicts zero radial velocity at that instant even though bending the elbow can eventually bring the tip inward. Sideways velocity is available immediately. This is why the robot is not stuck.

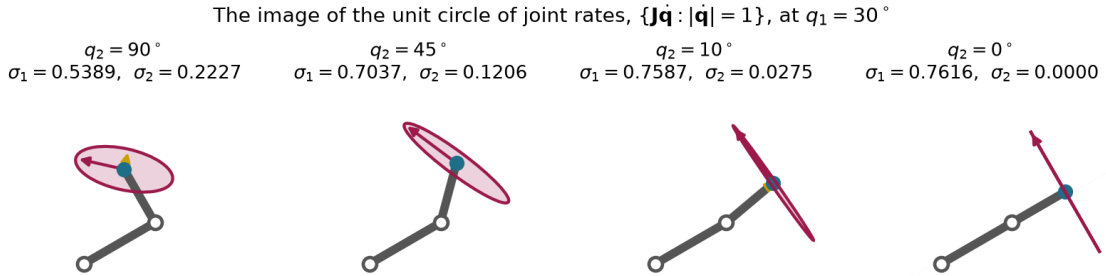


Figure 8.2. Position manipulability ellipses as q_2 decreases from 90° to 0° at $q_1 = 30^\circ$. The smaller semi-axis falls from 0.2227 to zero while the larger remains nonzero. At full extension the ellipse becomes a segment perpendicular to the arm.

8.3 Singular values as local gains near a singularity

Singular values give a quantitative explanation. Away from the singularity, a small $\sigma_{\min}$ means the arm produces very little velocity in its weakest direction per unit joint speed. To achieve a requested velocity component in that direction, the pseudo-inverse divides it by that small value. A request in a stronger direction need not demand large rates. For the fixed world-up command $(0, 0.1)$ m/s at $q_1 = 30^\circ$, the joint-rate norm rises from 0.67 rad/s at $q_2 = 45^\circ$ to 2.10 at 10° and 18.47 at 1° . At the singularity itself, the pseudo-inverse discards the unreachable component instead of dividing by zero. Near-singular amplification and exact-singularity projection are different behaviors; the two-link module shows both.

Rank counts independent directions; $\sigma_{\min}$ measures the weakest local gain. It is not a distance to a singular configuration in meters or joint angles, nor does it alone tell the cost of every command. Its numerical value depends on units and task scaling. For a fixed position task and units it is a useful warning of weak motion authority; for a full pose, angular and linear components must be scaled deliberately. Damping reduces amplification; explicit constraints enforce limits.

8.4 Requests at a rank-deficient configuration

The selection rule (12) of §7 behaves as follows at each rank.

- At full row rank ($\text{rank } \mathbf{J} = m$), every requested task velocity is achievable. For a redundant arm, (12) picks the minimum-norm vector among infinitely many exact solutions; for an invertible square matrix it agrees with (11).

- If $\text{rank } \mathbf{J} < m$, **some** requests are impossible. A request within the span of the columns is still achieved exactly. Otherwise (12) keeps the reachable component and drops the perpendicular, unreachable component.

For example, stretch the two-link arm along world x , so $\mathbf{q} = (0, 0)$. Its position Jacobian is $\begin{bmatrix} 0 & 0 \\ 0.7 & 0.3 \end{bmatrix}$. A request $(0, 0.1)$ m/s can be met because $0.7\dot{q}_1 + 0.3\dot{q}_2 = 0.1$ has solutions. A request $(0.1, 0)$ m/s cannot be met: the first row always outputs zero. The pseudo-inverse returns zero joint rates for this latter request, since extra joint motion cannot reduce its error. Thus “rank deficient” does not mean “every command fails”. The two-link module’s radial and tangential buttons provide optional exploration of precisely these two cases.

In the ellipse picture, $\mathbf{J}$ stretches joint-rate directions by the singular values; $\mathbf{J}^+$ reverses each nonzero stretch by dividing by it. This stretch-and-reverse description is the singular value decomposition (SVD) of $\mathbf{J}$, from which the long reading §9.2 derives $\mathbf{J}^+$. A zero-gain direction is unavailable and its requested component is discarded. In code, `np.linalg.pinv(J)` computes this map. All closeness and norm statements use the chosen coordinates and scaling: a pose task mixes angular and linear units, so their relative weight must be specified, just as in §3.

Why it appears in robot learning. Even a modest end-effector action can be difficult to execute at a poorly conditioned configuration. Evaluate the achieved motion and the joint commands, not just the network’s output. Training may include such configurations, but one cannot assume it does merely because the demonstrations complete the task. A controller can use damping or constraints, and a planner can prefer configurations with better motion authority.

This section is optional; §7 holds the required ideas.

Recognize: singular values as local gains, the tip speed per unit joint-rate norm along each principal direction; the condition number as the ratio of the largest to the smallest singular value; $\sigma_{\min}$ as a scale-dependent warning rather than a distance; the collapsed position ellipse; damping as one response, distinct from hard limits.

Deferred: determinant derivations, force duality, and the optional Panda rank-5 example (long reading §10). Gimbal lock (§2) is a coordinate defect; a robot’s geometric-Jacobian singularity is a limitation of its instantaneous motion for that task.

9. Where this appears in robot learning

The objects of §§1–8 sit in three places in a learning system.

In the data. A demonstration may include joint angles, images and measured or computed end-effector poses. Forward kinematics can turn joint readings into poses. If orientations are stored as quaternions, check component order and sign handling. Frame conventions and units are part of the data specification even when the file does not make them explicit.

In the action. The action space is the meaning assigned to the policy’s output numbers. A policy that emits joint targets asks a different question of the robot-side stack than one that emits a desired gripper velocity (Table 1).

The policy emits	Typical robot-side conversion	What still depends on the robot
joint positions or deltas	Track the targets; add deltas to the current joints. No IK conversion is required for this action.	Joint count, ordering, limits and the effect of each joint on the hand.
end-effector pose or pose change	Form a target pose in the specified frame and solve inverse kinematics.	Reachability, posture selection, geometry and collision constraints.
end-effector velocity	Solve differential IK, with damping or constraints as needed; integrate if using a position interface.	The current Jacobian, limits and the motion actually achieved.

Table 1. The action specifies an interface, not a complete controller. A joint-space policy need not explicitly learn forward and inverse kinematics, and a task-space policy is not automatically independent of embodiment. Task-space commands can share a physical meaning across robots, while each robot supplies its own conversion and feasibility checks.

What “delta” must specify. A delta is an increment from a stated reference. “Move 1 cm forward” is ambiguous until “forward” names world, camera or gripper axes. With current pose $\mathbf{X} = {}^W\mathbf{X}^E$, a body-frame rigid-motion increment composes on the right, $\mathbf{X}_{\text{next}} = \mathbf{X} \Delta \mathbf{X}$; a world-frame rigid-motion increment composes on the left, $\mathbf{X}_{\text{next}} = \Delta \mathbf{X} \mathbf{X}$. These are different commands. Some interfaces instead add a world-frame position delta and update orientation separately; that convention must also be stated. A policy’s output needs its frame, units, update rule and control period to become an executable command.

In the model. A differentiable kinematic layer evaluates φ and passes a task-loss gradient back through its Jacobian (§6). A learned dynamics model has a different map, but uses the same local-linearization idea. The existence of a Jacobian does not make a model accurate outside the states used to fit it.

The three course questions now have concrete meanings: *what does the learned component replace* (for example, choosing joint targets), *what does it keep* (for example, a tracking controller and the robot's mechanics), and *what does it assume* (frames, embodiment, reachable goals and training coverage)? The answers depend on the actual architecture and action interface.

10. Minimum Robotics Knowledge Checklist

After this note the reader should be able to recognize and explain each of the following. The first group is what must be understood; the second is what should be recognized when a paper uses it.

Understand.

1. A frame is an origin with right-handed axes; ${}^A\mathbf{p}_C^B$ is the position of B relative to A written in C 's axes; the same arrow has different coordinates in different frames.
2. A rotation matrix ${}^A\mathbf{R}^B$ has the axes of B as its columns, satisfies $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$, and is inverted by transposition; an orientation has three degrees of freedom.
3. ${}^A\mathbf{p}^P = {}^A\mathbf{p}^B + {}^A\mathbf{R}^B {}^B\mathbf{p}^P$ changes the reference frame of a point.
4. The pose ${}^A\mathbf{X}^B \in \text{SE}(3)$ is one 4×4 matrix holding a rotation and a position; it locates B and converts B -coordinates to A -coordinates.
5. Poses compose by matrix product with matching inner letters, ${}^A\mathbf{X}^C = {}^A\mathbf{X}^B {}^B\mathbf{X}^C$, and invert as $[\mathbf{R}^\top, -\mathbf{R}^\top \mathbf{p}; \mathbf{0}^\top, 1]$.
6. For these fixed-base arms with hinge/slider joints, the configuration has one number per joint; task space has dimension m ; forward kinematics $\varphi : \mathbf{q} \mapsto {}^W\mathbf{X}^E(\mathbf{q})$ is a product of joint and fixed transforms, smooth, cheap and differentiable. For the two-link arm, derive its position from the two link directions; the second link's world heading is the sum of both joint angles. Explain every intermediate transform, including why a local shift must be rotated before it is added to a world position. Distinguish the flange from the tool center, and construct a desired tool pose from an object pose and a supplied grasp.

7. Inverse kinematics, find $\mathbf{q}$ with $\varphi(\mathbf{q}) = \mathbf{y}^*$, may have no solution, one, several or a continuum. For a position task, an error objective is $\min_{\mathbf{q}} \|\varphi(\mathbf{q}) - \mathbf{y}^*\|^2$.
8. For a vector task, $\mathbf{J}(\mathbf{q}) = \partial\varphi/\partial\mathbf{q}$ has m rows for task components and n columns for joints; $\Delta\mathbf{x} \approx \mathbf{J}\Delta\mathbf{q}$ and $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$. For full 3D motion, the geometric Jacobian maps joint rates to three angular and three linear velocity components. Both are local maps.
9. For a hinge joint, the angular-velocity column of the geometric Jacobian is the joint's positive axis direction in world coordinates. Given the axes and the joint velocities, the end-effector's angular velocity is the weighted sum of those directions, as in the FR3 two-joint example of §6.
10. $\dot{\mathbf{q}} = \mathbf{J}^+(\mathbf{q})\dot{\mathbf{x}}$ makes two decisions in order. It first matches the requested end-effector velocity as closely as possible, exactly when the request is reachable and otherwise by dropping the unreachable component. Among the joint velocities that match equally well, it then chooses the one with the smallest joint-rate norm.
11. Two valid answers to the same task request differ by a joint velocity that produces zero motion in that task. Null-space motion is defined relative to the stated task, and preserving a task does not mean keeping the whole tip still.
12. One local step toward a static position target uses the remaining displacement and the local model of §6, solved for the update:

$$\mathbf{e} = \mathbf{y}^* - \varphi(\mathbf{q}), \quad \mathbf{J}(\mathbf{q}) \Delta\mathbf{q} = \mathbf{e}.$$

It updates a guess and has no control period, and forward kinematics at the updated configuration checks it; in the worked example 0.17 m of error remains. Whether and how to repeat it is Optimization I's question; Appendix C shows the repeated steps.

13. A singularity is the loss of an instantaneous motion direction for the chosen task, a rank drop relative to that task's usual maximum rank. Near one, a request along the nearly lost direction can demand large joint speed; at one, the pseudo-inverse drops the unreachable component.
14. A policy's action space (joint positions, end-effector pose, end-effector velocity) decides which kinematic conversion the stack performs; it does not establish what the network has learned internally. Deltas also require a frame and an update convention.
15. A position interface needs a joint-position command. Convert a joint rate into a small position increment using the control period:

$$\mathbf{q}_{\text{next}} = \mathbf{q} + \Delta t \dot{\mathbf{q}}.$$

Recognize.

16. $\text{SO}(3)$ and $\text{SE}(3)$; quaternions, rotation vectors and Euler angles as alternative spellings of $\mathbf{R}$, with gimbal lock as the Euler-angle defect and the two quaternion orderings as the common bug.
17. The kinematic tree, URDF and `.g` files; revolute and prismatic joints.
18. The angular-first stacking of the $6 \times n$ geometric Jacobian and of the spatial velocity ${}^W\mathbf{V}^E = (\boldsymbol{\omega}, \mathbf{v})$; singular values and the manipulability ellipse, with $\sigma_{\min}$ as a scale-dependent local gain (§8); the singular value decomposition (SVD) as the construction behind $\mathbf{J}^+$ (long reading §9.2).
19. The null-space equation $\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}^+\mathbf{J})\mathbf{z}$ and “redundancy resolution”; the damped step $(\mathbf{J}^\top\mathbf{J} + \rho\mathbf{I})^{-1}\mathbf{J}^\top$ and “damped least squares” (Appendix C derives both).
20. The paper spellings of Table A.1 in Appendix A: T , g , H for a pose; $J^\dagger$ for $\mathbf{J}^+$; θ for $\mathbf{q}$. Velocity symbols such as $\mathcal{V}$ and ξ still require a check of reference point, expression frame and stacking order.
21. A relative transform describes a motion error; a matrix norm can define a scalar loss (Appendix B). Express a geometric pose-error command in the Jacobian’s frame.
22. The transpose propagates a task-space loss gradient back to joint coordinates:

$$\nabla_{\mathbf{q}}\mathcal{L} = \mathbf{J}^\top \nabla_{\mathbf{x}}\mathcal{L}.$$

This is a sensitivity calculation. It is different from selecting joint rates with the pseudo-inverse to achieve a requested task velocity.

These are the tools needed to interpret the equations above. Their full derivations and the additional machinery for dynamics, trajectory generation and contact belong to the long reading and later material.

Check the ideas before looking at the answers

Use the existing examples. Questions 1–4 and 7–14 test understanding, and questions 9–14 can be answered from §§6–7 alone. Question 5 is optional practice with §8. Question 6 tests recognition of the gradient rule and understanding of action conventions. The answers follow so you can identify which step needs another read.

1. The tray origin is at world position (0.40, 0.20) m, and the attached mug is at tray coordinates (0.15, 0.05) m. Turn the tray to 90°. Give the tray-to-mug arrow in world axes and the mug’s world position. Which local coordinates stay unchanged?

2. Given the world-to-tray and tray-to-mug poses, write the product that locates the mug in the world. Which matrix acts first on a mug-coordinate column? Should a direction vector use a final homogeneous coordinate of one or zero?
3. At joint angles 30° and 45° , give each link's world heading. Use the link lengths 0.4 m and 0.3 m to recover the tip position. Why would using 45° as the second link's world heading give a wrong result?
4. The worked upward request gives joint rates $(0.3415, -0.5772)$ rad/s. Over 0.01 s, what joint increment is commanded and what tip displacement does the Jacobian predict? Does the linear prediction guarantee the exact finite displacement?
5. This question is optional and uses §8. At the stretched arm with shoulder angle 30° , compare a radial request with a tangent request, each of magnitude 0.1 m/s. Which can be met? What does the pseudo-inverse do with an unreachable component, and can finite bending still move the tip inward?
6. A task loss gradient is a two-vector and the position Jacobian is two by seven. What product gives the seven joint sensitivities? Separately, a policy returns "move by $(0.01, 0, 0)$." Name the missing conventions before treating that output as a command.
7. At construction stage 2 in §4.1, the origin has reached the elbow. Why does the next factor change the axes but leave that origin unchanged? Why does the final shift point at 75° in the world even though its local translation column is $(0.3, 0)$?
8. The FR3 flange is at height 0.5902820523 m, and its local z points down. Its tool center is 0.1034 m along that axis. Find the tool height. The desired grasp has the same downward z direction at world height 0.10 m. Which local z offset places the approach 0.10 m above it? Does computing this target establish that a grasp will succeed?
9. Two hinges have the world axis directions $(0, 0, 1)$ and $(0, 1, 0)$ at the current configuration, and they turn at 0.3 and -0.1 rad/s while every other joint is still. Give the gripper's angular velocity and its magnitude. Is the result a sequence of two rotations?
10. Stretch the two-link arm along world x, so that both joints are at 0° . Which tip-velocity direction has no instantaneous answer, and why is the arm not stuck there?
11. At $(30^\circ, 45^\circ)$ the horizontal-only request of 0.1 m/s has the two answers $A = (-0.2042, 0)$ and $B = (0, -0.3451)$ rad/s. Show that their midpoint $(-0.1021, -0.1725)$ rad/s also satisfies the request, give its vertical tip velocity, and compare its joint-rate norm with the minimum-norm answer's 0.1757 rad/s.

12. A seven-joint arm holds its flange's full pose at a regular configuration. How many independent coordinated motion patterns remain? If it holds only the tool-center position instead, how many? Name the two things that changed between the two demonstrations.
13. Two joint changes were computed at $(30^\circ, 45^\circ)$: $(0.003415, -0.005772)$ rad from the upward velocity request over 0.01 s, and $(-0.920116, 1.293088)$ rad from the target $(0.5, 0.2)$ m. For each, say what was given (a velocity or a displacement), what role time played, what the Jacobian's inverse multiplied, and how the result is checked.
14. The update of §7 solved $\mathbf{J}\Delta\mathbf{q} = \mathbf{e}$ exactly, yet 0.17 m of error remained. Say which of these explains it: the target is unreachable, the equation was solved wrongly, or the linear model was used far outside the small changes for which it is accurate. What would you compute next, and what must be recomputed first?

Answers and reasons

1. The world-expressed arrow is $(-0.05, 0.15)$ m, so the mug's world position is $(0.35, 0.35)$ m. The attached mug's tray coordinates remain $(0.15, 0.05)$ m. Rotating the arrow does not change its reference point; adding the tray origin does.

2. The matching-frame product is

$${}^W\mathbf{X}^M = {}^W\mathbf{X}^T T \mathbf{X}^M.$$

The rightmost factor first converts mug coordinates to tray coordinates. A direction uses zero so translation has no effect.

3. The headings are 30° and 75° . Adding the two link vectors gives

$$\mathbf{p} = \begin{bmatrix} 0.4 \cos 30^\circ + 0.3 \cos 75^\circ \\ 0.4 \sin 30^\circ + 0.3 \sin 75^\circ \end{bmatrix} \approx \begin{bmatrix} 0.4241 \\ 0.4898 \end{bmatrix} \text{ m.}$$

The second joint angle is measured relative to the first link, whose heading is already 30° .

4. The increment is $(0.003415, -0.005772)$ rad. The Jacobian predicts $(0, 0.001)$ m to rounding precision. The actual finite motion can differ because this is a local approximation.
5. The radial request lies outside the Jacobian's range and its projection is zero. The tangent request is achievable. The pseudo-inverse drops the unreachable component and then chooses the least joint-rate norm. Finite bending can produce second-order inward displacement even though the instantaneous radial velocity is zero.

6. Multiply by the transpose as in item 22; its dimensions are seven by two. This is not an inverse-motion request. For the policy output, specify whether it is a displacement or a velocity, its units, its reference point and expression frame, its update rule, and the control period if converting between a displacement and a rate.
7. The third factor is a rotation with zero translation, so the origin stays at the elbow. Its axes turn by the relative joint angle 45° from a frame already pointing at 30° . The last factor's local translation is rotated by those current axes before being added to the elbow position. Its world direction is therefore 75° .
8. The tool height is 0.4868820523 m. For the approach, the local offset is -0.10 m along z , which becomes $+0.10$ m along world z , so the approach height is 0.20 m. These are pose calculations. Reachability, collision avoidance, finger contact and grasp stability remain to be checked.
9. The angular velocity is the weighted sum of the axis directions:

$$\boldsymbol{\omega} = 0.3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0.1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.1 \\ 0.3 \end{bmatrix} \text{ rad/s},$$

$$\|\boldsymbol{\omega}\| = \sqrt{0.1} \approx 0.3162 \text{ rad/s}.$$

It is one instantaneous rotation about the axis along $(0, -0.1, 0.3)$, with the two contributions added as vectors in world coordinates. It is not two successive finite rotations, and the order of the joints plays no role in the sum.

10. Both columns of the position Jacobian at $(0^\circ, 0^\circ)$ point along world y , so no combination of joint rates gives a velocity along the arm; the along-arm direction is lost at that instant. The arm is not stuck. Sideways velocity is available at once, and bending the elbow brings the tip inward through a finite motion, which the instantaneous picture of the Jacobian does not show.
11. The first row of (9) applied to the midpoint gives the requested horizontal velocity:

$$\begin{aligned} -0.4898 \cdot (-0.1021) - 0.2898 \cdot (-0.1725) &\approx 0.0500 + 0.0500 \\ &= 0.1 \text{ m/s}. \end{aligned}$$

The valid answers form a straight line, and any point between A and B lies on it. The second row gives the vertical tip velocity, which the task leaves free:

$$0.4241 \cdot (-0.1021) + 0.0776 \cdot (-0.1725) \approx -0.0567 \text{ m/s}.$$

The midpoint's norm is 0.2005 rad/s, larger than 0.1757 rad/s; the midpoint minus the minimum-norm answer is a null-space motion of the horizontal task. The script `code/kinematics_note.py` prints these numbers in its section 7(e).

12. Seven joints minus six held components leave one pattern, and seven minus three leave four, at full task rank. Two things changed: the task, from the full pose with six components to the tool-center position with three, and the held reference point, from the flange to the tool center 10.3 cm beyond it. These counts describe independent joint-velocity patterns. They do not identify one or four particular joints that can all turn freely.
13. The first change was given a velocity in m/s. Time entered as the control period 0.01 s; the Jacobian's inverse multiplied the velocity to give joint rates, and the rates times the period gave the increment. Forward kinematics at the incremented configuration checks it, and differs from the predicted 1 mm rise by about 3 μm . The second change was given a displacement in meters, the target minus the tip, with no time at all; the same inverse multiplied the displacement to give a change of the configuration guess. Forward kinematics at the updated configuration checks it and leaves 0.17 m. The same matrix acted on different inputs and produced different objects, a command for the next control period and an update to a guess.
14. The third explanation applies. The target lies inside the reachable ring, and $\mathbf{J}\Delta\mathbf{q} = \mathbf{e}$ holds exactly, so the first two do not apply; the model is a first-order prediction, and a 74° elbow turn is far outside its accurate range. Next, evaluate the forward kinematics and the Jacobian at the updated configuration, form the new displacement $\mathbf{e}_{\text{new}}$, and solve the local equation again, possibly for a shortened step. Optimization I says how far to trust a step and when to stop; Appendix C.3 shows the repeated steps reaching the target.

Reproduce the local calculations. This complete example needs Python and NumPy, not a robot simulator. It reverses the change of frame of §3, rebuilds the arm and checks its Jacobian against finite differences, solves the upward request and integrates one short command, proposes the joint-position update of §7 and checks it with forward kinematics, and compares a reachable and an unreachable request at full extension. Run it as a script or in a notebook cell. Angles are in radians after the initial conversion, lengths in meters, and every error is written target minus tip. The same code is the file `code/reading_example.py` beside this reading, and a check keeps the two identical. Copying the code out of the PDF has not been checked. A page break can change the indentation, so run the file instead.

```

import numpy as np

np.set_printoptions(precision=6, suppress=True)

def position(q):
    a, b = q
    return np.array([0.4*np.cos(a) + 0.3*np.cos(a+b),
                    0.4*np.sin(a) + 0.3*np.sin(a+b)])

def jacobian(q):
    a, b = q
    return np.array([[-0.4*np.sin(a)-0.3*np.sin(a+b),
                    -0.3*np.sin(a+b)],
                    [ 0.4*np.cos(a)+0.3*np.cos(a+b),
                    0.3*np.cos(a+b)]]

# 1. The inverse round-trip of section 3: frame B at (0.40, 0.20) m
#     turned by 30 deg, and the point P at (0.15, 0.05) m in B's axes.
th = np.deg2rad(30.0)
R = np.array([[np.cos(th), -np.sin(th)], [np.sin(th), np.cos(th)]]
p_AB = np.array([0.40, 0.20])
p_BP = np.array([0.15, 0.05])
p_AP = p_AB + R @ p_BP          # equation (3): B-coordinates to A
print("P in A [m]:", p_AP)
# equation (5): subtract the offset, then rotate back
print("P back in B [m]:", R.T @ (p_AP - p_AB))
row = np.array([[0.0, 0.0, 1.0]])
X = np.block([[R, p_AB[:, None]], [row]])
X_inv = np.block([[R.T, -(R.T @ p_AB)[:, None]], [row]])
print("X times its inverse is the identity:",
      np.allclose(X @ X_inv, np.eye(3)))
print("plain transpose of X applied to (P in A, 1):",
      X.T @ np.append(p_AP, 1.0))

# 2. The Jacobian at (30 deg, 45 deg), checked entry by entry
#     against central finite differences.
q = np.deg2rad([30.0, 45.0])
J = jacobian(q)
h = 1e-6
J_fd = np.column_stack([(position(q + h*e) - position(q - h*e)) /
                        (2*h)
                        for e in np.eye(2)])
print("tip position [m]:", position(q))
print("position Jacobian [m/rad]:\n", J)
print("largest |analytic - finite difference| [m/rad]:",
      np.abs(J - J_fd).max())

# 3. Differential IK for an upward request, then one integrated
#     command: a rate times a time.
requested = np.array([0.0, 0.1])    # world frame, meters per second

```

```

rates = np.linalg.solve(J, requested)  # the square case (11)
print("joint rates [rad/s]:", rates,
      " achieved velocity [m/s]:", J @ rates)
dt = 0.01  # seconds
step = dt * rates
actual = position(q + step) - position(q)
print("joint increment [rad]:", step, " = [deg]:", np.rad2deg(step))
print("actual tip increment [m]:", actual,
      " model error [m]:", np.linalg.norm(actual - dt*requested))

# 4. One joint-position update toward a static target: a displacement
#    in meters, no time step. Every error is target minus tip.
y_star = np.array([0.5, 0.2])
e = y_star - position(q)
dq = np.linalg.solve(J, e)  # the local equation J dq = e
q_new = q + dq
e_new = y_star - position(q_new)
print("remaining displacement e [m]:", e, " |e| [m]:",
      np.linalg.norm(e))
print("update dq [rad]:", dq, " = [deg]:", np.rad2deg(dq))
print("tip after the update [m]:", position(q_new))
print("e_new [m]:", e_new, " |e_new| [m]:", np.linalg.norm(e_new))

# 5. At full extension the pseudo-inverse keeps a reachable request
#    and drops an unreachable one.
J_straight = jacobian(np.zeros(2))
for command in ([0.0, 0.1], [0.1, 0.0]):
    rates_straight = np.linalg.pinv(J_straight) @ command
    print("straight-arm request [m/s]:", command,
          " achieved [m/s]:", J_straight @ rates_straight)

```

The printed values are the numbers of §3 and §7. NumPy prints the arrays with six decimals and the scalars in full; the table rounds the scalars. A printed `-0.` is a zero computed as a tiny negative number, not a sign error.

Printed line	Value
P in A; P back in B	(0.504904, 0.318301) m; (0.15, 0.05) m, and <code>True</code> for the identity check
plain transpose of X applied to (P in A, 1)	(0.59641, 0.023205, 1.265622)
tip position; position Jacobian	(0.424056, 0.489778) m; rows $(-0.489778, -0.289778)$ and $(0.424056, 0.077646)$ m/rad
largest difference, analytic against finite-difference Jacobian	about 4×10^{-11} m/rad, which is floating-point round-off in the difference quotient at $h = 10^{-6}$; the central difference's own truncation error is far smaller
joint rates; achieved velocity	(0.341506, -0.577209) rad/s; (0, 0.1) m/s
joint increment	(0.003415, -0.005772) rad, which is (0.195669, -0.330716) in degrees
actual tip increment; model error	(-0.000002 , 0.000998) m against the predicted (0, 0.001) m; the norm of actual minus predicted is 3.0×10^{-6} m
remaining displacement $\mathbf{e}$; its norm	(0.075944, -0.289778) m; 0.299564 m
update $\Delta \mathbf{q}$	(-0.920116 , 1.293088) rad, which is (-52.718747 , 74.088475) in degrees
tip after the update; $\mathbf{e}_{\text{new}}$; its norm	(0.335682, 0.143665) m; (0.164318, 0.056335) m; 0.173707 m
straight-arm requests	(0, 0.1) m/s is achieved exactly; (0.1, 0) m/s gives zero achieved velocity

The tip increment differs from its prediction because the Jacobian changes as the joints move; the update leaves 0.17 m for the same reason, magnified by a change of 74° . The script `code/kinematics_note.py` prints the same values and makes the number-by-number comparison.

Explore the demonstrations. The figures and the printed numbers are complete without a simulator. The demonstrations below show the same objects moving, in simulation only. Each link opens a file of the companion folder that comes with this PDF. If your viewer does not follow a link, the folder's `README.html` has the same link. Browser modules open in any browser. The Drake demonstrations need Python with Drake and a browser for Meshcat. A launcher route such as `/run/self-motion` works only on a computer that holds the course checkout and runs its launcher, `scripts/demo_server.py`, which also serves the slide buttons of Decks 02 and 03.

Browser module, and how to open it	Start, then what to watch	Static fallback
Frames module (§1, §3): open the module, guide . It opens in any browser and needs no installation.	It starts on the tray example with the readouts (0.150, 0.050), (0.105, 0.118) and (0.505, 0.318). Turning the tray to 90° keeps the tray-axes numbers and changes the world-axes numbers (self-check 1).	Figure 1
Rotations module (§2): open tab 3 , open the gimbal-lock view, guide . It needs WebGL (browser 3D graphics) and downloads one library from a content delivery network (CDN) the first time.	Tab 3 shows the tray's 30° about z as a matrix, as axis-angle and as a quaternion. Tick “hand this quaternion to a library that expects (x, y, z, w)” to see the orange ghost of the component-order misread. The gimbal-lock view on tab 2 shows two axes aligning.	the list of encodings in §2
Two-link module (§§6–8): open at the worked pose, guide . It opens in any browser.	It starts at the worked pose with the two column arrows, $\det \mathbf{J} = 0.0849$ and the ellipse. Dragging joint 2 to 0° makes the arrows parallel and collapses the ellipse; the radial and tangential buttons show a dropped and a met request (§8.4).	the tangents figure of §6.1; Figures 8.1 and 8.2
Gauss–Newton IK module (Appendix C.3): open on the worked example, guide . It opens in any browser.	It starts with the arm at $(30^\circ, 45^\circ)$ and the target (0.5, 0.2) m. Pressing Run shows five steps reaching the target on the residual chart, and the readout says “solved”. Without the query string the target is the unreachable (0.75, 0.05) m; with Armijo ticked, Run ends with the readout “stalled”.	Figure C.1

Drake demonstration, and how to run it	Start, then what to watch	Static fallback
Panda forward kinematics and Jacobian (§6): run <code>fk_jacobian_demo.py</code> with <code>--meshcat</code> , guide ; launcher route <code>/run/fk-jacobian</code> .	It starts at the ready pose on the sliders' 0.01 rad grid, with rank 6 and $\sigma_{\min} = 0.2246$ (0.2244 at the exact pose). The zero-configuration preset gives rank 5 and a lost rotation about world x (long reading §10.5).	Deck 03's picture of the zero configuration
FR3 chain and grasp target (§4.2, §5): run <code>fr3_grasp_chain.py</code> with <code>--meshcat</code> , guide ; launcher route <code>/run/fr3-chain</code> .	It starts at the ready configuration with the mug, grasp and approach frames fixed. Changing one joint moves the downstream frames and leaves the targets in place; Appendix D.3 describes the link colors and the opacity sliders.	the FR3 ready-pose figure of §4.2; the tables of Appendix D
Self-motion with the full pose held (§7.4): run <code>self_motion_panda.py</code> with <code>--meshcat</code> ; launcher route <code>/run/self-motion</code> .	The arm starts elbow up, and its one slider sets the arc length s along the curve. Moving it sweeps the elbow while the flange-pose error stays near 10^{-15} m and the rank stays 6.	Deck 03's self-motion picture
Fingertip-only app (§7.4): run <code>tip_only_panda.py</code> with <code>--app</code> , guide ; launcher route <code>/run/tip-only-app</code> .	The app opens paused at elbow up, and “Open Controls” shows the joint sliders in degrees. Dragging joint 4 changes the gripper tilt while the fingertip error stays near zero; “Random pose, same tip” shows other members of the family.	Deck 03's fingertip picture

Appendix A. Alternative notation used in papers

This appendix is optional reference material. Table A.1 lists, for each object of the core, the spelling this course uses, the spellings papers and libraries use, and the convention that a spelling may signal. For velocities and Jacobians, a change of name can also mean a change of reference point, expression frame or stacking order, so check those before comparing numbers.

Object	This course	Related spellings	Convention or source
position of B relative to A	${}^A\mathbf{p}^B$	$\mathbf{p}_{AB}, \mathbf{p}_{ab}, \mathbf{t}, \mathbf{o}_B, \mathbf{p_AB}$	Lynch & Park p_{ab} ; Drake code $\mathbf{p_AB}$
rotation of B relative to A	${}^A\mathbf{R}^B$	$\mathbf{R}_{AB}, \mathbf{R}_{ab}, {}^A_B\mathbf{R}, \mathbf{R}_B^A$	Lynch & Park R_{ab} ; Craig ${}^A_B R$
pose (rigid transform) of B relative to A	${}^A\mathbf{X}^B$	$\mathbf{T}_{AB}, {}^A\mathbf{T}_B, \mathbf{T}(\mathbf{q}), \mathbf{g}, \mathbf{H}, T_{W \rightarrow A}$	Lynch & Park T_{ab} ; most papers T ; RAI $T_{W \rightarrow A}$
configuration	$\mathbf{q} \in \mathbb{R}^n$	$\boldsymbol{\theta}, \mathbf{s}_{\text{joint}}$, “joint state”	Lynch & Park $\boldsymbol{\theta}$
forward kinematics	$\varphi(\mathbf{q}), {}^W\mathbf{X}^E(\mathbf{q})$	$f(\mathbf{q}), \text{FK}(\mathbf{q}), \mathbf{T}(\mathbf{q}), \mathbf{x} = f(\mathbf{q})$	most papers f or FK
task-space velocity	$\dot{\mathbf{x}}$	$\mathbf{v}, \mathcal{V}, \boldsymbol{\xi}, \boldsymbol{\nu}$, “twist”, “spatial velocity”	Check the task, reference point, expression frame and stacking order. Our geometric velocity uses the end-effector origin; Lynch & Park’s space twist uses a different linear component.
Jacobian	$\mathbf{J}(\mathbf{q})$	$J, J_a, J_g, J_b, J_s, \partial f / \partial \mathbf{q}$	Match its output convention to the requested motion. A pose-coordinate derivative and a geometric Jacobian need not have the same entries.
pseudo-inverse	$\mathbf{J}^+$	$J^\dagger, J^\#, \text{pinv}(J)$	most papers $J^\dagger$
damped pseudo-inverse	$\mathbf{J}^\#$ with ρ	$J_\lambda^\dagger, J^\top (J J^\top + \lambda I)^{-1}$	control papers, “damped least squares”
orientation in a dataset	$\mathbf{R}$	unit quaternion (w, x, y, z) or (x, y, z, w) , rotation vector, Euler angles, “6D rotation”	most manipulation datasets store position plus quaternion; the ordering differs by library and must be checked per dataset

Table A.1. Common names and related conventions. A notation match alone does not make two velocity vectors interchangeable. $\mathbf{T}$ is the pose in most papers but the planar translation $\mathbf{T}(a, b)$ in §4.1; T is a time horizon in the learning block.

Appendix B. Pose-error and orientation-loss variants

This appendix is optional reference material. Section 3 builds a motion error as a relative transform and scores a mismatch with a scalar loss, and §7 converts a desired pose change into a velocity request. The variants and frame conventions behind both are collected here.

B.1 The rotation angle from the trace, and losses on matrix entries

Section 3 scores an orientation mismatch by the angle θ of the relative rotation $\mathbf{R}^{*\top}\mathbf{R}$. For a 3D rotation through θ , the trace (the sum of the diagonal entries) is $1 + 2 \cos \theta$, giving

$$\theta = \arccos\left(\frac{\text{tr}(\mathbf{R}^{*\top}\mathbf{R}) - 1}{2}\right), \quad 0 \leq \theta \leq \pi.$$

An identical orientation has trace 3 and angle 0; a 90° mismatch has trace 1. In numerical code, clip the argument of `arccos` to $[-1, 1]$ to handle roundoff.

Matrix differences are mathematically valid as **losses**, even though they are not rigid motions. For example, the Frobenius norm (the square root of the sum of squared matrix entries) satisfies $\|\mathbf{R} - \mathbf{R}^*\|_F^2 = 4(1 - \cos \theta)$, a legitimate orientation penalty. A norm on the whole pose matrix implicitly combines that penalty with squared position error. Such a loss is allowed; choose its weighting deliberately and say what it measures.

B.2 Geometric pose errors for a Jacobian

Section 7 builds a full-pose velocity request in the Jacobian’s frame: the position displacement and the rotation-vector error, both divided by the control period, angular part first. This subsection gives the two common frame choices and the rule for a finite orientation-error residual.

With world-expressed geometric velocities, the displacement is $\mathbf{p}^* - \mathbf{p}$ and the world-expressed rotation error comes from $\mathbf{R}^*\mathbf{R}^\top$. The body-expressed error of §3 needs a frame conversion first. These quantities choose an instantaneous velocity request. When optimizing a finite orientation-error residual, use its derivative. With the target-minus-current convention above, the derivative near agreement is the negative of the world-expressed geometric angular block; with Optimization I’s current-minus-target residual, the sign is positive.

Appendix C. Null-space projection, damping and repeated steps

This appendix is optional reference material. It derives two results that §7 names and shows a third on the two-link example. None of them is needed to follow the core.

C.1 Projecting a preferred motion into the null space

Section 7 shows null-space motion as the difference of two valid answers to the same task request. The equations below convert a preferred joint motion into null-space motion.

To build one from a preferred joint-rate vector $\mathbf{z}$, first find the task velocity it would cause, $\mathbf{J}\mathbf{z}$. Then subtract the minimum-norm joint rates that would cause that same task velocity, $\mathbf{J}^+\mathbf{J}\mathbf{z}$. What remains has zero task velocity. Hence the full family of least-squares solutions is

$$\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}^+\mathbf{J})\mathbf{z}.$$

Here $\mathbf{I}$ is the $n \times n$ identity matrix. The second term can pursue a preferred posture without changing the first term's achieved task velocity. Preservation is instantaneous: a finite step can drift away from the target. The Drake self-motion demos recompute the null space from the singular value decomposition at each step and then correct the task error, the full pose or the tip position; one fixed null-space vector cannot trace the whole curved family of configurations. Null-space motion alone also does not guarantee obstacle avoidance or feasibility within joint limits.

C.2 The damped step, derived

Section 7 names damping and what it trades. This subsection states the damped least-squares step and derives it from the cost it minimizes.

Near a singularity, a small singular value means little tip motion per unit joint speed. Reversing that weak gain can demand enormous joint rates. Instead of insisting on the closest possible velocity at any joint speed, **damped least squares** balances velocity error against joint speed:

$$\min_{\dot{\mathbf{q}}} \|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|^2 + \rho\|\dot{\mathbf{q}}\|^2, \quad \rho > 0.$$

Differentiating both squared terms contributes the same factor of two. Canceling it and setting the derivative to zero gives $\mathbf{J}^\top(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}) + \rho\dot{\mathbf{q}} = 0$, so $(\mathbf{J}^\top\mathbf{J} + \rho\mathbf{I})\dot{\mathbf{q}} = \mathbf{J}^\top\dot{\mathbf{x}}$ and

$$\dot{\mathbf{q}} = (\mathbf{J}^\top\mathbf{J} + \rho\mathbf{I})^{-1}\mathbf{J}^\top\dot{\mathbf{x}}.$$

The inverse exists because positive damping gives positive cost curvature in every nonzero joint-rate direction:

$$\mathbf{z}^\top (\mathbf{J}^\top \mathbf{J} + \rho \mathbf{I}) \mathbf{z} = \|\mathbf{J}\mathbf{z}\|^2 + \rho \|\mathbf{z}\|^2 > 0 \quad \text{for } \mathbf{z} \neq \mathbf{0}.$$

Increasing ρ discourages joint motion more strongly and accepts more tracking error. Along a singular direction this replaces the gain $1/\sigma_i$ with $\sigma_i/(\sigma_i^2 + \rho)$, which stays finite and tends to zero as $\sigma_i \rightarrow 0$. This is a soft tradeoff, not a hard speed cap: explicit velocity and joint-limit constraints are separate requirements. The optimization block develops the method further.

C.3 Repeating the local step

Section 7 proposes one joint-position update from the local model and checks it by forward kinematics. This subsection shows what repeating it does on the two-link example; Optimization I derives the step, controls it and says when it fails.

At the new guess, recompute the residual and the Jacobian, then repeat (Figure C.1). This is the **Gauss–Newton** step for position least squares: it solves a new linear approximation at each guess. It is not a promise of convergence. A large step can invalidate the approximation; damping or a shorter step can help. Near an exact solution with an invertible Jacobian, the two-link example can converge quadratically. Unreachable targets, singularities and poor starting guesses do not have that guarantee. A different start may find a different solution or fail.

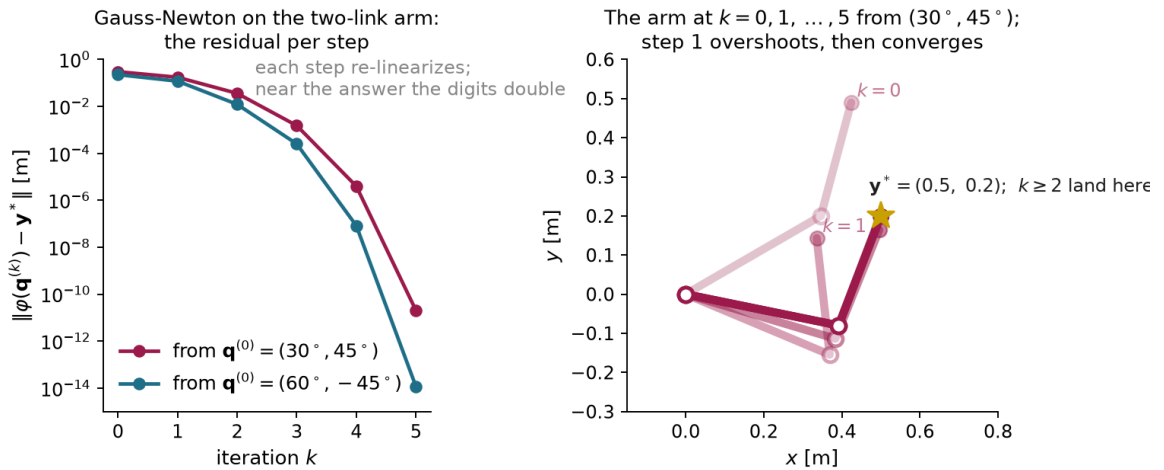


Figure C.1. Gauss–Newton inverse kinematics on the two-link arm: the residual per step from two starting configurations (left), and successive arm configurations from $(30^\circ, 45^\circ)$ toward $(0.5, 0.2)$ (right). Each step solves a local displacement problem with a freshly evaluated Jacobian.

Appendix D. Reproducing the FR3 chain

This appendix is optional reference material. Section 4.2 works one edge of the FR3 chain by hand. The constants for all seven edges, the intermediate origins that check a full multiplication, and the checks and viewer controls of `drake/01c-fr3-grasp-chain/fr3_grasp_chain.py` are collected here.

D.1 The constant factors of all seven joints

Each row supplies the constant translation $\mathbf{t}_i$ and the fixed turn α_i of the edge formula in §4.2, together with the measured angle at the ready configuration.

Joint i	Constant translation $\mathbf{t}_i$ in parent axes (m)	Fixed turn α_i	Measured turn q_i
1	(0, 0, 0.333)	0°	0°
2	(0, 0, 0)	−90°	−45°
3	(0, −0.316, 0)	90°	0°
4	(0.0825, 0, 0)	90°	−135°
5	(−0.0825, 0.384, 0)	−90°	0°
6	(0, 0, 0)	90°	90°
7	(0.088, 0, 0)	90°	45°

After link 7, a fixed shift of 0.107 m along its local z reaches the flange E ; that constant has no joint of its own.

D.2 Checkpoints along the chain

Continuing the multiplication of §4.2 through all seven joints and the flange shift gives the following independent checkpoints. Some model frames share an origin; the zero displacement between them does not imply that the corresponding physical links have zero size.

Frame reached	World position (m), rounded
link 1	(0, 0, 0.3330)
link 2	(0, 0, 0.3330)
link 3	(−0.2234, 0, 0.5564)
link 4	(−0.1651, 0, 0.6148)
link 5	(0.2189, 0, 0.6973)
link 6	(0.2189, 0, 0.6973)
link 7	(0.3069, 0, 0.6973)
flange E	(0.3069, 0, 0.5903)

D.3 Checks and viewer controls

The independent NumPy implementation reads the model’s fixed poses and joint axes and forms the chain itself. At nine tested configurations, its 90 cumulative frame poses agree with Drake to less than 4×10^{-16} in the matrix entries. That checks the implementation against the nominal model; it does not calibrate the physical robot. In the viewer (`--meshcat`), the **Toggle distinct link colors** control identifies physical links 0–7 with a legend; the color choice changes only the display. The **Link opacity** slider makes these meshes translucent. Around 0.35, stage 1 shows the link shells around solid internal frame markers. An opacity of 1 restores solid links. The separate **Mug opacity** slider makes the mug body and handle translucent. Around 0.25, you can see the selected frame through the surface. **Show mug frame M** and **Show grasp frame G^*** isolate the two frames of §5, and **Compare M and G^*** shows both: M is drawn with longer dashed axes and G^* with shorter solid axes. These are display styles; both frames keep their true poses. The mug pose and targets stay fixed.

Open questions and weak points.

- Full-pose control still requires careful frame conversion and angular/linear weighting; the numerical example verifies position control only.
 - The pseudo-inverse and damping equations do not enforce limits or collisions. Those require the constrained formulations in the optimization block.
 - The examples assume rigid links, calibrated geometry and a fixed base. Contact and model error require additional reasoning; a correct kinematic calculation alone cannot guarantee a grasp.
-

Sources. The long reading `notes.md` in this folder supplies the extended derivations and worked examples; Lynch & Park, *Modern Robotics* (2017), Chapters 2–6, and Tedrake, *Robotic Manipulation* (manipulation.mit.edu), Chapter 3, are the references behind it. Figures 1, 3, 8.2 and C.1 are drawn by `code/make_figures.py`; Figures 2, 4 and 8.1, the FR3 ready-pose figure, the tangent figure and the two joint-velocity plots are Deck 03's.