ELEC0129

1 Intro

1.1 Basic Notation

Position vector (point): $P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$.
Names are attached as a subscript: P_{home}.

Frame: $\{\cdot\}$, $\cdot$ is uppercase/numeric name.
"*Base* frame" (of a robot) is conventionally $\{0\}$.

Position vector in frame $\{\text{ref}\}$: ${}^{\text{ref}}P$.

Origin of frame $\{X\}$: $P_{X_{\text{org}}}$.

Rotation matrix of $\{B\}$ wrt. $\{A\}$: ${}^{A}_{B}R$.

Homogeneous transform matrix of $\{B\}$ wrt. $\{A\}$: ${}^{A}_{B}T$.

1.2 Transformations

Note that while ${}^{A}_{B}R$ "measures" $\{B\}$'s rotation in $\{A\}$ (${}^{A}_{B}R\hat{X}_A = \hat{X}_B$), when applied as a transformation (on position vectors), the "direction" is reversed, producing a $\{B\}$-to-$\{A\}$ frame substitution (${}^{A}P = {}^{A}_{B}R\,{}^{B}P$).
The same goes for other transforms, e.g., homogeneous transforms T.

> For frames $\{A\}$ and $\{B\}$:
> if they differ only by a **rotation**, then ${}^{A}P = {}^{A}_{B}R\,{}^{B}P$;
> if they differ only by a **translation**, then ${}^{A}P = {}^{B}P + {}^{A}P_{B_{\text{org}}}$ (where ${}^{A}P_{B_{\text{org}}} = P_{B_{\text{org}}} - P_{A_{\text{org}}}$);
> otherwise if **both**, then ${}^{A}P = {}^{A}_{B}T\,{}^{B}P$.

1.2.1 Rotations

Remember to use the right-hand corkscrew rule to figure which way is the positive angle.

1.2.1.1 Matrix Repr.

The rotation of $\{B\}$ wrt. $\{A\}$ can be represented as a 3×3 matrix:

$$ {}^{A}_{B}R = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} $$

where $\hat{X}, \hat{Y}, \hat{Z}$ are the orthonormal basis vectors of a frame.

Rotations about a given axis by θ:

$$ R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} $$

$$ R_Y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} $$

$$ R_Z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

Rotation matrices are orthogonal: ${}^{B}_{A}R = {}^{A}_{B}R^{-1} = {}^{A}_{B}R^{\top}$.

1.2.1.2 Roll-pitch-yaw Repr.

Aka. fixed-angle representation.

> Three rotations about a fixed set of axes (e.g. the base frame's).
> Modern terminology calls this *extrinsic* Euler angles.

For XYZ $\to \gamma\ \beta\ \alpha$:

$$ R = R_Z(\alpha)R_Y(\beta)R_X(\gamma) $$

$$ R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} $$

Useful inverses:

$$ \beta = \operatorname{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) $$

$$ \alpha = \operatorname{atan2}\left(\tfrac{r_{21}}{\cos\beta}, \tfrac{r_{11}}{\cos\beta}\right) $$

$$ \gamma = \operatorname{atan2}\left(\tfrac{r_{32}}{\cos\beta}, \tfrac{r_{33}}{\cos\beta}\right) $$

A second solution exists from the $-$`sqrt` branch ($\beta' = \pi - \beta$, with corresponding α', γ').

Singularity when $\cos\beta = 0 \iff \beta = \left(n + \frac{1}{2}\right)\pi$; only $\alpha \pm \gamma$ is recoverable.

1.2.1.3 Euler Angles Repr.

> Three rotations about a moving set of axes (that attach to the rotating object).
> Modern terminology calls this *intrinsic* Euler angles.

Note that extrinsic and intrinsic Euler angles are equivalent via a reversal of axis order;
For ZYX $\to \alpha\ \beta\ \gamma$:

$$ R = R_Z(\alpha)R_Y(\beta)R_X(\gamma) $$

As this is identical to the roll-pitch-yaw representation, the same formula applies, but with the angles pertaining to different axes.

1.2.1.4 Angle-Axis Repr.

Aka. Axis-Angle representation.

Rotation is described by a vector + a scalar:
an axis of rotation k and an angle of rotation θ about that axis.

$$ k = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} $$

Invariant: $\|k\| = 1$.

Problems:
- 0-vector is ambiguous: the axis becomes undefined.
- 0-degree rotations are problematic: they result in a 0-vector.

$$ R = \begin{bmatrix} k_x^2\,\operatorname{siv}\theta + \cos\theta & k_x k_y\,\operatorname{siv}\theta - k_z\sin\theta & k_x k_z\,\operatorname{siv}\theta + k_y\sin\theta \\ k_y k_x\,\operatorname{siv}\theta + k_z\sin\theta & k_y^2\,\operatorname{siv}\theta + \cos\theta & k_y k_z\,\operatorname{siv}\theta - k_x\sin\theta \\ k_z k_x\,\operatorname{siv}\theta - k_y\sin\theta & k_z k_y\,\operatorname{siv}\theta + k_x\sin\theta & k_z^2\,\operatorname{siv}\theta + \cos\theta \end{bmatrix} $$

where $\operatorname{siv}(\theta) = 1 - \cos\theta$.

Useful inverses:

$$ \theta = \arccos\left(\tfrac{1}{2}(r_{11} + r_{22} + r_{33} - 1)\right) $$

$$ k = \tfrac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} $$

Singularity when $\sin\theta = 0 \iff \theta = n\pi$.

Fallback for $\theta = \pi$: read $k_i^2 = \frac{r_{ii}+1}{2}$ from the diagonal; signs fixed by off-diagonals $r_{ij} = 2k_i k_j$ ($i \neq j$).

1.2.1.5 Euler Parameters Repr.

We can avoid singularities of three-parameter representations by, well, using four parameters.

For a rotation with axis represented by a unit vector k and angle θ, its four Euler parameters are:

$$ \varepsilon_1 = k_x \sin\tfrac{\theta}{2} $$
$$ \varepsilon_2 = k_y \sin\tfrac{\theta}{2} $$
$$ \varepsilon_3 = k_z \sin\tfrac{\theta}{2} $$
$$ \varepsilon_4 = \cos\tfrac{\theta}{2} $$

Invariant: $\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$.

$$ R = \begin{bmatrix} 1 - 2(\varepsilon_2^2 + \varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2 - \varepsilon_3\varepsilon_4) & 2(\varepsilon_1\varepsilon_3 + \varepsilon_2\varepsilon_4) \\ 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\varepsilon_4) & 1 - 2(\varepsilon_1^2 + \varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3 - \varepsilon_1\varepsilon_4) \\ 2(\varepsilon_1\varepsilon_3 - \varepsilon_2\varepsilon_4) & 2(\varepsilon_2\varepsilon_3 + \varepsilon_1\varepsilon_4) & 1 - 2(\varepsilon_1^2 + \varepsilon_2^2) \end{bmatrix} $$

Useful inverses:

$$ \varepsilon_4 = \tfrac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} $$
$$ \varepsilon_1 = \tfrac{1}{2}\sqrt{1 + r_{11} - r_{22} - r_{33}} $$
$$ \varepsilon_2 = \tfrac{1}{2}\sqrt{1 - r_{11} + r_{22} - r_{33}} $$
$$ \varepsilon_3 = \tfrac{1}{2}\sqrt{1 - r_{11} - r_{22} + r_{33}} $$

... shortcut if $\varepsilon_4 \neq 0$:

$$ \varepsilon_1 = \tfrac{r_{32} - r_{23}}{4\varepsilon_4} $$
$$ \varepsilon_2 = \tfrac{r_{13} - r_{31}}{4\varepsilon_4} $$
$$ \varepsilon_3 = \tfrac{r_{21} - r_{12}}{4\varepsilon_4} $$

No singularities by construction.

1.2.2 Translations

Translations can be represented by vector-vector addition; no matrices are needed.

1.2.3 Homogeneous Transforms

Essentially, a combination of rotations and translations. Can be used to represent any "movement" of a frame or rigid body (or a point, but they do not have an orientation).

${}^{A}P = {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{B_{\text{org}}}$
"P in frame A = P in frame B, first rotated by B's orientation in A, then translated by B's origin's offset in frame A"

This can be represented as a 4×4 *homogeneous transform* matrix:

$$ \left[\begin{array}{c} {}^{A}P \\ \hline 1 \end{array}\right] = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{B_{\text{org}}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} {}^{B}P \\ \hline 1 \end{array}\right] $$

By a slight abuse of notation, this is symbolically written as ${}^{A}P = {}^{A}_{B}T\,{}^{B}P$.

The homogeneous transform matrix is invertible. Multiple homogeneous transforms can be trivially converted to a single transform by computing the product of their respective matrices.

Because the rotation matrix is orthogonal, these transformation matrices are relatively easy to invert:

$$ {}^{A}_{B}T^{-1} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{B_{\text{org}}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R^{\top} & & -{}^{A}_{B}R^{\top}\,{}^{A}P_{B_{\text{org}}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = {}^{B}_{A}T $$

2 Terminology and Concepts

2.1 Joints

Revolute: rotation about an axis.

Prismatic: translation along an axis.

Both types of joints have an axis. For revolute joints, this is the axis of rotation; for prismatic joints, this is the axis of translation. The axes also have an orientation, determining the positive angle (using the right-hand corkscrew rule) or translation direction.
The choice of orientation is arbitrary, and usually are done to make the angle/translation signs more intuitive.

2.2 Frames

A *frame* is a local coordinate system that rigidly attaches to some object, usually a link or part of a robot.

Conventionally, frame 0, notated $\{0\}$, is the *base* frame of a robot, used to describe the robot's position and orientation in the world. Then, for a robot with n joints, we have frames $\{1\}$ to $\{n\}$ rigidly attached to each link (frame $\{i\}$ on link L_i). The end-effector gets a special additional frame, the *tool center point* (TCP).

3 Kinematics

3.1 Denavit-Hartenberg Parameters (modified)

In D-H convention, the transformation from one frame to another is only represented by 4 parameters; two degrees of freedom are removed (one translational, one rotational).
This works because the D-H convention **restricts** how a frame can be placed relative to its adjacent frames (in a linkage setup). With proper placement of frames, the convention can still capture any link geometry and model both revolute and prismatic joints.

Note that this course specifically uses the *Modified* D-H parameters, and so will all the notes below.

3.1.1 Frame Placement

Frame i rigidly attaches to link i. (i is 1-indexed, but the robot base frame is frame 0.)

One way to determine frame placements is by forward-progressing the constraints. We start with some *sensible*[1] placement of the base frame $\{0\}$, then for each subsequent frame $\{i\}$, its placement is constrained by how $\{i-1\}$ was placed.

Let x_i, y_i, z_i be the axes of frame $\{i\}$ (lines in 3d space with an arrow).
Let L_i be *link* i, and J_i be *joint* i.
Let n_i be the *common normal* between z_i and z_{i+1}: the line perpendicular to (and intersecting) both z_i and z_{i+1} (and if they are parallel, any such solution).

Frame axis assignment:
- z_i is the axis of J_i (the joint connecting L_{i-1} and L_i)
  - for revolute, oriented so the positive angle follows the right-hand corkscrew rule
  - for prismatic, oriented in the positive direction
- x_i is (parallel to) n_i
  - oriented from z_i to z_{i+1}
- y_i can then be deduced uniquely
  - oriented to form a right-handed coordinate system

The origin of the frame $\{i\}_{\text{org}}$ is of course the intersection point of z_i and x_i.

One consequential property of the placement rules is that x_i intersects and is perpendicular to z_{i+1} (and z_i). Also, z_i is collinear to the common normal between x_{i-1} and x_i, just like how x axes are to z axes.

3.1.1.1 Conventions/Tips

On base frame placement:
While $\{0\}$ has some freedom in its placement (z_0 is unconstrained since there is no joint 0), a strategic placement can simplify the D-H parameters. One common strategy is to place $\{0\}$ to coincide with $\{1\}$ when the J_1 is at its "home" or zero position, so that $a_0 = 0$ and $\alpha_0 = 0$. Another is to align z_0 with z_1 and place x_0 to 1. align with x_1 while J_1 is in its "home" state and 2. so $\{0\}_{\text{org}}$ is on the base/mount surface.

Frame $\{n\}$ placement:
- orient x_n to align with x_{n-1} when joint n is at its zero position
- a_{n-1} and α_{n-1} are then determined as usual from axes $n-1$ and n.

TCP frame: typically a generic 6-DOF transform from $\{n\}$ (tool geometry rarely fits D-H constraints).
- align z with the end-effector's *approach* direction (e.g. the shaft of a drill)
- align x with the end-effector's *lateral* direction, if it has one (e.g. the gripping direction of a gripper)

3.1.2 Transformations

After the frames are placed, the transformation T_i from $\{i-1\}$ to $\{i\}$ can be represented by the 4 (Modified) D-H parameters:

Param	Symbol	Description	Alternative Description
Link length	a_{i-1}	distance from z_{i-1} to z_i (along x_{i-1})	$\lvert(P_i - P_{i-1}) \cdot (\hat{z}_{i-1} \times \hat{z}_i)\rvert / \lVert\hat{z}_{i-1} \times \hat{z}_i\rVert$
Link twist	α_{i-1}	angle from z_{i-1} to z_i (about x_{i-1})	$\operatorname{atan2}((\hat{z}_{i-1} \times \hat{z}_i) \cdot \hat{x}_{i-1}, \hat{z}_{i-1} \cdot \hat{z}_i)$
Joint offset	d_i	distance from x_{i-1} to x_i along z_i	displacement, if prismatic
Joint angle	θ_i	angle from x_{i-1} to x_i about z_i	angle, if revolute

Intuitively, a_{i-1} and α_{i-1} describe the geometry of link i (z_{i-1} and z_i's relative position), while d_i and θ_i describe the configuration of joint i (how L_i moves relative to L_{i-1} by J_i).

Additional notes:
- α specifically measures the angles between *projections* of the z axes onto a plane normal to x (with sign given by the right-hand corkscrew rule). Same idea for θ
- when the orientation of x_i has multiple solutions (when z_{i-1} and z_i intersect or are collinear), x_i is conventionally oriented to make α_{i-1} be zero or positive
- when the placement of x_i has multiple solutions (when z_{i-1} and z_i are parallel), x_i is conventionally placed to pass through $\{i-1\}_{\text{org}}$

$$
{}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

3.2 Inverse Kinematics

Given desired end-effector pose, find joint parameters. Generally non-linear; no general algorithm exists.

3.2.1 Existence and Multiplicity

Workspace: volume reachable by the end-effector.
- *Reachable workspace*: all points reachable in at least one orientation.
- *Dexterous workspace*: subset where all orientations are reachable.
- Interior points typically admit multiple orientations; boundary points only one.
- *Joint limits* shrink workspace volume or reduce orientations per point.

Multiplicity: a single target pose often has several IK solutions.
- Planar RRR: 2 solutions (elbow-up / elbow-down).
- PUMA 560 (6R): 8 solutions = 4 arm configurations × 2 wrist-flips, where flipping is ${\theta_4}' = \theta_4 + \pi, {\theta_5}' = -\theta_5, {\theta_6}' = \theta_6 + \pi$.
- $n > 6$ DOF: infinitely many solutions – *redundant manipulator.*

Multiple solutions are useful for collision avoidance; the controller must still pick one (heuristic: smallest joint movement from current configuration).

Solution methods: *closed-form* (algebraic or geometric) and *numerical.* Only closed-form is in scope.

3.2.2 Geometric Solution to Planar RRR

For a 3-link planar manipulator (link lengths L_1, L_2, L_3) reaching $\{3\}$-frame pose (x, y, φ), solve via the base–elbow–wrist triangle.

> Given a desired end-effector pose (x_e, y_e, φ), back off by L_3 along the EE's orientation to get the wrist target: $x = x_e - L_3 \cos\varphi, \quad y = y_e - L_3 \sin\varphi$
> φ is unchanged since link 3 is rigidly oriented with $\{3\}$.

1. θ_2

<u>*law of cosines*</u> on the L_1-L_2 triangle (cos term sign flip as the interior elbow angle is $\pi - \theta_2$):

$$x^2 + y^2 = L_1^2 + L_2^2 + 2 L_1 L_2 \cos\theta_2$$

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2 L_1 L_2}\right)$$

Elbow-up: ${\theta_2}' = -\theta_2$.

2. θ_1

Let $\beta = \operatorname{atan2}(y, x)$ (angle from base to wrist) and ψ be the interior angle at the base of the same triangle:

$$L_2^2 = x^2 + y^2 + L_1^2 - 2 L_1 \sqrt{x^2 + y^2} \cos\psi$$

then $\theta_1 = \beta \pm \psi$ (elbow-up / elbow-down).

3. θ_3

Trivially, orientation closes out: $\theta_3 = \varphi - \theta_1 - \theta_2$.

This trick is planar-specific; non-planar arms require their own derivation.

3.2.3 Algebraic Solution to Planar RRR

For a planar RRR robot, we have

$$
{}^{0}_{3}T = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 & L_2\cos(\theta_1+\theta_2) + L_1\cos\theta_1 \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 & L_2\sin(\theta_1+\theta_2) + L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & x \\ \sin\varphi & \cos\varphi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

For a target $\{3\}$ pose of (x, y, φ), we want to solve for the joint angles θ_1, θ_2, and θ_3.

Clearly, we have
$\cos\varphi = \cos(\theta_1 + \theta_2 + \theta_3)$
$\sin\varphi = \sin(\theta_1 + \theta_2 + \theta_3)$
$x = L_2 \cos(\theta_1 + \theta_2) + L_1 \cos\theta_1$
$y = L_2 \sin(\theta_1 + \theta_2) + L_1 \sin\theta_1$

Firstly, θ_2

<u>*Square and sum*</u> the equations for x and y
$x^2 + y^2 = L_1^2 + 2L_1L_2(\cos\theta_1\cos(\theta_1+\theta_2) + \sin\theta_1\sin(\theta_1+\theta_2)) + L_2^2$
$\quad = L_1^2 + 2L_1L_2\cos\theta_2 \quad + L_2^2$
$\cos\theta_2 = \frac{x^2+y^2-L_1^2-L_2^2}{2L_1L_2}$
A clean way to obtain both solutions is to use atan2:
$\theta_2 = \operatorname{atan2}\left(\pm\sqrt{1-\cos^2\theta_2}, \cos\theta_2\right)$

Secondly, θ_1

With knowledge of $\cos\theta_2$ and $\sin\theta_2$, we can trig identity our way into reduced equations for x and y:
$x = L_2\cos(\theta_1+\theta_2) + L_1\cos\theta_1 \quad = L_2(\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2) + L_1\cos\theta_1$
$\quad = (L_2\cos\theta_2 + L_1)\cos\theta_1 - (L_2\sin\theta_2)\sin\theta_1$
$y = L_2\sin(\theta_1+\theta_2) + L_1\sin\theta_1 \quad = L_2(\sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2) + L_1\sin\theta_1$
$\quad = (L_1 + L_2\cos\theta_2)\sin\theta_1 + (L_2\sin\theta_2)\cos\theta_1$

Extract the common, now-known coefficients:
$K_1 = L_2\cos\theta_2 + L_1$
$K_2 = L_2\sin\theta_2$

Now, once we have
$x = K_1\cos\theta_1 - K_2\sin\theta_1$
$y = K_1\sin\theta_1 + K_2\cos\theta_1$

we perform <u>*trigonometric substitution*</u>: define γ and r as the angle (opposite K_2) and the hypotenuse of the right triangle with legs K_1 and K_2
$r = \sqrt{K_1^2 + K_2^2}$
$\gamma = \operatorname{atan2}(K_2, K_1)$

and so
$\frac{x}{r} = \cos\gamma\cos\theta_1 - \sin\gamma\sin\theta_1 = \cos(\gamma + \theta_1)$
$\frac{y}{r} = \cos\gamma\sin\theta_1 + \sin\gamma\cos\theta_1 = \sin(\gamma + \theta_1)$

finally
$\theta_1 = \operatorname{atan2}\left(\frac{y}{r}, \frac{x}{r}\right) - \gamma$

Lastly, θ_3

Observe that $\theta_1 + \theta_2 + \theta_3 = \operatorname{atan2}(\sin\varphi, \cos\varphi)$, so
$\theta_3 = \operatorname{atan2}(\sin\varphi, \cos\varphi) - \theta_1 - \theta_2$

3.3 Jacobians

Forward kinematics are functions of "joint angles" $\mapsto$ "end-effector pose".
Jacobians are the derivatives of these functions wrt. time; Jacobians map joint velocities to cartesian velocity (of end-effector).

It is the "full" derivative of a vector-valued multivariate function, represented as a matrix of partial derivatives.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ has a Jacobian J of size $m \times n$.
In robotics, our FK functions have output dimension $m = 6$ (3 for position + 3 for orientation), and input dimension n is the number of joints.

Sometimes $m = 3$ when only dealing with position (or orientation)… or even $m = 2$ for planar robots.

We have two main methods to compute the Jacobian of a FK function: velocity propagation and direct differentiation.

3.3.0.1 Notation

Absolute (wrt. $\{0\}$) angular velocity of $\{i\}$: ω_i.

Absolute (wrt. $\{0\}$) linear velocity of $\{i\}$: v_i.

These measures, as usual, can be expressed in any frame, e.g. ${}^{A}\omega_i$.
Without a TL-script, the frame of expression is implicitly $\{0\}$.

Angular velocity is represented as a vector; direction is axis and magnitude is speed.

Note that the cross-product $\omega \times P$ gives the tangential linear velocity at P due to w.

Generalized joint parameter (represents either θ or d): q_i

$\boldsymbol{q} = \begin{bmatrix} q_1 \\ \dots \\ q_n \end{bmatrix}$

Jacobian (linear + angular): $\boldsymbol{J}$
Liner-velocity Jacobian: $\boldsymbol{J}_v$
Angular-velocity Jacobian: $\boldsymbol{J}_\omega$

3.3.1 Velocity Propagation

Quite intuitive; roughly,
1. describe each link's "own" (relative to its own frame) velocity as a function of its joint velocity
2. start from base and "propagate" the velocities down the chain using the transformations between frames
3. express the end-effector velocity in the base frame
4. extract the Jacobian by matching coefficients of $\dot{q}_i$

This works for both linear and angular velocities.

Start with a static base:

$$
\begin{aligned}
{}^{0}\omega_0 &= 0 \\
{}^{0}v_0 &= 0
\end{aligned}
$$

For **revolute** joint $i+1$:

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\hat{Z} \\
{}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\,({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1})
\end{aligned}
$$

For **prismatic** joint $i+1$:

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i \\
{}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\,({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + \dot{d}_{i+1}\hat{Z}
\end{aligned}
$$

End-effector $\{e\}$ rigidly attached to $\{n\}$ at offset ${}^{n}P_e$:

$$
\begin{aligned}
{}^{n}\omega_e &= {}^{n}\omega_n \\
{}^{n}v_e &= {}^{n}v_n + {}^{n}\omega_n \times {}^{n}P_e
\end{aligned}
$$

Re-express in the base frame:

$$
\begin{aligned}
{}^{0}v_e &= {}^{0}_{n}R\,{}^{n}v_e \\
{}^{0}\omega_e &= {}^{0}_{n}R\,{}^{n}\omega_e
\end{aligned}
$$

3.3.2 Direct Differentiation

> This method, unlike velocity propagation, does not work well for angular velocities – differentiating a rotation matrix is technically possible ($\dot{R} = [\omega]_\times R$) but extracting the angular Jacobian column-by-column this way is painful.

1. obtain the FK function $[x, y, z] = \boldsymbol{f}(q_1, …, q_n)$
2. differentiate f wrt. t; compute all its partial derivatives $\frac{\partial \boldsymbol{f}}{\partial q_i}$
3. arrange partials in a matrix to obtain (linear-velocity-only) Jacobian $\boldsymbol{J}_v$

$$
\begin{aligned}
\dot{x} &= \frac{\partial f_x}{\partial q_1}\dot{q}_1 + \dots + \frac{\partial f_x}{\partial q_n}\dot{q}_n \\
\dot{y} &= \frac{\partial f_y}{\partial q_1}\dot{q}_1 + \dots + \frac{\partial f_y}{\partial q_n}\dot{q}_n \\
\dot{z} &= \frac{\partial f_z}{\partial q_1}\dot{q}_1 + \dots + \frac{\partial f_z}{\partial q_n}\dot{q}_n
\end{aligned}
$$

$$
\boldsymbol{J}_v = \begin{bmatrix} | & | & & | \\ \frac{\partial \boldsymbol{f}}{\partial q_1} & \frac{\partial \boldsymbol{f}}{\partial q_2} & \cdots & \frac{\partial \boldsymbol{f}}{\partial q_n} \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \leftarrow & \nabla^\top f_x & \rightarrow \\ \leftarrow & \nabla^\top f_y & \rightarrow \\ \leftarrow & \nabla^\top f_z & \rightarrow \end{bmatrix} = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \frac{\partial f_x}{\partial q_2} & \cdots & \frac{\partial f_x}{\partial q_n} \\ \frac{\partial f_y}{\partial q_1} & \frac{\partial f_y}{\partial q_2} & \cdots & \frac{\partial f_y}{\partial q_n} \\ \frac{\partial f_z}{\partial q_1} & \frac{\partial f_z}{\partial q_2} & \cdots & \frac{\partial f_z}{\partial q_n} \end{bmatrix}
$$

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$$

3.4 other stuff

Singularities values of $\boldsymbol{q}$ that result in $\det(\boldsymbol{J}) = 0$

Statics

The Jacobian also maps end-effector forces to joint torques:

$$\boldsymbol{\tau} = \boldsymbol{J}^\top \boldsymbol{F}$$

Derivation:
consider mechanical power in both joint space and cartesian space:

$$
\begin{aligned}
\text{Power} &= \boldsymbol{F} \cdot \dot{\boldsymbol{x}} = \boldsymbol{F}^\top \dot{\boldsymbol{x}} \\
\text{Power} &= \boldsymbol{\tau} \cdot \dot{\boldsymbol{q}} \ = \boldsymbol{\tau}^\top \dot{\boldsymbol{q}}
\end{aligned}
$$

then, by the conservation of energy,

$$
\begin{aligned}
\boldsymbol{\tau}^\top \dot{\boldsymbol{q}} &= \boldsymbol{F}^\top \dot{\boldsymbol{x}} \\
\boldsymbol{\tau}^\top \dot{\boldsymbol{q}} &= \boldsymbol{F}^\top (\boldsymbol{J}\dot{\boldsymbol{q}}) \\
\boldsymbol{\tau}^\top &= \boldsymbol{F}^\top \boldsymbol{J} \\
\boldsymbol{\tau} &= \boldsymbol{J}^\top \boldsymbol{F}
\end{aligned}
$$

Above uses $\boldsymbol{J}_v$ only – assumes no end-effector moment. For a full wrench $\mathcal{F} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{N} \end{bmatrix}$, the $6 \times n$ Jacobian is required.

At a singularity, the same duality means certain end-effector forces require *zero* joint torque (the structure carries them directly through the links). E.g. fully extended arm supporting a vertical load.

4 Dynamics and Control

4.1 Trajectory Planning

Design a *time profile* $u = f_t(t)$ for the robot to follow between waypoints; $\dot{u}$ and $\ddot{u}$ then follow by differentiation. Not concerned with the geometric path / route – only the temporal evolution.

Given u_0, u_f, t_f, design $u(t)$. u can be either Cartesian or joint-space.

Joint space: design $f_\theta(t)$ for each joint independently.
- Pro: cheap (no IK at runtime); free of singularity / workspace issues.
- Con: Cartesian path is non-linear, may collide.

Cartesian space: design $f_x(t), f_y(t),...$ directly.
- Pro: enforces geometric path shape and end-effector orientation.
- Con: must solve IK at every time step; intermediate points may leave workspace, switch IK branches, or pass near singularities.

4.1.1 Straight Line

$u(t) = \frac{u_f - u_0}{t_f} t + u_0$ for $t < t_f$, else u_f.

Velocity discontinuous at endpoints → jerky motion, vibrations, wear. Rarely usable directly.

4.1.2 Cubic Polynomial

$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

4 conditions: $u(0) = u_0$, $u(t_f) = u_f$, $\dot{u}(0) = 0$, $\dot{u}(t_f) = 0$.

$a_0 = u_0$. $a_1 = 0$. $a_2 = \frac{3}{t_f^2}(u_f - u_0)$. $a_3 = -\frac{2}{t_f^3}(u_f - u_0)$

Boundary accelerations are nonzero ($\ddot{u}(0) = 2a_2$); acceleration is discontinuous at $t = 0, t_f$.

4.1.3 Quintic Polynomial

Adds $\ddot{u}(0) = 0$ and $\ddot{u}(t_f) = 0$ for 6 total conditions.

$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$

$a_0 = u_0$. $a_1 = 0$. $a_2 = 0$

$a_3 = \frac{10}{t_f^3}\Delta$. $a_4 = -\frac{15}{t_f^4}\Delta$. $a_5 = \frac{6}{t_f^5}\Delta$

where $\Delta = u_f - u_0$. Smooth acceleration throughout.

4.1.4 Linear with Parabolic Blends (LSPB)

Three segments:
1. parabolic blend $[0, t_b]$ at constant acceleration $\ddot{u}$
2. linear (constant-velocity) middle $[t_b, t_f - t_b]$
3. parabolic deceleration $[t_f - t_b, t_f]$ at $-\ddot{u}$

Assumptions:
- equal blend durations t_b (symmetric)
- symmetric about midpoint: $(t_b, u_b) = \left(\frac{t_f}{2}, \frac{u_0+u_f}{2}\right)$
- velocity continuous at end of first blend: $\dot{u}t_b = \frac{u_f - u_0}{t_f - t_b}$

In the first blend, $\dot{u} = \ddot{u}t$ and $u = u_0 + \frac{1}{2}\ddot{u}t^2$.

Substituting symmetry into the velocity-matching condition yields the quadratic $\ddot{u}t_b^2 - \ddot{u}t_f t_b + (u_f - u_0) = 0$

Taking the minus branch ($t_b \leq \frac{t_f}{2}$):

$t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0)}}{2\ddot{u}}$

$u_b = u_0 + \frac{1}{2}\ddot{u}t_b^2$

Existence: requires $\ddot{u} \geq 4\frac{u_f - u_0}{t_f^2}$

(else discriminant is negative – chosen acceleration too small for the linear region to exist).

Piecewise trajectory:

$$u(t) = \begin{cases} u_0 + \frac{1}{2}\ddot{u}t^2 & t < t_b \\ \frac{u_b - u_0}{t_b - t_b}(t - t_b) + u_b & t_b \leq t < t_f - t_b \\ u_f - \frac{1}{2}\ddot{u}(t_f - t)^2 & t \geq t_f - t_b \end{cases}$$

4.1.5 Multi-Segment Trajectories

Concatenate per-segment trajectories with constant-u dwell intervals between moves. Each segment uses one of the schemes above.

4.2 Manipulator Dynamics

$$M(\mathbf{q})\ddot{\mathbf{q}} + V(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q}) = \boldsymbol{\tau}$$

$M(\mathbf{q})$ is the $n \times n$ *mass matrix*;
$V(\mathbf{q}, \dot{\mathbf{q}})$ is an n-dim vector of *centrifugal* ($\propto \dot{q}_i^2$) and *Coriolis* ($\propto \dot{q}_i\dot{q}_j, i \neq j$) terms;
$G(\mathbf{q})$ is an n-dim vector of gravitational forces.

M is positive-definite and symmetric, hence invertible.
V can be derived from M (out-of-scope) and vanishes if M is constant.

4.2.1 Newton-Euler Formulation

$$\begin{aligned} \boldsymbol{F} &= m\ddot{\boldsymbol{s}} \\ N &= {}^I I\dot{\omega} + \omega \times {}^I I\omega \end{aligned}$$

${}^I I$ is the *inertia tensor* at the body's center-of-mass frame $\{c\}$.

Two passes:
1. *Outward iteration* ($\{0\} \to \{e\}$): propagate velocities and accelerations, then compute each link's net force/moment at its CoM.
2. *Inward iteration* ($\{e\} \to \{0\}$): solve for the inter-link wrench transmitted across each joint; extract joint torque/force by projection onto the joint axis.

4.2.1.1 Outward Iteration

Initial conditions: ${}^0\omega_0 = 0$, ${}^0\dot{\omega}_0 = 0$, ${}^0\ddot{v}_0 = -g$.

> Gravity trick: setting ${}^0\ddot{v}_0$ to the opposite of the gravity vector is a clean substitution for the G component of the dynamics equation.

For each link $i = 0, ..., n-1$:

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}R\,{}^i\omega_i + \dot{\theta}_{i+1}\hat{Z} \\ {}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}R\,{}^i\dot{\omega}_i + \left({}^{i+1}R\,{}^i\omega_i \times \dot{\theta}_{i+1}\hat{Z}\right) + \ddot{\theta}_{i+1}\hat{Z} \\ {}^{i+1}\ddot{v}_{i+1} &= {}^{i+1}R\left({}^i\dot{\omega}_i + {}^i\omega_i \times {}^iP_{i+1} + {}^i\omega_i \times ({}^i\omega_i \times {}^iP_{i+1})\right) \\ &\quad + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\hat{Z} + \ddot{d}_{i+1}\hat{Z} \end{aligned}$$

Prismatic terms ($\dot{d}, \ddot{d}$) drop for revolute joints; rotational $\dot{\theta}, \ddot{\theta}$ terms drop for prismatic.

Propagate the linear acceleration to the CoM:

$${}^{i+1}\ddot{v}_{C_{i+1}} = {}^{i+1}\ddot{v}_{i+1} + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}})$$

Net wrench at the CoM:

$$\begin{aligned} {}^{i+1}F_{i+1} &= m_{i+1}\,{}^{i+1}\ddot{v}_{C_{i+1}} \\ {}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times ({}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1}) \end{aligned}$$

4.2.1.2 Inward Iteration

Initial: ${}^{n+1}f_{n+1}$ and ${}^{n+1}n_{n+1}$ set to the external forces.

For $i = n, ..., 1$:

$$\begin{aligned} {}^i f_i &= {}_{i+1}^{\ i}R\,{}^{i+1}f_{i+1} + {}^iF_i \\ {}^i n_i &= {}_{i+1}^{\ i}R\,{}^{i+1}n_{i+1} + {}^iP_{i_c} \times {}^iF_i + {}^iP_{i+1} \times ({}_{i+1}^{\ i}R\,{}^{i+1}f_{i+1}) + {}^iN_i \end{aligned}$$

Extract the joint torque/force by projection onto the joint axis:

$$\tau_i = \begin{cases} {}^i n_i \cdot \hat{Z} & \text{if revolute} \\ {}^i f_i \cdot \hat{Z} & \text{if prismatic} \end{cases}$$

4.2.2 Inertia Tensor

For a body with a frame $\{A\}$ at a chosen reference point:

$${}^A I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

Mass moments of inertia (diagonal):

$$I_{xx} = \iiint_V (y^2 + z^2)\rho\, dV$$

$$I_{yy} = \iiint_V (x^2 + z^2)\rho\, dV$$

$$I_{zz} = \iiint_V (x^2 + y^2)\rho\, dV$$

Mass products of inertia (off-diagonal):

$$I_{xy} = \iiint_V xy\rho\, dV$$

$$I_{xz} = \iiint_V xz\rho\, dV$$

$$I_{yz} = \iiint_V yz\rho\, dV$$

Principal axes: a frame orientation for which all products of inertia vanish; diagonal entries are then the *principal moments*. For dynamics, place the frame at each link's COM, giving ${}^C I_i$.

E.g. **rectangular block** ($l \times w \times h$), frame at the centre, axes aligned with edges (all products vanish by symmetry):

$$I_{xx} = \frac{m}{12}(w^2 + h^2), \quad I_{yy} = \frac{m}{12}(l^2 + h^2), \quad I_{zz} = \frac{m}{12}(l^2 + w^2)$$

4.2.3 Cartesian-Space Dynamics

Premultiply joint-space dynamics by $\boldsymbol{J}^{-T}$ and use $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F}$, $\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$:

$$M_x\ddot{\boldsymbol{x}} + V_x + G_x = \boldsymbol{F}$$

where

$$M_x = \boldsymbol{J}^{-T} M \boldsymbol{J}^{-1}$$

$$V_x = \boldsymbol{J}^{-T}\left(V - M\boldsymbol{J}^{-1}\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right)$$

$$G_x = \boldsymbol{J}^{-T} G$$

Useful when forces are naturally expressed in task space (e.g. polishing).
One may also use the Jacobian expressed in $\{e\}$ so the force aligns with tool axes regardless of pose: ${}^e\boldsymbol{J}_e = {}_0^e\boldsymbol{R}\,{}^0\boldsymbol{J}_e$

4.2.4 Friction

$$M(\mathbf{q})\ddot{\mathbf{q}} + V(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q}) = \boldsymbol{\tau} - \boldsymbol{\tau}_f$$

where $\boldsymbol{\tau}_f$ is the friction forces.

Common models:
- *Viscous*: $\tau_f = k\dot{\mathbf{q}}$.
- *Coulomb*: $\tau_f = c\,\mathrm{sgn}(\dot{\mathbf{q}})$ (constant magnitude, sign of velocity).
- *Combined*: $\tau_f = c\,\mathrm{sgn}(\dot{\mathbf{q}}) + k\dot{\mathbf{q}}$.

4.3 Manipulator Control

4.3.1 Second-Order Linear Systems

E.g., **Mass-Spring System**

Frictionless:

$$m\ddot{x} + kx = 0$$

$$x = R\cos\left(\sqrt{\frac{k}{m}}t - \varphi\right)$$

With viscous friction b:

$$m\ddot{x} + b\dot{x} + kx = 0$$

Characteristic equation:

$$m\lambda^2 + b\lambda + k = 0$$

$$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mk}}{2m}$$

- Underdamped: complex conjugate roots
  $x = e^{\sigma t}(c_1\cos(\omega_1 t) + c_2\sin(\omega_1 t))$
  where $\lambda_1 = p \pm q\mathrm{i}$
- Critically damped: repeated real root
  $x = (c_1 + c_2 t)e^{\lambda_1 t}$
- Overdamped: two real roots
  $x = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$

In a control system that models a mass-spring-damper system, $k_p = k$ acts as the *stiffness* of the system...

For some arbitrary choice of k_p, critical damping is achieved by $k_v = b = 2\sqrt{mk_p}$.

Note that this is a PD (proportional-derivative) controller.

4.3.2 Control Law Partitioning

One can partition the controller into two parts, a model-based compensator and a servo controller, s.t. the servo controller's parameters are completely independent of the non-design physical parameters.

1. Given system dynamics $m\ddot{x} + b\dot{x} + kx = F$
2. Design model-based compensator:
   - delegate highest-order control (f): $F = mf + b\dot{x} + kx$
   - system is reduced to unit mass ($\ddot{x} = f$)
3. Design servo controller
   - $f = -k_v\dot{x} - k_p x$

I.e.,
$m\ddot{x} + b\dot{x} + kx = F = m(-k_v\dot{x} - k_p x) + b\dot{x} + kx$
$\ddot{x} + k_v\dot{x} + k_p x = 0$

Unit-mass removes m from the critical damping condition, so $k_v = 2\sqrt{k_p}$.

4.3.3 Nonzero Setpoint

Following a partitioned controller, the servo controller can be trivially parameterized:
$f = -k_v\dot{x} - k_p(x - x_d)$
where x_d is the desired setpoint.

4.3.4 Trajectory Following

Once again, only the servo controller changes:
$f = \ddot{x}_d - k_v(\dot{x} - \dot{x}_d) - k_p(x - x_d)$
where x_d, $\dot{x}_d$, and $\ddot{x}_d$ are the desired position, velocity, and acceleration at the current time step.

Note that if we define the *tracking error* as $e = x - x_d$, then the closed-loop dynamics of the error is
$\ddot{e} + k_v\dot{e} + k_p e = 0$
which is a mass-spring-damper system in of itself!

4.3.5 Motor Physics

For a permanent magnet DC motor:
- $\tau_m = k_m i_a$
- $V_{emf} = k_e\dot{\theta}_m$

where k_m is the motor's torque constant; k_e is its electrical constant

A simple circuit model of one motor is

$$L_a\frac{di_a}{dt} + R_a i_a = V_a - V_{emf}$$

$$= V_a - k_e\dot{\theta}_m$$

...but let's assume the inductance L_a is negligible:

$$R_a i_a = V_a - k_e\dot{\theta}_m$$

For a rotary shaft (τ: gear) we have its moment of inertia I and viscous friction b.
Consider a motor-load-gear-gear system...

Motor-side dynamics:

$$I_m\ddot{\theta}_m = \tau_m - b_m\dot{\theta}_m - F\tau_m$$

Load-side dynamics (no additional τ as it is not motorized):

$$I\ddot{\theta} = F\tau - b\dot{\theta}$$

With $\eta = \frac{\tau_o}{\tau_m}$, the combined dynamics can be expressed in load-side terms as

$$(I + I_m\eta^2)\ddot{\theta} + (b + b_m\eta^2)\dot{\theta} = \tau$$

which once again is a mass-spring-damper system (driven by τ, with zero proportional component).

A partitioned control system:
- $\tau = (I + I_m\eta^2)f + (b + b_m\eta^2)\dot{\theta}$
- $f = \ddot{\theta}_d - k_v(\dot{\theta} - \dot{\theta}_d) - k_p(\theta - \theta_d)$
- set desired stiffness k_p, then $k_v = 2\sqrt{k_p}$

Finally, $\tau_m = \frac{1}{\eta}\tau$ and $\tau_m = k_m i_a$ so

$$V_a = R_a i_a + k_e\dot{\theta}_m$$

$$= R_a\left(\frac{\tau}{k_m\eta}\right) + k_e\eta\dot{\theta}$$

A function of voltage in terms of velocity and desired torque.

4.3.6 Multi-joint Control

A model-based compensator could also isolate individual joints in a multi-joint system!

Take the manipulator dynamics model $\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + V(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q})$
- $\boldsymbol{\tau} = M(\mathbf{q})\boldsymbol{f} + V(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q})$
- $\boldsymbol{f} = \ddot{\mathbf{q}}_d - K_v(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) - K_p(\mathbf{q} - \mathbf{q}_d)$

note that K_v and K_p are now matrices, but they are diagonal for decoupled control:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_n \end{bmatrix} - \begin{bmatrix} k_{v1} & 0 & \cdots & 0 \\ 0 & k_{v2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_{vn} \end{bmatrix}\begin{bmatrix} \dot{q}_1 - \dot{q}_{d1} \\ \dot{q}_2 - \dot{q}_{d2} \\ \vdots \\ \dot{q}_n - \dot{q}_{dn} \end{bmatrix} - \begin{bmatrix} k_{p1} & 0 & \cdots & 0 \\ 0 & k_{p2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_{pn} \end{bmatrix}\begin{bmatrix} q_1 - q_{d1} \\ q_2 - q_{d2} \\ \vdots \\ q_n - q_{dn} \end{bmatrix}$$