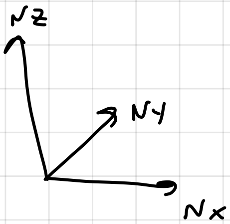
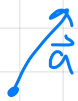
Attitude / Orientation (relative orientation of 2 frames)

vectors





dot products

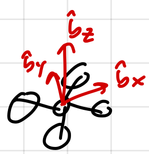
$\vec{a}$ expressed in N is $\begin{bmatrix} a \cdot n_x \\ a \cdot n_y \\ a \cdot n_z \end{bmatrix}$

where $\vec{c} \cdot \vec{d} = |c||d| \cos\theta$

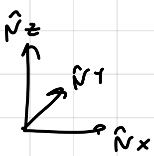
dot products

We will always assume a frame/basis to have three "right handed" unit vectors that are orthogonal. If we are given a generic vector a, we can express this vector in the basis N by taking the dot products with each unit vector, and storing them in the length 3 vector. The dot product between vectors in the geometric sense is the product of the vector lengths with the cosine of the angle between them.

Attitude



"body frame"



"world frame"

$$a \cdot b = \cancel{|a||b|} \cos\theta$$

A rotation matrix/direction cosine matrix (DCM) is simply a collection of dot products between unit vectors in two frames. Here we have N for a world frame, and B for a quadrotor body frame. Since these unit vectors are unit norm, the 9 entries in the DCM are simply the cosines of the angles between all of these.

$$^N Q^B = \begin{array}{c c} & \begin{array}{ccc} b_x & b_y & b_z \end{array} \\ \begin{array}{c} n_x \\ n_y \\ n_z \end{array} & \begin{bmatrix} (n_x \cdot b_x) & (n_x \cdot b_y) & (n_x \cdot b_z) \\ (n_y \cdot b_x) & (n_y \cdot b_y) & (n_y \cdot b_z) \\ (n_z \cdot b_x) & (n_z \cdot b_y) & (n_z \cdot b_z) \end{bmatrix} \end{array} \in SO(3)$$

- "Special" $\det(Q) = 1$
- "orthogonal" $^N Q^B = [^B Q^N]^{-1} = [^B Q^N]^T$

each row/column is norm 1

$$a_N = {}^N Q^B \, a_B$$

expressed in N

expressed in B

Here is how we take a vector expressed in B, and express it in N.

DCM's are in the group SO(3). Special for det = 1, orthogonal since all rows/columns are unit norm, and transpose = inverse. This should make intuitive sense given the construction of the matrix, in order to have B - N instead of N - B, you would just switch columns with rows (transpose).

$$\hat{a}_N = {}^N Q^B \, \hat{a}_B \, ({}^N Q^B)^T$$

This is also true, but kind of useless. We really only write it down to show parallels between DCM's and quaternions later.

$$\begin{bmatrix} \vec{a} \cdot n_x \\ \vec{a} \cdot n_y \\ \vec{a} \cdot n_z \end{bmatrix} = {}^N Q^B \begin{bmatrix} \vec{a} \cdot b_x \\ \vec{a} \cdot b_y \\ \vec{a} \cdot b_z \end{bmatrix}$$

Since these unit vectors are orthogonal ($n_x \cdot n_x = 1$, $n_x \cdot n_z = 0$)

You should multiply this out by hand and see this for yourself. It works out since we know that any unit vector dotted with itself is 1, and dotted with any of the other orthogonal unit vectors is 0.

Compositions

remember: $(ABC)^T = C^T B^T A^T$

$$ {}^{N}Q^{D} = {}^{N}Q^{B}\,{}^{B}Q^{C}\,{}^{C}Q^{D} $$

We can "compose" these DCM's with matrix multiplication, making sure to align the frames in the following manner.

$$ \left[{}^{D}Q^{N}\right]^T = \left[{}^{C}Q^{D}\right]^T \left[{}^{B}Q^{C}\right]^T \left[{}^{N}Q^{B}\right]^T $$

$$ {}^{N}Q^{D} = {}^{D}Q^{C}\;{}^{C}Q^{B}\;{}^{B}Q^{N} $$

We can transpose this whole expression and see that everything still makes sense.

Skew/hat/cross product matrix

Skew(A), $[A\times]$ $A^{\times}$ $A_{\times}$ $\tilde{A}$ $\hat{A}$

$$ \hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \qquad a \times b = \hat{a}\, b $$

$\hat{a}^T = -\hat{a}$

This "hat" map takes in a length 3 vector, and turns it into a skew-symmetric 3x3 matrix. Skew symmetric means transpose equals the negative of the matrix. This can also be used to take a cross product with matrix-vector multiplication. You will see this same matrix written out many different ways in textbooks/papers.

also define an "un-hat" $\vee$ to take a 3x3 and return a 3 vector

$$ \hat{x}_N = {}^{N}Q^{B}\,\hat{x}_B\,\left({}^{N}Q^{B}\right)^T $$ ← useless but will show a nice property later

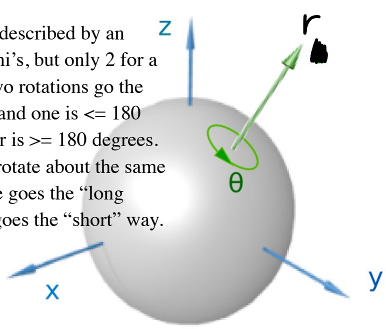
Again restating this method for converting a vector xb to xn that is useless, but paralells a similar operation with quaternions that we will go into later.

DCM as a rotation

Euler said between any 2 frames there is a simple rotation to get from 1 to the other

Euler figured out that between any 2 frames exists a simple rotation. This means instead of storing 9 dot products, I can just store a length 3 "Axis-angle" vector that describes the axis and angle required to rotate from one frame to the other. This representation is useful for describing rotations, but it is a poor choice for simulation since the kinematics are poorly define around 0, and you reach a singularity at 180 where you divide by a zero.

Any rotation can be described by an infinite number of phi's, but only 2 for a theta < 2pi. These two rotations go the opposite directions, and one is <= 180 degrees and the other is >= 180 degrees. Basically they both rotate about the same axis vector r, but one goes the "long way" and the other goes the "short" way.



unit vector

$$ \underset{\text{axis-angle}}{\phi} = \underset{\text{axis}}{r}\;\underset{\text{angle}}{\theta} $$

- not defined at 0
- singular at 180
- expensive/nonlinear

- Kinematics are awful, ${}^{N}\dot{\phi}^{B} = f({}^{N}\phi^{B}, {}^{N}\omega^{B})$
- for any attitude, we have 2 ϕ's under 2π
- singularity at $\theta = \pi$

Main Idea: Let's store the rotation between N and B instead of ${}^{N}Q^{B}$

DCM ↔ axis-angle

We can convert to and from DCM - axis angle with matrix exponential and matrix log. These operations can be special cased for the specific matrix types we see here (SO(3) for exp, skew symmetric for log).

$$\underbrace{{}^N\phi^B = \log({}^NQ^B)}$$

$${}^NQ^B = \exp(\widehat{{}^N\phi^B}) \quad , \text{ if } {}^N\phi^B \text{ is small}, \; {}^NQ^B \approx I + \hat{\phi}$$

"Rodrigues Formula"

there are "special" log, exp for these matrices

Rotations - 3 DOF

DCM -	9 terms -	6 cons	= 3 dof
ϕ	3	- 0	= 3 dof
q	4	- 1	= 3 dof

DCM 9 parameters - 6 constraints = 3 DOF. All attitude parameterizations are like this, no matter how many parameters, the number of constraints brings the DOF down to 3.

quaternion

always $\|q\| = 1$

$\mathbb{H}$ for quaternions

S^3 for unit quaternions

$${}^Nq^B = \begin{bmatrix} \cos\theta/2 \\ r\sin\frac{\theta}{2} \end{bmatrix} \begin{matrix} \text{Scalar} \\ \text{Vector} \end{matrix} \in \mathbb{R}^4 \; (\mathbb{H})$$

, you'll see this called $q = \mathrm{Exp}_q(\phi)$

$$q = \exp_q\left(\begin{bmatrix} 0 \\ \phi \end{bmatrix}\right)$$

${}^Nq^B, -{}^Nq^B$ describe the same attitude, 1 has a positive scalar (<180°)

same attitude, different rotations "double cover"

q and -q are same attitude, but different rotations. One goes <= 180 one goes >= 180, same as axis angle.

- below in red are elemental quaternion operations written using normal matrix-vector math

This section describes the elemental quaternion operations, then in red shows an alternative notation where all these operations are done using simple matrix/vector math.

- quaternion multiplication

$${}^Nq^C = {}^Nq^B \odot {}^Bq^C$$

$$\begin{bmatrix} s_1 \\ v_1 \end{bmatrix} \odot \begin{bmatrix} s_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} s_1 s_2 - v_1^T v_2 \\ s_1 v_2 + s_2 v_1 + v_1 \times v_2 \end{bmatrix} \begin{matrix} \text{scalar part} \\ \text{vector part} \end{matrix}$$

$$= \underbrace{\left[\begin{array}{c|c} s_1 & -v_1^T \\ \hline v_1 & s_1 I_3 + \hat{v}_1 \end{array}\right]}_{L(q)} \begin{bmatrix} s_2 \\ v_2 \end{bmatrix}$$

$$= \underbrace{\left[\begin{array}{c|c} s_2 & -v_2^T \\ \hline v_2 & s_2 I_3 - \hat{v}_2 \end{array}\right]}_{R(q)} \begin{bmatrix} s_1 \\ v_2 \end{bmatrix}$$

$$L(q^+) = L(q)^T = L(q)^{-1}$$

$$R(q^+) = R(q)^T = R(q)^{-1}$$

$$q_1 \odot q_2 = \underline{L(q_1)\, q_2}$$

$$= \underline{R(q_2)\, q_1}$$

quaternion conjugate

$${}^Nq^B = ({}^Bq^N)^+ = \begin{bmatrix} s \\ -v \end{bmatrix}$$

$${}^Nq^B = \underbrace{\begin{bmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix}}_{T} {}^Bq^N$$

for a $w \in \mathbb{R}^3$, quat "hat map"

$$\hat{\hat{w}} = \begin{bmatrix} 0 \\ w \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} 0\;0\;0 \\ I_3 \end{bmatrix}}_{H} w$$

$$\check{\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}} = \begin{bmatrix} q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\check{q} = \underline{H^T q}$$

unhat V

DCM

Here we show how similar the DCM and quaternion operations look with our notation. Then in red we show the quaternion stuff using the matrix/vector notation for everything.

quat

$$\hat{x}_N = {}^NQ^B \, \hat{x}_B \, ({}^NQ^B)^T \qquad \bar{x}_N = {}^Nq^B \odot \bar{x}_B \odot ({}^Nq^B)^\dagger \qquad H x_N = L({}^Nq^B) \, H x_B \, T \, {}^Nq^B$$

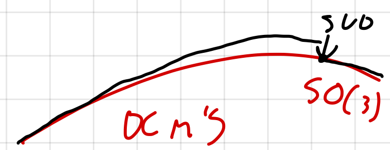
$${}^NQ^C = {}^NQ^B \, {}^BQ^C \qquad {}^Nq^C = {}^Nq^B \odot {}^Bq^C \qquad {}^Nq^C = L({}^Nq^B)\, {}^Bq^C = R({}^Bq^C)\, {}^Nq^B$$

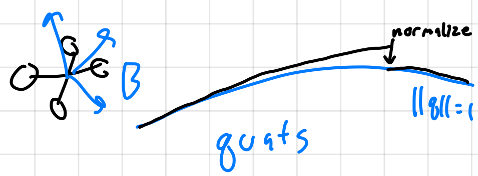
$${}^NQ^B = ({}^BQ^N)^T \qquad {}^Nq^B = ({}^Bq^N)^\dagger \qquad {}^Nq^B = T \, {}^Bq^N$$

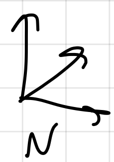
$${}^N\dot{Q}^B = {}^NQ^B \, \widehat{{}^N\omega_B^B} \qquad {}^N\dot{q}^B = \frac{1}{2} {}^Nq^B \odot \bar{\omega} \qquad {}^N\dot{q}^B = \frac{1}{2} L({}^Nq^B) H \omega$$

DCM from q

$${}^N\dot{Q}^B = {}^NQ^B \left[\widehat{{}^N\omega_B^B}\right]$$







$${}^N\omega_B^B$$

$$\bar{x}_N = {}^Nq^B \odot \bar{x}_B \odot ({}^Nq^B)^\dagger$$

$$H x_N = L(q) \left(R(q)^T H x_B \right)$$

$$x_N = \underbrace{H^T L(q) R(q)^T H}_{{}^NQ^B} \, x_B$$

Computing a DCM from a quaternion.

For simulation, both the DCM and quaternion have nice and simple kinematics equations without singularities. They are both great for simulation in this sense, but quaternions win out on the following two points:

1. the quaternion has 4 parameters < DCM's 9
2. numerical roundoff error will cause the DCM's to drift from orthogonality and an expensive SVD has to be used to project them back onto the SO(3) group. Quaternions on the other hand will drift away from their unit norm requirement, but renormalizing is significantly less computation than an SVD.

$$x = \begin{bmatrix} \text{position} \\ \text{velocity} \\ \text{quatern} \\ \omega \end{bmatrix}$$

$${}^{N}\dot{Q}^{B}$$

Kind of $\, {}^{B_t}\phi^{B_{t+\Delta t}} = \int_0^{dt} \omega \, dt \approx dt \cdot \omega$

For short periods (small dt), the following is approximately true. At the limit dt going to 0, this is true. This is useful for deriving the kinematics of attitude parameterizations.

$$\begin{aligned}
{}^{N}Q^{B_{t+1}} &= {}^{N}Q^{B_t}\,{}^{B_t}Q^{B_{t+1}} \\
&= {}^{N}Q^{B_t} \exp(dt \cdot \hat{\omega}) \\
&= {}^{N}Q^{B_t} (I + dt \cdot \hat{\omega})
\end{aligned}$$

$${}^{N}Q^{B_{t+1}} = {}^{N}Q^{B_t} + dt \cdot {}^{N}Q^{B_t}\,\hat{\omega}$$

$$\frac{{}^{N}Q^{B_{t+1}} - {}^{N}Q^{B_t}}{dt} = {}^{N}Q^{B_t}\,\hat{\omega}$$

Here is the DCM kinematics derivation.

$$\boxed{{}^{N}\dot{Q}^{B} = {}^{N}Q^{B}\,\hat{\omega}}$$

$3\times3 \quad 3\times3 \quad 3\times3$

Rodrigues parameter "Gibbs Vector"

$g = \frac{v}{s}$, so $\in \mathbb{R}^3$ rp_from_quat

we said $q, -q$ describe same attitude, g doesn't care

$g = \frac{v}{s} = \frac{-v}{-s}$

STILL SINGULAR AT 180°

Cayley map

$q = \text{normalize}\left(\begin{bmatrix} 1 \\ g \end{bmatrix}\right)$ quat_from_rp

Rodrigues parameters are used a lot in this class, they are simple, and don't have any "short" or "long" way ambiguities. All Rodrigues parameters describe rotations less than 180 (which covers all rotations, think about this if it's not obvious to you). However, RP's go singular at 180 degrees just like the axis angle.

Optimizing over attitude

We are looking to optimize over attitude, and below a set of conventions to modify our derivatives is laid out. This is all explained in more detail in the following paper: https://roboticexplorationlab.org/papers/planning_with_attitude.pdf

We think of things additively, if we have a small change to x called Δx

$$x + \Delta x$$

add : $x + \Delta x$

mult : $q \odot \text{qust\_frm\_mrp}(g)$

but this doesn't quite work for quaternions

$$q + \Delta q$$

Δq doesn't mean anything if it's not unit norm

but Taylor series are setup such that we add stuff. How do we compose a small rotation?

$$L(q)\,\text{quat\_from\_rp}(g)$$

Small changes are applied to q through quaternion multiplication, here we are representing our small rotation with a RP g, and converting it to a quaternion for multiplication.

if g is small, we have

$$L(q)\,\text{quat\_from\_rp}(g) \approx L(q)\begin{bmatrix}1\\ g\end{bmatrix} \approx L(q)\begin{bmatrix}1\\ 0\\ 0\\ 0\end{bmatrix} + L(q)Hg$$

$$\approx q + \underbrace{L(q)H}_{G(q)}g$$

"attitude Jacobian" $G(q) \in \mathbb{R}^{4\times 3}$

this shows we can get away with add g's to q's if we put a $L(q)H$ in front

another perspective

$$\left.\frac{\partial f(x+\Delta x)}{\partial \Delta x}\right|_{\Delta x = 0} = \frac{\partial f}{\partial x}$$

Here is another derivation of the G(q) matrix.

$$\left.\frac{\partial f(L(q)\,\text{q-from-rp}(g))}{\partial g}\right|_{g=0} = \frac{\partial f}{\partial q}G(q)$$

Taylor series for $f : \mathbb{H} \rightarrow \mathbb{R}^p$

Function that takes in a quaternion and outputs a vector (not a quaternion).

$$f(L(q)\,\text{quat-from-rp}(g)) \approx \underset{p\times 1}{f(q)} + \underset{p\times 4}{\frac{\partial f}{\partial q}}\,\underset{4\times 3}{G(q)}\,\underset{3\times 1}{g}$$

This means we just modify our jacobians with $G(q)$ on the right side to to make them compatible with g as the Δ

Let's use Newtons method with this:

In red is normal newton's method, in black is out modified newton's method when we are dealing with a quaternion input to our function.

	- we want $f(q)=0$
$f(x)=0$	
	- begin loop
$res = f(x)$	$res = f(q_k)$
$jac = \frac{\partial f}{\partial x}$	$jac = \frac{\partial f}{\partial q} G(q_k)$ — we modify to use g as Δ
$\Delta x = -jac^{-1} res$	$g = -jac^{-1} res$
$\alpha = LS$	$\alpha =$ linesearch
$x_{k+1} = x_k + \alpha \Delta x$	$q_{k+1} = L(q_k)\,\text{quat\_from\_rp}(\alpha \cdot g)$
	-end loop

- Jacobian of quaternion valued function $f: \mathbb{H}^4 \to \mathbb{H}^4$

$q' = f(q)$ — quat in, quat out

we modify this jacobian to be the following:

$$\underbrace{G(q')^T}_{3\times4} \underbrace{\frac{\partial f}{\partial q}}_{4\times4} \underbrace{G(q)}_{4\times3}$$

- Hessian of scalar $f: \mathbb{H} \to \mathbb{R}$

$$\underbrace{G(q)^T}_{3\times4} \underbrace{\nabla_q^2 f}_{4\times4} \underbrace{G(q)}_{4\times3} - \underbrace{I_3}_{3\times3} \underbrace{(q^T \nabla_q f)}_{\text{scalar}}$$

Review:

Jacobians

$f(q) : \mathbb{H} \rightarrow \mathbb{R}^p$ quat in, vector out

$$\frac{\partial f}{\partial q} G(q)$$

$q' = f(q) : \mathbb{H} \rightarrow \mathbb{H}$ quat in quat out

$$G(q')^T \frac{\partial f}{\partial q} G(q)$$

Hession

$f(q) : \mathbb{H} \rightarrow \mathbb{R}$ quat in, scalar out

$$G(q)^T \nabla_q^2 f \, G(q) - I_3 (q^T \nabla_q f)$$

Newtons method for unconstrained minimization

Example of Newton's method for unconstrained optimization of a scalar valued function with quaternion input.

$$\min_q \quad f(q)$$

- begin loop

T's because $\nabla_q f = \left(\frac{\partial f}{\partial q}\right)^T$ by convention

modified gradient $\quad \text{grad} = \left[(\nabla_q f(q_k))^T G(q_k)\right]^T$

modified Hessian $\quad H = G(q_k)^T \nabla_q^2 f(q_k) G(q_k) - I_3 (q_k^T \nabla_q f(q_k))$

$$g = -H^{-1} \text{grad}$$

$$\alpha = \text{linesearch}$$

$$q_{k+1} = L(q_k) \, \text{quat\_from\_rp}(\alpha \cdot g)$$

- end loop

$${}^N Q^N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Identity (no rot.)

$${}^N q^N = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

All attitude parameterizations have an "identity" denoting no rotation.

${}^{\sim}Q^{B}$ kinematics

if ${}^{\sim}\omega^{B}$ is constant and dt is small

$$ {}^{B_t}Q^{B_{t+1}} = \exp(\hat{\omega}\cdot dt) \underbrace{\approx I + dt\cdot\hat{\omega}}_{\text{and dt is small}} $$

$$ {}^{\sim}Q^{B_{t+1}} = {}^{\sim}Q^{B_t}\,{}^{B_t}Q^{B_{t+1}} $$

$$ = {}^{\sim}Q^{B_t}\exp(dt\cdot\hat{\omega}) $$

$$ = {}^{\sim}Q^{B_t}(I + dt\cdot\hat{\omega}) $$

$$ {}^{\sim}Q^{B_{t+1}} = {}^{\sim}Q^{B_t} + dt\cdot{}^{\sim}Q^{B_t}\hat{\omega} $$

$$ \frac{{}^{\sim}Q^{B_{t+1}} - {}^{\sim}Q^{B_t}}{dt} = {}^{\sim}Q^{B_t}\hat{\omega} $$

$$ \boxed{{}^{\sim}\dot{Q}^{B} = {}^{\sim}Q^{B}\hat{\omega}} $$

${}^{\sim}q^{B}$ kinematics

$$ {}^{\sim}q^{B_{t+1}} = {}^{\sim}q^{B_t}\odot{}^{B_t}q^{B_{t+1}} $$

$$ = {}^{\sim}q^{B_t}\odot\begin{bmatrix}1\\ \frac{dt}{2}\omega\end{bmatrix} $$

$$ {}^{\sim}q^{B_{t+1}} = {}^{\sim}q^{B_t} + \frac{1}{2}dt\,{}^{\sim}q^{B_t}\begin{bmatrix}0\\ \omega\end{bmatrix} $$

$$ \boxed{{}^{\sim}\dot{q}^{B} = \frac{{}^{\sim}q^{B_{t+1}} - {}^{\sim}q^{B_t}}{dt} = \frac{1}{2}{}^{\sim}q^{B}\odot\begin{bmatrix}0\\ \omega\end{bmatrix}} $$