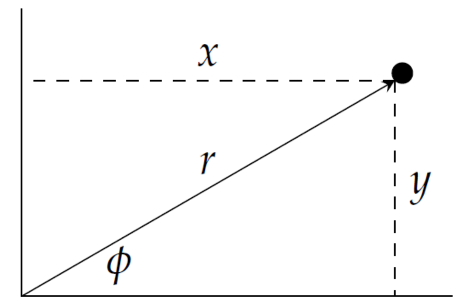
rotation in two dimensions

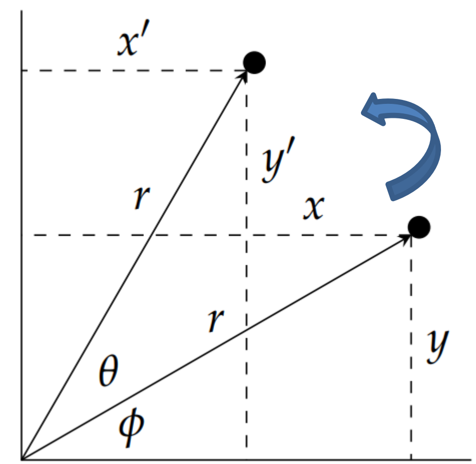
- Cartesian coordinate (x,y)
- Polar coordinate $(r, \emptyset)$

$$
\begin{aligned}
x &= r\cos\phi \\
y &= r\sin\phi \\
r &= \sqrt{x^2+y^2} \\
\phi &= \tan^{-1}\frac{y}{x}\,.
\end{aligned}
$$



- rotating a vector
  - suppose the vector is rotated by the angle θ
  - the new vector's polar coordinates : $(r, \emptyset + \theta)$

$$
\begin{aligned}
x' &= r\cos(\phi+\theta) \\
&= r(\cos\phi\cos\theta - \sin\phi\sin\theta) \\
&= (r\cos\phi)\cos\theta - (r\sin\phi)\sin\theta \\
&= x\cos\theta - y\sin\theta,
\end{aligned}
$$



rotation in two dimensions

- similarly, y' coordinate is given by

$$\begin{aligned} y' &= r\sin(\phi+\theta) \\ &= r(\sin\phi\cos\theta+\cos\phi\sin\theta) \\ &= (r\sin\phi)\cos\theta+(r\cos\phi)\sin\theta \\ &= y\cos\theta+x\sin\theta \\ &= x\sin\theta+y\cos\theta\,. \end{aligned}$$

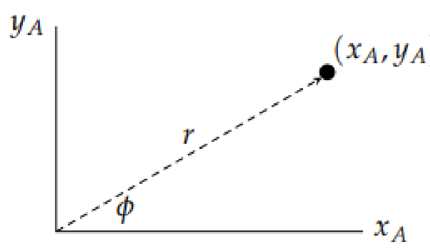
- In matrix form, we have

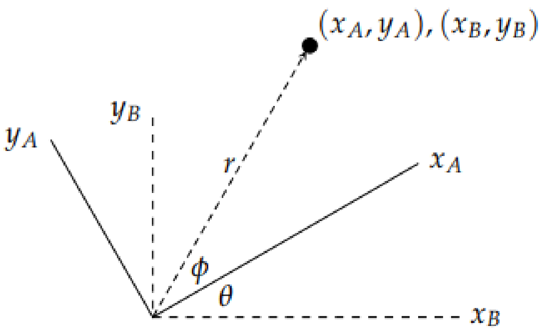
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- rotating a vector
- rotating the frame of reference

rotating the frame of reference

- Now the vector **is not** rotating. The frame of reference **is** rotating





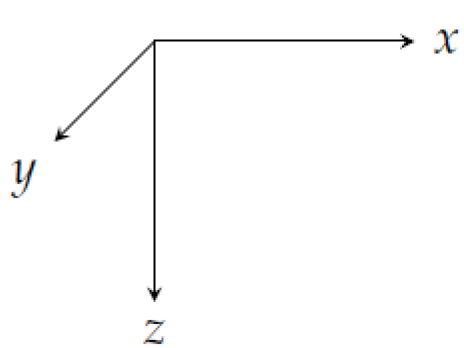
- relationship between two coordinate frames

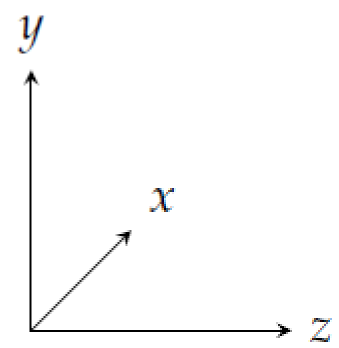
$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_A \\ y_A \end{bmatrix}$$

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_B \\ y_B \end{bmatrix}$$

rotation in three dimensions

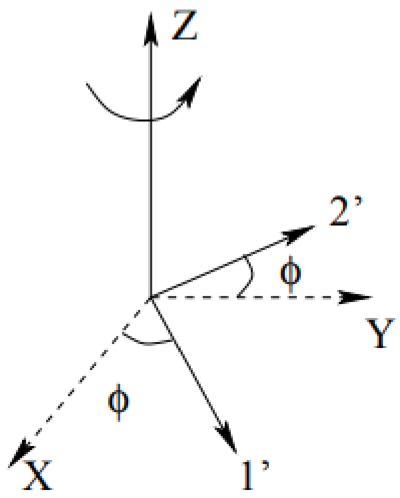
- three dimensional coordinate frame
  - right hand rule: For the familiar x- and y-axes on paper, curl your fingers on the short 90 path from the x-axis to the y-axis (not on the long 270 path from the x-axis to the y-axis). When you do so your thumb points out of the paper and this is taken as the positive direction of the z-axis.





rotation around z axis (frame rotation)

- rotation around z axis (Ø)



$$R_{old}^{new} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[r]_{new} = R_{old}^{new}[r]_{old}$$

- Ex: $\frac{\pi}{2}$ rotation

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_{new} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_{old}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{old}$$

rotation around x and y axis

- rotation around x axis (Ø)

- rotation around y axis (Ø)

rotation around x and y axis

- rotation around x axis (Ø)

$$R_{old}^{new} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

- rotation around y axis (Ø)

$$R_{old}^{new} = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}$$

Euler angles

- For general applications in 3D, often need to perform 3 separate rotations to relate our “inertial frame” to our “body frame”
  - Especially true for aircraft problems
- Standard: start with the body frame (x, y, z) aligned with the inertial (X, Y, Z), and then perform 3 rotations to reorient the body frame.

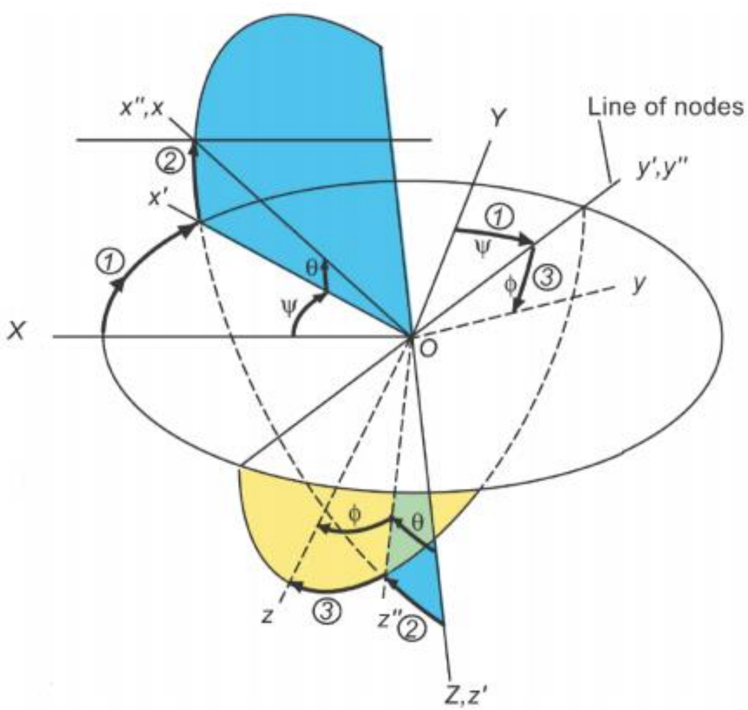
① Rotate by ψ about $Z \Rightarrow x', y', z'$

② Rotate by θ about $y' \Rightarrow x'', y'', z''$

③ Rotate by ϕ about $x'' \Rightarrow x, y, z$

Euler angles:

- $\psi \sim$ Heading/yaw
- $\theta \sim$ Pitch
- $\phi \sim$ Roll



Euler angles: ψ, θ, ϕ

- three rotations : $\psi \to \theta \to \phi$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = T_3(\psi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = T_2(\theta) \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = T_1(\phi) \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix}$$

body coordinate system

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = T_1(\phi) T_2(\theta) T_3(\psi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

inertial coordinate system

$$= \underbrace{\begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix}}_{C} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

rotation matrix to Euler angles

- rotation matrix C

$$\begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix}$$

$$C(1,3) = -\sin\theta \rightarrow \theta = -\sin^{-1}(C(1,3))$$

$$\frac{C(2,3)}{C(3,3)} = \frac{\sin\phi}{\cos\phi} = \tan\phi$$

$$\frac{C(1,2)}{C(1,1)} = \tan\psi$$

```
theta = -asin(C(1,3));

phi = atan2(C(2,3),C(3,3));

psi = atan2(C(1,2),C(1,1));
```

- P = atan2(Y,X) returns the four-quadrant inverse tangent

3D attitude estimation using inertial sensors

- three sensors (accelerometer, gyroscope, magnetic sensor)
  - accelerometer: $y_a \in R^3$, gyroscope: $y_g \in R^3$, magnetic sensor: $y_m \in R^3$

- attitude estimation using IMU (inertial measurement unit)

$$
\begin{aligned}
y_a &= C\tilde{g} \\
y_m &= C\tilde{m}
\end{aligned}
\qquad
\tilde{g} = \begin{bmatrix} 0 \\ 0 \\ 9.8 \end{bmatrix}
\qquad
\tilde{m} = m \begin{bmatrix} \cos\alpha \\ 0 \\ -\sin\alpha \end{bmatrix}
$$

triad algorithm (order is important)

Suppose unit vecotrs a_i and b_i $(i = 1, 2)$ are given and we want to find a rotation matrix C satisfying

$$a_1 = Cb_1, \quad a_2 = Cb_2. \tag{1}$$

If a_i and b_i are not unit vectors, make them unit vectors.

three orthnormal vecotrs

$$r_1 = b_1, \quad r_2 = \frac{b_1 \times b_2}{\|b_1 \times b_2\|}, \quad r_3 = r_1 \times r_2$$

```
f = cross(b1,b2);
f = f / norm(foo)
```

$$s_1 = a_1, \quad s_2 = \frac{a_1 \times a_2}{\|a_1 \times a_2\|}, \quad s_3 = s_1 \times s_2$$

Now there exists a unique orthogonal matrix C which satisfies

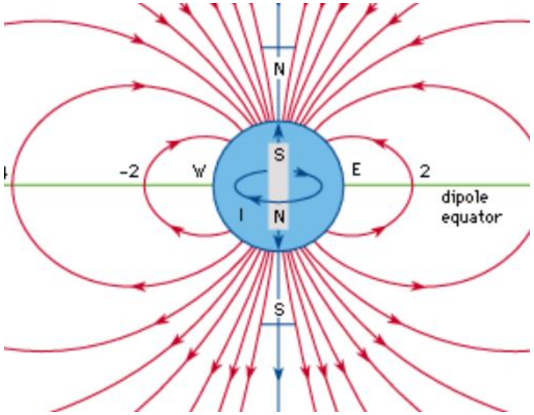
$$Cr_i = s_i, \quad i = 1, 2, 3.$$

The rotation matrix C is given by

$$C = s_1 r_1' + s_2 r_2' + s_3 r_3'.$$

attitude estimation using triad algorithm

- magnetic dip angle α : Ulsan (about 50 deg)
  - latitude : 35.34, longitude: 129.19
  - dip angle is computed using the program in USGS (United States Geological Survery's) (http://geomag.usgs.gov/)
  - world magnetic model (global) 2000.6.11 data
  - $\alpha = 50$ deg $= 0.8727$ rad



$$\tilde{m} = \begin{bmatrix} m_{n,x} \\ 0 \\ m_{n,z} \end{bmatrix} = \begin{bmatrix} \cos\alpha \\ 0 \\ -\sin\alpha \end{bmatrix}$$

- Now use the TRIAD algorithm
  - given y_a and $y_m \Rightarrow$ normalize y_a and y_m

$$\bar{y}_a = \frac{y_a}{\|y_a\|}, \quad \bar{y}_m = \frac{y_m}{\|y_m\|}$$

  - Use the TRIAD with

$$a_1 = \bar{y}_a, \quad b_1 = \frac{\tilde{g}}{\|\tilde{g}\|}, \quad a_2 = \bar{y}_m, \quad b_2 = \tilde{m}.$$

attitude estimation using matlab

- ya = [-1.9470, 0.9589, 9.5567] ' , ym = [0.7540 , -0.2518,   -0.6067] '

- find the Euler angles
  - compute the rotation matrix C using the Triad algorithm
  - compute the Euler angles from C

attitude estimation using matlab

```
ya = [-1.9470  ;  0.9589  ;  9.5567];
ym = [ 0.7540 ; -0.2518 ;  -0.6067];
gtilde = [ 0 ; 0 ; 9.8];
alpha = 50 * D2R;
mtilde = [ cos(alpha) ; 0 ; -sin(alpha) ];
a1 = ya / norm(ya);
a2 = ym / norm(ym);
b1 = gtilde / norm(gtilde);
b2 = mtilde;
r1 = b1;
r2 = cross(b1,b2) / norm(cross(b1,b2));
r3 = cross(r1,r2);
```

```
s1 = a1;

s2 = cross(a1,a2) / norm(cross(a1,a2));

s3 = cross(s1,s2);


C = s1 * r1' + s2 * r2' + s3 * r3';

theta = -asin(C(1,3));

phi = atan2(C(2,3),C(3,3));

psi = atan2(C(1,2),C(1,1));
```

matlab function

```
function [euler] = dcm2euler(R)
% function euler = quaternion2euler(q)

euler = zeros(3,1);
euler(2) = -asin(R(1,3));
euler(1) = atan2(R(2,3),R(3,3));
euler(3) = atan2(R(1,2),R(1,1));
```

```
function [C] = dcmfromyaym(ya,ym)
% function [C] = dcmfromyaym(ya,ym)

ya = ya / norm(ya);
ym = ym / norm(ym);

foo = cross(ya,ym);
foo = foo / norm(foo);
C = [ -cross(ya,foo) , foo, ya ];
```

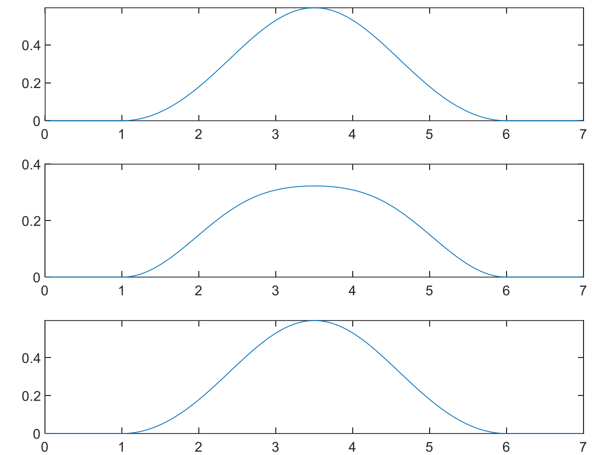
simple (but not good) attitude estimator

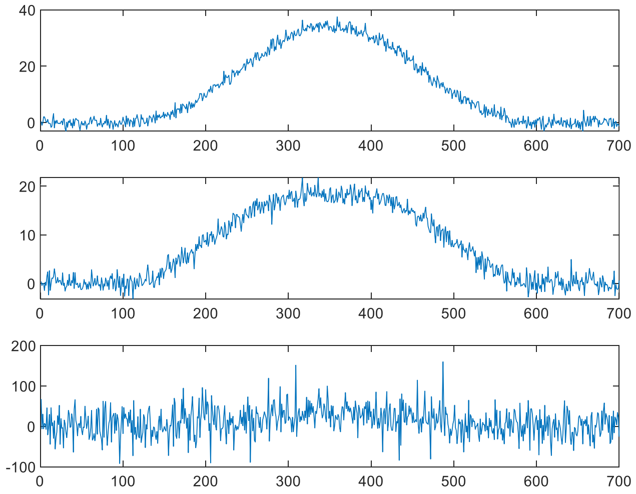
- attitude is estimated from ya and ym
  - “3dsim.mat” → ya, ym, yg (3dattitude2.zip)

```
load('3dsim.mat');

R2D = 180 / pi;
N = size(ya,2);
euler_hat = zeros(3,N);
for i = 1:N
    euler_hat(:,i) =
dcm2euler( dcmfromyaym( ya(:,i), ym(:,i)) );
end

subplot(3,1,1);
plot(R2D*euler_hat(1,:));
subplot(3,1,2);
plot(R2D*euler_hat(2,:));
subplot(3,1,3);
plot(R2D*euler_hat(3,:));
```





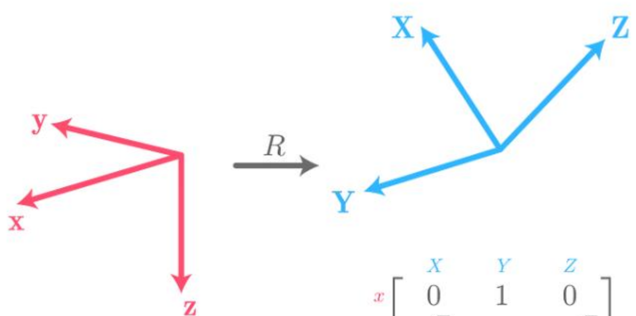
Euler's theorem

- Euler's theorem: any rotation is a rotation about a fixed axis

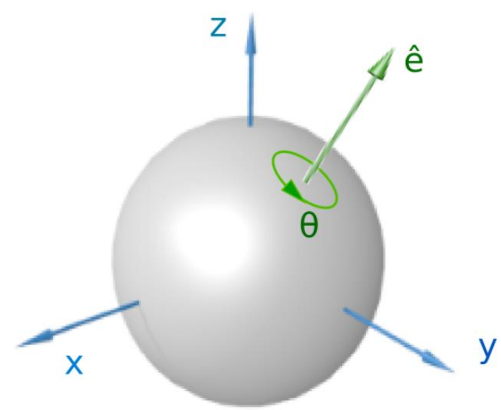
- rotation axis: e and angle θ

$$C(e,\theta) = (\cos\theta)I - \sin\theta[e\times] + (1-\cos\theta)ee^T$$

$$[a\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$



$$R = \begin{array}{c} \\ x \\ y \\ z \end{array}\!\!\begin{array}{c} \begin{array}{ccc} X & Y & Z \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{bmatrix} \end{array}$$



quaternion

- four parameter rotation representation

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \in R^4 \qquad \|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$$

- quaternion and rotation axis/angle

$$q = \begin{bmatrix} \cos\frac{\theta}{2} \\ e\sin\frac{\theta}{2} \end{bmatrix}$$

- quaternion to rotation matrix

$$C(q) = \begin{bmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{bmatrix}$$

rotation matrix to quaternion

- find q_j^2 from the following

$$
\begin{aligned}
q_0^2 &= \tfrac{1}{4}(1 + \mathrm{Tr}\, C) \\
q_1^2 &= \tfrac{1}{4}(C_{11} - C_{22} - C_{33} + 1) \\
q_2^2 &= \tfrac{1}{4}(C_{22} - C_{11} - C_{33} + 1) \\
q_3^2 &= \tfrac{1}{4}(C_{33} - C_{11} - C_{22} + 1)
\end{aligned}
$$

- Choose the largest q_j
- We can compute the quaternion from the following

$$
\begin{aligned}
C_{12} + C_{21} &= 4q_1q_2 \\
C_{13} + C_{31} &= 4q_1q_3 \\
C_{23} + C_{32} &= 4q_2q_3 \\
C_{12} - C_{21} &= 4q_0q_3 \\
C_{13} - C_{31} &= -4q_0q_2 \\
C_{23} - C_{32} &= 4q_0q_1
\end{aligned}
$$

$$
C(q) = \begin{bmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{bmatrix}
$$

quaternion to Euler angles

$$
\begin{aligned}
\psi &= \mathrm{atan2}(2q_1q_2 + 2q_0q_3, 2q_0^2 + 2q_1^2 - 1) \\
\theta &= -\sin^{-1}(2q_1q_3 - 2q_0q_2) \\
\phi &= \mathrm{atan2}(2q_2q_3 + 2q_0q_1, 2q_0^2 + 2q_3^2 - 1)
\end{aligned}
$$

quaternion dynamics

- derivative of quaternion

$$\dot{q} = \frac{1}{2}\Omega(\omega_b)q \qquad \Omega(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}$$

- simple discretization (after the integration, you have to normalized q)

$$q_{k+1} = (I + \frac{1}{2}\Omega T)q_k$$

- 2nd order approximation

$$q_{k+1} = (I + \frac{3}{4}\Omega_k T - \frac{1}{4}\Omega_{k-1} T - \frac{1}{8}\|\omega_k\|_2^2 T^2)q_k$$

attitude estimation (1^{st} order approximation)

- attitude computation by integrating the gyroscope outputs
  - true quaternion : q, T = 0.01

```
qhat = zeros(4,N);
eulerhat = zeros(3,N);
qhat(:,1) = q(:,1);
eulerhat(:,1) = quaternion2euler(q(:,1));

for i = 2:N
        wx = yg(1,i-1);
        wy = yg(1,i-1);
        wz = yg(1,i-1);
```

```
  Omega = [ 0 , -wx, -wy, -wz ; wx , 0 ,
wz , -wy ; wy , -wz, 0, wx ; wz , wy , -
wx , 0 ];

  qhat(:,i) = (eye(4) + 0.5 * Omega * T)
* qhat(:,i-1);

  qhat(:,i) = qhat(:,i) / norm(qhat(:,i));
  eulerhat(:,i) =
quaternion2euler(qhat(:,i));

    end
```

- Write the matlab code for the 2^{nd} order approximation and compare the results between 1^{st} order and 2^{nd} order approximation

Rotation comparison

D. Q. Huynh, Metrics for 3D Rotations: Comparison and Analysis (2009)

- How similar q_1 and q_2?

- The following is not a good measure $\quad \|q_1 - q_2\|$

$$q = \begin{bmatrix} \cos\frac{2\pi+\theta}{2} \\ e\sin\frac{2\pi+\theta}{2} \end{bmatrix} = \begin{bmatrix} -\cos\frac{\theta}{2} \\ -e\sin\frac{\theta}{2} \end{bmatrix} = -q$$

$$\begin{aligned} \cos(\pi+\tfrac{\theta}{2}) &= \cos\pi\cos\tfrac{\theta}{2} - \sin\pi\sin\tfrac{\theta}{2} = -\cos\tfrac{\theta}{2} \\ \sin(\pi+\tfrac{\theta}{2}) &= \sin\pi\cos\tfrac{\theta}{2} + \cos\pi\sin\tfrac{\theta}{2} = -\sin\tfrac{\theta}{2} \end{aligned}$$

- A measure using rotation matrices

$$\|1 - C(q_1)C(q_2)'\|_F$$

- Frobenius norm

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_i^m \sum_j^n |a_{ij}|^2} = \sqrt{\operatorname{trace}(A^* A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)},$$