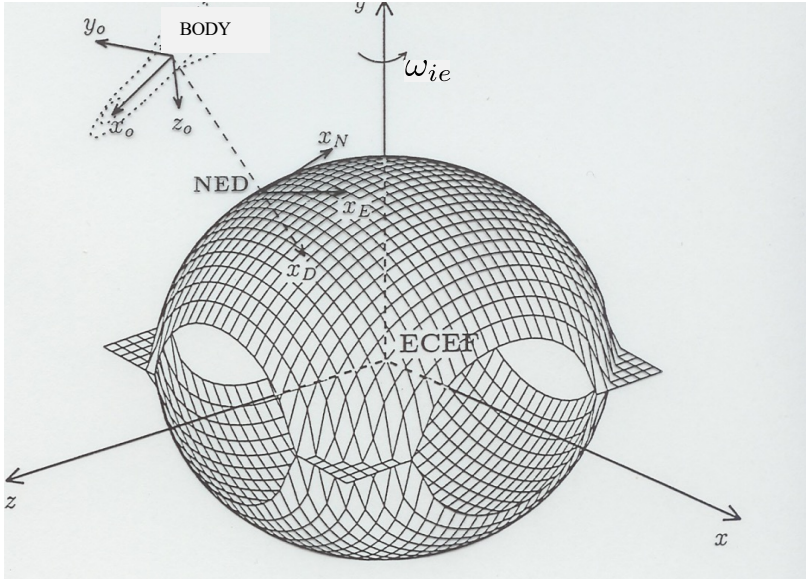
Chapter 2 - Kinematics

"The study of dynamics can be divided into two parts: kinematics, which treats only geometrical aspects of motion, and kinetics, which is the analysis of the forces causing the motion"



Overall Goal of Chapters 2 to 10

Represent the 6-DOF equations of motion in a compact matrix-vector form according to:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}$$

Chapter Goals

- Understand the geographic reference frame NED, the Earth-centered reference frame ECEF and the body-fixed reference frame BODY.
- Understand what FLOW axes are and why we use these axes for marine craft and aircraft
- Be able to write down the differential equations relating BODY velocities to NED positions, both for Euler angles and unit quaternions.
- Be able to:
  - Transform ECEF (x^{e}, y^{e}, z^{e}) positions to (longitude, latitude, height) and vice versa
  - Transform (longitude, latitude, height) to flat-Earth positions (x^{n}, y^{n}, z^{n}) and vice versa
- Define, visualize, and explain the use of:
  - Angle of attack α
  - Sideslip angle β
  - Vertical crab angle α_c
  - Horizontal crab angle β_c
  - Heading angle ψ
  - Course angle χ

2.1 Reference Frames

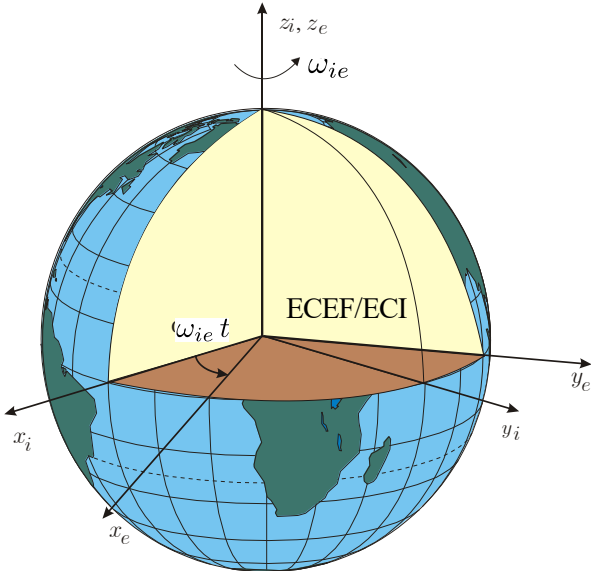
Earth-Centered Reference Frames

$\{i\}$: The Earth-centered inertial reference frame (ECI)

$\{i\} = (x_i, y_i, z_i)$ is an inertial frame for terrestrial navigation, that is a nonaccelerating reference frame in which Newton's laws of motion apply.

$\{e\}$: The Earth-centered Earth-fixed reference frame (ECEF)

$\{e\} = (x_e, y_e, z_e)$ has its origin o_e fixed to the center of the Earth but the axes rotate relative to the inertial frame ECI, which is fixed in space.



x_e - axis in the equatorial plane pointing towards the zero/prime meridian; same longitude as the Greenwich observatory

y_e - axis in the equatorial plane completing the right-hand frame

z_e - axis pointing along the Earth's rotational axis

The Earth rotation is $\omega_{ie} = 7.2921 \times 10^{-5}$ rad/s and the Earth's rotational vector expressed in $\{e\}$ is

$$\boldsymbol{\omega}_{ie}^{e} = [0, 0, \omega_{ie}]^{\top}$$

2.1 Reference Frames (cont.)

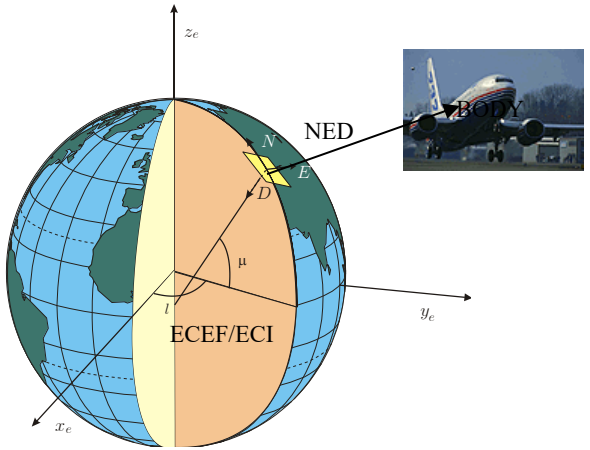
Geographic Reference Frames (Tangent Planes)

Geographical reference frames are usually chosen as tangent planes on the surface of the Earth.

- **Terrestrial navigation:** The tangent plane on the surface of the Earth moves with the craft and its location is specified by time-varying longitude-latitude values (l, μ). The tangent frame is usually rotated such that its axes points in the NED directions.

- **Local navigation:** The tangent plane is fixed at constant values (l_0 , μ_0) and the position is computed with respect to a local coordinate origin. The axes of the tangent plane are usually chosen to coincide with the NED axes.

  "Flat-Earth navigation": position is accurate to a smaller geographical area (10 km × 10 km).



{n}: The North-East-Down reference frame (NED)

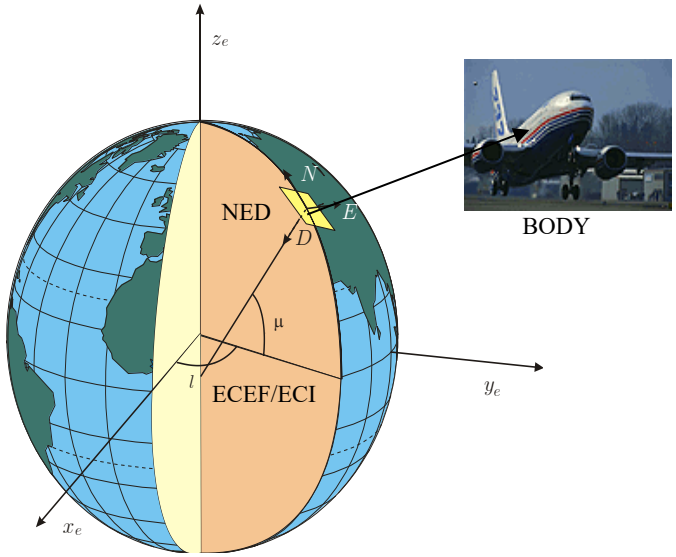
{n} = (x_n, y_n, z_n) has its origin o_n defined relative to the Earth's reference ellipsoid (WGS 84), usually as the tangent plane to the ellipsoid with

- x_n - axis points towards true North
- y_n - axis points towards East
- z_n - axis points downwards normal to the Earth's surface

2.1 Reference Frames (cont.)

Body-Fixed Reference Frames

The body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ with origin o_b is a moving coordinate frame that is fixed to the craft.



$\{b\}$: The body-fixed reference frame (BODY)

For marine craft, the body axes are chosen as:

- x_b - longitudinal axis (directed from aft to fore)
- y_b - transversal axis (directed to starboard)
- z_b -normal axis (directed from top to bottom)

$\{f\}$: The body-fixed flow axes reference frame (FLOW)

Flow axes are used to align the x-axis with the craft's velocity vector such that lift is perpendicular to the relative flow and drag is parallel. The transformation from FLOW to BODY axes is defined by two principal rotations where the rotation angles are the angle of attack α and the sideslip angle β. The main purpose of the flow axes is to simply the computations of lift and drag forces.

$$F_{\text{drag}} = \frac{1}{2}\rho V_r^2 S C_D(\alpha)$$

$$F_{\text{lift}} = \frac{1}{2}\rho V_r^2 S C_L(\alpha)$$

$$\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} -F_{\text{drag}} \\ -F_{\text{lift}} \end{bmatrix}$$

2.1 Body-Fixed Reference Points

The most important reference point

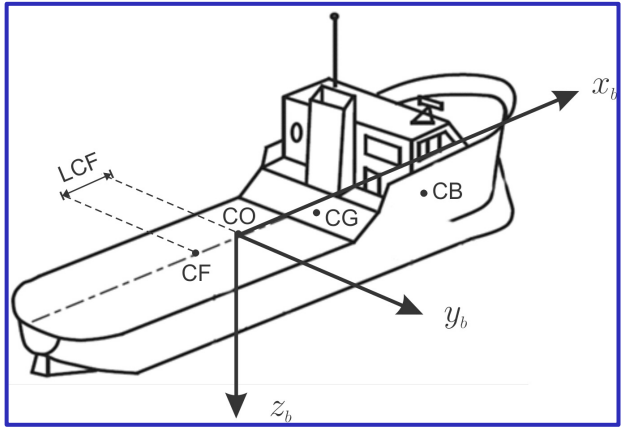
CO - Coordinate origin o_b of the body-fixed frame $\{b\}$. CO also represents the coordinate origin of the guidance, navigation and motion control systems.

The following time-varying points are expressed with respect to the CO

CG - Center of gravity located at $\boldsymbol{r}_{bg}^{b} = [x_g, y_g, z_g]^\top$ relative to the CO.

CB - Center of buoyancy located at $\boldsymbol{r}_{bb}^{b} = [x_b, y_b, z_b]^\top$ relative to the CO.

CF - Center of flotation located at $\boldsymbol{r}_{bf}^{b} = [x_f, y_f, z_f]^\top$ relative to the CO.



The **CF** is located a distance LCF from the **CO** in the x-direction

The center of flotation is the centroid of the water plane area A_{wp} in calm water. For small angles (linear theory), the vessel will roll and pitch about this point.

2.1 Generalized Coordinates

For a marine craft not subject to any motion constraints

Number of independent (generalized) coordinates = DOFs

The term generalized coordinates refers to the parameters that describe the configuration of the craft relative to some reference configuration.

For marine craft, the generalized position and velocity vectors are

$$\boldsymbol{\eta} = [x^n, y^n, z^n, \phi, \theta, \psi]^\top$$

$$\dot{\boldsymbol{\eta}} = [\dot{x}^n, \dot{y}^n, \dot{z}^n, \dot{\phi}, \dot{\theta}, \dot{\psi}]^\top$$

These quantities are all formulated in NED. It is advantageous to express the velocities of the craft in the BODY frame. In other words

$$\boldsymbol{\nu} = [u, v, w, p, q, r]^\top$$

The relationship between the velocity vector in NED and BODY will be derived on the subsequent pages.

2.1 Vector Notation (Fossen 2021)

Coordinate-free vector (arrow notation)

$$\vec{u} = u_1^n \vec{n}_1 + u_2^n \vec{n}_2 + u_3^n \vec{n}_3$$

$\vec{n}_i$ $(i = 1,2,3)$ are the unit vectors that define $\{n\}$

Coordinate form of $\vec{u}$ in $\{n\}$ (bold notation)

$$\mathbf{u}^n = [u_1^n, u_2^n, u_3^n]^\top$$ " Vector expressed in $\{n\}$"

$\boldsymbol{p}_{eb}^e$	position of the CO relative to o_e expressed in $\{e\}$
$\boldsymbol{p}_{nb}^n$	position of the CO relative to o_n expressed in $\{n\}$
$\boldsymbol{v}_{nb}^e$	linear velocity of the CO relative to o_n expressed in $\{e\}$
$\boldsymbol{v}_{nb}^n$	linear velocity of the CO relative to o_n expressed in $\{n\}$
$\boldsymbol{\omega}_{nb}^b$	angular velocity of $\{b\}$ with respect to $\{n\}$ expressed in $\{b\}$
$\boldsymbol{f}_b^n$	force with line of action through the point CO expressed in $\{n\}$
$\boldsymbol{m}_b^n$	moment about the point CO expressed in $\{n\}$
$\boldsymbol{\Theta}_{nb}$	Euler angles from $\{b\}$ to $\{n\}$
$\boldsymbol{q}_b^n$	Unit quaternion from $\{b\}$ to $\{n\}$

ECEF position $\boldsymbol{p}_{eb}^e = \begin{bmatrix} x^e \\ y^e \\ z^e \end{bmatrix} \in \mathbb{R}^3$

NED position $\boldsymbol{p}_{nb}^n = \begin{bmatrix} x^n \\ y^n \\ z^n \end{bmatrix} \in \mathbb{R}^3$

Body-fixed linear velocity $\boldsymbol{v}_{nb}^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$

Body-fixed force $\boldsymbol{f}_b^b = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3$

Longitude and latitude $\boldsymbol{\Theta}_{en} = \begin{bmatrix} l \\ \mu \end{bmatrix} \in \mathbb{T}^2$

Attitude (Euler angles) $\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{T}^3$

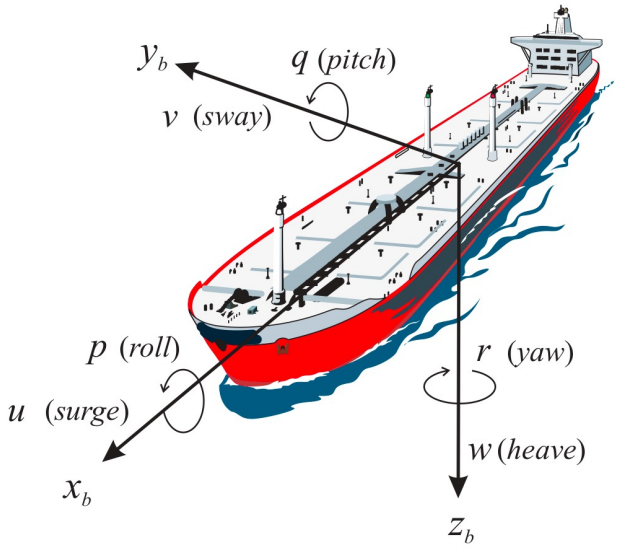
Body-fixed angular velocity $\boldsymbol{\omega}_{nb}^b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$

Body-fixed moment $\boldsymbol{m}_b^b = \begin{bmatrix} K \\ M \\ N \end{bmatrix} \in \mathbb{R}^3$

2.1 Six Degrees-of-Freedom (6-DOF) Motions

6-DOF refers to the freedom of movement of a rigid body in 3-D space. It describes how the object can translate and rotate relative to its own body-fixed axes. It is not used for NED coordinates.

	BODY			NED
DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	Motions in the x_b-direction (surge)	X	u	x^n
2	Motions in the y_b-direction (sway)	Y	v	y^n
3	Motions in the z_b-direction (heave)	Z	w	z^n
4	Rotation about the x_b-axis (roll)	K	p	ϕ
5	Rotation about the y_b-axis (pitch)	M	q	θ
6	Rotation about the z_b-axis (yaw)	N	r	ψ



The notation is adopted from:

SNAME (1950). Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No. 1-5, April 1950, pp. 1-15.*

2.1 Summary: 6-DOF Vectors

ECEF position	$\boldsymbol{p}_{eb}^{e} = \begin{bmatrix} x^e \\ y^e \\ z^e \end{bmatrix} \in \mathbb{R}^3$	Longitude and latitude	$\boldsymbol{\Theta}_{en} = \begin{bmatrix} l \\ \mu \end{bmatrix} \in \mathbb{T}^2$
NED position	$\boldsymbol{p}_{nb}^{n} = \begin{bmatrix} x^n \\ y^n \\ z^n \end{bmatrix} \in \mathbb{R}^3$	Attitude (Euler angles)	$\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{T}^3$
Body-fixed linear velocity	$\boldsymbol{v}_{nb}^{b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$	Body-fixed angular velocity	$\boldsymbol{\omega}_{nb}^{b} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$
Body-fixed force	$\boldsymbol{f}_{b}^{b} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3$	Body-fixed moment	$\boldsymbol{m}_{b}^{b} = \begin{bmatrix} K \\ M \\ N \end{bmatrix} \in \mathbb{R}^3$

6-DOF generalized position, velocity and force vectors expressed in $\{b\}$

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}_{nb}^{n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}_{nb}^{b} \\ \boldsymbol{\omega}_{nb}^{b} \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}_{b}^{b} \\ \boldsymbol{m}_{b}^{b} \end{bmatrix}$$

2.2 Transformations between BODY and NED

Orthogonal matrices of order 3:

$$O(3) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{3\times 3}, \quad \mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}\}$$

Special orthogonal group of order 3:

$$SO(3) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{3\times 3}, \quad \mathbf{R} \text{ is orthogonal and } \det\mathbf{R} = 1\}$$

Rotation matrix:

$$\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}, \qquad \det\mathbf{R} = 1$$

Since **R** is orthogonal, $\mathbf{R}^{-1} = \mathbf{R}^\top$

Example:

$$\boldsymbol{v}^{\text{to}} = \boldsymbol{R}^{\text{to}}_{\text{from}}\boldsymbol{v}^{\text{from}}$$

2.2 Transformations between BODY and NED (cont.)

Cross-product operator as matrix-vector multiplication: $\boldsymbol{\lambda} \times \mathbf{a} := \mathbf{S}(\boldsymbol{\lambda})\mathbf{a}$

$$\mathbf{S}(\boldsymbol{\lambda}) = -\mathbf{S}^{\top}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

where $\mathbf{S} = -\mathbf{S}^{\top}$ is a skew-symmetric matrix

The inverse operator is denoted: $\boldsymbol{\lambda} = \text{vex}\,(\boldsymbol{S}(\boldsymbol{\lambda}))$

Matlab:
The cross-product operator is included in the MSS toolbox as `Smtrx.m`. Hence, the cross-product $\boldsymbol{b} = \boldsymbol{S}(\boldsymbol{\lambda})\boldsymbol{a}$ can be computed as

```
S = Smtrx(lambda);
b = S * a;
```

The inverse operation is

```
lambda = vex(S);
```

2.2 Transformations between BODY and NED (cont.)

Euler's theorem on rotation

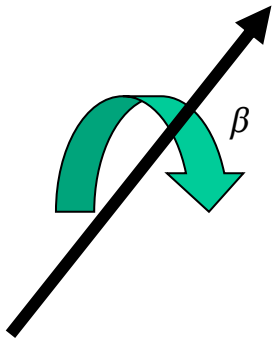
$$\mathbf{R}_{\lambda,\beta} = \mathbf{I}_{3\times 3} + \sin\beta\ \mathbf{S}(\boldsymbol{\lambda}) + (1-\cos\beta)\ \mathbf{S}^2(\boldsymbol{\lambda})$$

$$\mathbf{v}^n_{b/n} = \mathbf{R}^n_b \mathbf{v}^b_{b/n}, \quad \mathbf{R}^n_b := \mathbf{R}_{\lambda,\beta}$$

where

$$\begin{aligned}
R_{11} &= (1-\cos\beta)\ \lambda_1^2 + \cos\beta \\
R_{22} &= (1-\cos\beta)\ \lambda_2^2 + \cos\beta \\
R_{33} &= (1-\cos\beta)\ \lambda_3^2 + \cos\beta \\
R_{12} &= (1-\cos\beta)\ \lambda_1\lambda_2 - \lambda_3 \sin\beta \\
R_{21} &= (1-\cos\beta)\ \lambda_2\lambda_1 + \lambda_3 \sin\beta \\
R_{23} &= (1-\cos\beta)\ \lambda_2\lambda_3 - \lambda_1 \sin\beta \\
R_{32} &= (1-\cos\beta)\ \lambda_3\lambda_2 + \lambda_1 \sin\beta \\
R_{31} &= (1-\cos\beta)\ \lambda_3\lambda_1 - \lambda_2 \sin\beta \\
R_{13} &= (1-\cos\beta)\ \lambda_1\lambda_3 + \lambda_2 \sin\beta
\end{aligned}$$

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^\top, \quad |\boldsymbol{\lambda}| = 1$$



2.2 Euler Angle Transformation

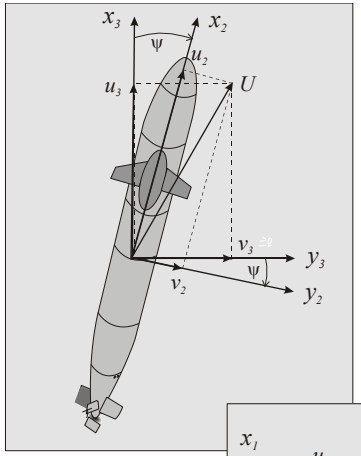
Three principal rotations

$$\boldsymbol{\lambda} = [0, 0, 1]^\top \qquad \beta = \psi$$
$$\boldsymbol{\lambda} = [0, 1, 0]^\top \qquad \beta = \theta$$
$$\boldsymbol{\lambda} = [1, 0, 0]^\top \qquad \beta = \phi$$

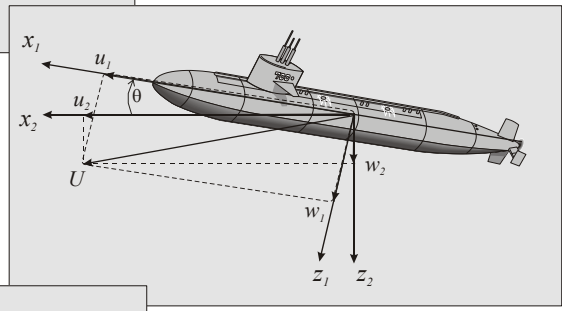
$$\mathbf{R}_{z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

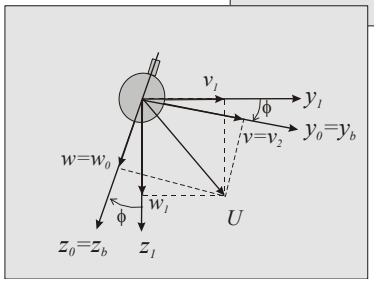
$$\mathbf{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$$



(1) Rotation over yaw angle ψ about z_3. Note that $w_3=w_2$.



(2) Rotation over pitch angle θ about y_2. Note that $v_2=v_1$.



(3) Rotation over roll angle ϕ about x_1. Note that $u_1=u_2$

.

2.2 Euler Angle Transformation (cont.)

Linear velocity transformation (zyx convention)

$$\dot{\boldsymbol{p}}_{nb}^{n} = \boldsymbol{R}(\boldsymbol{\Theta}_{nb})\boldsymbol{v}_{nb}^{b}$$

$$\boldsymbol{R}_{b}^{n} := \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi}$$

$$(\boldsymbol{R}_{b}^{n})^{-1} = \boldsymbol{R}_{n}^{b} = \boldsymbol{R}_{x,\phi}^{\top}\boldsymbol{R}_{y,\theta}^{\top}\boldsymbol{R}_{z,\psi}^{\top}$$

Euler angle rotation matrix

$$\boldsymbol{R}_{b}^{n} := \boldsymbol{R}(\boldsymbol{\Theta}_{nb})$$

$$\boldsymbol{R}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix}$$

Matlab:
The rotation matrix $\boldsymbol{R}(\boldsymbol{\Theta}_{nb})$ is implemented in the MSS toolbox as

```
R = Rzyx(phi,theta,psi)
```

Small-angle approximation

$$\boldsymbol{R}(\delta\boldsymbol{\Theta}_{nb}) \approx \boldsymbol{I}_3 + \boldsymbol{S}(\delta\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & -\delta\psi & \delta\theta \\ \delta\psi & 1 & -\delta\phi \\ -\delta\theta & \delta\phi & 1 \end{bmatrix}$$

2.2 Euler Angle Transformation (cont.)

NED positions (continuous time and discrete time)

$$\dot{\boldsymbol{p}}_{nb}^{n} = \boldsymbol{R}(\boldsymbol{\Theta}_{nb})\boldsymbol{v}_{nb}^{b}$$

Euler's method with sampling time h

$$\boldsymbol{p}_{nb}^{n}[k+1] = \boldsymbol{p}_{nb}^{n}[k] + h\boldsymbol{R}\left(\boldsymbol{\Theta}_{nb}[k]\right)\boldsymbol{v}_{nb}^{b}[k]$$

Component form

$$\begin{aligned}
\dot{x}^n &= u\cos(\psi)\cos(\theta) + v[\cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi)] \\
&\quad + w\left[\sin(\psi)\sin(\phi) + \cos(\psi)\cos(\phi)\sin(\theta)\right] \\
\dot{y}^n &= u\sin(\psi)\cos(\theta) + v[\cos(\psi)\cos(\phi) + \sin(\phi)\sin(\theta)\sin(\psi)] \\
&\quad + w[\sin(\theta)\sin(\psi)\cos(\phi) - \cos(\psi)\sin(\phi)] \\
\dot{z}^n &= -u\sin(\theta) + v\cos(\theta)\sin(\phi) + w\cos(\theta)\cos(\phi)
\end{aligned}$$

2.2 Euler Angle Transformation (cont.)

Angular velocity transformation (zyx convention)

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}^b_{nb} \qquad \boldsymbol{\omega}^b_{nb} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}^\top_{x,\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}^\top_{x,\phi}\boldsymbol{R}^\top_{y,\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} := \boldsymbol{T}^{-1}(\boldsymbol{\Theta}_{nb})\dot{\boldsymbol{\Theta}}_{nb}$$

where

$$\boldsymbol{T}^{-1}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & 0 & -\mathrm{s}\theta \\ 0 & \mathrm{c}\phi & \mathrm{c}\theta\mathrm{s}\phi \\ 0 & -\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \quad \Longrightarrow \quad \boldsymbol{T}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix}$$

Small angle approximation

$$\mathbf{T}_\Theta(\delta\boldsymbol{\Theta}_{nb}) \approx \begin{bmatrix} 1 & 0 & \delta\theta \\ 0 & 1 & -\delta\phi \\ 0 & \delta\phi & 1 \end{bmatrix}$$

Singular point at $\theta = \pm 90^o$

$\mathbf{T}^{-1}_\Theta(\boldsymbol{\Theta}_{nb}) \neq \mathbf{T}^\top_\Theta(\boldsymbol{\Theta}_{nb})$

2.2 Euler Angle Transformation (cont.)

Euler angle attitude representations

ODE for Euler angles

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{nb}^{b}$$

Component form

$$\dot{\phi} = p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta)$$
$$\dot{\theta} = q\cos(\phi) - r\sin(\phi)$$
$$\dot{\psi} = q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)}, \quad \theta \neq \pm 90^{\circ}$$

ODE for rotation matrix

$$\dot{\boldsymbol{R}}_{b}^{n} = \boldsymbol{R}_{b}^{n}\boldsymbol{S}(\boldsymbol{\omega}_{nb}^{b})$$

where

$$\boldsymbol{S}(\boldsymbol{\omega}_{nb}^{b}) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

Must be combined with an algorithm for computation of the Euler angles $\boldsymbol{\Theta}_{nb} = [\phi, \theta, \psi]^{\top}$ from the rotation matrix $\boldsymbol{R}_{b}^{n}$

Matlab:
The transformation matrix $\boldsymbol{T}(\boldsymbol{\Theta}_{nb})$ is implemented in the MSS toolbox as

```
T = Tzyx(phi,theta)
```

2.2 Euler Angle Transformation (cont.)

Summary: 6-DOF kinematic equations

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu}$$

$$\Updownarrow$$

$$\begin{bmatrix} \dot{\boldsymbol{p}}^{n}_{nb} \\ \dot{\boldsymbol{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{b}_{nb} \\ \boldsymbol{\omega}^{b}_{nb} \end{bmatrix}$$

3-parameter representation

$\boldsymbol{\Theta}_{nb} = [\phi, \theta, \psi]^{\top}$

with singularity at $\theta = \pm 90^{o}$

Component form

$$\begin{aligned}
\dot{x}^n &= u\cos(\psi)\cos(\theta) + v[\cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi)] \\
&\quad + w\left[\sin(\psi)\sin(\phi) + \cos(\psi)\cos(\phi)\sin(\theta)\right] \\
\dot{y}^n &= u\sin(\psi)\cos(\theta) + v[\cos(\psi)\cos(\phi) + \sin(\phi)\sin(\theta)\sin(\psi)] \\
&\quad + w[\sin(\theta)\sin(\psi)\cos(\phi) - \cos(\psi)\sin(\phi)] \\
\dot{z}^n &= -u\sin(\theta) + v\cos(\theta)\sin(\phi) + w\cos(\theta)\cos(\phi) \\
\dot{\phi} &= p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\
\dot{\theta} &= q\cos(\phi) - r\sin(\phi) \\
\dot{\psi} &= q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)}, \quad \theta \neq \pm 90^{\circ}
\end{aligned}$$

```
phi = deg2rad(10)
theta = deg2rad(20)
psi = deg2rad(30)

R = Rzyx(phi, theta, psi)

T =  Tzyx(phi, theta)
```

```
phi =
    0.1745
theta =
    0.3491
psi =
    0.5236
R =
    0.8138   -0.4410    0.3785
    0.4698    0.8826    0.0180
   -0.3420    0.1632    0.9254
T =
    1.0000    0.0632    0.3584
         0    0.9848   -0.1736
         0    0.1848    1.0480
```

2.2 Unit Quaternions

4-parameter representation

$$Q = \{\mathbf{q} | \mathbf{q}^\top \mathbf{q} = 1, \mathbf{q} = [\eta, \boldsymbol{\varepsilon}^\top]^\top, \quad \boldsymbol{\varepsilon} \in \mathbb{R}^3 \text{ and } \eta \in \mathbb{R}\} \qquad \boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \varepsilon_3]^\top$$

- avoids the representation singularity of the 3-parameter Euler angle representation known as gimbal lock
- involves fewer trigonometric calculations , enhancing stability
- Quaternion interpolation results in smoother transitions compared to Euler angles
- Quaternion multiplication is more efficient and stable than combining Euler angles

Unit quaternion (Euler parameter) rotation matrix (Chou 1992)

$$\mathbf{R}_{\beta,\boldsymbol{\lambda}} = \mathbf{I}_{3\times 3} + \sin\beta\ \mathbf{S}(\boldsymbol{\lambda}) + (1 - \cos\beta)\ \mathbf{S}^2(\boldsymbol{\lambda})$$

$$\eta = \cos\frac{\beta}{2}$$

$$\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \varepsilon_3]^\top = \boldsymbol{\lambda} \sin\frac{\beta}{2}$$

$$\mathbf{q} = \begin{bmatrix} \eta \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\beta}{2} \\ \boldsymbol{\lambda} \sin\frac{\beta}{2} \end{bmatrix} \in Q$$

$$\boldsymbol{R}(q_b^n) := \boldsymbol{I}_3 + 2\eta \boldsymbol{S}(\boldsymbol{\varepsilon}) + 2\boldsymbol{S}^2(\boldsymbol{\varepsilon})$$

Chou, J. C. K. (1992). Quaternion Kinematic and Dynamic Differential Equations. IEEE Transactions on Robotics and Automation 8(1), 53–64.

2.2 Unit Quaternions (cont.)

Linear velocity transformation

$$\dot{\boldsymbol{p}}_{nb}^{n} = \boldsymbol{R}(\boldsymbol{q}_b^n)\boldsymbol{v}_{nb}^{b} \qquad \boldsymbol{R}^{-1}(\boldsymbol{q}_b^n) = \boldsymbol{R}^{\top}(\boldsymbol{q}_b^n)$$

$$\boldsymbol{R}(\boldsymbol{q}_b^n) = \begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\eta) & 1-2(\varepsilon_1^2+\varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\eta) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\eta) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\eta) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix}$$

Component form (NED positions)

$$\begin{aligned} \dot{x}^n &= u(1-2\varepsilon_2^2-2\varepsilon_3^2)+2v(\varepsilon_1\varepsilon_2-\varepsilon_3\eta)+2w(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ \dot{y}^n &= 2u(\varepsilon_1\varepsilon_2+\varepsilon_3\eta)+v(1-2\varepsilon_1^2-2\varepsilon_3^2)+2w(\varepsilon_2\varepsilon_3-\varepsilon_1\eta) \\ \dot{z}^n &= 2u(\varepsilon_1\varepsilon_3-\varepsilon_2\eta)+2v(\varepsilon_2\varepsilon_3+\varepsilon_1\eta)+w(1-2\varepsilon_1^2-2\varepsilon_2^2) \end{aligned}$$

Matlab:
The quaternion rotation matrix is easily computed by using the MSS toolbox commands

```
q = [eta,eps1,eps2,eps3];
R = Rquat(q);
```

Must be integrated under the constraint

$$\eta^2+\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2 = 1$$

2.2 Unit Quaternions (cont.)

Angular velocity transformation

$$\dot{\boldsymbol{q}}_b^n = \frac{1}{2}\begin{bmatrix} -\boldsymbol{\varepsilon}^\top \\ \eta \boldsymbol{I}_3 + \boldsymbol{S}(\boldsymbol{\varepsilon}) \end{bmatrix} \boldsymbol{\omega}_{nb}^b := \boldsymbol{T}(\boldsymbol{q}_b^n)\boldsymbol{\omega}_{nb}^b$$

$$\boldsymbol{T}(\boldsymbol{q}_b^n) = \frac{1}{2}\begin{bmatrix} -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\ \eta & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \eta & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \eta \end{bmatrix}, \qquad \boldsymbol{T}^\top(\boldsymbol{q}_b^n)\boldsymbol{T}(\boldsymbol{q}_b^n) = \frac{1}{4}\boldsymbol{I}_3$$

Nonsingular to the price of one more parameter but unit quaternions are numerical efficient

Alternative representation using the quaternion product

$$\dot{\boldsymbol{q}}_b^n = \frac{1}{2}\boldsymbol{q}_b^n \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{nb}^b \end{bmatrix} \qquad \boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \begin{bmatrix} \eta_1\eta_2 - \boldsymbol{\varepsilon}_1^\top\boldsymbol{\varepsilon}_2 \\ \eta_1\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1 + \boldsymbol{S}(\boldsymbol{\varepsilon}_1)\boldsymbol{\varepsilon}_2 \end{bmatrix}$$

Matlab:
The transformation matrix $\boldsymbol{T}(\boldsymbol{q}_b^n)$ is implemented in the MSS toolbox as

```
T = Tquat(q)
```

```
quat = euler2q(phi, theta, psi)

Rq = Rquat(quat)
Tq = Tquat(quat)
```

```
quat =
    0.9515
    0.0381
    0.1893
    0.2393
Rq =
    0.8138   -0.4410    0.3785
    0.4698    0.8826    0.0180
   -0.3420    0.1632    0.9254
Tq =
   -0.0191   -0.0947   -0.1196
    0.4758   -0.1196    0.0947
    0.1196    0.4758   -0.0191
   -0.0947    0.0191    0.4758
```

2.2 Unit Quaternions (cont.)

Summary: 6-DOF kinematic equations (7 ODEs)

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_q(\boldsymbol{\eta})\boldsymbol{\nu}$$
$$\Updownarrow$$
$$\begin{bmatrix} \dot{\boldsymbol{p}}_{nb}^n \\ \dot{\boldsymbol{q}}_b^n \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{q}_b^n) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{4\times 3} & \boldsymbol{T}(\boldsymbol{q}_b^n) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{nb}^b \\ \boldsymbol{\omega}_{nb}^b \end{bmatrix}$$

4-parameter representation

$$\mathbf{q} = [\eta, \varepsilon_1, \varepsilon_2, \varepsilon_3]^\top$$

Nonsingular but one more ODE is needed.

$$\boldsymbol{\eta} = [x^n, y^n, z^n, \eta, \varepsilon_1, \varepsilon_2, \varepsilon_3]^\top$$

Component form

$$\begin{aligned}
\dot{x}^n &= u(1 - 2\varepsilon_2^2 - 2\varepsilon_3^2) + 2v(\varepsilon_1\varepsilon_2 - \varepsilon_3\eta) + 2w(\varepsilon_1\varepsilon_3 + \varepsilon_2\eta) \\
\dot{y}^n &= 2u(\varepsilon_1\varepsilon_2 + \varepsilon_3\eta) + v(1 - 2\varepsilon_1^2 - 2\varepsilon_3^2) + 2w(\varepsilon_2\varepsilon_3 - \varepsilon_1\eta) \\
\dot{z}^n &= 2u(\varepsilon_1\varepsilon_3 - \varepsilon_2\eta) + 2v(\varepsilon_2\varepsilon_3 + \varepsilon_1\eta) + w(1 - 2\varepsilon_1^2 - 2\varepsilon_2^2)
\end{aligned}$$

$$\begin{aligned}
\dot{\eta} &= -\frac{1}{2}(\varepsilon_1 p + \varepsilon_2 q + \varepsilon_3 r) \\
\dot{\varepsilon}_1 &= \frac{1}{2}(\eta p - \varepsilon_3 q + \varepsilon_2 r) \\
\dot{\varepsilon}_2 &= \frac{1}{2}(\varepsilon_3 p + \eta q - \varepsilon_1 r) \\
\dot{\varepsilon}_3 &= \frac{1}{2}(-\varepsilon_2 p + \varepsilon_1 q + \eta r)
\end{aligned}$$

2.2 Unit Quaternions (cont.)

Discrete-time algorithm for unit quaternion normalization

$$(\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \eta^2 = 1$$

Algorithm 2.1 (Discrete-Time Unit Quaternion Normalization)

1. $k = 0$. *Compute the initial value of* $\boldsymbol{q}_b^n[0]$.
2. *Propagate the unit quaternion using Euler's method*

$$\boldsymbol{q}_b^n[k+1] = \boldsymbol{q}_b^n[k] + h\boldsymbol{T}(\boldsymbol{q}_b^n[k])\boldsymbol{\omega}_{nb}^b[k]$$

   where h *is the sampling time and* k *is the sampling index.*
3. *Normalization*

$$\boldsymbol{q}_b^n[k+1] = \frac{\boldsymbol{q}_b^n[k+1]}{\|\boldsymbol{q}_b^n[k+1]\|} = \frac{\boldsymbol{q}_b^n[k+1]}{\sqrt{\boldsymbol{q}_b^n[k+1]^\top \boldsymbol{q}_b^n[k+1]}}$$

4. *Let* $k = k+1$ *and return to Step 2.*

2.2 Unit Quaternions (cont.)

Discretization of the unit quaternion differential equation

```
% Exact discretization is much more acccurate than Euler's method
% MSS supports both methods thanks to Tquat.m
w = [p q r]';

% Euler intrgration
q = q + h * Tquat(q) * w;
q = q / sqrt(q' * q);

% Exact discretization
q = expm( Tquat(w) * h ) * q;
q = q / sqrt(q' * q);
```

$$\dot{q}_b^n = \frac{1}{2}\begin{bmatrix} -\varepsilon^\top \\ \eta \boldsymbol{I}_3 + \boldsymbol{S}(\varepsilon) \end{bmatrix} \omega_{nb}^b := \boldsymbol{T}(q_b^n)\omega_{nb}^b$$

It is possible to rewrite the expression

$\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\mathbf{w}$ such that $\dot{\mathbf{q}} = \bar{\mathbf{T}}(\mathbf{w})\mathbf{q}$

```
function T = Tquat(u)
% Tq = Tquat(q) computes the quaternion transformation matrix Tq of
%   dimension 4 x 3 for attitude such that q_dot = Tq * w
% Tw = Tquat(w) computes the quaternion transformation matrix Tw of
%   dimension 4 x 4 for attitude such that q_dot = Tw * q
%
% The standard inout is the unit quaternion q = [ eta eps1 eps2 eps3 ]' of
% dimenison 4. If the angular rates w ≈ [ p q r ]' of dimension 3 are used
% as input, the alternative matrix Tw is returned.

if (length(u) == 4)                % T = Tq and q = u

    eta  = u(1); eps1 = u(2); eps2 = u(3); eps3 = u(4);

    T = 0.5 * [...
        -eps1 -eps2 -eps3
         eta  -eps3  eps2
         eps3  eta  -eps1
        -eps2  eps1  eta   ];

elseif (length(u) == 3)            % T = Tw and w = u

    w = [u(1) u(2) u(3)]';

    T = 0.5 * [...
        0  -w'
        w  -Smtrx(w) ];

end
```

2.2 Unit Quaternions (cont.)

Continuous-time algorithm for unit quaternion normalization

$$\dot{\boldsymbol{q}}_b^n = \boldsymbol{T}(\boldsymbol{q}_b^n)\boldsymbol{\omega}_{nb}^b + \frac{\gamma}{2}\left(1 - (\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n\right)\boldsymbol{q}_b^n$$

If ***q*** is initialized as a unit vector, then it will remain a unit vector.

However, integration of the quaternion vector ***q*** from the differential equation will introduce numerical errors that will cause the length of ***q*** to deviate from unity.

This is avoided by introducing feedback:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left((\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n\right) = \underbrace{2(\boldsymbol{q}_b^n)^\top \boldsymbol{T}(\boldsymbol{q}_b^n)\boldsymbol{\omega}_{nb}^b}_{0 \text{ since } \boldsymbol{q}_b^n(0) \text{ is a unit vector}} + \gamma\left(1 - (\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n\right)(\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n$$

$$= \gamma\left(1 - (\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n\right)(\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n$$

A change of coordinates $x = 1 - (\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n$ and $\dot{x} = -\mathrm{d}/\mathrm{d}t\left((\boldsymbol{q}_b^n)^\top \boldsymbol{q}_b^n\right)$ yields

$$\dot{x} = -\gamma x(1 - x) \qquad \gamma \geq 0 \text{ (typically 100)}$$

$\dot{x} = -\gamma x(1 - x)$ linearization about $x = 0$ gives $\dot{x} = -\gamma x$

2.2 Unit Quaternion from Euler Angles

If the Euler angles $\boldsymbol{\Theta}_{nb} = [\phi, \theta, \psi]^\top$ are known the corresponding unit quaternion can be computed using the transformation (NASA 2013):

$$\boldsymbol{q}_b^n = \begin{bmatrix} \cos\left(\frac{1}{2}\psi\right)\cos\left(\frac{1}{2}\theta\right)\cos\left(\frac{1}{2}\phi\right) + \sin\left(\frac{1}{2}\psi\right)\sin\left(\frac{1}{2}\theta\right)\sin\left(\frac{1}{2}\phi\right) \\ \cos\left(\frac{1}{2}\psi\right)\cos\left(\frac{1}{2}\theta\right)\sin\left(\frac{1}{2}\phi\right) - \sin\left(\frac{1}{2}\psi\right)\sin\left(\frac{1}{2}\theta\right)\cos\left(\frac{1}{2}\phi\right) \\ \sin\left(\frac{1}{2}\psi\right)\cos\left(\frac{1}{2}\theta\right)\sin\left(\frac{1}{2}\phi\right) + \cos\left(\frac{1}{2}\psi\right)\sin\left(\frac{1}{2}\theta\right)\cos\left(\frac{1}{2}\phi\right) \\ \sin\left(\frac{1}{2}\psi\right)\cos\left(\frac{1}{2}\theta\right)\cos\left(\frac{1}{2}\phi\right) - \cos\left(\frac{1}{2}\psi\right)\sin\left(\theta\right)\sin\left(\frac{1}{2}\phi\right) \end{bmatrix}$$

Formula (2.82) is implemented in the MSS toolbox as `euler2q.m` and it is only valid for the zyx convention of the Euler angles. This formula can also be used to compute the initial values of the Euler parameters corresponding to Step 1 of Algorithm 2.1.

```
phi=10*(pi/180); psi = 30*(pi/180); theta=-20*(pi/180);
q = euler2q(phi,theta,psi)
q =
    0.9437
    0.1277
   -0.1449
    0.2685
>> norm(q)          % normalization test
ans =
     1
```

NASA (2013). Mission Planning and Analysis Division. Euler Angles, Quaternions, and Transformation Matrices. *https://ntrs.nasa.gov/api/citations/19770024290/downloads/19770024290.pdf*

2.2 Euler Angles from a Unit Quaternion

Since the rotation matrices of the two kinematic representations are equal:

$$\boldsymbol{R}(\boldsymbol{\Theta}_{nb}) := \boldsymbol{R}(\boldsymbol{q}_b^n) \qquad \boldsymbol{\Theta}_{nb} = [\phi, \theta, \psi]^\top \quad \mathbf{q} = [\eta, \varepsilon_1, \varepsilon_2, \varepsilon_3]^\top$$

$$\begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

Euler angle solutions:

$$\begin{aligned} \phi &= \mathrm{atan2}(R_{32}, R_{33}) \\ \theta &= -\mathrm{asin}(R_{31}), \qquad \theta \neq \pm 90^\circ \\ \psi &= \mathrm{atan2}(R_{21}, R_{11}) \end{aligned}$$

$$\begin{aligned} \phi &= \mathrm{atan2}\left(2(\varepsilon_2\varepsilon_3 + \varepsilon_1\eta),\, 1 - 2(\varepsilon_1^2 + \varepsilon_2^2)\right) \\ \theta &= -\mathrm{asin}\left(2(\varepsilon_1\varepsilon_3 - \varepsilon_2\eta)\right) \\ \psi &= \mathrm{atan2}\left(2(\varepsilon_1\varepsilon_2 + \varepsilon_3\eta),\, 1 - 2(\varepsilon_2^2 + \varepsilon_3^2)\right) \end{aligned}$$

where atan2(y,x) is the 4-quadrant inverse tangent confining the result to $(-\pi, \pi]$

Case Study: Numerical Integration of the Kinematic Differential Equations

```
% Numerical integration of the kinematic differential equations. The unit
% quaternions are integrated, and the computed quaternion is transformed into
% Euler angles and compared with the solution obtained by the numerical
% integration of the Euler angle differential equation.
clearvars;

h = 0.01;                       % Sampling time
t_final = 50;                   % Final simulation time
N = round(t_final/h);           % Number of samples

nu = [5 -3 3 0.1 0.02 -0.02]';  % Initial velocity vector
nu_ref = zeros(6,1);            % Desired velocity vector

% Initial values: Euler angle representation (subscript 1)
pos1 = [ 0 0 0]';
Theta1 = deg2rad( [ 10 5 1 ] )';

% Initial values: unit quaternion representation (subscript 2)
pos2 = pos1;
quat2 = euler2q(Theta1(1), Theta1(2), Theta1(3));
```

MAIN LOOP

```
sim.data = zeros(N+1,18);
for i = 1:N+1

    sim.time(i) = (i-1) * h;

    % Transform quaternion q2 to Euler angles Theta2, for comparison
    [phi2, theta2, psi2] = q2euler(quat2);
    Theta2 = [phi2 theta2 psi2]';

    % Store simulation data at time [k]
    sim.data(i,:) = [pos1' Theta1' pos2' Theta2' nu'];

    % Numerical integration of Euler angle differential equations [k+1]
    pos1 = pos1 + h * Rzyx(Theta1(1),Theta1(2),Theta1(3)) * nu(1:3); % Forward Euler
    Theta1 = Theta1 + h * Tzyx(Theta1(1),Theta1(2)) * nu(4:6);       % Forward Euler

    % Numerical integration of quaternion differential equations [k+1]
    pos2 = pos2 + h * Rquat(quat2) * nu(1:3);      % Forward Euler
    quat2 = expm( Tquat(nu(4:6) ) * h) * quat2;    % Exact discretization
    quat2  = quat2 / norm(quat2);                  % Normalization

    % Velocities at time [k+1]
    w_n = 0.1;              % LP-filter natural frequency
    nu = lowPassFilter(nu, nu_ref, w_n, h);

end
```

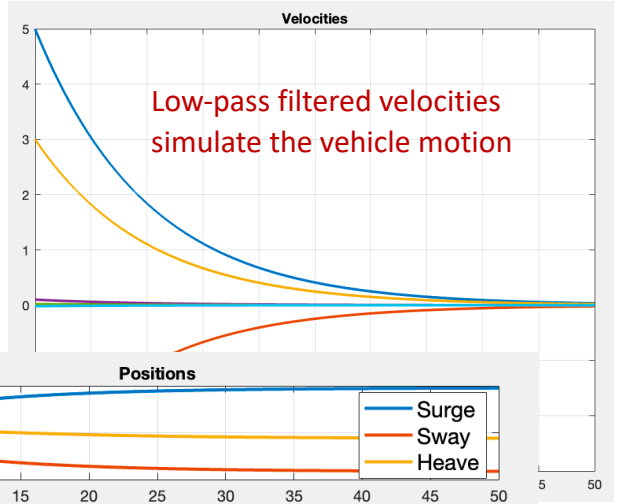
Case Study: Numerical Integration of the Kinematic Differential Equations (cont.)

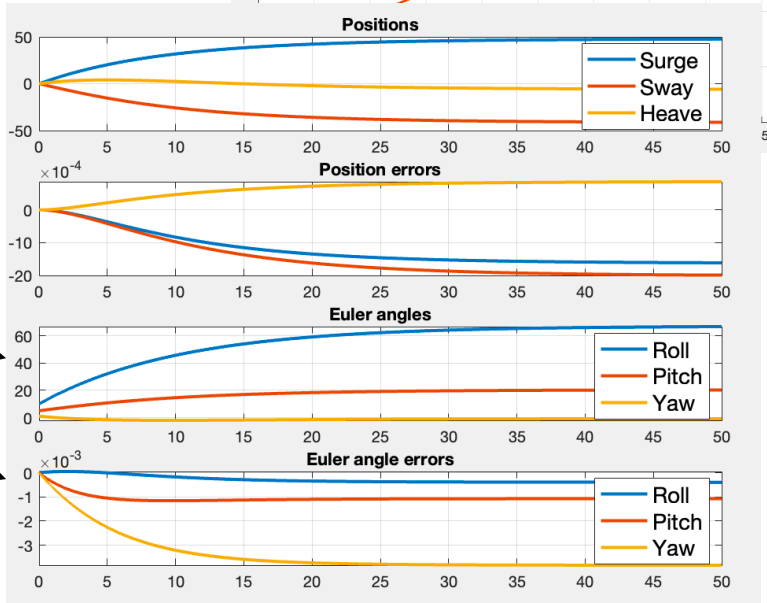
PLOTS

```
t = sim.time;
pos1 = sim.data(:,1:3);
Theta1 = rad2deg(sim.data(:,4:6));
pos2 = sim.data(:,7:9);
Theta2 = rad2deg(sim.data(:,10:12));
nu = sim.data(:,13:18);

figure(1); figure(gcf)
subplot(411); plot(t,pos1,'linewidth',2);
legend('Surge','Sway','Heave','FontSize',14)
grid;
title('Positions')
subplot(412); plot(t,pos1-pos2,'linewidth',2);
grid;
title('Position errors')
subplot(413); plot(t,Theta1,'linewidth',2);
legend('Roll','Pitch','Yaw','FontSize',14)
grid;
title('Euler angles')
subplot(414); plot(t,ssa(Theta1-Theta2),'linewidth',2);
legend('Roll','Pitch','Yaw','FontSize',14)
grid;
title('Euler angle errors')

figure(2); figure(gcf)
plot(t,nu,'linewidth',2)
grid;
title('Velocities')
```

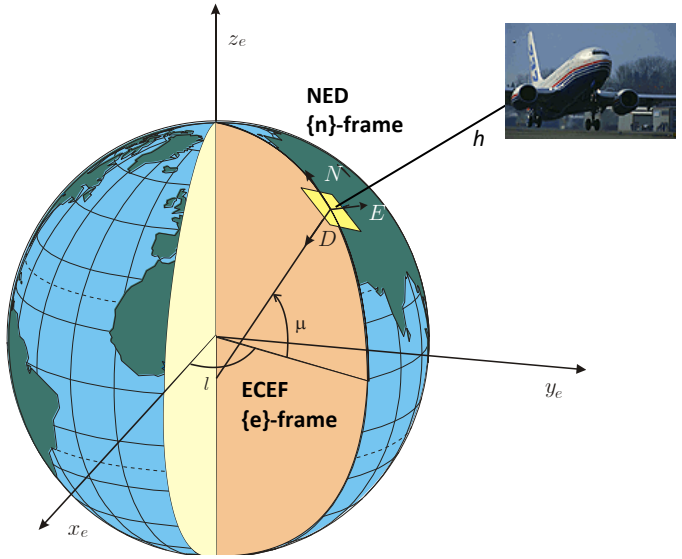




2.3 Transformation between ECEF and NED

A point on or above the Earth's surface is uniquely determined by:

Longitude: l (deg)
Latitude: μ (deg)
Ellipsoidal height: h (m)



NED axes definitions:

N – North axis is pointing North
E – East axis is pointing East
D – Down axis is pointing down in the normal direction to the Earth's surface

2.3 Longitude and Latitude Transformations

The transformation between the ECEF and NED velocity vectors is:

$$\dot{\boldsymbol{p}}^e_{eb} = \boldsymbol{R}^e_n \dot{\boldsymbol{p}}^n_{eb} = \boldsymbol{R}^e_n \boldsymbol{R}^n_b \boldsymbol{v}^b_{eb} \qquad \boldsymbol{\Theta}_{en} = [l, \mu]^\top$$

Two principal rotations:

1. a rotation *l* about the z-axis
2. a rotation $(-\mu - \pi/2)$ about the *y*-axis.

$$\begin{aligned}\boldsymbol{R}^e_n &:= \boldsymbol{R}_{z,l} \boldsymbol{R}_{y,-\mu-\frac{\pi}{2}} \\ &= \begin{bmatrix} \cos(l) & -\sin(l) & 0 \\ \sin(l) & \cos(l) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\left(-\mu-\frac{\pi}{2}\right) & 0 & \sin\left(-\mu-\frac{\pi}{2}\right) \\ 0 & 1 & 0 \\ -\sin\left(-\mu-\frac{\pi}{2}\right) & 0 & \cos\left(-\mu-\frac{\pi}{2}\right) \end{bmatrix}\end{aligned}$$

$$\boldsymbol{R}^e_n = \boldsymbol{R}(\boldsymbol{\Theta}_{en}) = \begin{bmatrix} -\cos(l)\sin(\mu) & -\sin(l) & -\cos(l)\cos(\mu) \\ -\sin(l)\sin(\mu) & \cos(l) & -\sin(l)\cos(\mu) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix}$$

Matlab:
The rotation matrix $\boldsymbol{R}^e_n$ is computed using the MSS toolbox command

```
R = Rll(l,mu)
```

2.3 Longitude/Latitude from ECEF Coordinates

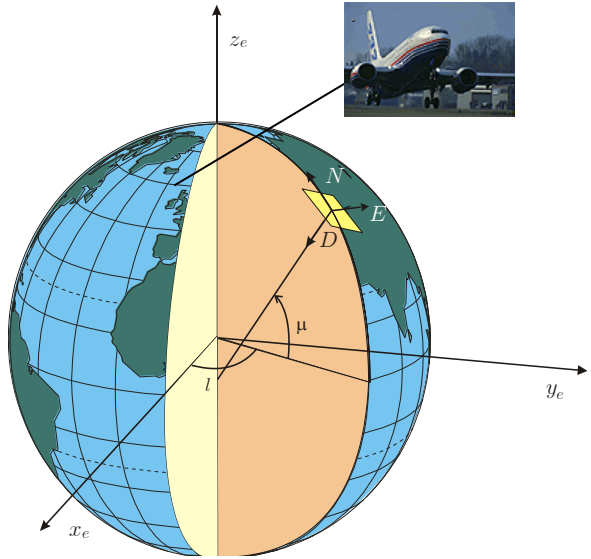
Satellite navigation system measurements are given in the ECEF frame:
Not to useful for the operator.

Presentation of terrestrial position data $\boldsymbol{p}_{eb}^{e} = [x^e, y^e, z^e]^\top$ is therefore made in terms of the ellipsoidal parameter's longitude *l*, latitude μ and height *h*.

$$\boldsymbol{p}_{eb}^{e} = [x^e, y^e, z^e]^\top$$

Transformation

l, μ and *h*



2.3 Longitude/Latitude from ECEF Coordinates (cont.)

Parameters	Comments
$r_e = 6\,378\,137$ m	Equatorial radius of ellipsoid (semimajor axis)
$r_p = 6\,356\,752$ m	Polar axis radius of ellipsoid (semiminor axis)
$\omega_e = 7.292115 \cdot 10^{-5}$ rad/s	Angular velocity of the Earth
$e = 0.0818$	Eccentricity of ellipsoid

WGS-84

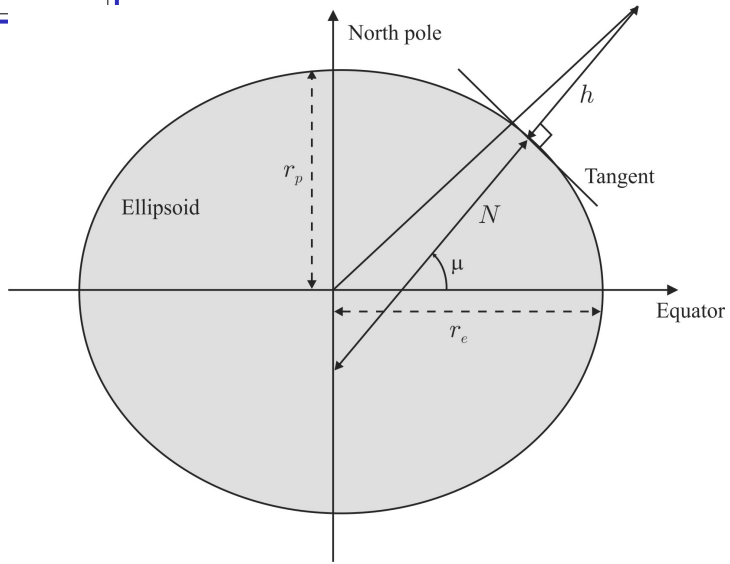
$$N = \frac{r_e^2}{\sqrt{r_e^2 \cos^2\mu + r_p^2 \sin^2\mu}} \qquad e = \sqrt{1 - \left(\frac{r_p}{r_e}\right)^2}$$

$$l = \text{atan}\left(\frac{y^e}{x^e}\right) \qquad \boldsymbol{p}_{eb}^e = [x^e, y^e, z^e]^\top$$

while latitude μ and height *h* are implicitly computed by

$$\tan(\mu) = \frac{z^e}{p}\left(1 - e^2 \frac{N}{N+h}\right)^{-1}$$

$$h = \frac{p}{\cos(\mu)} - N$$



World Geodetic System (1984). Its Definition and Relationships with Local Geoedtic Systems. DMA TR 8350.2, 2nd ed., Defence Mapping Agency, Fairfax, VA.

2.3 Longitude/Latitude from ECEF Coordinates (cont.)

Algorithm 2.2 (Transformation of (x^e, y^e, z^e) to (l, μ, h))

1. *Compute* $p = \sqrt{(x^e)^2 + (y^e)^2}$.
2. *Compute the approximate value* μ_0 *from*

$$\tan(\mu_0) = \frac{z^e}{p}(1 - e^2)^{-1}$$

3. *Compute an approximate value* N *from*

$$N = \frac{r_e^2}{\sqrt{r_e^2 \cos^2(\mu_0) + r_p^2 \sin^2(\mu_0)}}$$

4. *Compute the ellipsoidal height by*

$$h = \frac{p}{\cos(\mu_0)} - N$$

5. *Compute an improved value for the latitude by*

$$\tan(\mu) = \frac{z^e}{p}\left(1 - e^2 \frac{N}{N + h}\right)^{-1}$$

6. *Check for another iteration step: if* $|\mu - \mu_0| < \varepsilon$, *where* ε *is a small number, then the iteration is complete. Otherwise set* $\mu_0 = \mu$ *and continue with Step 3.*

Matlab:
Algorithm 2.2 is programmed in the MSS toolbox as a function

```
[l,mu,h] = ecef2llh(x,y,z)
```

Hofmann-Wellenhof, B., H. Lichtenegger and J. Collins (1994). Global Positioning System: Theory and Practice. 3rd ed. Springer Verlag. New York, NY.

2.3 ECEF Coordinates from Longitude/Latitude

The transformation from $\Theta_{en} = [l, \mu]^\top$ for given heights *h* to

$$\boldsymbol{p}_{eb}^{e} = [x^e, y^e, z^e]^\top$$

is given by (Heiskanen and Moritz 1967)

$$\begin{bmatrix} x^e \\ y^e \\ z^e \end{bmatrix} = \begin{bmatrix} (N+h)\cos(\mu)\cos(l) \\ (N+h)\cos(\mu)\sin(l) \\ \left(\frac{r_p^2}{r_e^2}N + h\right)\sin(\mu) \end{bmatrix}$$

Heiskanen, W. A. and H. Moritz (1967). Physical Geodesy. Freeman. London.

2.3 ECEF Coordinates from Longitude/Latitude (cont.)

Matlab:
The transformation from $\boldsymbol{\Theta}_{en} = [l, \mu]^\top$ to $\boldsymbol{p}^e_{eb} = [x^e, y^e, z^e]^\top$. Equation (2.96), is programmed in the MSS toolbox function

```
[x,y,z] = llh2ecef(l,mu,h)
```

Assume that $l = 10.3°$, $\mu = 63.0°$ and $h = 0$ m. Hence, the ECEF coordinates are computed to be

$$\begin{bmatrix} x^e \\ y^e \\ z^e \end{bmatrix} = \begin{bmatrix} 2\,856\,552 \text{ m} \\ 519\,123 \text{ m} \\ 5\,659\,978 \text{ m} \end{bmatrix}$$

using the MSS Matlab command

```
[x,y,z] = llh2ecef(10.3*(pi/180),63.0*(pi/180),0)
```

2.4 Transformations between ECEF and Flat-Earth Coordinates

For local flat-Earth navigation it can be assumed that the NED tangent plane is fixed on the surface of the Earth

Assume that the NED tangent plane is located at l_0 and μ_0 such that

$$\boldsymbol{R}_n^e = \boldsymbol{R}(\boldsymbol{\theta}_{en}) = \text{constant} \qquad \boldsymbol{\theta}_{en} = [l_0, \mu_0]^\top$$

The ECEF coordinates satisfy the differential equation

$$\dot{\boldsymbol{p}}_{eb}^e = \boldsymbol{R}_n^e \boldsymbol{R}_b^n \boldsymbol{v}_{eb}^b$$

Flat Earth is a good approximation for ships and floating structures operating in a limited region.

Flat Earth is a bad approximation for global waypoint tracking control systems for marine craft since (l, μ) will vary largely for vessels in transit between the different continents.

2.4 Longitude, Latitude and Height from Flat-Earth Coordinates

Given a local NED position (x^n, y^n, z^n) with coordinate origin (l_0, μ_0) and reference height h_{ref} in meters above the surface of the Earth, the change in longitude and latitude is (Equation 2.39, Farrell 2008)

$$\Delta l = \frac{y^n}{(R_N + h_{\text{ref}})\cos(\mu_0)}$$
$$\Delta\mu = \frac{x^n}{R_M + h_{\text{ref}}}$$

$$\Delta l := l - l_0$$
$$\Delta\mu := \mu - \mu_0$$

```
% WGS-84 parameters
a = 6378137;              % Semi-major axis (equitorial radius)
f = 1 / 298.257223563;    % Flattening
e = sqrt( 2*f - f^2 );    % Earth eccentricity
```

$$R_N = \frac{r_e}{\sqrt{1 - e^2\sin^2(\mu_0)}}$$
$$R_M = R_N\frac{1 - e^2}{1 - e^2\sin^2(\mu_0)}$$

$$l = \text{ssa}(l_0 + \Delta l)$$
$$\mu = \text{ssa}(\mu_0 + \Delta\mu)$$
$$h = h_{\text{ref}} - z^n$$

Matlab:
The ssa$(\cdot)$ function is implemented in the MSS toolbox as

```
angle = ssa(angle,type)
```

where `type` can be chosen as `'rad'` or `'deg'`.

Matlab:
The triplet (l, μ, h) corresponding to the NED coordinates (x^n, y^n, z^n) in a flat-Earth coordinate origin located at (l_0, μ_0) with reference height h_{ref} in meters above the ellipsoid is computed as

```
[l,mu,h] = flat2llh(x,y,z,l0,mu0,h_ref)
```

ssa is the smallest signed angle confining the argument to the interval $[-\pi, \pi)$

Smallest Signed Angle: Smallest Difference Between Two Angles

```
function angle = ssa(angle,unit)
% SSA is the "smallest signed angle" or the smallest difference between two
% angles. Examples:
%
% angle = ssa(angle) maps an angle in rad to the interval [-pi pi)
% angle = ssa(angle,'deg') maps an angle in deg to the interval [-180 180)
%
% For feedback control systems and state estimators used to control the
% attitude of vehicles, the difference of two angles should always be
% mapped to [-pi pi) or [-180 180] to avoid step inputs/discontinuties.
%
% Note that in many languages (C, C++, C#, JavaScript), the modulus
% operator mod(x,y) returns a value with the same sign as x.
% For these use a custom mod function: mod(x,y) = x - floor(x/y) * y
% For the Unity game engine use: Mathf.DeltaAngle
%
% Author:     Thor I. Fossen
% Date:       2018-09-21
% Revisions:  2020-03-04  Deafult rad, otional argument for deg
%_____________________________________________________________________

if (nargin == 1)
    angle = mod( angle + pi, 2 * pi ) - pi;
elseif strcmp(unit,'deg')
    angle = mod( angle + 180, 360 ) - 180;
end
```

```
Command Window
>> ssa(181,'deg')
ans =
  -179
>> ssa(179,'deg')
ans =
   179
fx >>
```

2.4 Flat-Earth Coordinates from Longitude, Latitude and Height

The NED positions (x_n, y_n, z_n) with respect to a flat-Earth coordinate system with origin (l_0, μ_0) and reference height h_{ref} are computed as (Equation 2.39, Farrell 2008)

$$\Delta l := l - l_0$$
$$\Delta \mu := \mu - \mu_0$$

$$x^n = \Delta\mu \, (R_M + h_{ref})$$
$$y^n = \Delta l \, (R_N + h_{ref}) \cos(\mu_0)$$
$$z^n = h_{ref} - h$$

$$R_N = \frac{r_e}{\sqrt{1 - e^2 \sin^2(\mu_0)}}$$
$$R_M = R_N \frac{1 - e^2}{1 - e^2 \sin^2(\mu_0)}$$

Matlab:
For a marine craft operating on the sea surface ($h = 0$) or submerged ($h < 0$) with longitude and latitude (l, μ) and a local coordinate origin at (l_0, μ_0, h_{ref}), the NED positions can be computed using the MSS toolbox command

```
[x,y,z] = llh2flat(l,mu,h,l0,mu0,h_ref)
```

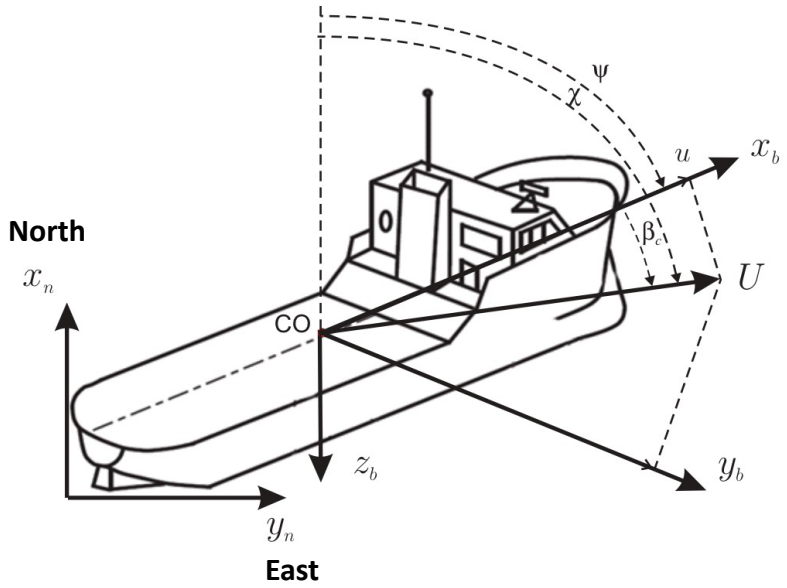
2.5 Definitions of Course, Heading and Crab Angles

The relationship between the course angle, heading angle and crab angle is important for maneuvering of a vehicle in the horizontal plane.

The terms course and heading are used interchangeably in much of the literature on guidance, navigation and control and this leads to confusion.

- **Course angle χ**
  The course angle χ of a vehicle is the cardinal direction in which the vehicle is moving.
  Measured using GNSS (or HPR under water)
- **Heading (yaw) angle ψ**
  The heading angle ψ, is the direction the craft's bow (x_b axis) is pointed.
  Measured using a compass
- **Crab angle β_c**
  Inspection of the figure confirms that the crab angle is the difference between the course angle and the heading angle:

$$\chi = \psi + \beta_c \qquad \beta_c = \text{atan2}\,(v, u)$$



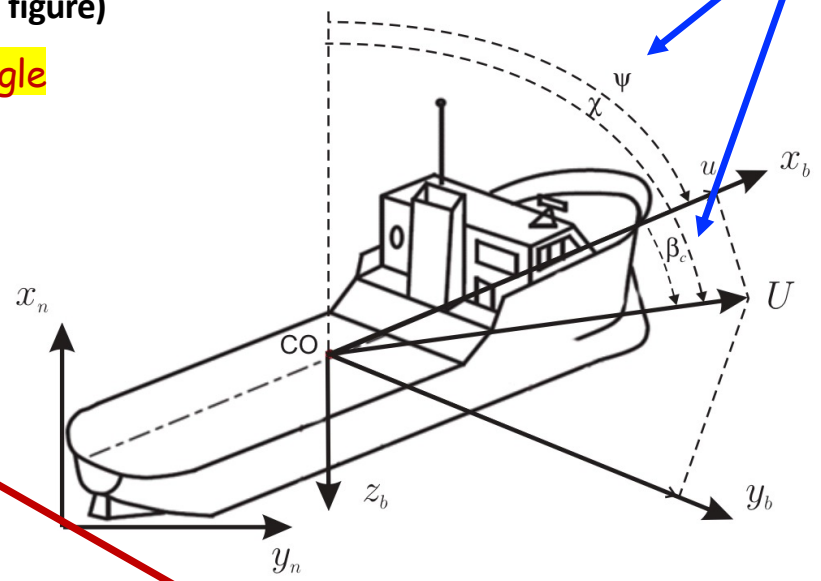
2.5 Crab Angle (Amplitude-Phase Form)

$$\chi = \psi + \beta_c$$

We want to prove the famous relationship (from the figure)

course angle = heading (yaw) angle + crab angle

$$\chi = \psi + \beta_c \qquad \beta_c = \tan^{-1}\left(\frac{v}{u}\right)$$



Proof:

$$\dot{x}^n = u\cos(\psi) - v\sin(\psi)$$
$$\dot{y}^n = u\sin(\psi) + v\cos(\psi)$$

These equations can be expressed in amplitude-phase form

$$\dot{x}^n = U\cos(\psi + \beta_c) := U\cos(\chi)$$
$$\dot{y}^n = U\sin(\psi + \beta_c) := U\sin(\chi)$$

Amplitude (speed):

$$U = \sqrt{u^2 + v^2}$$

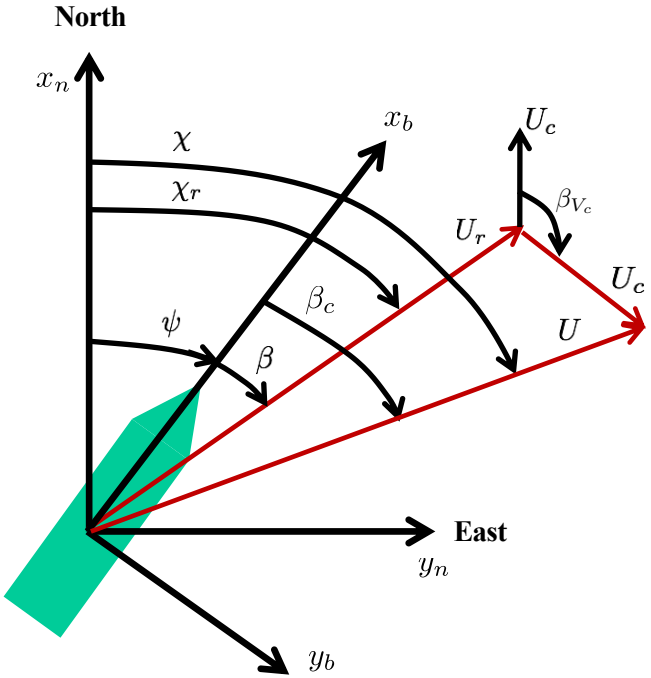
Phase (crab angle):

$$\beta_c = \tan^{-1}\left(\frac{v}{u}\right)$$

$$\begin{aligned} y_1 &= c_1\sin(x) + c_2\cos(x) \\ &= A\sin(x)\cos(\varphi) + A\cos(x)\sin(\varphi) \\ &= A\sin(x+\varphi) \end{aligned}$$

$$c_1 = A\cos(\varphi),\ c_2 = A\sin(\varphi) \qquad A = (c_1^2 + c_2^2)^{1/2} \qquad \varphi = \tan^{-1}(c_2/c_1)$$

2.5 Ocean Current Triangle: Horizontal Plane



Horizontal crab angle:	$\beta_c = \text{atan2}(v, u)$
Sideslip angle:	$\beta = \text{atan2}(v_r, u_r)$
Heading (yaw) angle:	ψ
Course angle:	$\chi = \psi + \beta_c$
Relative course angle:	$\chi_r = \psi + \beta$
Speed over ground:	$U = \sqrt{u^2 + v^2 + w^2}$
Relative speed:	$U_r = \sqrt{u_r^2 + v_r^2 + w_r^2}$
Ocean current speed:	$U_c = \sqrt{u_c^2 + v_c^2 + w_c^2}$
Horizontal ocean current direction:	β_{V_c}

GNSS measures course angle χ and speed over ground U
Compass measures heading angle ψ
Currents can be measured by an Acoustic Doppler Current Profiler (ADCP)

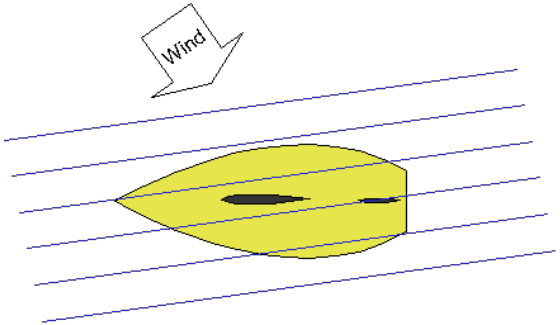
2.5 Crab Angle due to Sway Velocity

A B777 (as in the photo) would generally have a maximum crab angle of about 16 degrees when approaching and landing at the maximum demonstrated crosswind component of 38 knots.

$$\chi = \psi + \beta_c$$

$$\beta_c = \text{atan2}\,(v, u)$$

A marine craft exposed to wind, waves and ocean currents will also have a non-zero crab angle.



https://aviation.stackexchange.com/questions/58861/what-is-the-maximum-angle-between-an-airplane-and-runway-centerline-when-touchin

2.5 Extensions to Ocean Currents: Angle of Attack and Sideslip Angle

For a marine craft exposed to ocean currents, the concept of relative velocities is introduced. The relative velocities are

$$u_r = u - u_c$$
$$v_r = v - v_c$$
$$w_r = w - w_c$$

$$U_r = \sqrt{u_r^2 + v_r^2 + w_r^2}$$ Relative speed

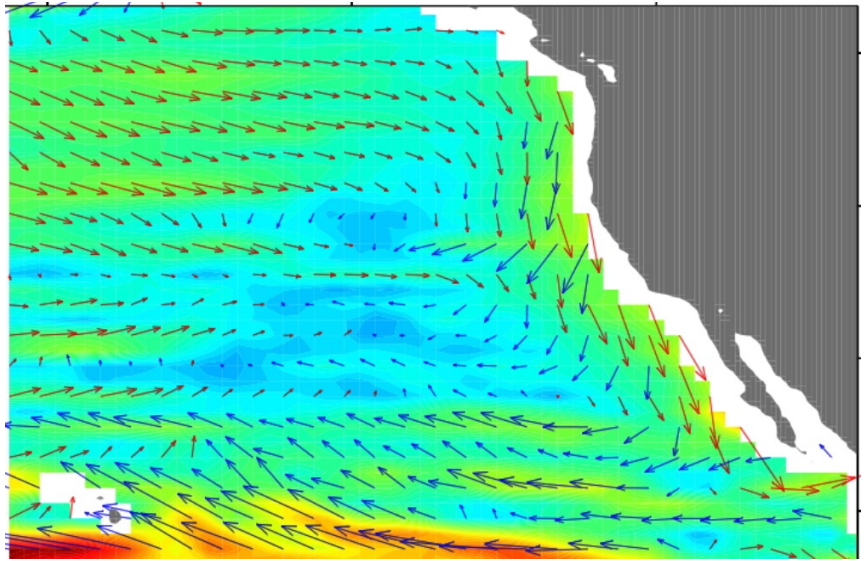
where u_c, v_c and w_c are the current velocities and

$$U_c = \sqrt{u_c^2 + v_c^2 + w_c^2}$$

is the ocean current speed.

Angle of attack and sideslip angle

$$\alpha = \text{atan2}(w_r, u_r)$$
$$\beta = \text{atan2}(v_r, u_r)$$



2.5 Crab Angle versus Sideslip Angle

Aircraft operate in the wind, while marine craft can be exposed to ocean currents, waves, and wind. The aerodynamic and hydrodynamic forces are functions of the relative velocity velocities

$$u_r = u - u_f$$
$$v_r = v - v_f$$

The subscript f denotes the flow due to wind, waves and/or ocean currents

Lift will be perpendicular and drag will be parallel to the relative flow. The 2-D linear relative velocities can be expressed as:

$$u_r = U_r \cos(\beta)$$
$$v_r = U_r \sin(\beta)$$
$$U_r = \left(u_r^2 + v_r^2\right)^{1/2}$$

Sideslip angle

$$\beta = \text{atan2}\left(v_r, u_r\right)$$

Note that crab angle is equal to the sideslip angle when $u_f = v_f = 0$

$$\beta_c = \text{atan2}\left(v, u\right)$$

Unlike **atan**, which only give results between $-\pi/2$ and $\pi/2$ and can be ambiguous, **atan2** returns results between $-\pi$ and π, covering all possible directions of the velocity vector.

- **Crab angle** is the angle between the direction χ the vehicle it is moving due to external forces such as ocean currents, waves or winds and the heading ψ of the vehicle satisfying $\beta_c = \chi - \psi$
- **Sideslip angle** is the the angle between the x_b-axis of the vehicle and the direction of the flow velocity.

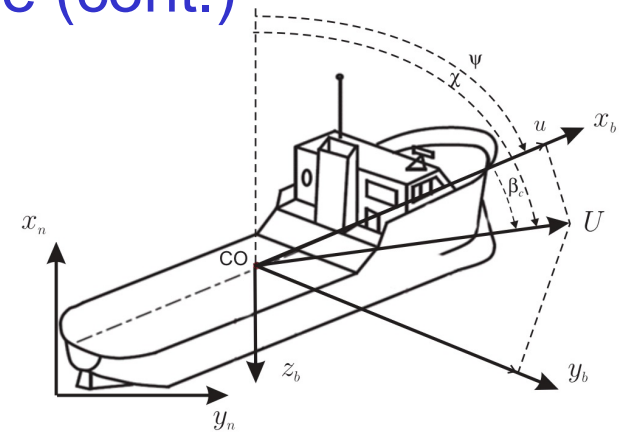
In the literature, the term sideslip angle is often used for the crab angle while we explicitly distinguish between the angles
The sideslip angle is used to compute the aero-/hydrodynamic coefficients, while the crab angle enters the LOS guidance law

2.5 Crab Angle versus Sideslip Angle (cont.)

Some interesting observations

1) A vehicle moving on a straight line in calm water ($U > 0$ and $v = 0$) will have a zero-crab angle $\beta_c = 0$

$$\beta_c = \text{atan}\left(\frac{v}{u}\right) = \sin^{-1}\left(\frac{v}{U}\right) \quad \overset{\beta_c \text{small}}{\Rightarrow} \quad \beta_c \approx \frac{v}{U}$$

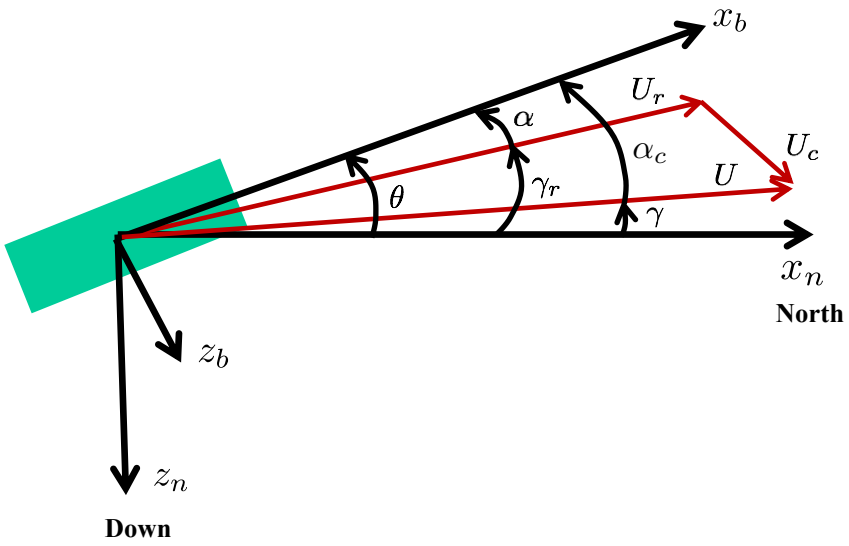


2) As soon as you start to turn, the sway velocity will be non-zero and consequently, $\beta_c \neq 0$. The crab angle corresponds to the amount of correction a vehicle must be turned to maintain the desired course.

When the crab angle is nonzero, it indicates that the vehicle is moving sideways to some extent. This is commonly referred to as sideslipping. Note that sideslipping is caused by a nonzero crab angle (NOT the sideslip angle).

3) A flow velocity (wind/current) is induced if the vehicle is exposed to environmental forces. The environmental force will change both v and U through the equations of motion and, thus, the crab angle.

2.5 Ocean Current Triangle: Vertical Plane



Vertical crab angle:	$\alpha_c = \text{atan2}(w, u)$
Angle of attack:	$\alpha = \text{atan2}(w_r, u_r)$
Pitch angle:	θ
Flight path angle:	$\gamma = \theta - \alpha_c$
Relative flight path angle:	$\gamma_r = \theta - \alpha$
Speed over ground:	$U = \sqrt{u^2 + v^2 + w^2}$
Relative speed:	$U_r = \sqrt{u_r^2 + v_r^2 + w_r^2}$
Ocean current speed:	$U_c = \sqrt{u_c^2 + v_c^2 + w_c^2}$
Vertical ocean current direction:	α_{V_c}

GNSS measures flight path γ and speed over ground U
AHRS or INS measure pitch angle θ
Currents can be measured by an Acoustic Doppler Current Profiler (ADCP)

2.5 Transformation between BODY and FLOW

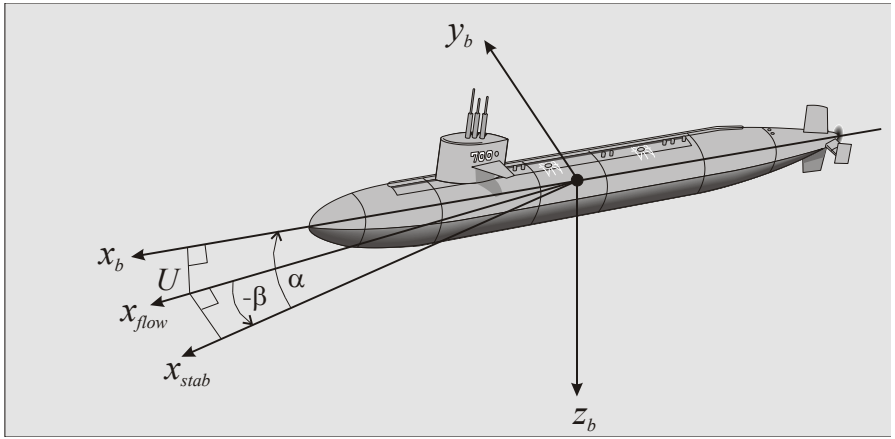
FLOW axes are often used to express hydrodynamic data. The FLOW axes are found by rotating the BODY axis system such that resulting x-axis is parallel to the freestream flow.

In FLOW axes, the x-axis directly points into the relative flow while the z-axis remains in the reference plane but rotates so that it remains perpendicular to the x-axis. The y-axis completes the right-handed system.

$$\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} -F_{\text{drag}} \\ -F_{\text{lift}} \end{bmatrix}$$

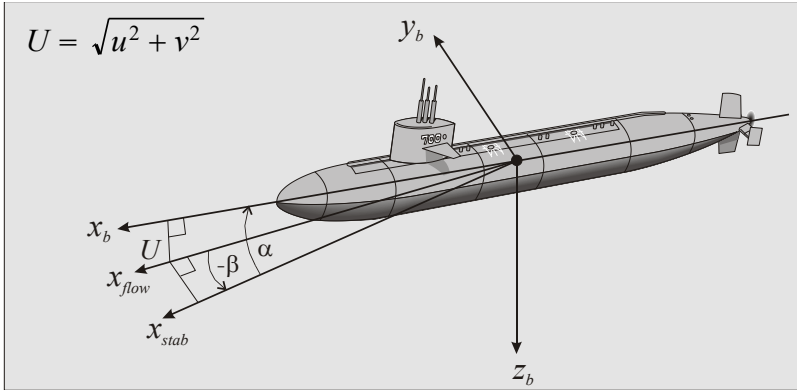
$$F_{\text{drag}} = \frac{1}{2}\rho V_r^2 S C_D(\alpha)$$

$$F_{\text{lift}} = \frac{1}{2}\rho V_r^2 S C_L(\alpha)$$



2.5 Rotation Matrix between BODY and FLOW

$$\boldsymbol{v}_r^{\text{stab}} = \boldsymbol{R}_{y,\alpha}\boldsymbol{v}_r^b$$
$$\boldsymbol{v}_r^{\text{flow}} = \boldsymbol{R}_{z,-\beta}\boldsymbol{v}_r^{\text{stab}}$$



Velocity transformation:

$$\begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \boldsymbol{R}_{y,\alpha}^{\top}\boldsymbol{R}_{z,-\beta}^{\top}\begin{bmatrix} U_r \\ 0 \\ 0 \end{bmatrix}$$

Principal rotations:

$$\mathbf{R}_{y,\alpha} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \quad \mathbf{R}_{z,-\beta} = \mathbf{R}_{z,\beta}^{\top} = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{v}_r^{\text{flow}} = \boldsymbol{R}_b^{\text{flow}}\boldsymbol{v}_r^b$$
$$\boldsymbol{v}_r^b = (\boldsymbol{R}_b^{\text{flow}})^{\top}\boldsymbol{v}_r^{\text{flow}}$$

$$\boldsymbol{R}_b^{\text{flow}} = \boldsymbol{R}_{z,-\beta}\boldsymbol{R}_{y,\alpha} = \begin{bmatrix} \cos(\beta)\cos(\alpha) & \sin(\beta) & \cos(\beta)\sin(\alpha) \\ -\sin(\beta)\cos(\alpha) & \cos(\beta) & -\sin(\beta)\sin(\alpha) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

2.5 Summary: Angle of Attack and Sideslip Angle

$$\boldsymbol{v}_r^{\text{flow}} = \boldsymbol{R}_b^{\text{flow}} \boldsymbol{v}_r^b \quad \rightarrow \quad \boldsymbol{v}_r^b = (\boldsymbol{R}_b^{\text{flow}})^\top \boldsymbol{v}_r^{\text{flow}}$$

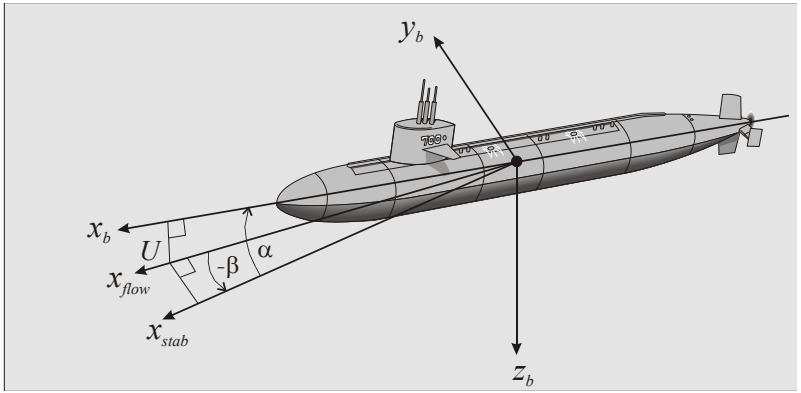
$$\boldsymbol{R}_b^{\text{flow}} = \boldsymbol{R}_{z,-\beta} \boldsymbol{R}_{y,\alpha} = \begin{bmatrix} \cos(\beta)\cos(\alpha) & \sin(\beta) & \cos(\beta)\sin(\alpha) \\ -\sin(\beta)\cos(\alpha) & \cos(\beta) & -\sin(\beta)\sin(\alpha) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

Relative velocities

$$\begin{aligned} u_r &= U_r \cos(\alpha)\cos(\beta) \\ v_r &= U_r \sin(\beta) \\ w_r &= U_r \sin(\alpha)\cos(\beta) \end{aligned}$$

Angle of attack and sideslip angle

$$\begin{aligned} \alpha &= \tan^{-1}\left(\frac{w_r}{u_r}\right) \\ \beta &= \sin^{-1}\left(\frac{v_r}{U_r}\right) \end{aligned}$$



Chapter Goals - Revisited

- Understand the geographic reference frame NED, the Earth-centered reference frame ECEF and the body-fixed reference frame BODY.
- Understand what FLOW axes are and why we use these axes for marine craft and aircraft
- Be able to write down the differential equations relating BODY velocities to NED positions, both for Euler angles and unit quaternions.
- Be able to:
  - Transform ECEF (x^{e}, y^{e}, z^{e}) positions to (longitude, latitude, height) and vice versa
  - Transform (longitude, latitude, height) to flat-Earth positions (x^{n}, y^{n}, z^{n}) and vice versa
- Define, visualize, and explain the use of:
  - Angle of attack α
  - Sideslip angle β
  - Vertical crab angle α_c
  - Horizontal crab angle β_c
  - Heading angle ψ
  - Course angle χ