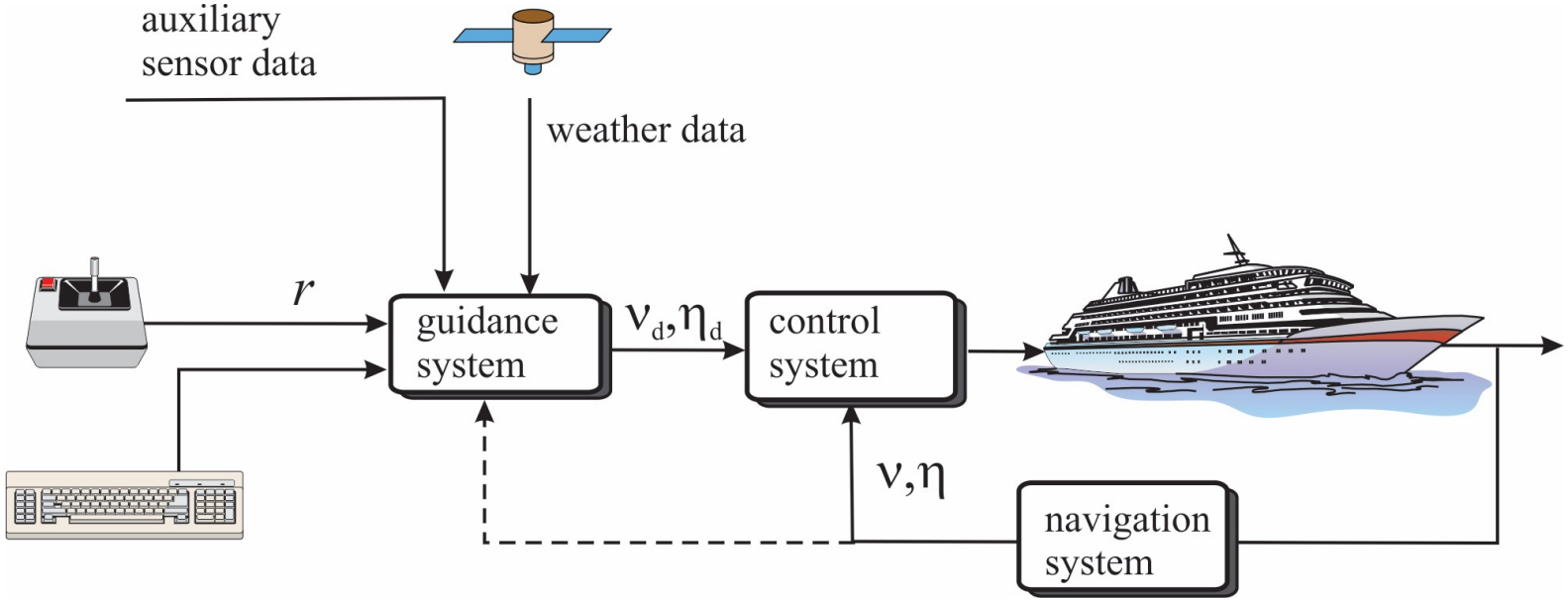
Chapter 12 – Guidance Systems

12.1 Trajectory Tracking
12.2 Guidance laws for Target Tracking
12.3 Linear Designs Methods for Path Following
12.4 LOS Guidance Laws for Path Following using Course Autopilots
12.5 LOS Guidance Laws for Path Following using Heading Autopilots
12.6 Curved-Path Path Following



Answers the question "Where do you want to go?"

Chapter Goals

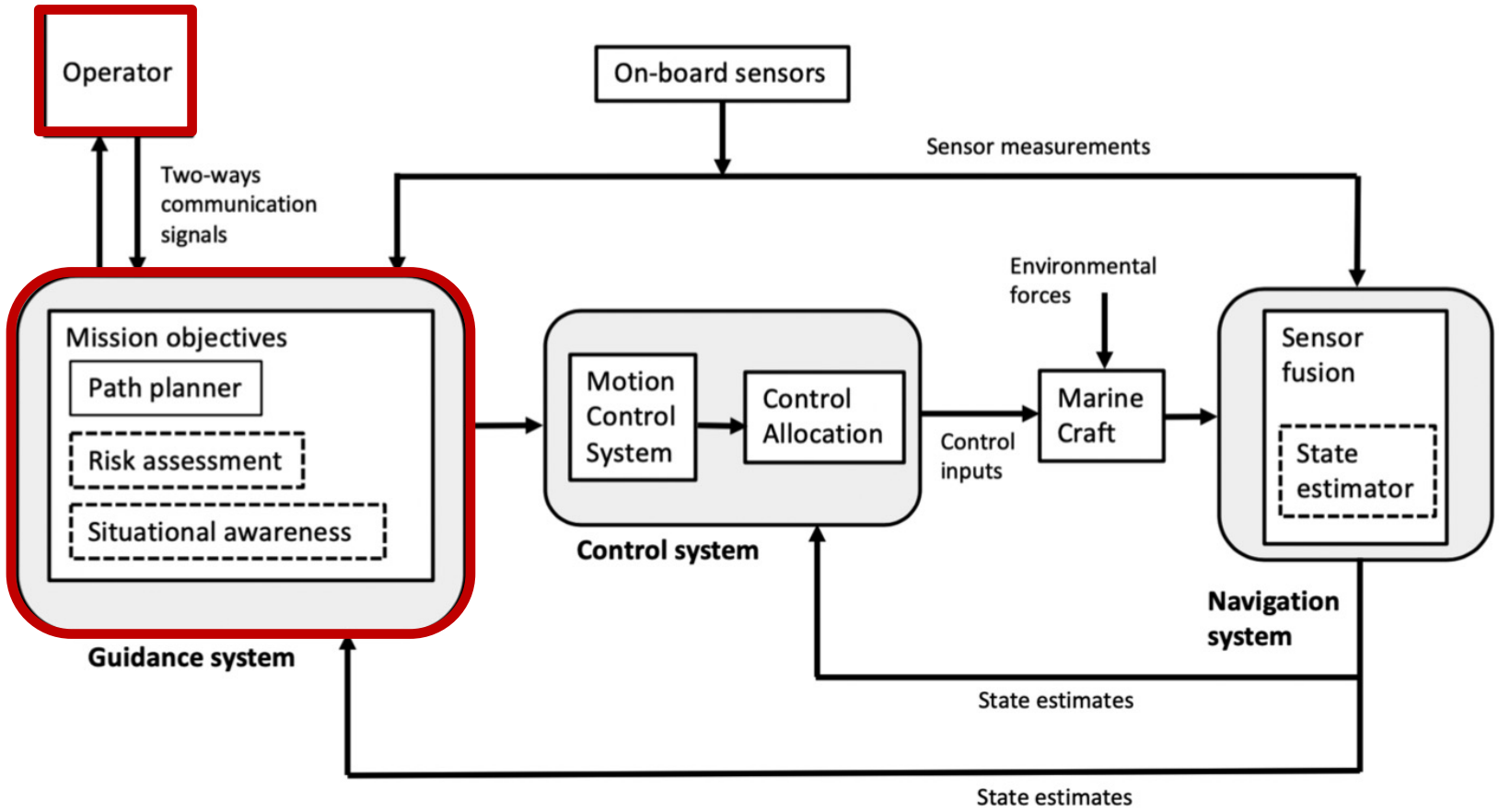
Guidance systems:

- Be able to design open-loop guidance systems using reference models (low-pass filters) for position and velocity control.
- Understand how to design a closed-loop guidance system using feedback control theory.

Guidance Laws:

- Be able to design guidance laws for target tracking, that is a moving object for which no future motion information, nor the path is known in advance.
- Be able to design a state-of-the-art linear path-following controller. This is the classical design method where the path is generated using waypoints to connect straight lines between the waypoints.
- Be able to design LOS guidance laws for path following using course autopilots.
- Be able to design LOS guidance laws for path following using heading autopilots, with extension to ILOS to compensate for sideslip and unmodelled dynamics.

Chapter 12 - Guidance Systems



Guidance refers to the determination of the desired path of travel (the "*trajectory*") from the craft's current location to a designated target, as well as desired changes in velocity, rotation and acceleration for following that path (main tool: dynamic optimization)

Guidance, Navigation and Control (GNC)

GNC is a branch of engineering dealing with the design of systems to control the movement of vehicles, especially, automobiles, drones, marine craft, aircraft, and spacecraft.

- **Guidance** focuses on the methods and systems used to follow a route or path to a destination once it has been identified. It's about steering, direction, and instruction on moving from one point to another.
  - How do I get there? is fundamentally a question of guidance.

Once you know your current location and your destination (navigation), guidance provides the step-by-step instructions or actions required to reach the destination.

- **Navigation** refers to the process of determining one's current position and planning a route to a desired destination.
  - Where am I? identifies your current location. This is the primary navigation question.
  - Where do I want to go? decides your destination
  - Which way should I go? plans the route to get there, considering various factors such as distance, time, obstacles, and preferences

- **Control** refers to the manipulation of the forces, by way of steering controls, thrusters, etc., needed to execute guidance commands whilst maintaining vehicle stability.

Chapter 12 - Guidance Systems

Charles Stark Draper, the father of inertial navigation, stated:
"Guidance depends upon fundamental principles and involves devices that are similar for vehicles moving on land, on water, under water, in air, beyond the atmosphere within the gravitational field of earth and in space outside this field"

Credit: National Air and Space Museum, Smithsonian Institution.

Charles Stark Draper (1901 – 1987) was an American scientist and engineer, known as the "father of inertial navigation". He was the founder and director of the MIT's Instrumentation Laboratory, later renamed the *Charles Stark Draper Laboratory*, which made the Apollo Moon landings possible through the Apollo Guidance Computer it designed for NASA.

Chapter 12 - Guidance Systems

Control objectives

In chapter 12, we will consider the following motion control scenarios:

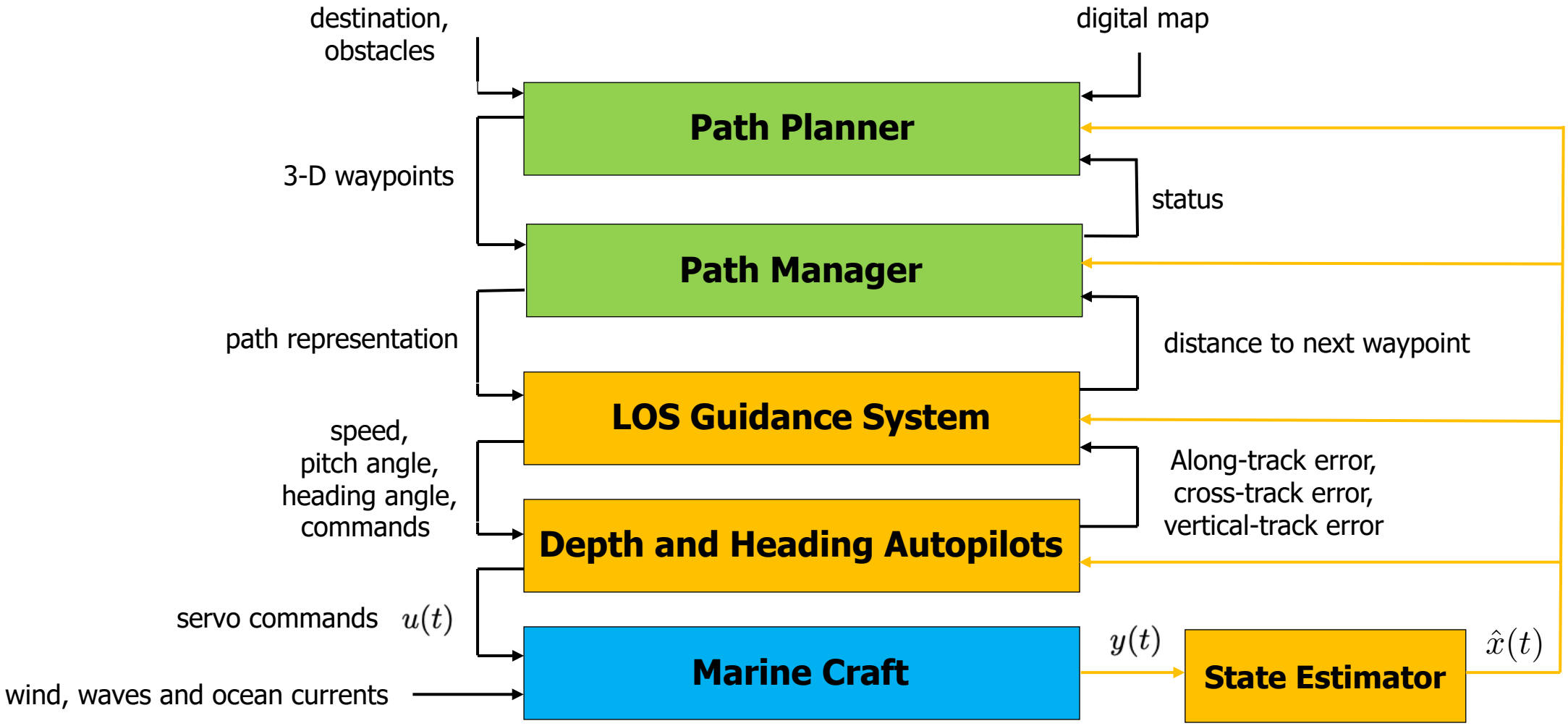
- **Setpoint regulation (point stabilization)** is a special case where the desired position and attitude are chosen to be constant.

- **Trajectory tracking** forces the system output $y(t)$ to track a desired output $y_d(t)$. The desired trajectory can be computed using reference models generated by low-pass filters, optimization methods or by simulating the marine craft motion.

- **Target tracking** methods for motion control scenarios then there are no information about the path and there is no trajectory to track. This is useful if the goal is to track a moving object such as a ship or an underwater vehicle, for which no future motion information is available. For this we use the target position and velocity as reference signals.

- **Path following** is following a predefined path independent of time. A popular approach is the Dubins path representing the shortest curve that connects two points in the 2-D Euclidean plane with a constraint on the curvature of the path and with prescribed initial and terminal tangents to the path. For path following no restrictions are placed on the temporal propagation along the path. Spatial constraints can, however, be added to represent obstacles and other positional constraints if they are known in advance.

Guidance System Architecture



12.1 Trajectory Tracking

Definition 12.1 (Trajectory Tracking)
A control system that forces the system output $\boldsymbol{y} \in \mathbb{R}^m$ *to track a desired output* $\boldsymbol{y}_d \in \mathbb{R}^m$ *solves a trajectory tracking problem.*

Tracking of a time-varying reference trajectory in 3 DOF (surge, sway and yaw) is achieved by minimizing the tracking error

$$\boldsymbol{e} := \boldsymbol{\eta} - \boldsymbol{\eta}_d = \begin{bmatrix} x^n - x_d^n \\ y^n - y_d^n \\ \psi - \psi_d \end{bmatrix}$$

Trajectory-tracking control laws are classified according to the number of available actuators.

Classification of Trajectory-Tracking Controllers:

- **Three or more controls:** This is referred to as a fully actuated dynamic positioning (DP) system where the configuration space is chosen as *surge, sway* and *yaw.* Typical applications are crab-wise motions (low-speed maneuvering) and stationkeeping where the goal is to drive the trajectory tracking error $e(t) = [e_x(t), e_y(t), e_\psi(t)]^T \rightarrow \mathbf{0}$.

This is the standard configuration for offshore DP vessels.

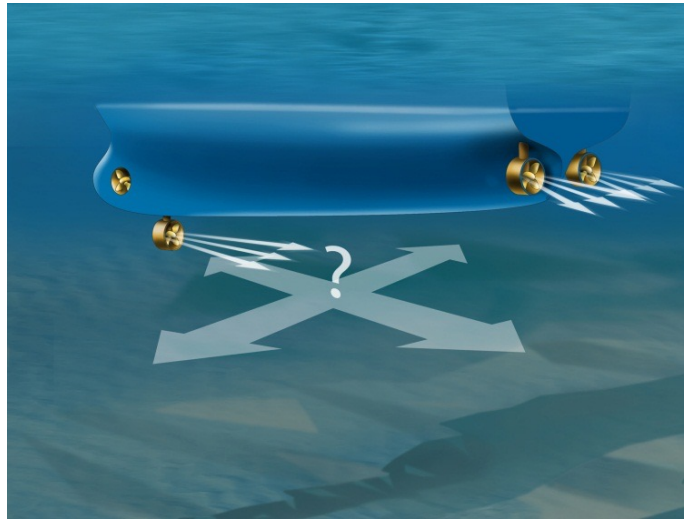
12.1 Trajectory Tracking

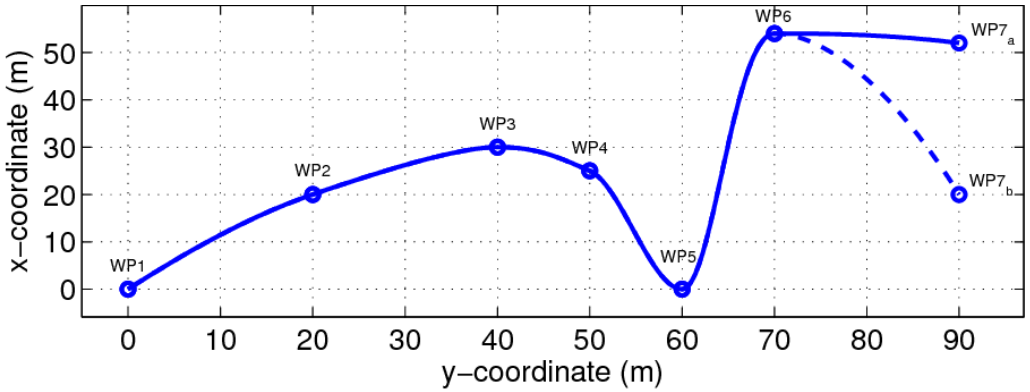
- **Two controls and trajectory tracking:** Trajectory-tracking in 3 DOFs, $\boldsymbol{e(t)} \in \mathbb{R}^3$, with only two controls, $\boldsymbol{u(t)} \in \mathbb{R}^2$, is an underactuated control problem which cannot be solved using linear theory. This problem has limited practical use if any!

- **Two controls and weather-optimal heading:**
If the ship is aligned up against the mean resulting force due to wind, waves and ocean currents a weathervaning controller can be designed such that only two controls, $\boldsymbol{u(t)} \in \mathbb{R}^2$, are needed to stabilize the ship in 3 DOFs. In this approach, the heading angle is left uncontrolled and allowed to vary automatically with the mean environmental forces.

- **Two controls and path following:** It is standard procedure to define a 2-D workspace (along-track and cross-track errors) and minimize the cross-track error by means of a LOS path-following controller. Hence, it is possible to follow a path by using only two controls (surge speed and yaw moment). For a conventional ship this is achieved by using a rudder and a propeller only. Since the input and output vectors are of dimension two, the 6-DOF system model must be internally stable.

- **One control:** It is impossible to design stationkeeping systems and trajectory-tracking control systems in 3 DOFs for a marine craft using only one control.





2-D path generated using waypoints in the xy plane

12.1.1 Reference Models for Trajectory Generation

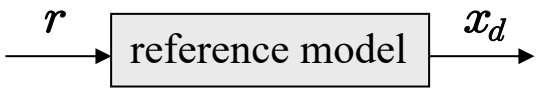
Simplest form is to use a *low-pass (LP) filter*

$$\frac{x_d}{r}(s) = h_{lp}(s)$$

where r is the command and x_d is the desired state

Choice of filter should reflect the dynamics of the craft feasible trajectory

- Speed limitations
- Acceleration limitations



Bandwidth of the reference model $<$ bandwidth of the vessel motion control system

12.1.1 Reference Models for Trajectory Generation

For marine craft it is convenient to use reference models motivated by the dynamics of *mass-damper-spring systems* to generate desired velocities

$$\frac{\nu_{d_i}}{r_i^b}(s) = \frac{\omega_{n_i}^2}{s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2}$$

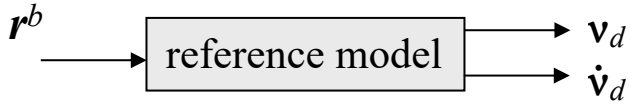
$\zeta_i \ (i = 1, \ldots, n)$ relative damping ratio
$\omega_{n_i} \ (i = 1, \ldots, n)$ natural frequency

This model can be written as a *MIMO mass-damper-spring system*

$$\dot{\boldsymbol{x}}_d = \boldsymbol{A}_d\boldsymbol{x}_d + \boldsymbol{B}_d\boldsymbol{r}^b \qquad \boldsymbol{x}_d := [\boldsymbol{\nu}_d^\top, \dot{\boldsymbol{\nu}}_d^\top]^\top \in \mathbb{R}^{2n}$$

$$\boldsymbol{A}_d = \begin{bmatrix} \boldsymbol{0}_{n\times n} & \boldsymbol{I}_n \\ -\boldsymbol{\Omega}^2 & -2\boldsymbol{\Delta}\boldsymbol{\Omega} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} \boldsymbol{0}_{n\times n} \\ \boldsymbol{\Omega}^2 \end{bmatrix}$$

$$\boldsymbol{\Delta} = \mathrm{diag}\{\zeta_1, \zeta_2, \ldots, \zeta_n\}$$
$$\boldsymbol{\Omega} = \mathrm{diag}\{\omega_{n_1}, \omega_{n_2}, \ldots, \omega_{n_n}\}$$



$$\lim_{t\to\infty} \boldsymbol{\nu}_d = \boldsymbol{r}^b$$

Note that a step in the command $\boldsymbol{r}^b$ will give a step in $\dot{\boldsymbol{\nu}}_d$

12.1.1 Reference Models for Trajectory Generation

The position and attitude reference model $\boldsymbol{\eta}_d$ is typically chosen of third order for filtering of steps in the command $\mathbf{r}^n$

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^2}{(1+T_i s)\left(s^2 + 2\zeta_i \omega_{n_i} s + \omega_{n_i}^2\right)}$$

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^3}{s^3 + (2\zeta_i + 1)\omega_{n_i} s^2 + (2\zeta_i + 1)\omega_{n_i}^2 s + \omega_{n_i}^3}$$

State-space representation

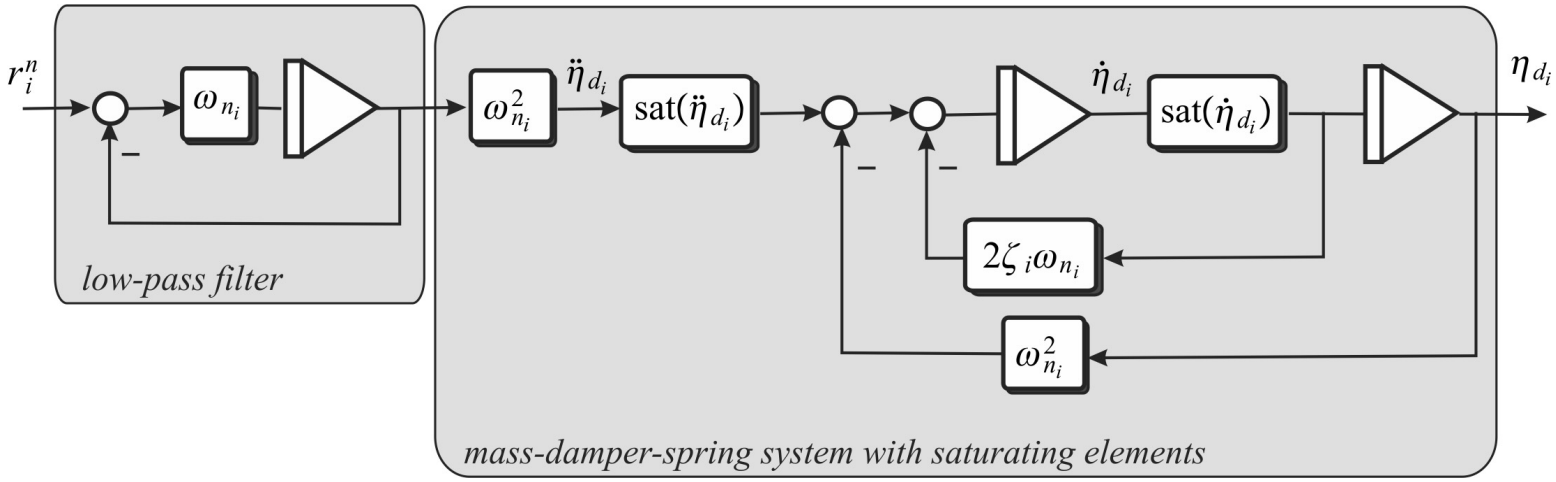
$$\dot{\boldsymbol{x}}_d = \boldsymbol{A}_d \boldsymbol{x}_d + \boldsymbol{B}_d \boldsymbol{r}^n \qquad \boldsymbol{x}_d := [\boldsymbol{\eta}_d^\top, \dot{\boldsymbol{\eta}}_d^\top, \ddot{\boldsymbol{\eta}}_d^\top]^\top \in \mathbb{R}^{3n}$$

$$\boldsymbol{A}_d = \begin{bmatrix} \mathbf{0}_{n\times n} & \boldsymbol{I}_n & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & \mathbf{0}_{n\times n} & \boldsymbol{I}_n \\ -\boldsymbol{\Omega}^3 & -(2\boldsymbol{\Delta} + \boldsymbol{I}_n)\boldsymbol{\Omega}^2 & -(2\boldsymbol{\Delta} + \boldsymbol{I}_n)\boldsymbol{\Omega} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} \\ \boldsymbol{\Omega}^3 \end{bmatrix}$$

$$\lim_{t\to\infty} \boldsymbol{\eta}_d = \boldsymbol{r}^n$$

12.1.1 Reference Models for Trajectory Generation

One *drawback* with a *linear reference model* is that the time constants in the model often yields a satisfactory response for one operating point of the system while the response for other amplitudes of the operator input r_i results in a completely different behavior. This is due to the exponential convergence of the signals in a linear system.



The performance of the linear reference model can also be improved by including saturation elements for *velocity* and *acceleration*

$$\text{sat}(x) = \begin{cases} \text{sgn}(x)x^{\max} & \text{if } |x| \geq x^{\max} \\ x & \text{else} \end{cases}$$

12.1.1 Reference Models for Trajectory Generation

Nonlinear damping can also be included in the reference model to reduce the velocity for large amplitudes or step inputs r^n. This suggests the modification

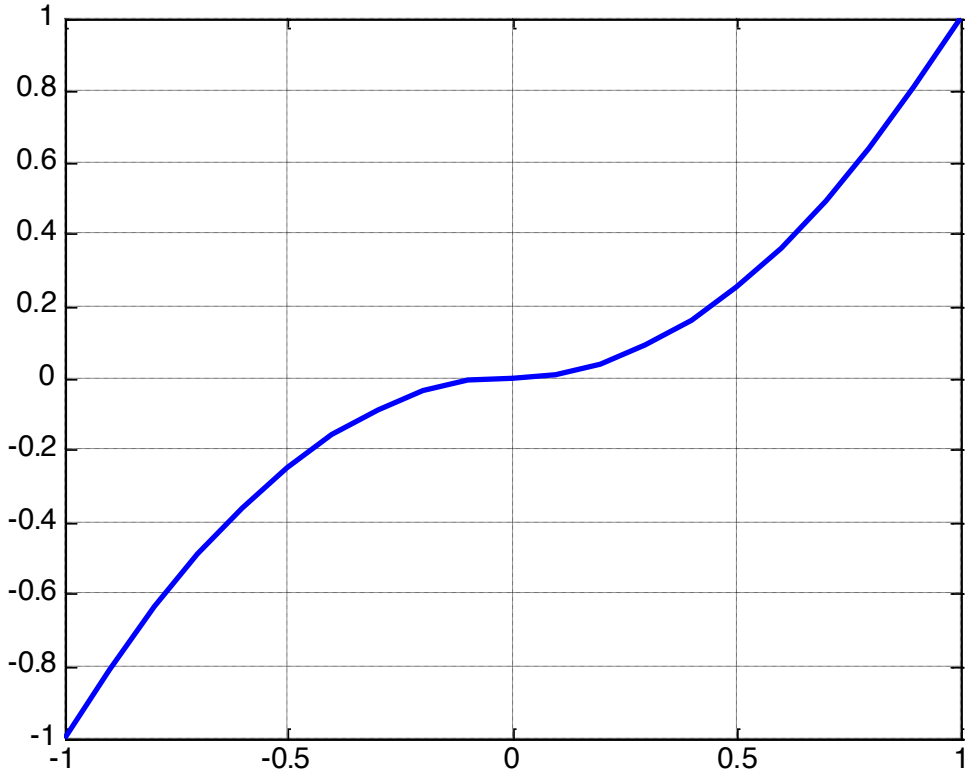
$$\boldsymbol{\eta}_d^{(3)} + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \mathbf{d}(\dot{\boldsymbol{\eta}}_d) + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\mathbf{r}^n$$

where the nonlinear function could be chosen as:

$$d_i(\dot{\eta}_{d_i}) = \sum_j \delta_{ij}|\dot{\eta}_{d_i}|^{p_j}\dot{\eta}_{d_i}, \quad (i = 1, \ldots, n)$$

where $\delta_{ij} > 0$ are design parameters

and $p_j > 0$ are some integers.



12.1.1 Reference Models for Trajectory Generation

Example: Reference Model Consider the *mass-damper-spring* reference model

$$\dot{\eta}_d = \nu_d$$

$$\dot{\nu}_d + 2\zeta\omega_n\nu_d + \delta\,|\nu_d|\,\nu_d + \omega_n^2\eta_d = \omega_n^2 r$$

$$\zeta = \omega_n = 1$$

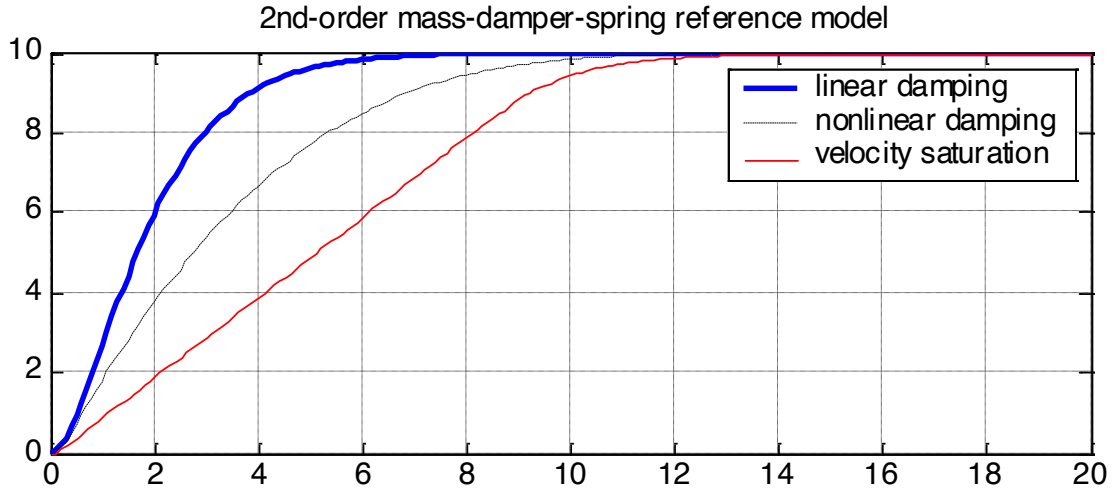
In the figure, we compare responses using:

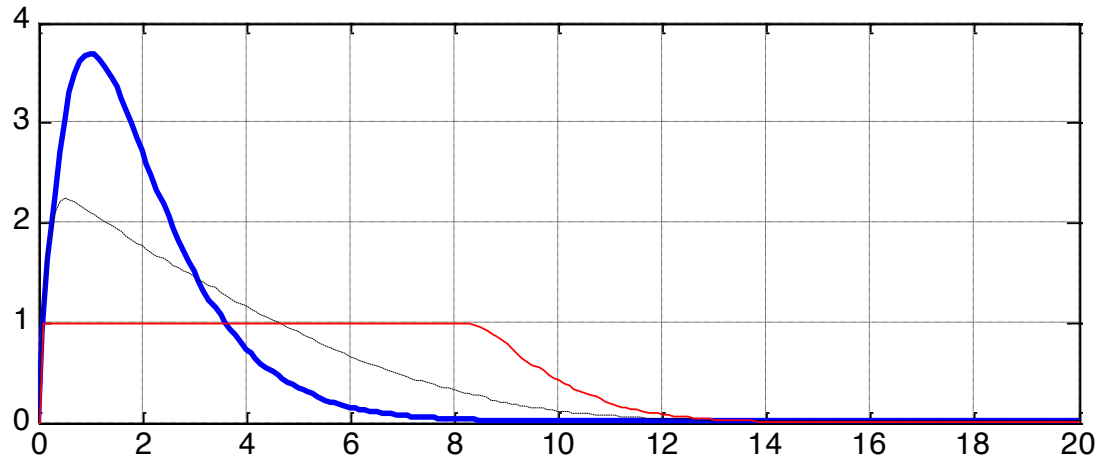
$$\delta = 0, \delta = 1$$

and a saturating element:

$$v_{\max} = 1$$

for an operator step input *r = 1*.



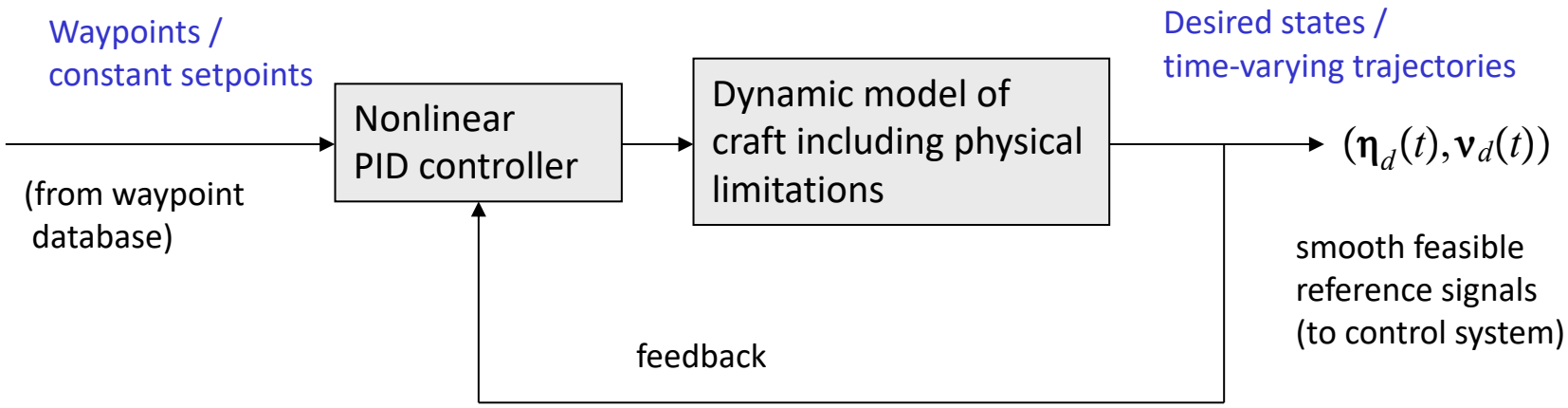


Desired position and velocity for a step input *r = 10*.

MSS Toolbox: ExRefMod.m

12.1.2 Trajectory Generation using a Marine Craft Simulator

Guidance System: dynamic model + guidance controller



Saturation elements for velocity and acceleration should be included to keep these quantities within their physical limits.

A switching strategy between the set-points (waypoints) must also be adopted.

12.1.2 Trajectory Generation using a Marine Craft Simulator

Generate a time-varying reference trajectory using a closed-loop model of the craft where the time constants, relative damping ratios and natural frequencies are chosen to reflect physical limitations of the craft.

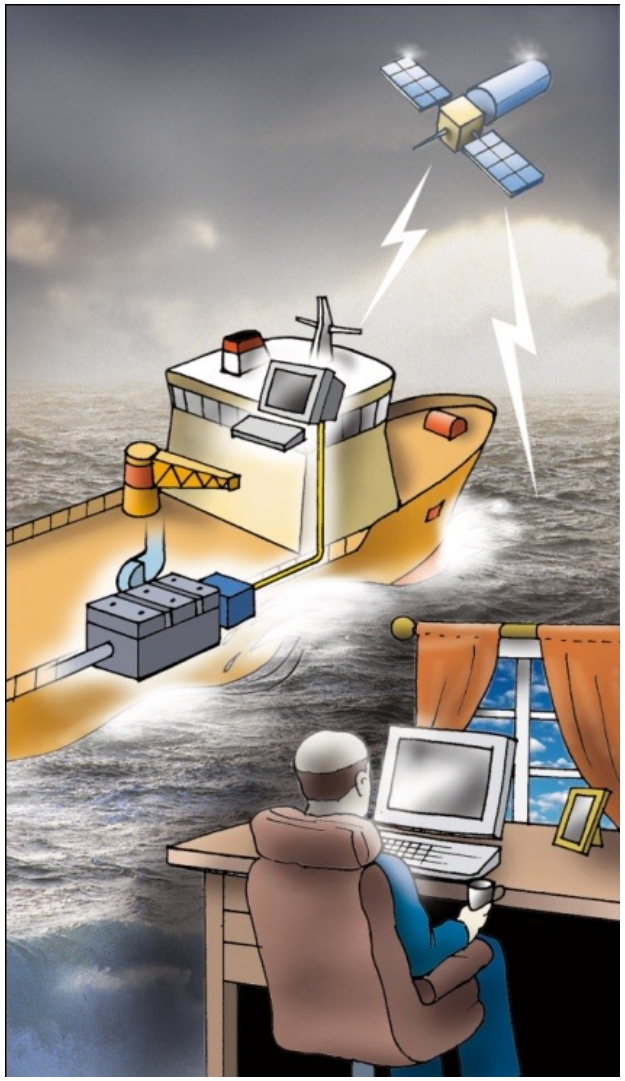
$$\dot{\boldsymbol{\eta}}_d = \boldsymbol{J}_\Theta(\boldsymbol{\eta}_d)\boldsymbol{\nu}_d$$
$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_d + \boldsymbol{D}\boldsymbol{\nu}_d + \boldsymbol{g}(\boldsymbol{\eta}_d) = \boldsymbol{\tau}$$

$$\boldsymbol{D} = \mathrm{diag}\{d_1, \ldots, d_6\} > 0$$

$$\boldsymbol{\tau} = \boldsymbol{g}(\boldsymbol{\eta}_d) - \boldsymbol{J}_\Theta^\top(\boldsymbol{\eta}_d)\left(\boldsymbol{K}_p(\boldsymbol{\eta}_d - \boldsymbol{\eta}_{\mathrm{ref}}) + \boldsymbol{K}_d\dot{\boldsymbol{\eta}}_d\right)$$

where $\boldsymbol{\eta}_{ref}$ is the desired reference and $(\boldsymbol{\eta}_d, \boldsymbol{v}_d)$ represents the desired states.
The PD control law is a guidance controller since it is applied to the reference model.

In addition to this, it is smart to include saturation elements for velocity and acceleration to keep these quantities within their physical limits.



Copyright © Bjarne Stenberg

12.1.2 Trajectory Generation using a Marine Craft Simulator

Example: Generation of Reference Trajectories using a Marine Craft Simulator

The kinematics of a ship can be modeled as

$$\dot{x}_d^n = u_d \cos(\psi_d)$$
$$\dot{y}_d^n = u_d \sin(\psi_d)$$

with speed dynamics

$$(m - X_{\dot{u}})\dot{u}_d + \frac{1}{2}\rho C_d A |u_d| u_d = T_{|n|n} |n| n$$

mass + quadratic drag formula

The yaw dynamics is chosen as

$$\dot{\psi}_d = r_d$$
$$T\dot{r}_d + r_d = K\delta$$

Nomoto model

where *K* and *T* are design parameters.

The guidance system has two inputs, *thrust* and *rudder angle*

$$\tau = -K_{p_u}(u_d - u_{ref}) - K_{i_u}\int_0^t (u_d - u_{ref})\mathrm{d}\tau, \qquad n = \mathrm{sgn}(\tau)\sqrt{|\tau|}$$
$$\delta = -K_{p_\psi}(\psi_d - \psi_{ref}) - K_{i_\psi}\int_0^t (\psi_d - \psi_{ref})\mathrm{d}\tau - K_{d_\psi} r_d$$

$$\psi_{ref} = \mathrm{atan2}\left(y_{los} - y_d,\ x_{los} - x_d\right)$$

Numerical integration of the ODEs with feedback yields a smooth reference trajectory

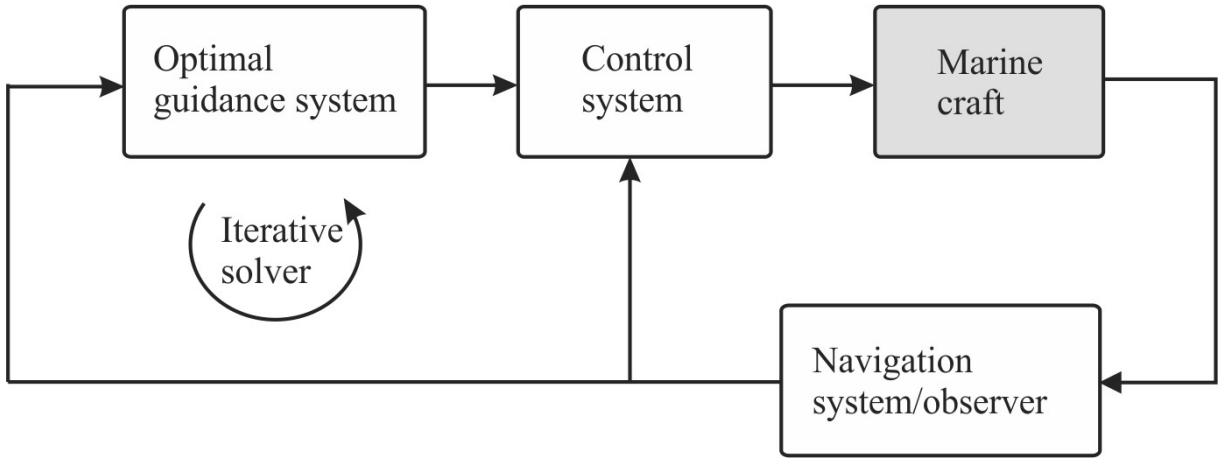
12.1.3 Optimal Trajectory Generation

Optimization methods can be used for trajectory and path generation. This gives a systematic method for inclusion of static and dynamic constraints. However, the price to be paid is that an optimization problem must be solved on-line in order to generate a feasible time-varying trajectory.

Implementation and solution of optimization problems can be done using linear programming (LP), quadratic programming (QP) and nonlinear methods. All these methods require that you have a solver that can be implemented in your program.

$$J = \min_{\boldsymbol{\eta}_d, \boldsymbol{\nu}_d} \{\text{power, time}\}$$

$U \leq U^{\max}$ (maximum speed)
$|r| \leq r^{\max}$ (maximum turning rate)
$|u_i| \leq u_i^{\max}$ (saturating limit of control u_i)
$|\dot{u}_i| \leq \dot{u}_i^{\max}$ (saturating limit of rate $\dot{u}_i$)



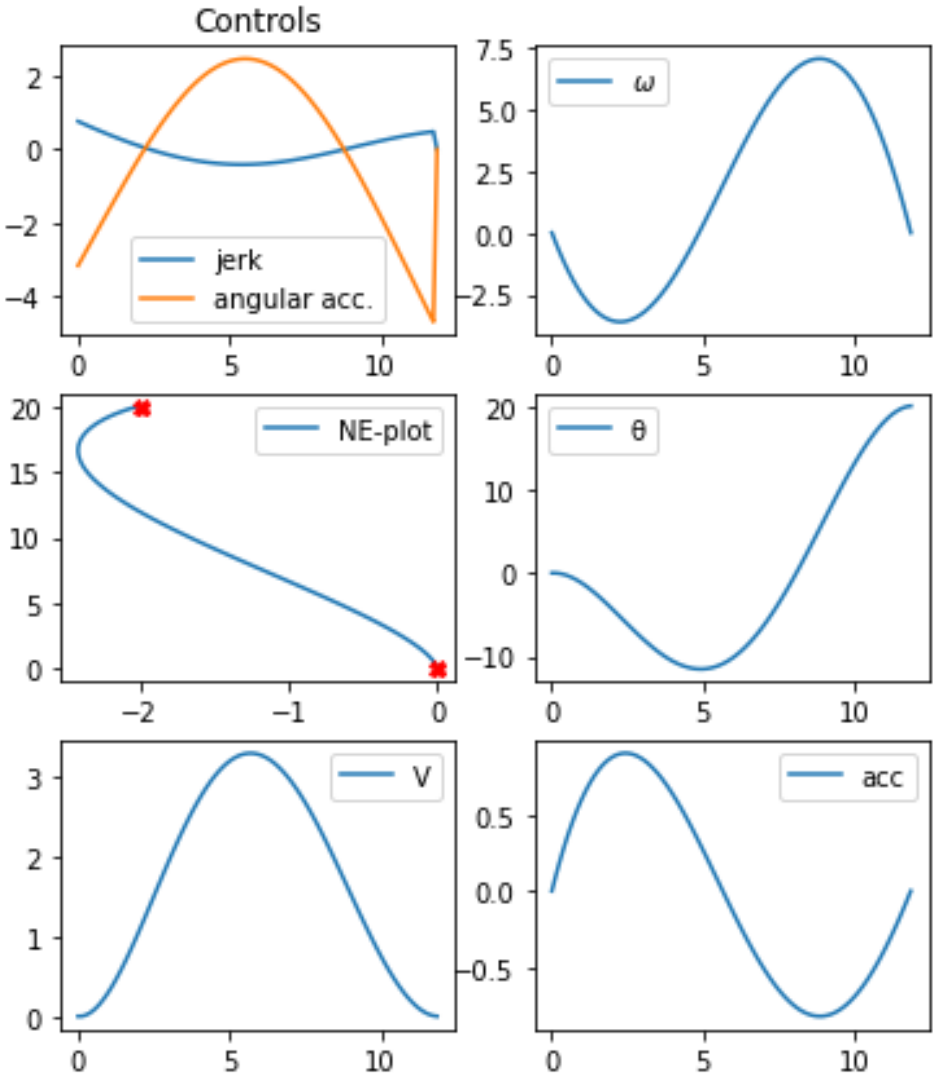
12.1.3 Path Generation using Dynamic Optimization

$$
\begin{aligned}
\dot{x} &= V\cos(\theta), & x(0) &= x_0 \\
\dot{y} &= V\sin(\theta), & y(0) &= y_0 \\
\dot{\theta} &= \omega, & \theta(0) &= \theta_0 \\
\dot{V} &= a, & V(0) &= V_0 \\
\dot{a} &= u_1, & \dot{a}(0) &= 0 \\
\dot{\omega} &= u_2, & \dot{\omega}(0) &= 0
\end{aligned}
$$

$$
\text{Minimize} \quad J = w_1 t_f + w_2 \int_0^{t_f} u_1^2\, dt + w_3 \int_0^{t_f} u_2^2\, dt
$$

Subject to

$$
\begin{aligned}
x(t_f) &= x_{\text{final}} \\
y(t_f) &= y_{\text{final}} \\
\theta(t_f) &= \theta_{\text{final}} \\
V(t_f) &= V_{\text{final}} \\
\mathrm{a}(t_f) &= \mathrm{a}_{\text{final}} \\
\omega(t_f) &= \omega_{\text{final}} \\
0 \le V &\le V_{\max}
\end{aligned}
$$



https://web.casadi.org

12.2 Guidance Laws for Target Tracking

Guided missiles have been operational since World War II, so organized research on guidance theory has been conducted almost if organized research on control theory.

Consequently, the most rich and mature literature on guidance is probably found within the guided missile community.

U.S. Air Force MQ-9A Reaper
From Wikipedia, the free Encyclopaedia

ADM-141C Improved Tactical Air Launched Decoy (ITALD)
https://www.navair.navy.mil/product/TALDITALD

Guidance laws for target tracking can be used in marine operations such as underway replenishment (UNREP) operations and formation control. UNREP operations involve cargo transfer between two or more cooperating surface craft in transit. The task of the so-called guide ship is to maintain a steady course and speed while the approach ship moves up alongside the guide or target ship to receive fuel, munitions and personnel.

12.2 Guidance Laws for Target Tracking

An intercepting ship can control its speed U and course angle χ using a conventional autopilot system.

Kinematic transformation: Desired speed/course angle to North-East velocities

$$u_d^n = U_d \cos(\chi_d)$$
$$v_d^n = U_d \sin(\chi_d)$$

$$\Longleftrightarrow$$

$$\chi_d = \text{atan2}(v_d^n,\, u_d^n)$$
$$U_d = \sqrt{(u_d^n)^2 + (v_d^n)^2}$$

This suggests that the desired velocities (u_d, v_d) obtained by a target tracking guidance law can be transformed to autopilot reference signals (χ_d , U_d) to achieve target tracking.

It is convenient to distinguish between maneuvering (accelerating) and nonmaneuvering (nonaccelerating) objects. An interceptor typically undergoes three operational phases; launch, midcourse and terminal.

We will consider three terminal guidance strategies:

- Line-of-Sight (LOS) Guidance
- Pure Pursuit (PP) Guidance
- Constant Bearing (CB) Guidance

12.2 Guidance Laws for Target Tracking

For surface craft, the 2-D position and velocity of the target is denoted by

$$\boldsymbol{p}_t^n = [x_t^n, y_t^n]^\top$$

$$\boldsymbol{v}_t^n = \dot{\boldsymbol{p}}_t^n = [u_t^n, v_t^n]^\top \qquad U_t = \|\boldsymbol{v}_t^n\| = \sqrt{(u_t^n)^2 + (v_t^n)^2}$$

The control objective of a target-tracking scenario can be formulated as

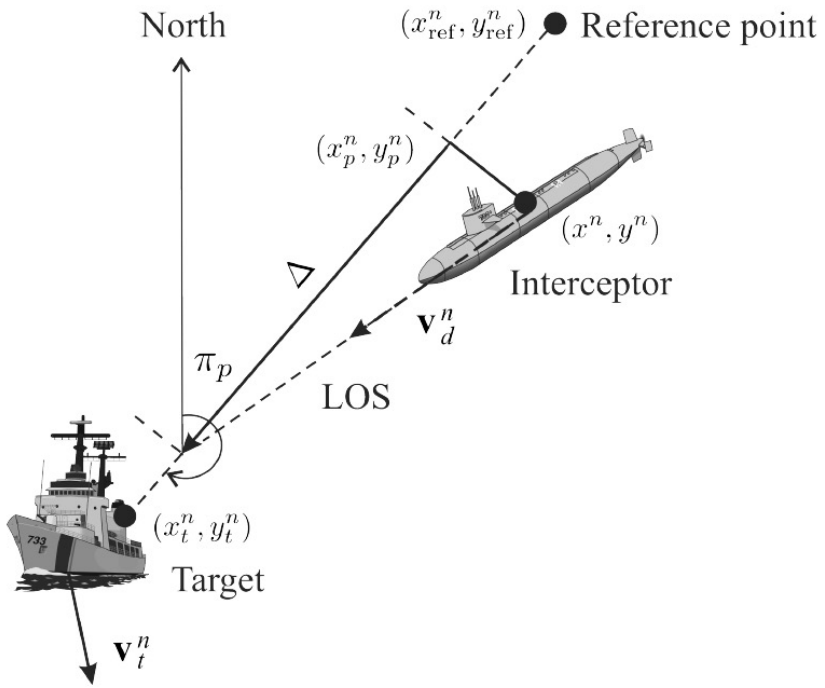
$$\lim_{t \to \infty} (\boldsymbol{p}^n - \boldsymbol{p}_t^n) = \boldsymbol{0}$$

The guidance laws are based on the following measurements

- Target position
- Target velocity (optional)

The target velocity can also be estimated by numerical differentiation of the target position.

12.2.1 Line-of-Sight Guidance Law



- 3-point guidance scheme
- The interceptor must constrain its motion along the reference-target line of sight
- Typically employed for surface-to-air missiles (beam-rider guidance)

The interceptor (approach) velocity is pointed to the LOS vector to obtain the desired velocity $\mathbf{v}_d^n$

$$u_d^n = U_d \cos(\chi_d)$$
$$v_d^n = U_d \sin(\chi_d)$$

Path-tangential angle

$$\pi_p = \text{atan2}(y_t^n - y_{\text{ref}}^n,\ x_t^n - x_{\text{ref}}^n)$$

Desired course angle

$$\chi_d = \pi_p - \tan^{-1}\left(\frac{y_e^p}{\Delta}\right)$$

$\Delta > 0$ (lookahead distance)

These equations have three unknowns:

- intersection point $(x_p^n,\ y_p^n)$
- cross-track error y_e^p

See next page for geometric solution

12.2.1 Line-of-Sight Guidance Law

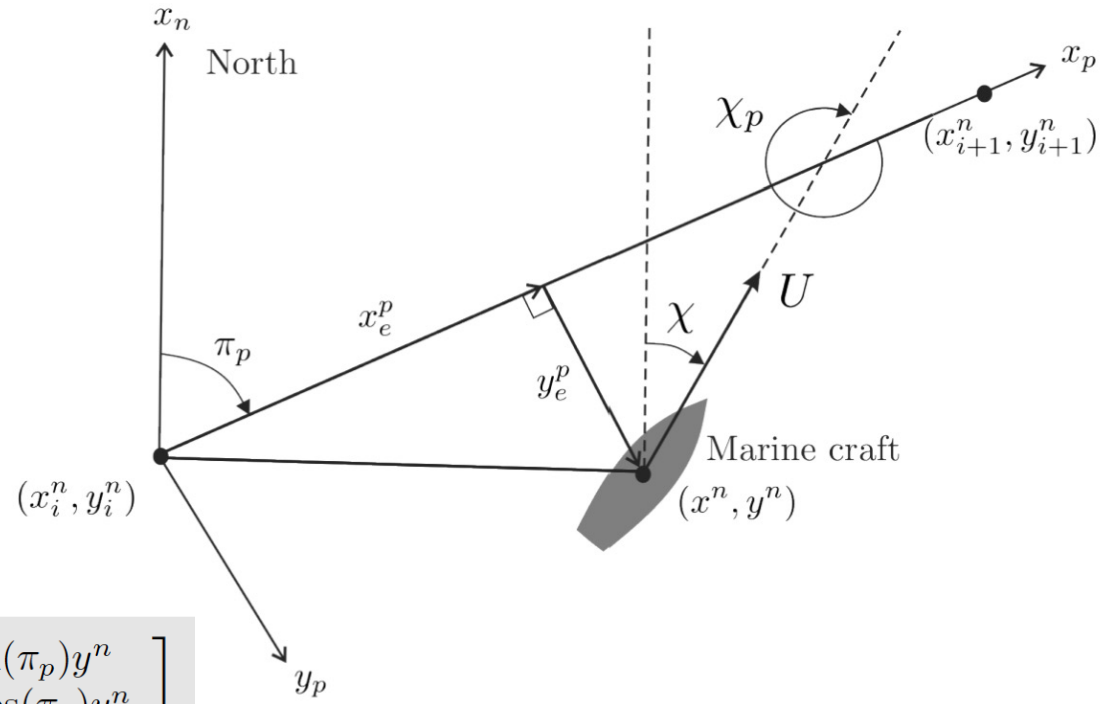
Computation of the cross-track error

For a craft located at the position (x^n, y^n), the along-track and cross-track errors $(x_e^p,\ y_e^p)$ are computed as the orthogonal distance to the path-tangential reference frame by setting $x_e^p \equiv 0$

$$\begin{bmatrix} 0 \\ y_e^p \end{bmatrix} = \boldsymbol{R}_p^n(\pi_p)^\top \left(\begin{bmatrix} x^n \\ y^n \end{bmatrix} - \begin{bmatrix} x_p^n \\ y_p^n \end{bmatrix} \right)$$

$$\tan(\pi_p) = \frac{y_t^n - y_p^n}{x_t^n - x_p^n}$$

$$\boldsymbol{R}_p^n(\pi_p) = \begin{bmatrix} \cos(\pi_p) & -\sin(\pi_p) \\ \sin(\pi_p) & \cos(\pi_p) \end{bmatrix} \in SO(2)$$



Solution:

$$\underbrace{\begin{bmatrix} \cos(\pi_p) & \sin(\pi_p) & 0 \\ -\sin(\pi_p) & \cos(\pi_p) & 1 \\ \tan(\pi_p) & -1 & 0 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_p^n \\ y_p^n \\ y_e^p \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \cos(\pi_p)x^n + \sin(\pi_p)y^n \\ -\sin(\pi_p)x^n + \cos(\pi_p)y^n \\ \tan(\pi_p)x_t^n - y_t^n \end{bmatrix}}_{\boldsymbol{b}}$$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

MSS Toolbox

```
function [x_p, y_p, y_e] = crosstrack(x_t, y_t, x_ref, y_ref, x, y, flag)
%  [x_p, y_p, y_e] = crosstrack(x_t, y_t, x_ref, y_ref, x, y, flag) computes
% the coordinate origin (x_p, y_p) of the path-tangential reference fram from
% the reference point (x_ref, y_ref) to the target position (x_p, y_p), and
% cross-track error y_e for a craft at position (x, y).
%
% Input:    (x_t,y_t) North-East target positions
%           (x_ref,y_ref) North-East reference positions
%           (x,y) crfat North-East positions
%           flag (optionally) = 1 (plot geometry)
%
% Outputs:  (x_p,y_p) origin of path-tangential frame expressed in NED
%           y_e cross-track error
```

```
% path-tanegntial angle
pi_p = atan2(y_t-y_ref, x_t-x_ref);

cp = cos(pi_p);
sp = sin(pi_p);
tp = tan(pi_p);

% A * z = b  -->  x = inv(A) * b
A = [cp sp  0
    -sp cp 1
     tp -1 0 ];

b = [ cp*x + sp*y
     -sp*x + cp*y
      tp*x_t - y_t ];

z = inv(A) * b;
x_p = z(1);
y_p = z(2);
y_e = z(3);
```

$$\tan(\pi_p) = \frac{y_t^n - y_p^n}{x_t^n - x_p^n}$$

$$\underbrace{\begin{bmatrix} \cos(\pi_p) & \sin(\pi_p) & 0 \\ -\sin(\pi_p) & \cos(\pi_p) & 1 \\ \tan(\pi_p) & -1 & 0 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_p^n \\ y_p^n \\ y_e^p \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \cos(\pi_p)x^n + \sin(\pi_p)y^n \\ -\sin(\pi_p)x^n + \cos(\pi_p)y^n \\ \tan(\pi_p)x_t^n - y_t^n \end{bmatrix}}_{\boldsymbol{b}}$$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

MSS Toolbox

Computation of the cross-track error

Matlab:
The cross-track error can be computed using the MSS toolbox function

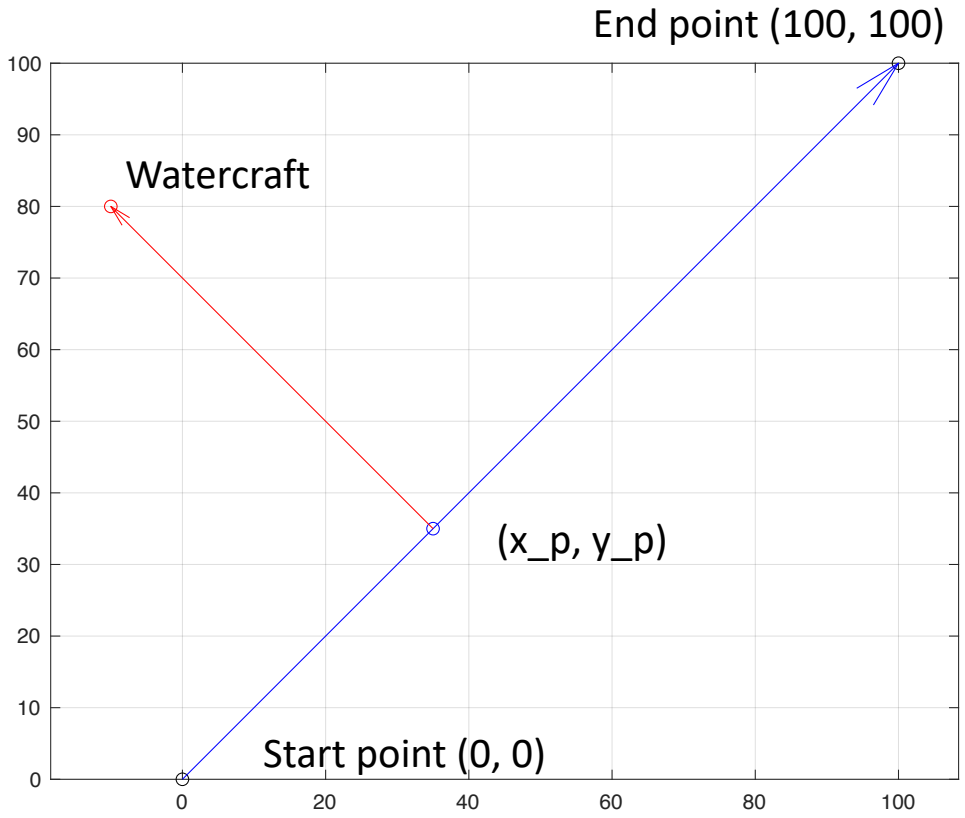
```
[x_p, y_p, y_e] = crosstrack(x_t, y_t, x_ref, y_ref, x, y, flag)
```

where `flag = 1` is an optional input for plotting the geometry of the problem.

```
>> [x_p, y_p, y_e] = crosstrack(100, 100, 0, 0, 80, -10, 1)

x_p =
   35.0000
y_p =
   35.0000
y_e =
  -63.6396
```

The cross-track error is -63.6396 m



Simplified version: compute the cross-track error for a path given by the waypoints (0,0) and (100, 100)

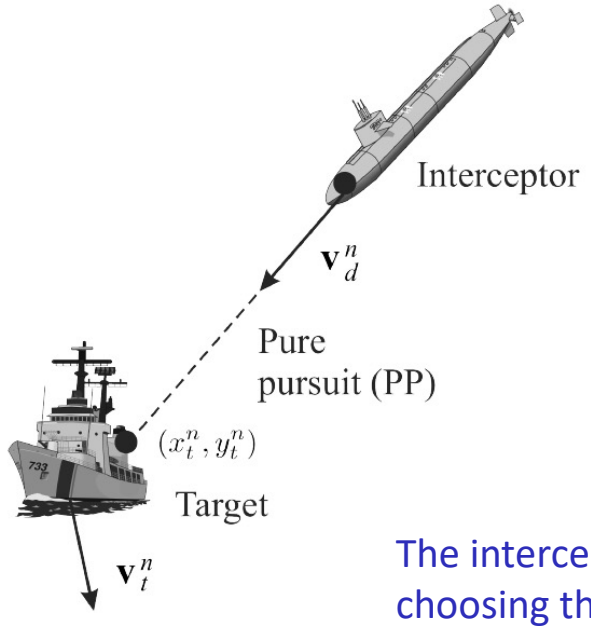
```
>> y_e = crosstrackWpt(100, 100, 0, 0, 80, -10)

y_e =
  -63.6396
```

```
% path-tanegntial angle
pi_p = atan2(y2-y1, x2-x1);

% cross-track error
y_e = -(x-x1) * sin(pi_p) + (y-y1) * cos(pi_p);
```

12.2.2 Pure Pursuit Guidance Law



- 2-point guidance scheme
- The interceptor must align its linear velocity along the interceptor-target line of sight
- Equivalent to a predator chasing a prey in the animal world
- Typically employed for air-to-surface missiles
- Deviated pursuit guidance is a variant of PP guidance (also called fixed-lead guidance)

The interceptor aligns its velocity along the LOS vector between the interceptor and the target by choosing the desired velocity as:

$$\boldsymbol{v}_d^n = [u_d^n,\, v_d^n]^\top$$

$$\boldsymbol{v}_d^n = -\kappa \frac{\tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|}$$

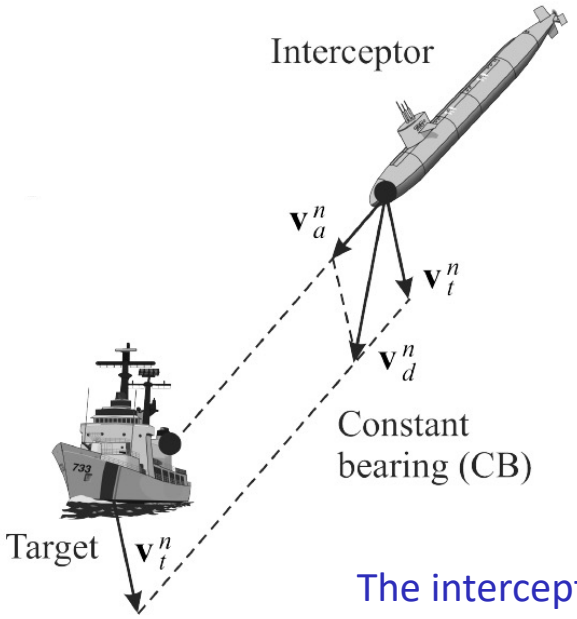
$$\tilde{\boldsymbol{p}}^n := \boldsymbol{p}^n - \boldsymbol{p}_t^n \qquad \kappa > 0$$

Autopilot commands:

$$\chi_d = \text{atan2}(v_d^n,\, u_d^n)$$

$$U_d = \sqrt{(u_d^n)^2 + (v_d^n)^2}$$

12.2.3 Constant Bearing Guidance Law



- 2-point guidance scheme
- Often referred to as parallel navigation
- Typically employed for air-to-air missiles
- Used for centuries by mariners to avoid collisions at sea
- Proportional navigation is the most common implementation method (optimal for scenarios involving nonmaneuvering targets)

The interceptor aligns the interceptor-target velocity along the LOS vector between the interceptor and the target.

$$\boldsymbol{v}_d^n = \boldsymbol{v}_t^n + \boldsymbol{v}_a^n$$
$$\boldsymbol{v}_a^n = -\kappa \frac{\tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|}$$

$$\kappa = U_a^{\max} \frac{\|\tilde{\boldsymbol{p}}^n\|}{\sqrt{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n + \Delta^2}}$$

12.2.3 Constant Bearing Guidance Law

Lyapunov function candidate (LFC)

$$V = \frac{1}{2}(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n > 0, \quad \forall \tilde{\boldsymbol{p}}^n \neq \mathbf{0}$$

Assume that the ship autopilot guarantees that $\boldsymbol{v}^n = \boldsymbol{v}_d^n$. Time differentiation of V along the trajectories of $\tilde{\boldsymbol{p}}^n = \boldsymbol{p}^n - \boldsymbol{p}_t^n$ gives

$$\begin{aligned}
\dot{V} &= (\tilde{\boldsymbol{p}}^n)^\top (\boldsymbol{v}^n - \boldsymbol{v}_t^n) \\
&= -\kappa \frac{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|} \\
&= -U_a^{\max} \frac{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n}{\sqrt{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n + \Delta^2}} \\
&< 0, \quad \forall \tilde{\boldsymbol{p}}^n \neq \mathbf{0}
\end{aligned}$$

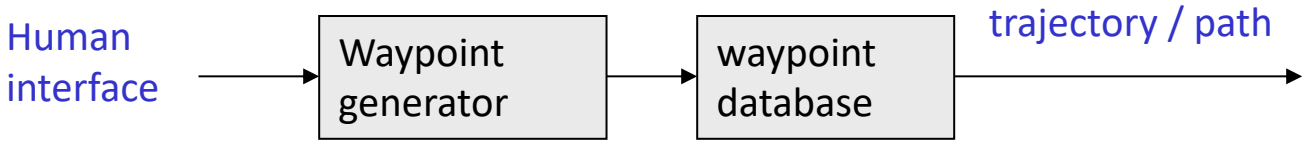
The LFC is positive definite and radially unbounded, while its derivative with respect to time is negative definite when adhering to

$$U \geq U_a^{\max} > 0$$

Hence, by standard Lyapunov arguments the origin is USGES as shown in **Appendix A.2.3**

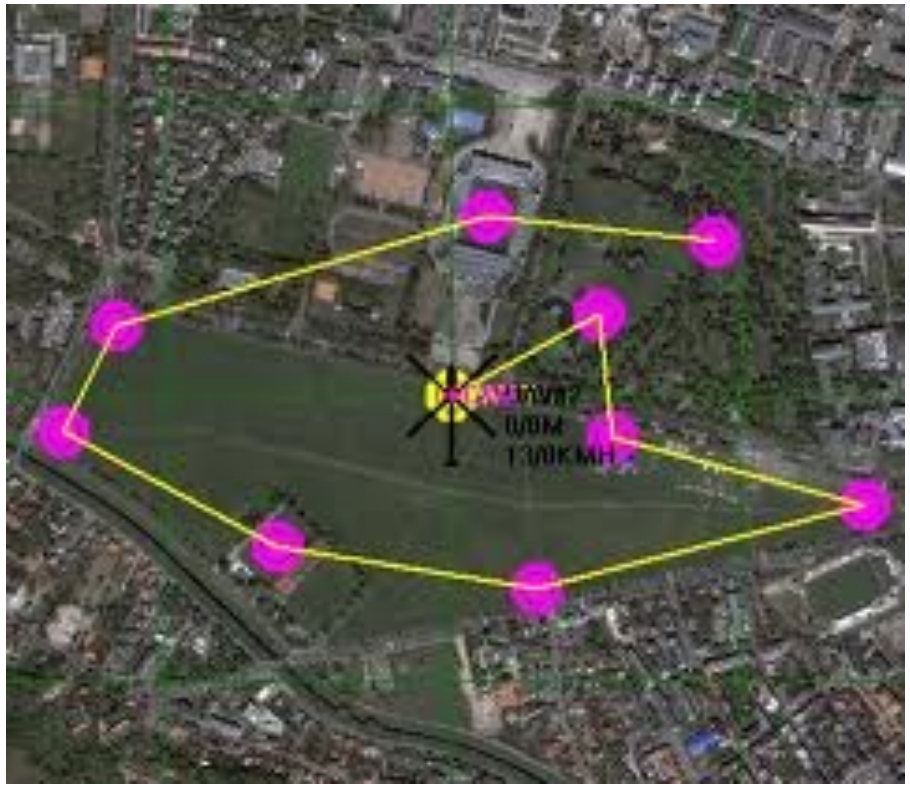
12.3 Linear Design Methods for Path Following

Waypoints are used to indicate changes in direction, speed and altitude along a desired path. Systems for waypoint generation are used both for ships and underwater vehicles. These systems consist of a waypoint generator with a human interface.



The selected waypoints are stored in a waypoint database and used for generation of a trajectory or a path for the moving craft to follow.

Sophisticated features like *weather routing*, *obstacle avoidance* and *mission planning* can be incorporated in the design of waypoint guidance systems.



Wikipedia:
A waypoint is an intermediate point or place on a route or line of travel, a stopping point or point at which course is changed.

12.3.1 Waypoints

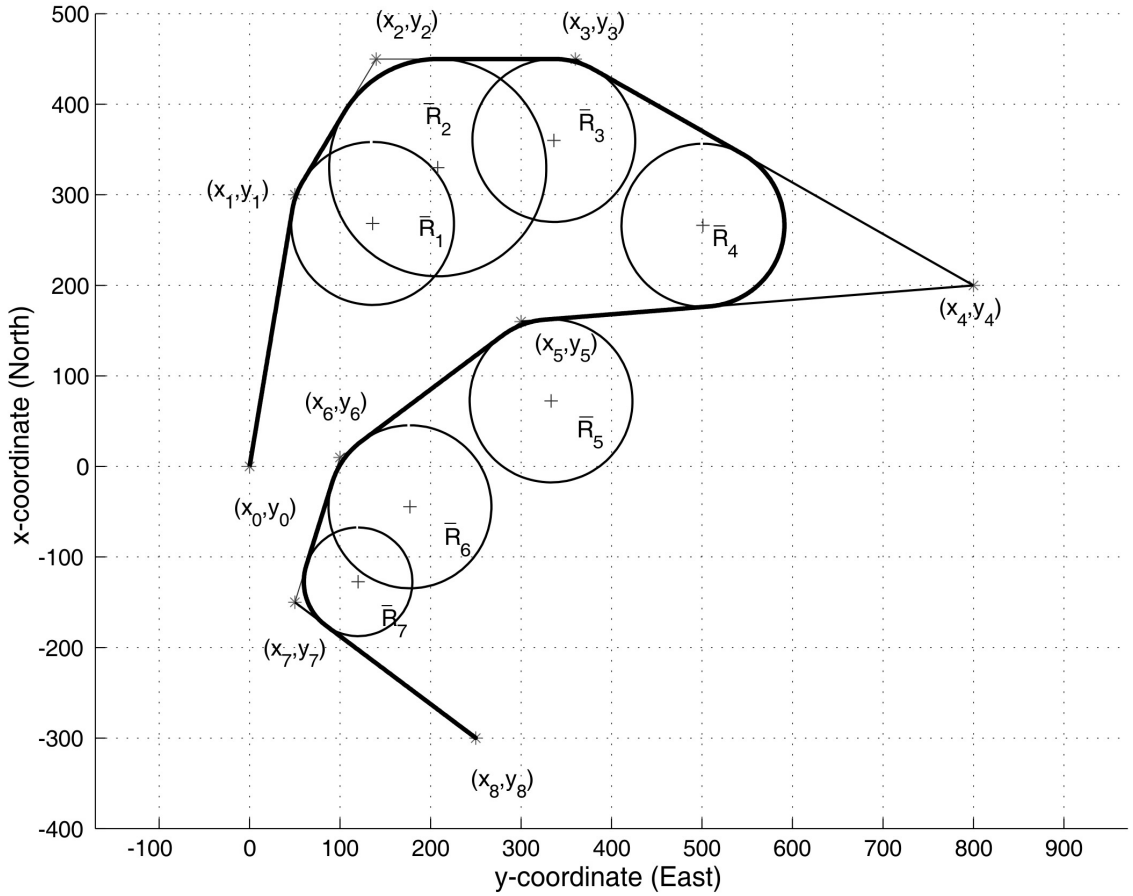
The route of a ship or an underwater vehicle is usually specified in terms of waypoints. Each waypoint is defined using Cartesian coordinates (x_i, y_i, z_i) for i=1,…,n. The *waypoint database* can be represented by a waypoint data structure:

$$\text{wpt.pos} = \{(x_0, y_0, z_0), (x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$$

and other properties such as

$$\text{wpt. speed} = \{U_0, U_1, \ldots, U_n\}$$
$$\text{wpt. heading} = \{\psi_0, \psi_1, \ldots, \psi_n\}$$

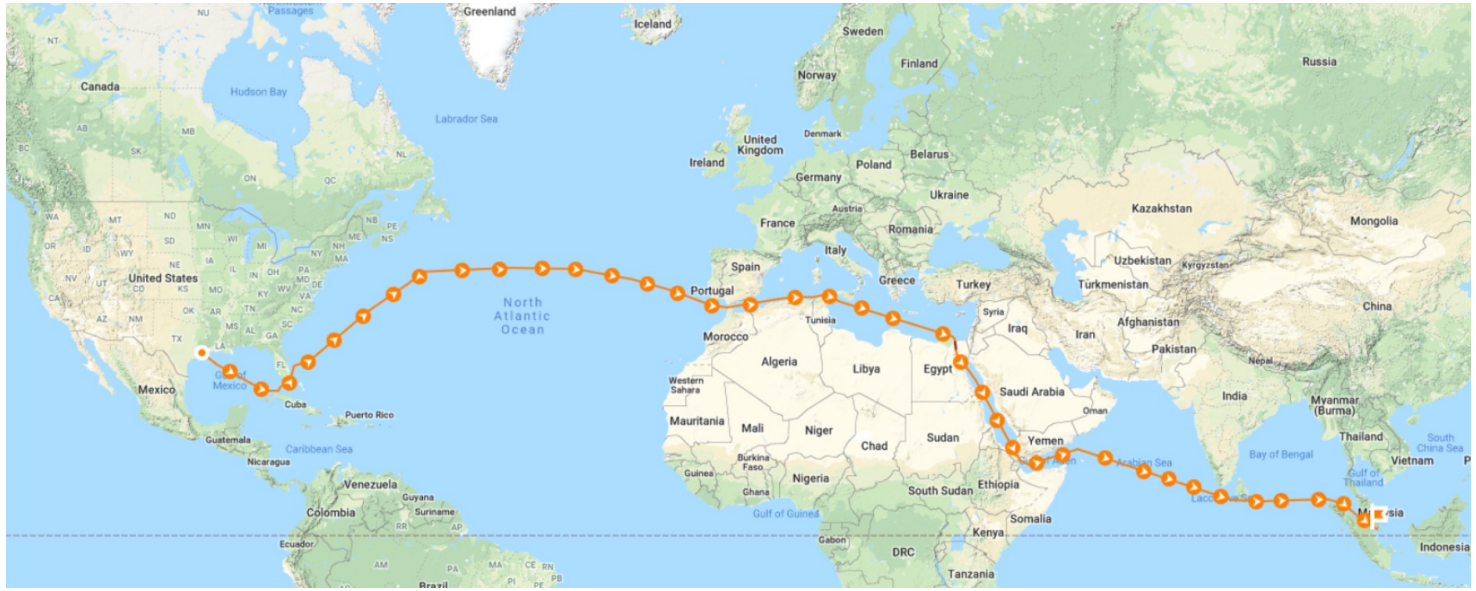
The three states (x_i, y_i, ψ_i) are also called the pose and they describe the start and end configurations of the craft given by two waypoints



12.3.1 Waypoints

The waypoint database can be generated using the following criteria:

- **Mission:** the vessel should move from some starting point (x_o, y_o, z_o) to the terminal point (x_n, y_n, z_n) via the waypoints (x_i, y_i, z_i).
- **Environmental data:** information about wind, waves, and currents can be used for energy optimal routing (or avoidance of bad weather for safety reasons)
- **Collision avoidance:** avoiding moving vessels close to your own route by introducing safety margins.
- **Feasibility:** each waypoint must be feasible; in that it must be possible to maneuver to the next waypoint without exceeding maximum speed and turning rate.
- **Geographical data:** information about shallow waters, islands etc. should be included.
- **Obstacles:** floating constructions and other obstacles must be avoided.



https://www.marinetraffic.com

12.3.2 Path Generation using Straight Lines and Inscribed Circles

In practice it is common to represent the desired path using **straight lines** and **circle arcs** to connect the waypoints (**Dubins path**). The famous result of Dubins (1957) states that:

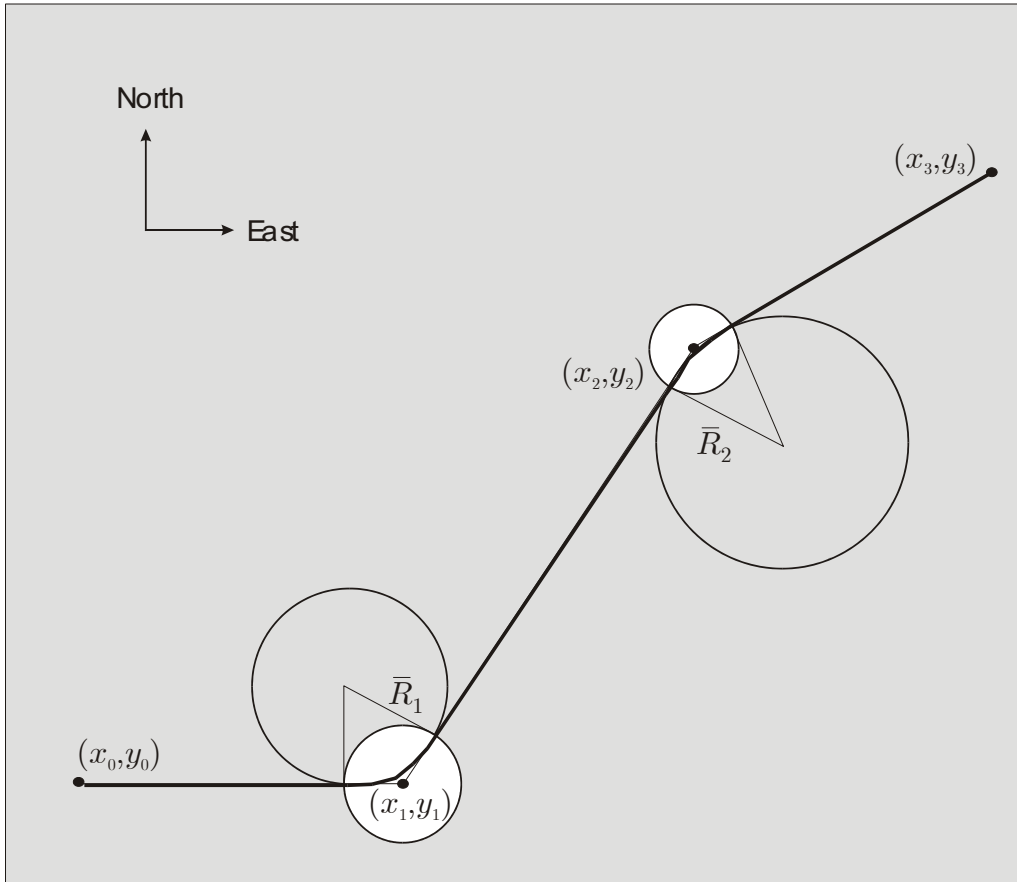
"The shortest path (minimum time) between two configurations (x^n, y^n, ψ) of a craft moving at constant speed U is a path formed by straight lines and circular arc segments"

The drawback, in comparison to a cubic interpolation strategy, is that a jump in the desired yaw rate r_d is experienced.

The desired yaw rate along the straight line is $r_d = 0$ while it is r_d = constant on the circle arc during steady turning. Hence, there will be a jump in the desired yaw rate during transition from the straight line to the circle arc.

The human operator usually specifies a circle with radius R_i around each waypoint (white circles)

$$\text{wpt. radius} = \{R_0, R_1, \ldots, R_n\}$$



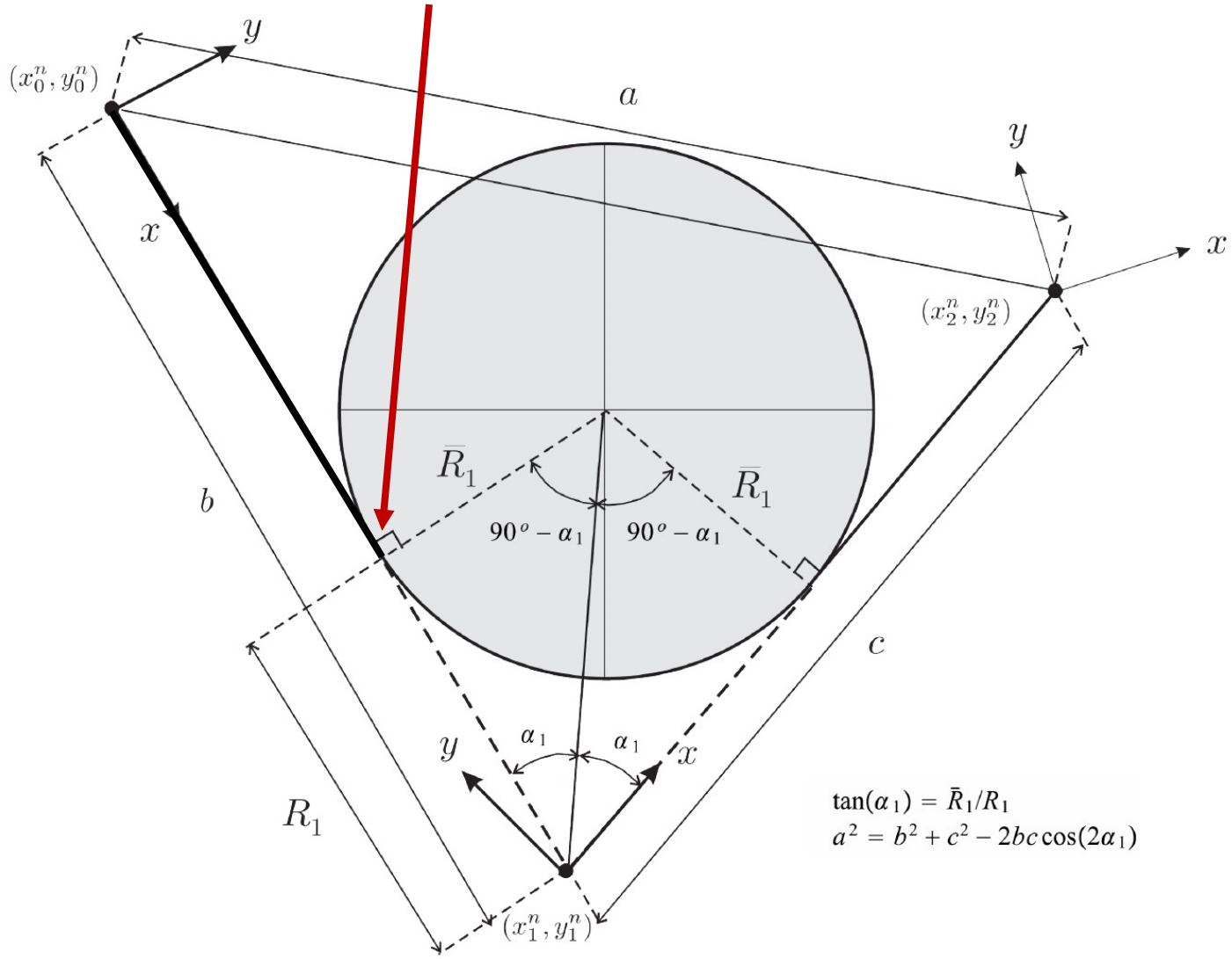
12.3.2 Path Generation using Straight Lines and Inscribed Circles

The point where the circle arc intersects the straight line represent the turning point of the ship.

The radius of the inscribed circle can be computed from R_i as

$$\bar{R}_i = R_i \tan\alpha_i$$

where α_i is defined in the figure.



12.3.3 Straight-Line Paths based on Circles of Acceptance

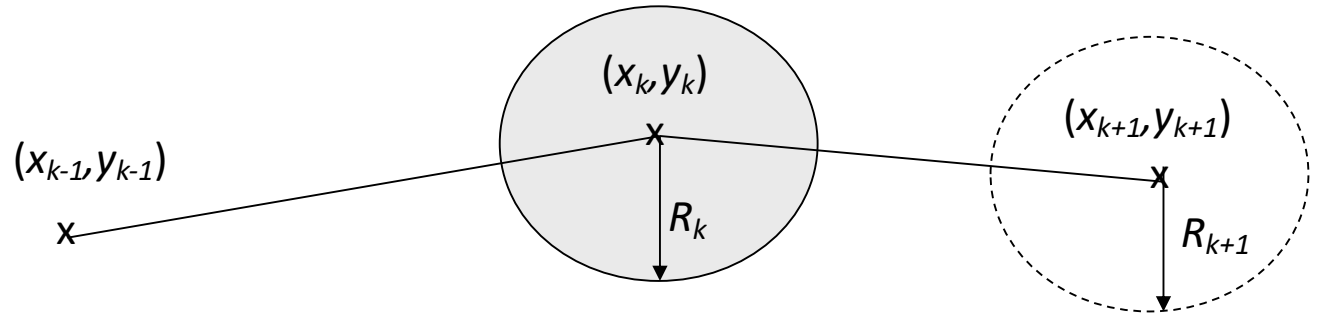
A more effective method for computer implementation is to drop the inscribed circles of Section 12.3.2 and only use straight-line segments going through the waypoints. When moving along a piecewise linear path made up of n-1 line segments connected by n waypoints, a switching mechanism for selecting the next waypoint is needed.

Circle of Acceptance for Surface Craft

When moving along the path a *switching mechanism for selecting the next waypoint* is needed. Waypoint (x_{i+1}, y_{i+1}) can be selected on a basis of whether the craft lies within a circle of acceptance with radius R_i around waypoint (x_i, y_i). If

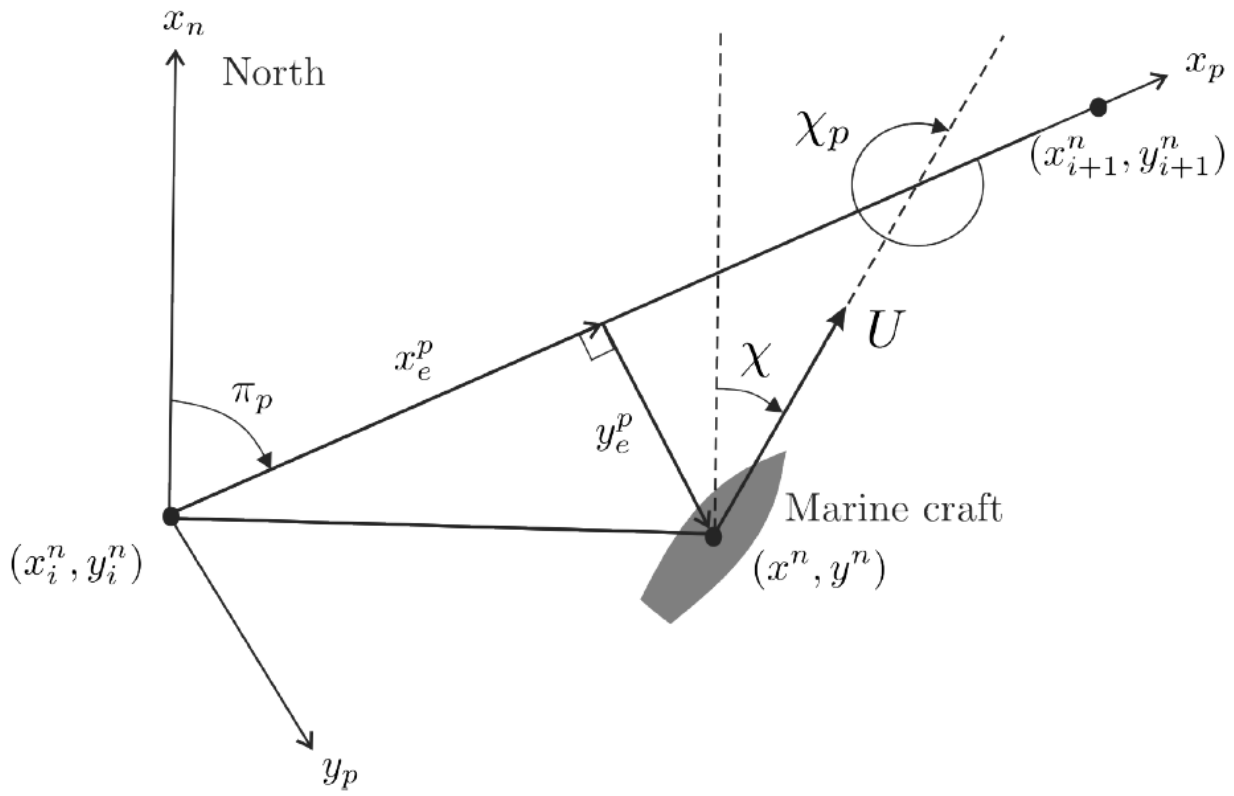
$$(x_{i+1}^n - x^n)^2 + (y_{i+1}^n - y^n)^2 \leq R_{i+1}^2$$

the next waypoint (x_{i+1}, y_{i+1}) should be selected (*i* should be incremented to *i=i+1*)
For ships, a guideline could be to choose R_{i+1} equal to two ship lengths, that is $R_{i+1} = 2L$.



If the craft position (x, y) is inside circle, choose next waypoint.

12.3.3 Straight-Line Paths based on Circles of Acceptance



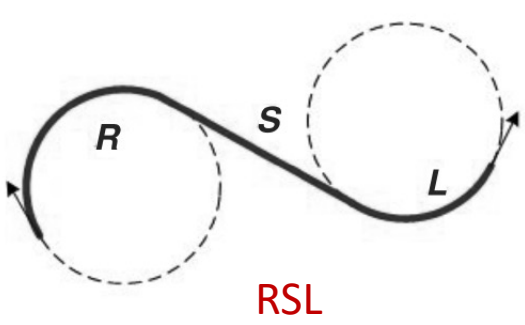
$$\pi_p = \text{atan2}(y^n_{i+1} - y^n_i,\, x^n_{i+1} - x^n_i)$$

$$\begin{bmatrix} x^p_e \\ y^p_e \end{bmatrix} = \boldsymbol{R}^n_p(\pi_p)^\top \left(\begin{bmatrix} x^n \\ y^n \end{bmatrix} - \begin{bmatrix} x^n_i \\ y^n_i \end{bmatrix} \right)$$

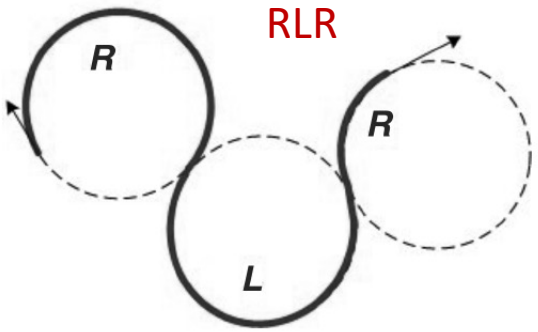
A perhaps more suitable switching criterion solely involves computation of the along-track distance, such that if the total along-track distance

$$d_{i+1} - |x^p_e| \leq R_{\text{switch}} \qquad R_{\text{switch}} > 0 \qquad d_{i+1} = \|\boldsymbol{p}^n_{i+1} - \boldsymbol{p}^n_i\|$$

12.3.3 Path Generation using Dubins Path



RSL



RLR

The desired yaw rate along the straight line is $r_d = 0$ while r_d = constant on the circle arc during steady turning.

The shortest path (minimum time) between two poses (x_i^n, y_i^n, ψ_i) and $(x_{i+1}^n, y_{i+1}^n, \psi_{i+1})$ *of a craft moving at constant speed* U *is a path formed by straight lines and circular arc segments.*

The drawback, in comparison to a cubic interpolation strategy, is that a jump in the desired yaw rate r_d is experienced.

The optimal path will be a combination of a right turn (R), left turn (L) or going straight (S). An optimal path will always be at least one of the six types: RSR, RSL, LSR, LSL, RLR, LRL.

The so-called Dubins path can also be proven by using Pontryagin's maximum principle.

In the case of time-varying speed, a dynamic optimization problem including the AUV dynamics must be solved.

L. E. Dubins (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents". American Journal of Mathematics. 79 (3): 497–516

12.3.5 Transfer Function Models for Straight-Line Path Following

Assume that the path-tangential coordinate system *{p}* has its origin at the waypoint $(x_i^n,\ y_i^n)$
The cross-track error expressed in *{p} satisfies*

$$\begin{aligned} \dot{y}_e^p &= U \sin(\chi_p) \\ &\approx U \chi_p \end{aligned}$$

GNSS course over ground (COG) measurement

$$\chi = \pi_p + \chi_p$$

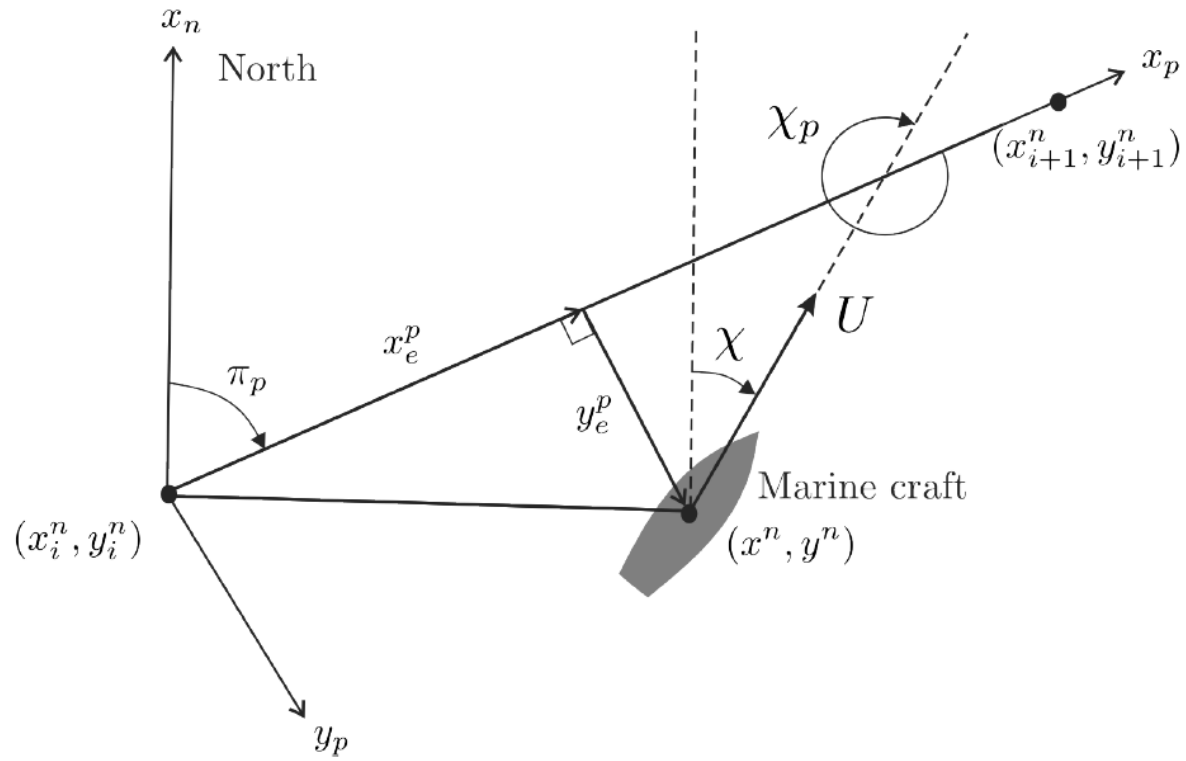
Transfer functions

$$y_e^p = \frac{U}{s}(\chi - \pi_p)$$

$$\psi = \frac{K}{s(Ts+1)}\delta + \frac{1}{s}d_r \qquad \text{Nomoto model}$$

Cross-track error transfer function for PID control design

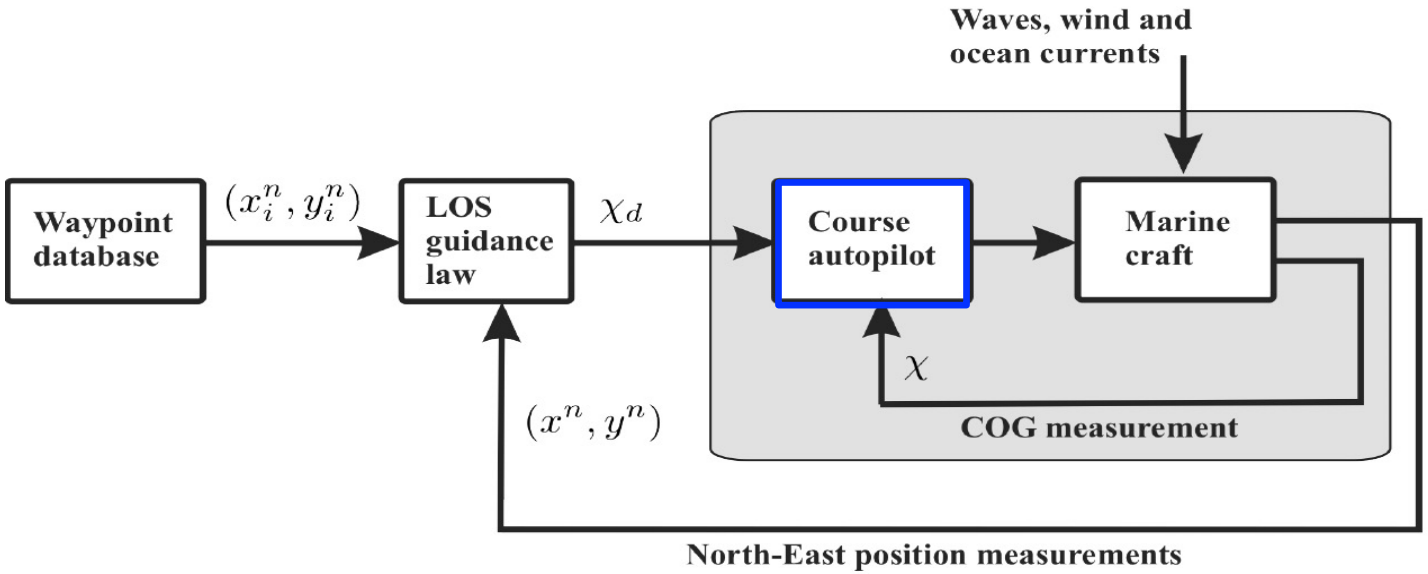
$$y_e^p = \frac{KU}{s^2(Ts+1)}\delta + d_y \qquad d_y = \frac{U}{s}(d_\chi - \pi_p)$$



12.4 LOS Guidance Laws for Path Following using Course Autopilots

Path following is the task of following a predefined path independent of time; that is there are no temporal constraints.

No restrictions are placed on the temporal propagation along the path. Spatial constraints, however, can be added to represent obstacles and other positional constraints.



The LOS guidance laws that can be used together with commercial course autopilots under the assumption that $\chi = \chi_d$

12.4.1 Vector Field Guidance Law

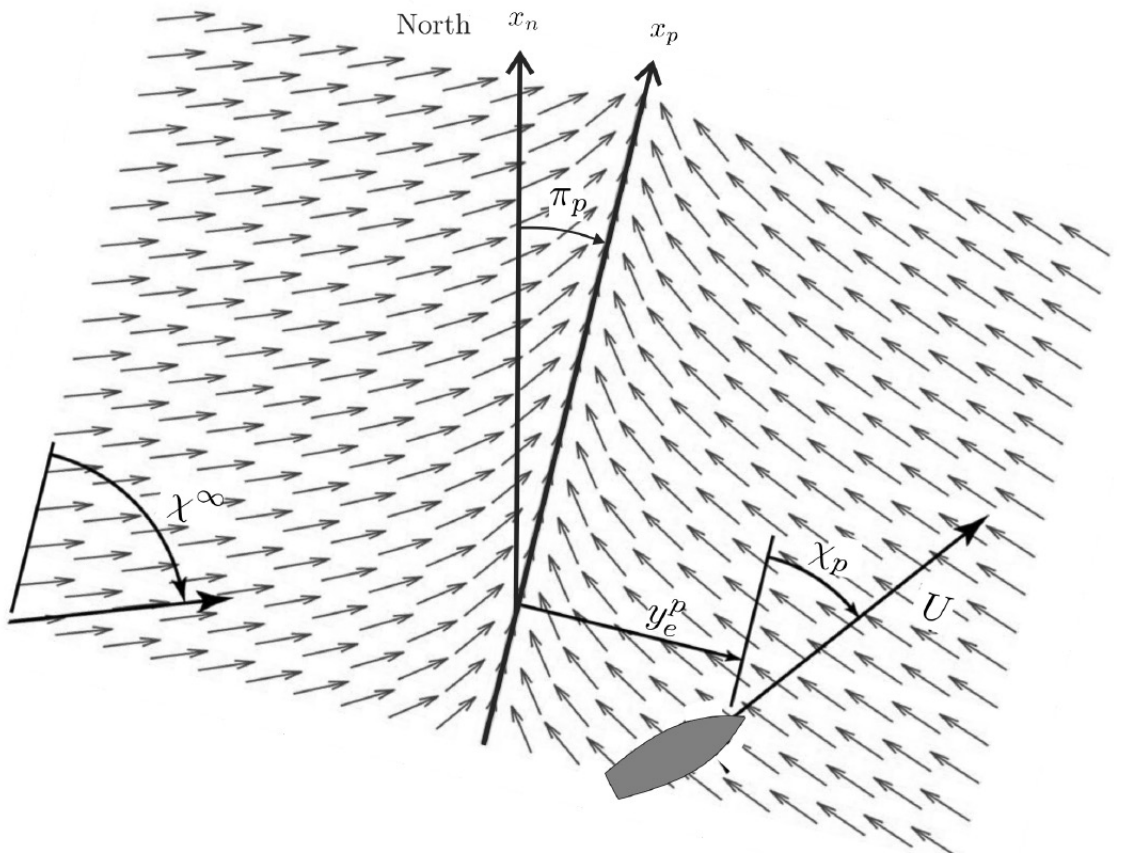
Vector field guidance law (Nelson et al. 2007)

$$\chi_d = \pi_p - \chi^\infty \frac{2}{\pi} \tan^{-1}(K_p\, y_e^p)$$

When $|y_e^p|$ is large, the craft is directed to approach the path with

$$\chi_p = \pm\chi^\infty \text{ for } \chi^\infty \in (0, \pi/2]$$

angle whereas when y_e^p approaches zero, the angle $\chi_p \to 0$



Vector field for straight-line path following

Lyapunov stability analysis

$$\dot{y}_e^p = U \sin(\chi - \pi_p)$$
$$= -U \sin\left(\chi^\infty \frac{2}{\pi} \tan^{-1}(K_p\, y_e^p)\right)$$
$$V = 1/2(y_e^p)^2$$
$$\dot{V} = -y_e^p\, U \sin\left(\chi^\infty \frac{2}{\pi} \tan^{-1}(K_p\, y_e^p)\right) \qquad -\frac{\pi}{2} < \chi^\infty \frac{2}{\pi} \tan^{-1}(K_p\, y_e^p) < \frac{\pi}{2}$$
$$< 0, \qquad \forall y_e^p \neq 0$$

Hence, the equilibrium point $y_e^p = 0$ is GAS

12.4.2 Proportional LOS Guidance Law

The vector-field guidance law in Section 12.4.1 is almost similar to a class of proportional LOS guidance laws used for marine craft path-following control. The difference is that we choose $\chi^{\infty} = \pi/2$ such that

$$\chi_d = \pi_p - \tan^{-1}\left(K_p\, y_e^p\right) \qquad K_p = \frac{1}{\Delta}$$

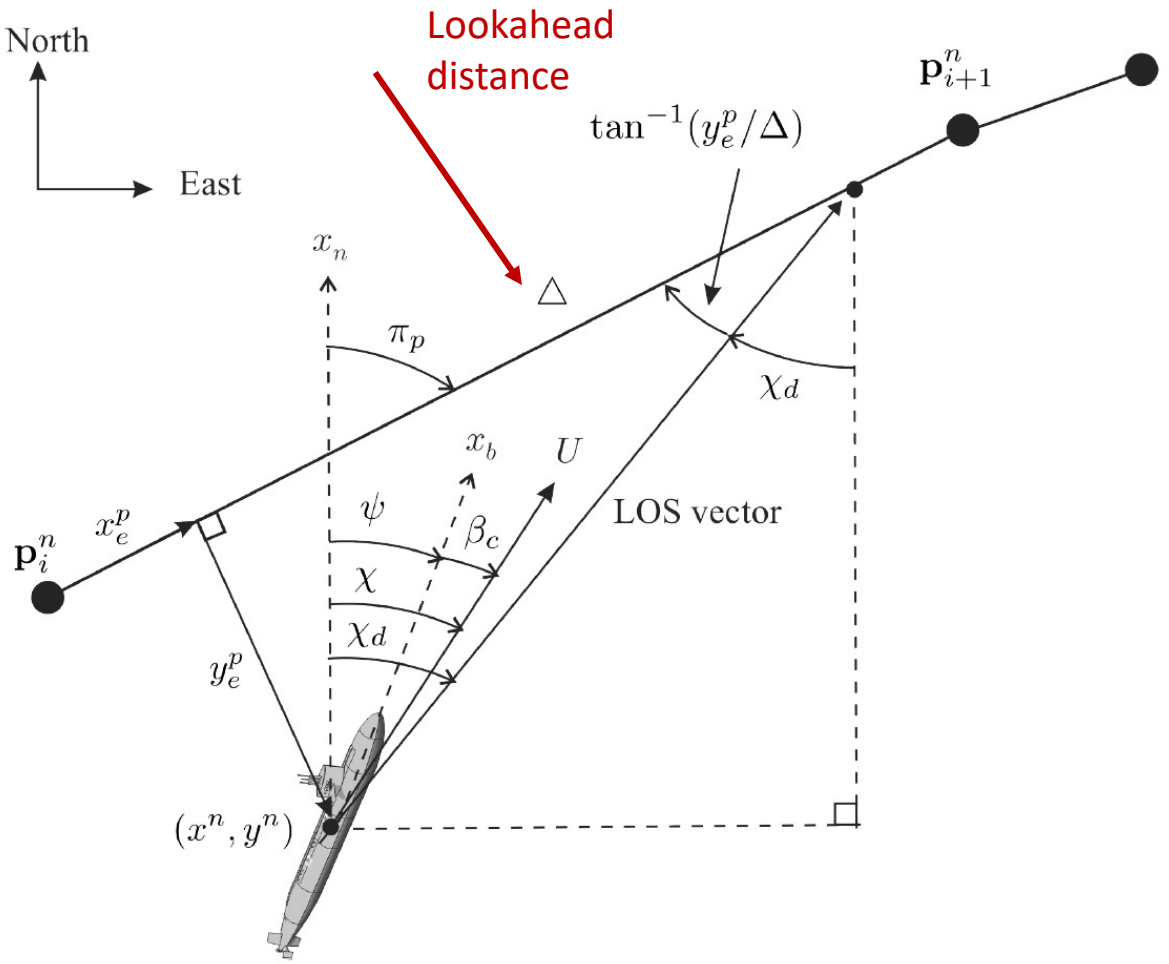
The desired course rate $\dot{\chi}_d = \omega_{\chi_d}$ is

$$\omega_{\chi_d} = -\frac{K_p \dot{y}_e}{(K_p y_e^p)^2 + 1} = -\frac{K_p U \sin(\chi - \pi_p)}{(K_p y_e^p)^2 + 1}$$

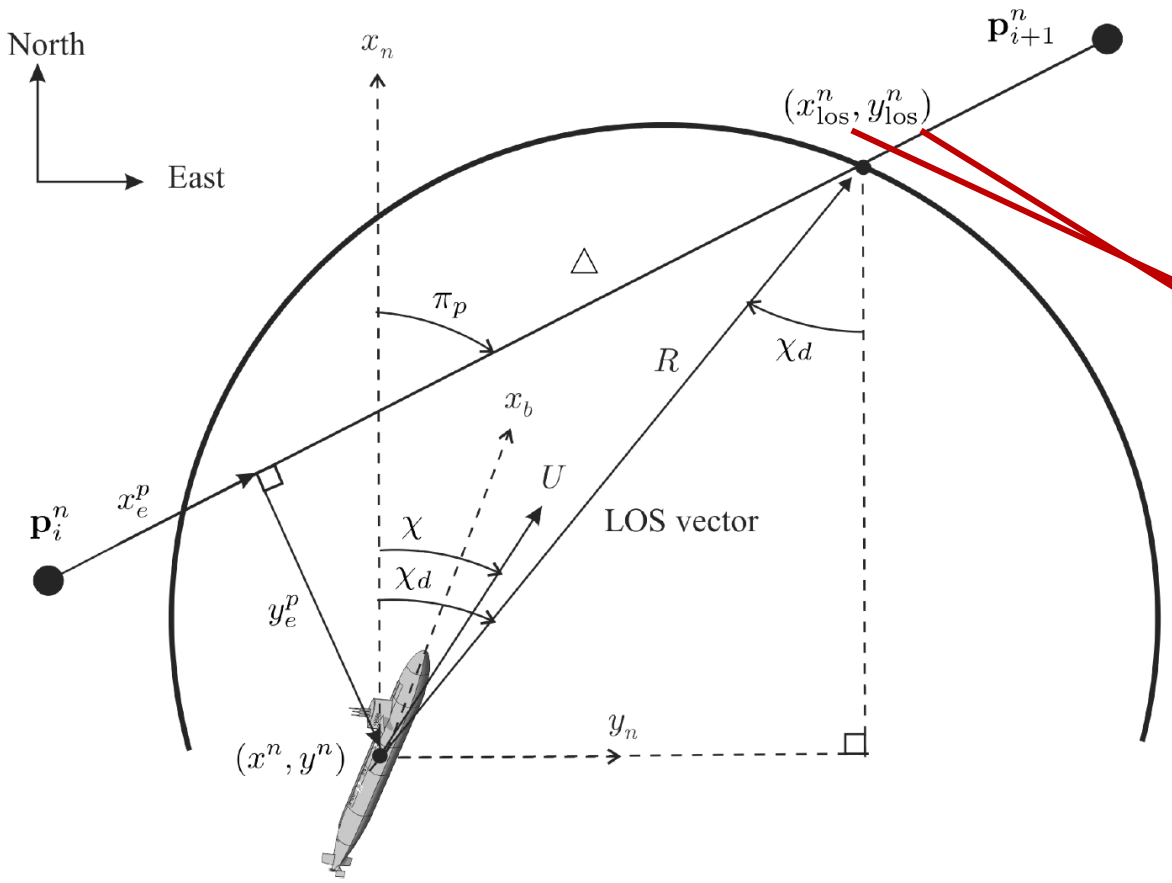
Cross-track error dynamics

$$\begin{aligned}
\dot{y}_e^p &= U \sin(\chi_p) \\
&= U \sin(\chi - \pi_p) \\
&= -U \sin\left(\tan^{-1}\left(y_e^p/\Delta\right)\right) \\
&= -\frac{U}{\sqrt{\Delta^2 + (y_e^p)^2}}\, y_e^p
\end{aligned}$$

The equilibrium point $y_e^p = 0$ is USGES as shown by Fossen and Pettersen (2014), see Appendix A.2.3.



12.4.3 Lookahead- and Enclosure-Based LOS Steering



The proportional LOS guidance laws can be classified according to (Breivik and Fossen 2009)

- Lookahead-based steering

$$\chi_d = \pi_p - \tan^{-1}\left(\frac{y_e^p}{\Delta}\right) \qquad \Delta > 0$$

- Enclosure-based steering

$$\chi_d = \text{atan2}\left(y_{\text{los}}^n - y^n, x_{\text{los}}^n - x^n\right)$$

$$\Delta(t) = \sqrt{R^2 - y_e^p(t)^2}$$

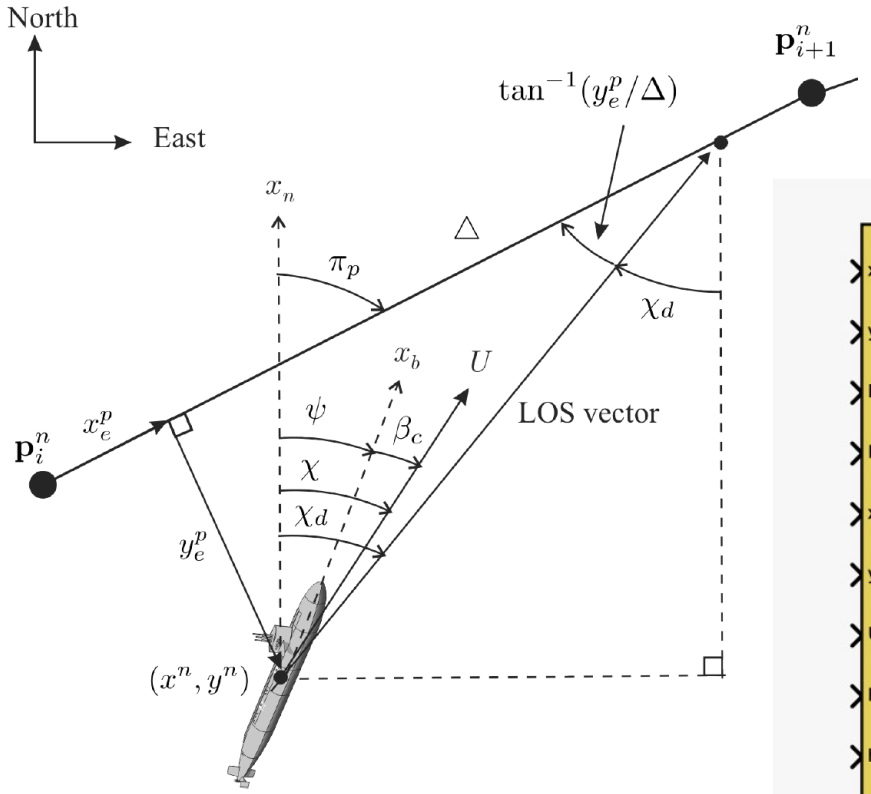
The two steering methods essentially operate by the same principle, but as will be made clear, the lookahead-based approach motivated by missile guidance has several advantages over the enclosure-based approach.

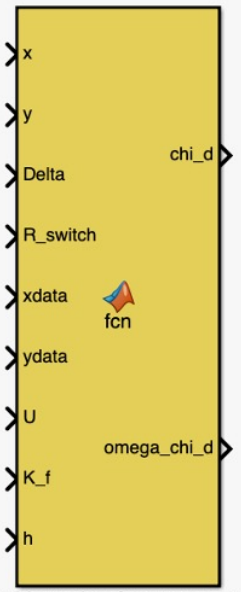
MSS Toolbox: LOSchi.m

```
x = 0; y= 0; Delta = 50; R_switch = 5;
wpt.pos.x = [0 100 200]';
wpt.pos.y = [0 100 200]';
U = 1, chi = 0
chi_d = LOSchi(x,y,Delta,R_switch,wpt,U,chi)
chi_d =
    0.7854
omega_psi_d =
    0.0141
```

$$d_{i+1} = \|\boldsymbol{p}_{i+1}^n - \boldsymbol{p}_i^n\|$$

Switch to next waypoint if $d_{i+1} - |x_e^p| \leq R_{\text{switch}}$





LOS guidance law for course autopilot

```
function [chi_d, omega_chi_d,y_e] = LOSchi(x,y,Delta,R_switch,wpt,U,K_f,h)
```

```
% [chi_d, omega_chi_d,y_e] = LOSchi(x,y,Delta,R_switch,wpt,U,K_f,h)
% LOSpsi computes thedesired course angle when the path is straight lines
% going through the waypoints  (wpt.pos.x, wpt.pos.y). The desired course
% angle chi_d and course rate d/dt chi_d = omega_chi_d used by course
% autopilot systems are computed using the roportional LOS guidance law:
%
%  chi_d = pi_h - atan( Kp * y_e ),    Kp = 1/Delta
%
%  omega_chi_d = -Kp * U * sin( chi - pi_h ) / ( 1 + (Kp * y_e)^2 )
%
% where pi_h is the path-tangential (azimuth) angle with respect to the North
% axis and y_e is the cross-track error. The observer/filter for the desired
% yaw angle psi_d is
%
%    d/dt psi_f = r_d + K_f * ssa( psi_d - psi_f ) )
%
% where the desired course rate omega_d = d(psi_d)/dt is computed by
%
%    d/dt y_e = -U * y_e / sqrt( Delta^2 + y_e^2 )
%    omega_chi_d = -Kp * dy_e/dt / ( 1 + (Kp * y_e)^2 )
%
% Initialization:
%   The active waypoint (xk, yk) where k = 1,2,...,n is a persistent
%   integer should be initialized to the first waypoint, k = 1, using
%   >> clear LOSchi
%
% Inputs:
%   (x,y): craft North-East positions (m)
%   Delta: positive look-ahead distance (m)
%   R_switch: go to next waypoint when the along-track distance x_e
%             is less than R_switch (m)
%   wpt.pos.x = [x1, x2,..,xn]': array of waypoints expressed in NED (m)
%   wpt.pos.y = [y1, y2,..,yn]': array of waypoints expressed in NED (m)
%   U: speed, vehicle cruise speed or time-varying measurement (m/s)
%   K_f: observer gain for desired yaw angle (typically 0.1-0-5)
%   h: sampling time (s)
```

12.4.3 Lookahead- and Enclosure-Based LOS Steering

The enclosure-based strategy for driving y_e^p to zero is then to direct the velocity toward the intersection point

$$\boldsymbol{p}_{\text{los}}^n = [x_{\text{los}}^n, y_{\text{los}}^n]^\top$$

that corresponds to the desired direction of travel. We can compute the intersection point from the following two equations

$$(x_{\text{los}}^n - x^n)^2 + (y_{\text{los}}^n - y^n)^2 = R^2$$

$$\tan(\pi_p) = \frac{y_{i+1}^n - y_i^n}{x_{i+1}^n - x_i^n} = \frac{y_{\text{los}}^n - y_i^n}{x_{\text{los}}^n - x_i^n}$$

These equations can be solved numerically or analytically for $\boldsymbol{p}_{\text{los}}^n = [x_{\text{los}}^n, y_{\text{los}}^n]^\top$

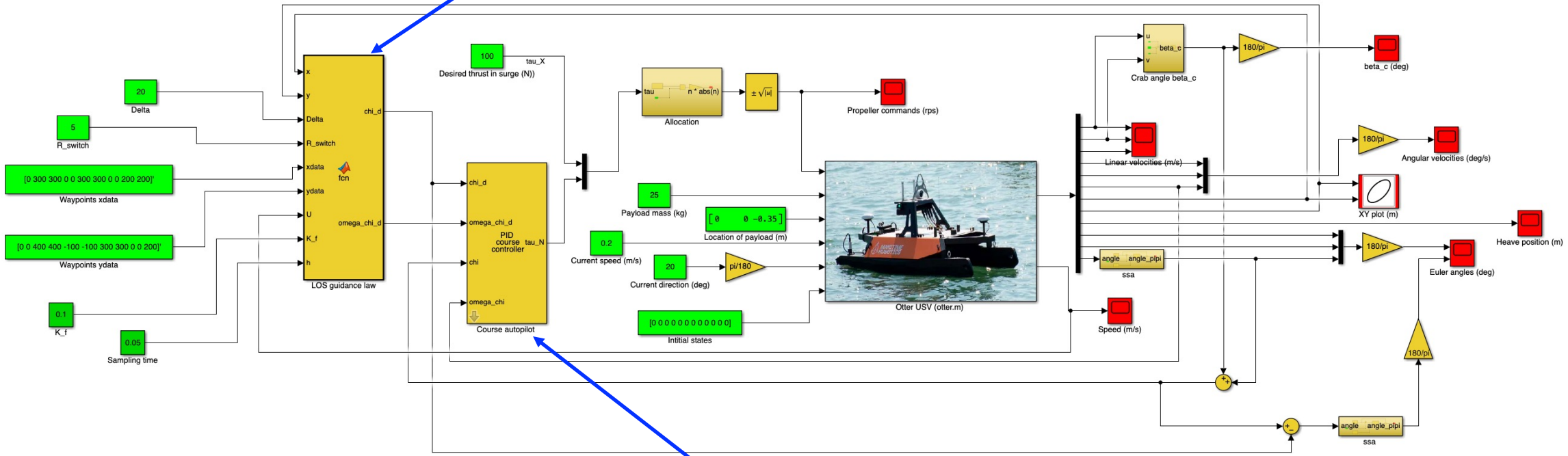
As can be immediately noticed, the lookahead-based steering scheme is less computationally intensive than the enclosure-based approach. It is also valid for all cross-track errors, whereas the enclosure-based strategy requires

$$R > |y_e^p|$$

MSS Simulink: demoOtterUSVPathFollowingCourseControl.slx

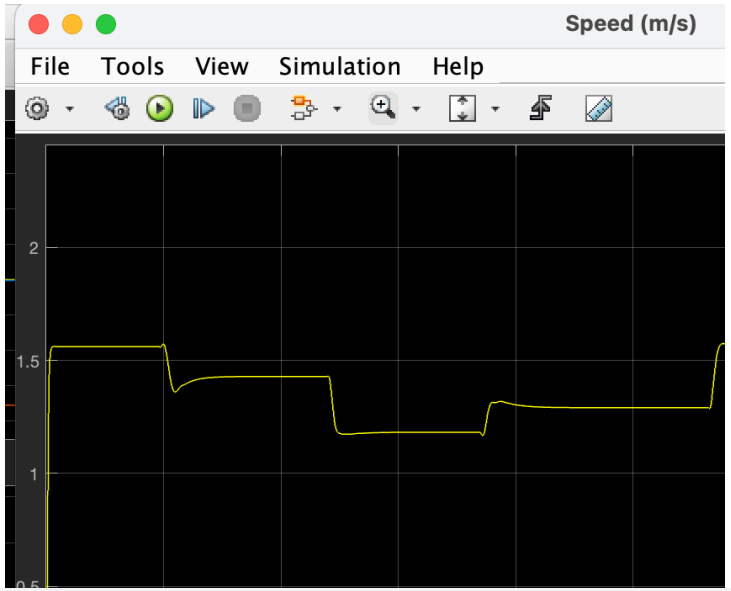
Simulink demo calling the MSS Matlab function LOSchi.m for LOS path-following control using a course autopilot

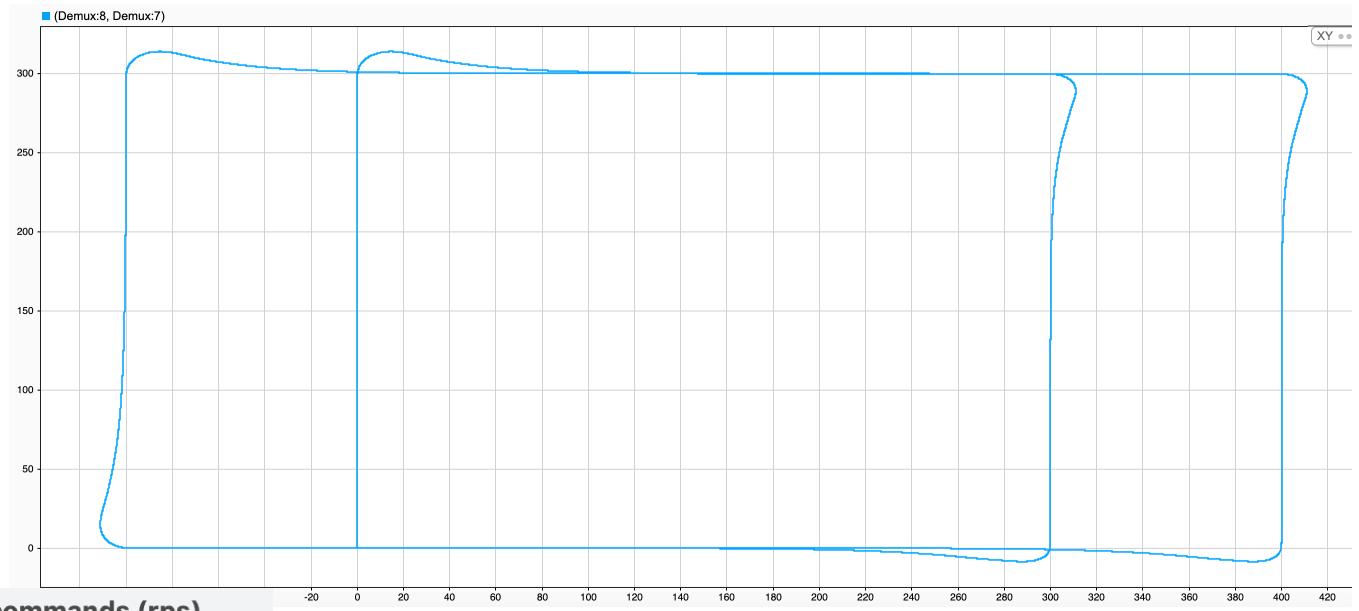
LOS guidance law for course autopilot

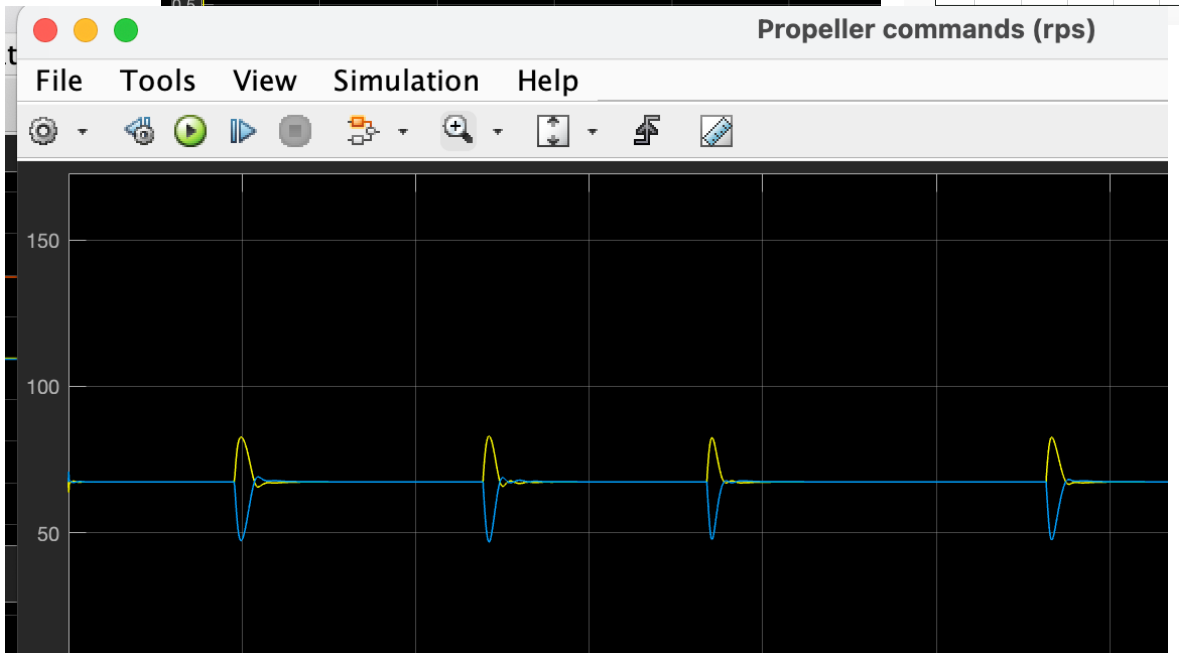


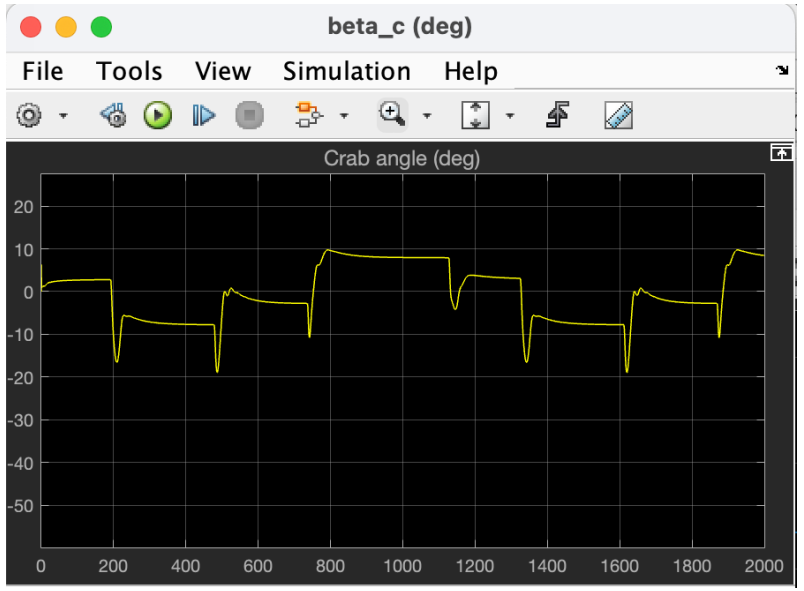
Course autopilot using course angle and course rate

MSS Simulink: demoOtterUSVPathFollowingCourseControl.slx









12.5 LOS Guidance Laws for Path Following using Heading Autopilots

It is important to stress the concepts for course and heading control since there are many conceptual misunderstandings regarding the course and heading of an aircraft and marine craft.

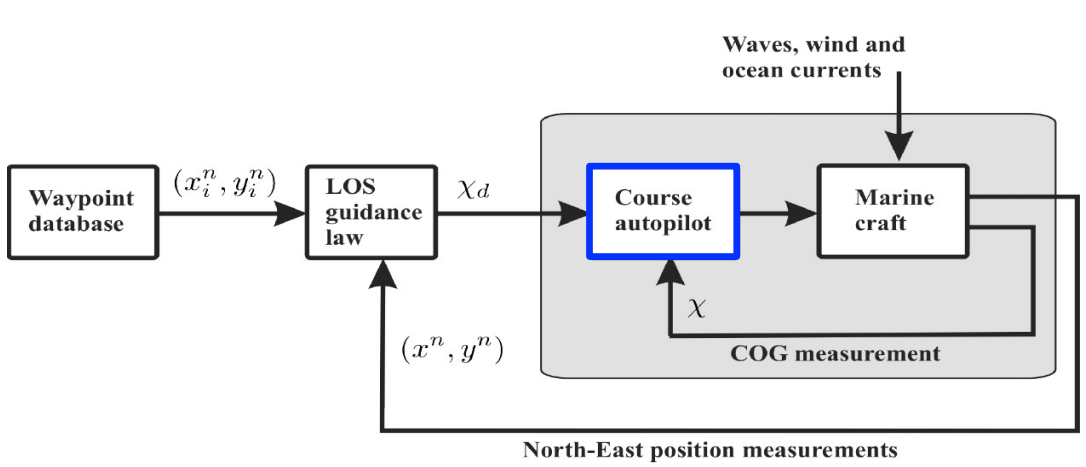
$$\chi = \psi + \beta_c$$

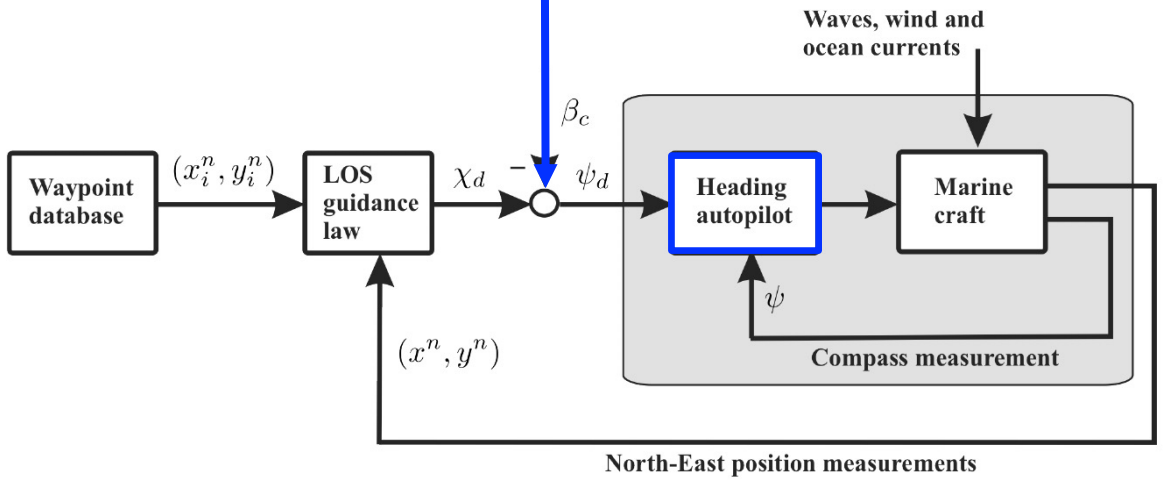
- The course angle χ of a vehicle is the cardinal direction in which the vehicle is moving,
- The heading angle ψ, is the direction the craft's bow (x_b axis) is pointed

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

The difference between the course and heading angles is the crab angle

If your vehicle is exposed to an environmental force, the sway velocity v will be non-zero. This gives a non-zero crab angle β_c and the vehicle is said to **sideslip**.





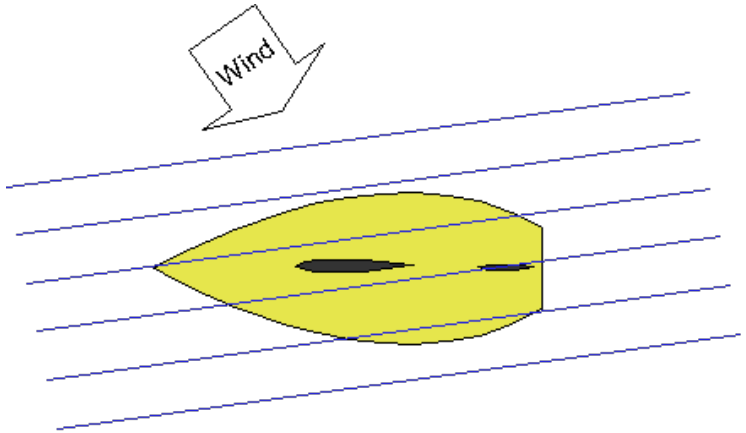
12.5.1 Crab Angle due to Crosswind/Crosscurrent Component

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.

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

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

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

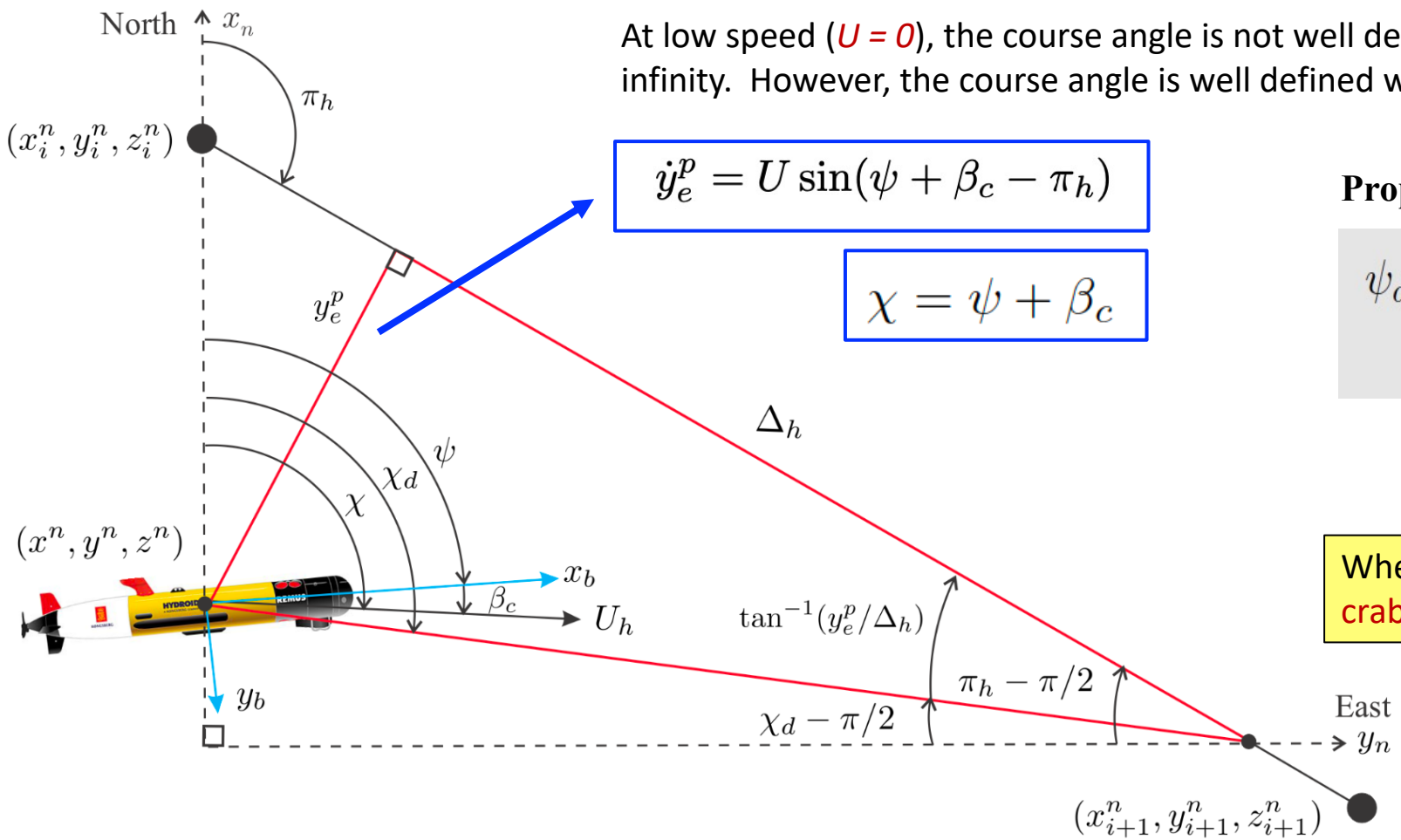


https://aviation.stackexchange.com/questions/58861/what-is-the-maximum-angle-between-an-airplane-and-runway-centerline-when-touchin

12.5.1 Proportional LOS Guidance Law for Heading Control

Marine craft often use compass measurements to measure the heading angle since they can operate at zero speed.

At low speed (*U = 0*), the course angle is not well defined since the crab angle can reach infinity. However, the course angle is well defined when moving from A to B (*U > 0*).



$$\dot{y}_e^p = U \sin(\psi + \beta_c - \pi_h)$$

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

Proportional LOS Guidance Law

$$\begin{aligned}\psi_d &= \chi_d - \beta_c \\ &= \pi_h - \tan^{-1}(K_p y_e^p) - \beta_c\end{aligned}$$

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

When designing the heading autopilot, the crab angle is treated as a disturbance in yaw

Crab Angle Compensation by Direct Measurements?

- At zero speed ($u = 0$), the course angle χ is not well defined
- Marine craft use compass measurements to measure the heading angle ψ, which is well defined at zero speed

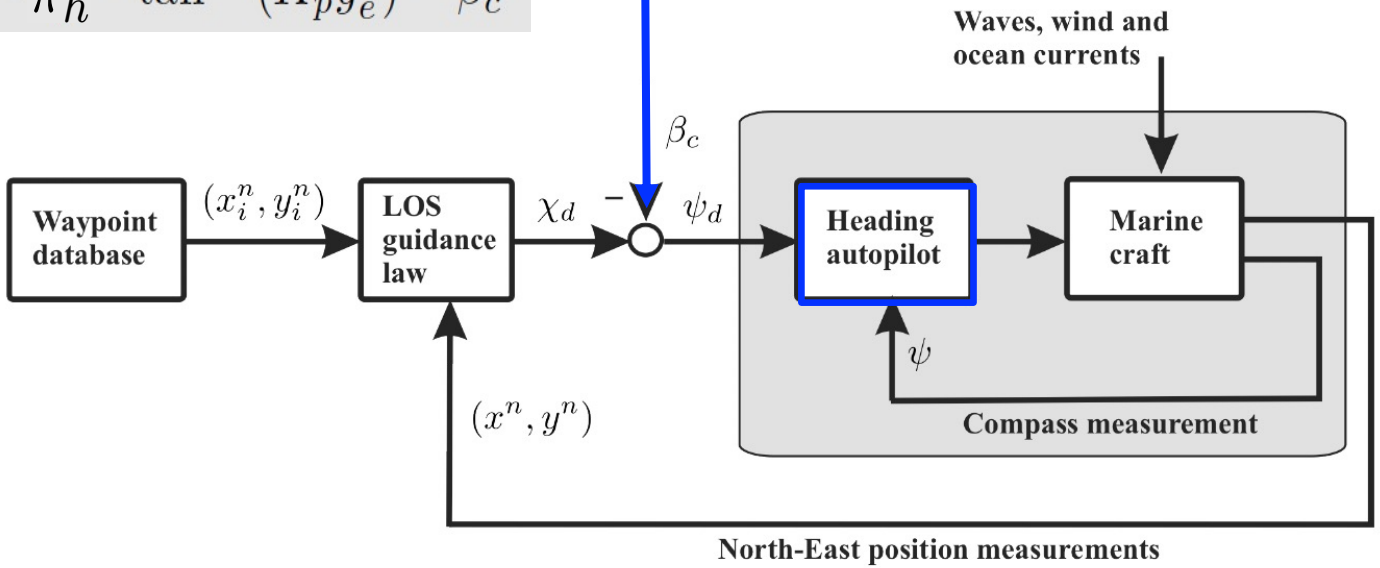
$$\dot{y}_e^p = U \sin(\psi + \beta_c - \pi_h)$$

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

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

$$\begin{aligned}\psi_d &= \chi_d - \beta_c \\ &= \pi_h - \tan^{-1}(K_p y_e^p) - \beta_c\end{aligned}$$

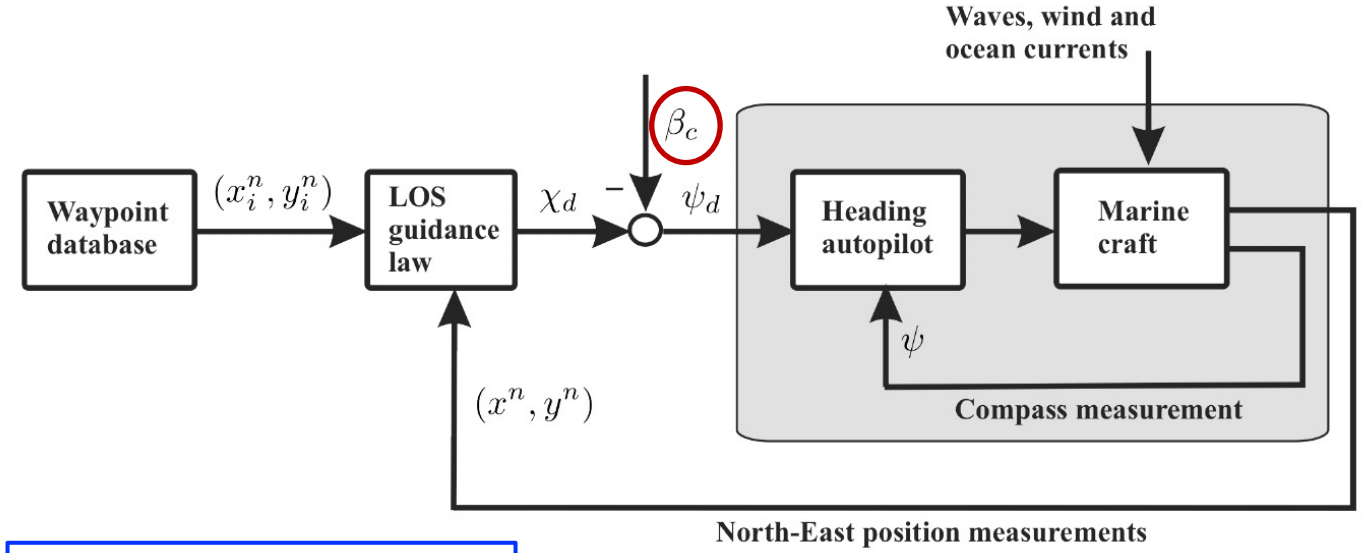
When designing the heading autopilot, the crab angle is treated as a disturbance



An estimate of the crab angle can be obtained by using velocity measurements. However, the formula for the crab angle is numerical ill-conditioned for small u and v.

12.5.2 Integral LOS (ILOS) for Compensation of β_c

The classical ILOS algorithm by Børhaug et al. (2008) is frequently used to compensate for the unknown drift term β_c



Børhaug et al. (2008) used Lyapunov stability theory to prove UGAS/ULES.

The stability analysis is based on a 3-DOF ship model (surge, sway and yaw) in cascade with the ILOS guidance law.

$$\dot{y}_e^p = U \sin(\psi + \beta_c - \pi_h)$$

Classical ILOS (Børhaug et al. 2008)

$$\psi_d = \pi_h - \tan^{-1}(K_p\, y_e^p + K_i\, y_{\text{int}}^p)$$

$$\dot{y}_{\text{int}}^p = \frac{\Delta y_e^p}{\Delta^2 + (y_e^p + \kappa\, y_{\text{int}}^p)^2}$$

where $K_p = 1/\Delta$, $K_i = \kappa K_p$ and $\kappa > 0$ are design parameters.

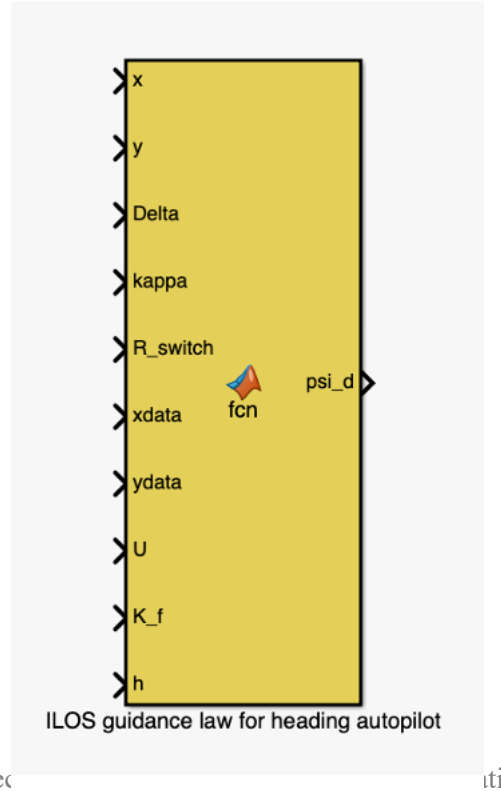
E. Børhaug, A. Pavlov and K. Y. Pettersen (2008). Integral LOS control for Path Following of Underactuated Marine Surface Vessels in the presence of Constant Ocean Currents, 47th IEEE Conference on Decision and Control (CDC), pp. 4984–4991, Cancun, Mexico.

MSS Toolbox

Classical ILOS (Børhaug et al. 2008)

$$\psi_d = \pi_h - \tan^{-1}(K_p\, y_e^p + K_i\, y_{\text{int}}^p)$$

$$\dot{y}_{\text{int}}^p = \frac{\Delta y_e^p}{\Delta^2 + (y_e^p + \kappa\, y_{\text{int}}^p)^2}$$



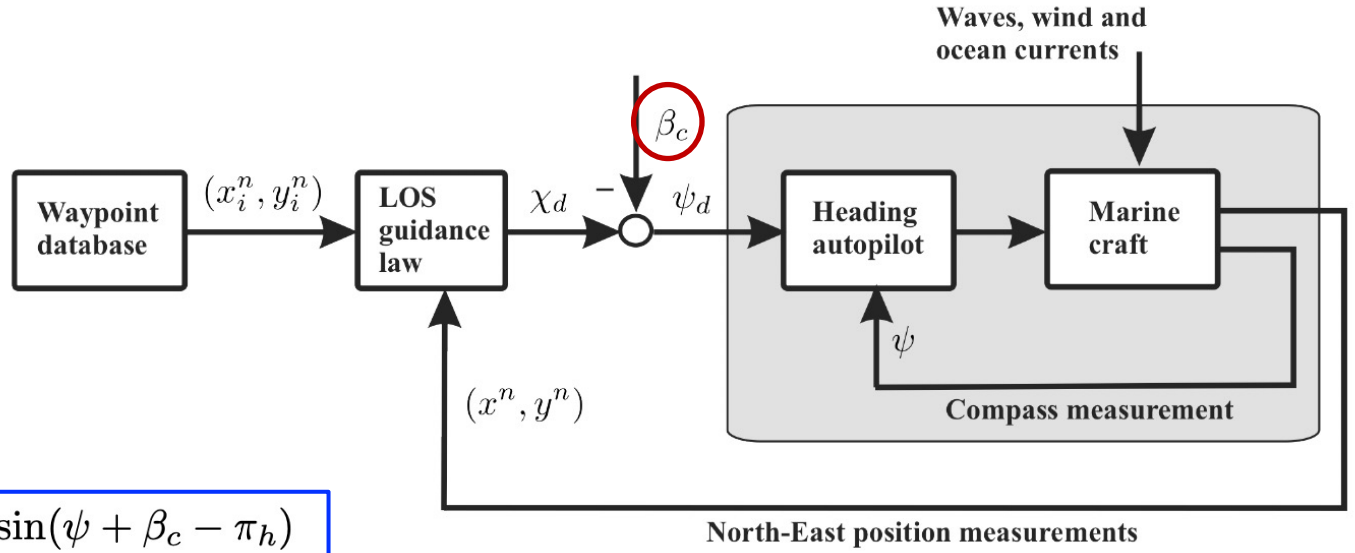
ILOS guidance law for heading autopilot

```
function [psi_d, r_d, y_e] = ILOSpsi(x,y,Delta,kappa,h,R_switch,wpt,U,K_f)
```

```
% [psi_d, r_d, y_e] = ILOSpsi(x,y,Delta,kappa,h,R_switch,wpt,U)
% ILOSpsi computes the desired heading angle psi_d, desired yaw rate r_d
% and cross-track error y_e when the path is  straight lines going through
% the waypoints (wpt.pos.x, wpt.pos.y). The desired heading angle computed
% using the classical ILOS guidance law by Børhaug et al. (2008).
%
%    psi_d = pi_h - atan( Kp * (y_e + kappa * y_int) ),  Kp = 1/Delta
%
%    d/dt y_int = Delta * y_e / ( Delta^2 + (y_e + kappa * y_int)^2 )
%
% where pi_h is the path-tangential (azimuth) angle with respect to the
% North axis and y_e is the cross-track error. The observer/filter for the
% desired yaw angle psi_d is
%
%    d/dt psi_f = r_d + K_f * ssa( psi_d - psi_f ) )
%
% where the desired yaw rate r_d = d(psi_d)/dt is computed by
%
%    d/dt y_e = -U * y_e / sqrt( Delta^2 + y_e^2 )
%    r_d = -Kp * dy_e/dt / ( 1 + (Kp * y_e)^2 )
%
% Initialization:
%   The active waypoint (xk, yk) where k = 1,2,...,n is a persistent
%   integer should be initialized to the first waypoint, k = 1, using
%   >> clear ILOSpsi
%
% Inputs:
%   (x,y): craft North-East positions (m)
%   Delta: positive look-ahead distance (m)
%   kappa: positive integral gain constant, Ki = kappa * Kp
%   h: sampling time (s)
%   R_switch: go to next waypoint when the along-track distance x_e
%             is less than R_switch (m)
%   wpt.pos.x = [x1, x2,..,xn]' array of waypoints expressed in NED (m)
%   wpt.pos.y = [y1, y2,..,yn]' array of waypoints expressed in NED (m)
%   U: speed, vehicle cruise speed or time-varying measurement (m/s)
%   K_f: observer gain for desired yaw angle (typically 0.1-0-5)
```

Adaptive LOS (ALOS) for Compensation of β_c

The ALOS algorithm by Fossen (2023) estimates the unknown drift term β_c as a (nearly) constant parameter and compensates β_c by direct matching. This is equivalent to adding integral action.



$$\dot{y}_e^p = U \sin(\psi + \beta_c - \pi_h)$$

ALOS (Fossen 2023)

$$\psi_d = \pi_h - \hat{\beta}_c - \tan^{-1}\left(\frac{y_e^p}{\Delta}\right)$$

$$\dot{\hat{\beta}}_c = \gamma \frac{\Delta}{\sqrt{\Delta^2 + (y_e^p)^2}} y_e^p$$

The stability proof guarantees that the equilibrium point is USGES.

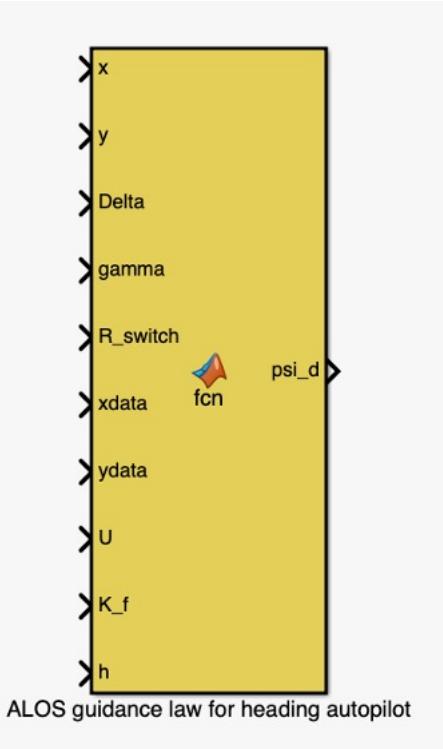
T. I. Fossen (2023). An Adaptive Line-of-sight (ALOS) Guidance Law for Path Following of Aircraft and Marine Craft. *IEEE Transactions on Control Systems Technology* 31(6), 2887-2894.

MSS Toolbox

ALOS (Fossen 2023)

$$\psi_d = \pi_h - \hat{\beta}_c - \tan^{-1}\left(\frac{y_e^p}{\Delta}\right)$$

$$\dot{\hat{\beta}}_c = \gamma \frac{\Delta}{\sqrt{\Delta^2 + (y_e^p)^2}} y_e^p$$



ALOS guidance law for heading autopilot

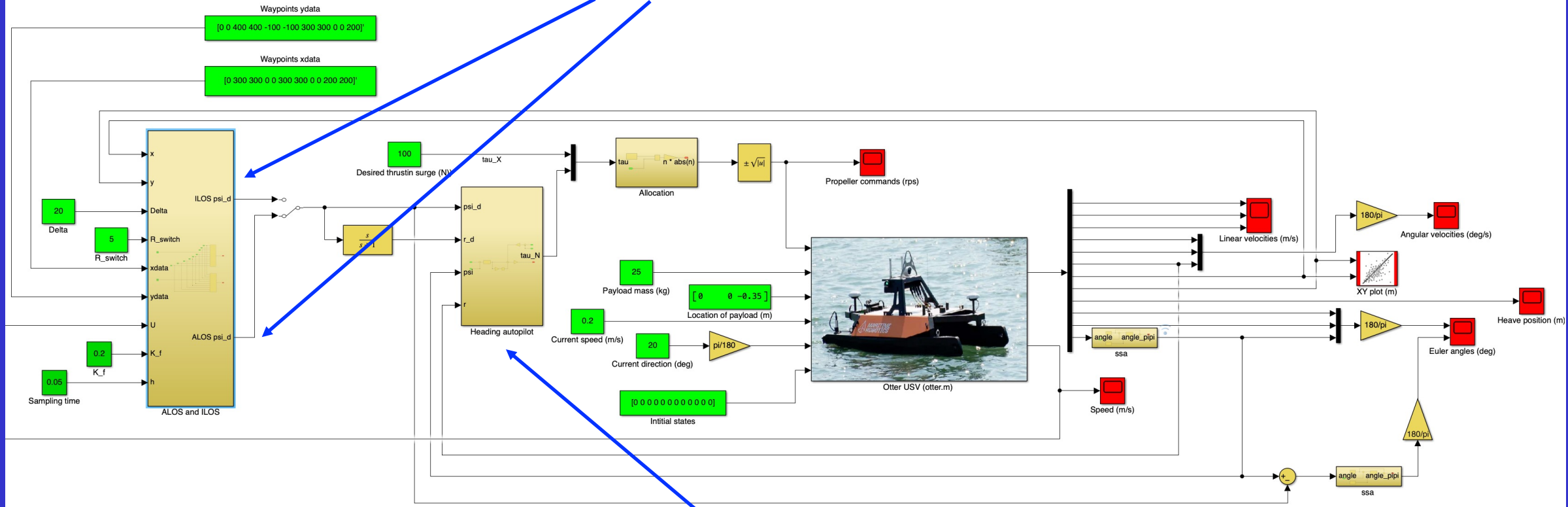
```
function [psi_d, r_d, y_e] = ALOSpsi(x,y,Delta,gamma,h,R_switch,wpt,U,K_f)
```

```
% [psi_d, r_d, y_e] = ALOSpsi(x,y,Delta,gamma,h,R_switch,wpt,U)
% ALOSpsi computes the desired heading angle psi_d, desired yaw rate r_d
% and cross-track error y_e when the path is  straight lines going through
% the waypoints (wpt.pos.x, wpt.pos.y). The desired heading angle computed
% using the adaptive LOS guidance law by Fossen (2023).
%
%    psi_d = pi_h - beta_hat - atan( Kp * y_e ),  Kp = 1/Delta
%
%    d/dt beta_hat = gamma * Delta * y_e / sqrt( Delta^2 + y_e^2 )
%
% where pi_h is the path-tangential (azimuth) angle with respect to the
% North axis and y_e is the cross-track error. The observer/filter for the
% desired yaw angle psi_d is
%
%    d/dt psi_f = r_d + K_f * ssa( psi_d - psi_f ) )
%
% where the desired yaw rate r_d = d(psi_d)/dt is computed by
%
%    d/dt y_e = -U * y_e / sqrt( Delta^2 + y_e^2 )
%    r_d = -Kp * dy_e/dt / ( 1 + (Kp * y_e)^2 )
%
% Initialization:
%   The active waypoint (xk, yk) where k = 1,2,...,n is a persistent
%   integer should be initialized to the first waypoint, k = 1, using
%   >> clear ALOSpsi
%
% Inputs:
%   (x,y): craft North-East positions (m)
%   Delta: positive look-ahead distance (m)
%   gamma: positive adaptive gain constant
%   h: sampling time (s)
%   R_switch: go to next waypoint when the along-track distance x_e
%             is less than R_switch (m)
%   wpt.pos.x = [x1, x2,..,xn]' array of waypoints expressed in NED (m)
%   wpt.pos.y = [y1, y2,..,yn]' array of waypoints expressed in NED (m)
%   U: speed, vehicle cruise speed or time-varying measurement (m/s)
%   K_f: observer gain for desired yaw angle (typically 0.1-0-5)
```

MSS Simulink: demoOtterUSVPathFollowingHeadingControl.slx

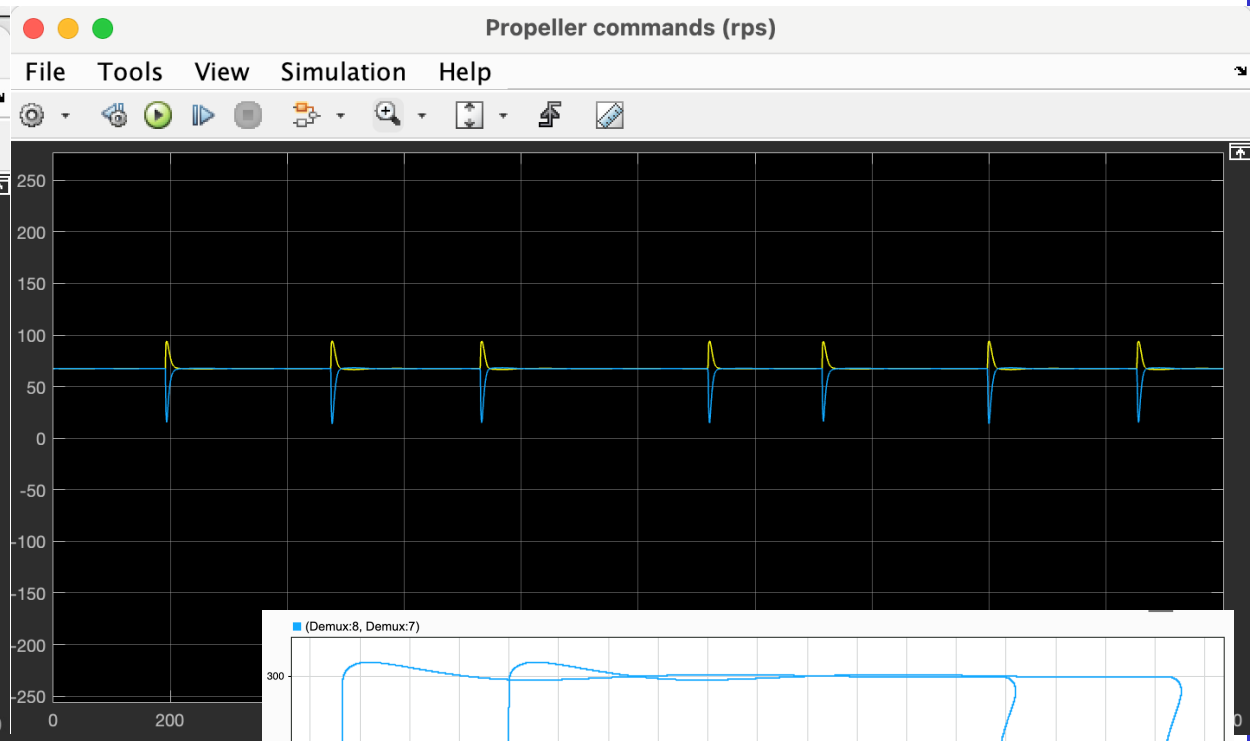
Simulink demo calling the MSS Matlab functions ALOSpsi.m and ILOSpsi.m for LOS path-following control using a heading autopilot

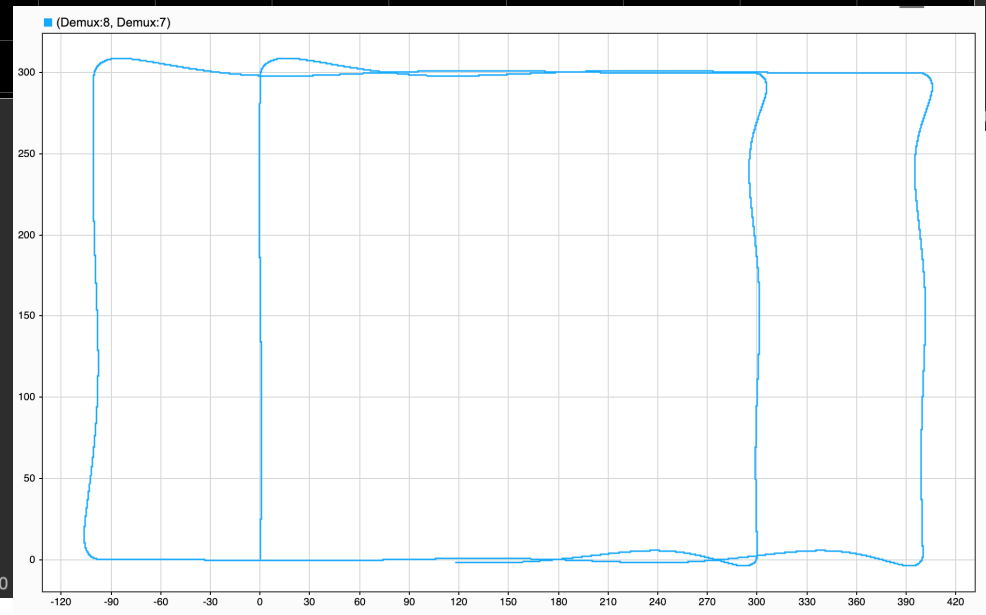
ALOS and ILOS guidance law for course autopilot



Heading autopilot using yaw angle and yaw rate

MSS Simulink: demoOtterUSVPathFollowingHeadingControl.slx





Adaptive Line-of-Sight Guidance Laws for 3-D Path Following

Compensation of Drift in Heading and Depth Autopilots using the ALOS Guidance Law for Path Following

Cross-Track Error Expressed in in the Path-Tangential Frame

$$\dot{y}_e^p = U \sin(\psi + \beta_c - \pi_h)$$

The crab angle β_c is treated as a disturbance in yaw

The yaw angel ψ is the control input

(We assume that the heading autopilot achieves perfect tracking such that $\psi = \psi_d$)

Vertical-Track Error Expressed in the Path-Tangential frame

$$\dot{z}_e^p = -U_v \sin(\theta - \alpha_c - \pi_v) + \frac{U_h \sin(\pi_v)}{\sqrt{1+\tan^2(\beta_c)}} \left(\sqrt{1+\tan^2(\beta_c)}\, \cos(\psi + \beta_c - \pi_h) - 1 \right)$$

α_c is treated as a disturbance in pitch

The pitch angel θ is the control input

(We assume that the pitch angle autopilot achieves perfect tracking such that $\theta = \theta_d$)

T. I. Fossen and A. P Aguiar (2024). A Uniform Semiglobal Exponential Stable Adaptive Line-of-Sight (ALOS) Guidance Law for 3-D Path Following. *Automatica.*

ALOS Guidance Laws ALOS for Compensation of α_c and β_c

$$\psi_d = \pi_h - \hat{\beta}_c - \tan^{-1}\left(\frac{y_e^p}{\Delta_h}\right)$$

$$\dot{\hat{\beta}}_c = \gamma_h \frac{\Delta_h}{\sqrt{\Delta_h^2 + (y_e^p)^2}} y_e^p$$

$$\theta_d = \pi_v + \hat{\alpha}_c + \tan^{-1}\left(\frac{z_e^p}{\Delta_v}\right)$$

$$\dot{\hat{\alpha}}_c = \gamma_v \frac{\Delta_v}{\sqrt{\Delta_v^2 + (z_e^p)^2}} z_e^p$$

$$\dot{y}_e^p = U_h \sin(\psi + \beta_c - \pi_h)$$

$$\dot{z}_e^p = -U_v \sin(\theta - \alpha_c - \pi_v) + \frac{U_h \sin(\pi_v)}{\sqrt{1 + \tan^2(\beta_c)}} \left(\sqrt{1 + \tan^2(\beta_c)}\cos(\psi + \beta_c - \pi_h) - 1\right)$$

Fossen and Aguiar (2004) have shown that the origins of the cross-track and vertical-track errors are USGES if

- $\Delta_h > 0$ and $\Delta_v > 0$ (positive look-ahead distances)
- $\gamma_h > 0$ and $\gamma_v > 0$ (positive adaptation gains)
- The heading and depth autopilots guarantees that $\psi = \psi_d$ and $\theta = \theta_d$
- u is kept constant by the speed autopilot

T. I. Fossen and A. P Aguiar (2024). A Uniform Semiglobal Exponential Stable Adaptive Line-of-Sight (ALOS) Guidance Law for 3-D Path Following. *Automatica.*

AUV Case Study

The Remus 100 at the Applied Underwater Robotics Laboratory (AUR-Lab) at NTNU.

The 6-DOF mathematical model of the Remus 100 AUV is available in the MATLAB MSS toolbox (Fossen and Perez, 2004). The script *remus100.m* describes an AUV of length 1.6 m, diameter of 19 cm and mass 31.9 kg.

The vehicle's maximum speed, 2.5 m/s, is obtained by running the propeller at 1525 rpm when there are no ocean currents. Depth is controlled by using the stern plane δ_s, while the tail rudder δ_r controls the yaw angle.

The heading and depth autopilots are implemented as PID controllers

$$\delta_s = -k_{p_\theta}\,\mathrm{ssa}(\tilde{\theta}) - k_{d_\theta}\dot{\theta} - k_{i_\theta}\int_0^t \mathrm{ssa}(\tilde{\theta})\mathrm{d}\tau$$

$$\delta_r = -k_{p_\psi}\,\mathrm{ssa}(\tilde{\psi}) - k_{d_\psi}\dot{\psi} - k_{i_\psi}\int_0^t \mathrm{ssa}(\tilde{\psi})\mathrm{d}\tau$$

MSS Toolbox: remus100.m

```
function [xdot,U] = remus100(x,ui,Vc,betaVc,w_c)
% The length of the Remus 100 AUV is 1.6 m, the cylinder diameter is 19 cm
% and the mass of the vehicle is 31.9 kg. The maximum speed of 2.5 m/s is
% obtained when the propeller runs at 1525 rpm in zero currents. The
% function calls are:
%   [xdot,U] = remus100(x,ui,Vc,betaVc,alphaVc,w_c)  3-D ocean currents
%   [xdot,U] = remus100(x,ui,Vc,betaVc,alphaVc)      horizontal ocean currents
%   [xdot,U] = remus100(x,ui)                        no ocean currents
% The function returns the time derivative xdot of the state vector:
%   x = [ u v w p q r x y z phi theta psi ]',     alternatively
%   x = [ u v w p q r x y z eta eps1 eps2 eps3 ]'
% in addition to the speed U in m/s (optionally). The state vector can be
% of dimension 12 (Euler angles) or 13 (unit quaternions):
%
%   u:       surge velocity          (m/s)
%   v:       sway velocity           (m/s)
%   w:       heave velocity          (m/s)
%   p:       roll rate               (rad/s)
%   q:       pitch rate              (rad/s)
%   r:       yaw rate                (rad/s)
%   x:       North position          (m)
%   y:       East position           (m)
%   z:       downwards position      (m)
%   phi:     roll angle              (rad)
%   theta:   pitch angle             (rad)
%   psi:     yaw angle               (rad)
%
% For the unit quaternion representation, the last three arguments of the
% x-vector, the Euler angles (phi, theta, psi), are replaced by the unit
% quaternion q = [eta, eps1, eps2, eps3]'. This increases the dimension of
% the state vector from 12 to 13.
```

```
% The control inputs are one tail rudder, two stern planes and a single-screw
% propeller:
%
%   ui = [ delta_r delta_s n ]'  where
%
%    delta_r:   rudder angle (rad)
%    delta_s:   stern plane angle (rad)
%    n:         propeller revolution (rpm)
%
% The arguments Vc (m/s), betaVc (rad), w_c (m/s) are optional arguments for
% ocean currents
%
%    v_c = [ Vc * cos(betaVc - psi), Vc * sin( betaVc - psi), w_c ]
```


```
% SIMremus100 User editable script for simulation of the Remus 100 AUV
%             (remus100.m) under feedback control (depth and heading control
%             when exposed to ocean currents). Both the Euler angle and unit
%             quaternion representations of the Remus 100 model can be used.
%
% Calls:      remus100.m
%
% Simulink:   demoAUVdepthHeadingControl.slx
%
% Author:     Thor I. Fossen
% Date:       2021-06-28
% Revisions:  2022-02-01 redesign of the autopilots
%             2022-05-06 retuning for new version of remus100.m
%             2022-05-08 added compability for unit quaternions

clearvars;
```

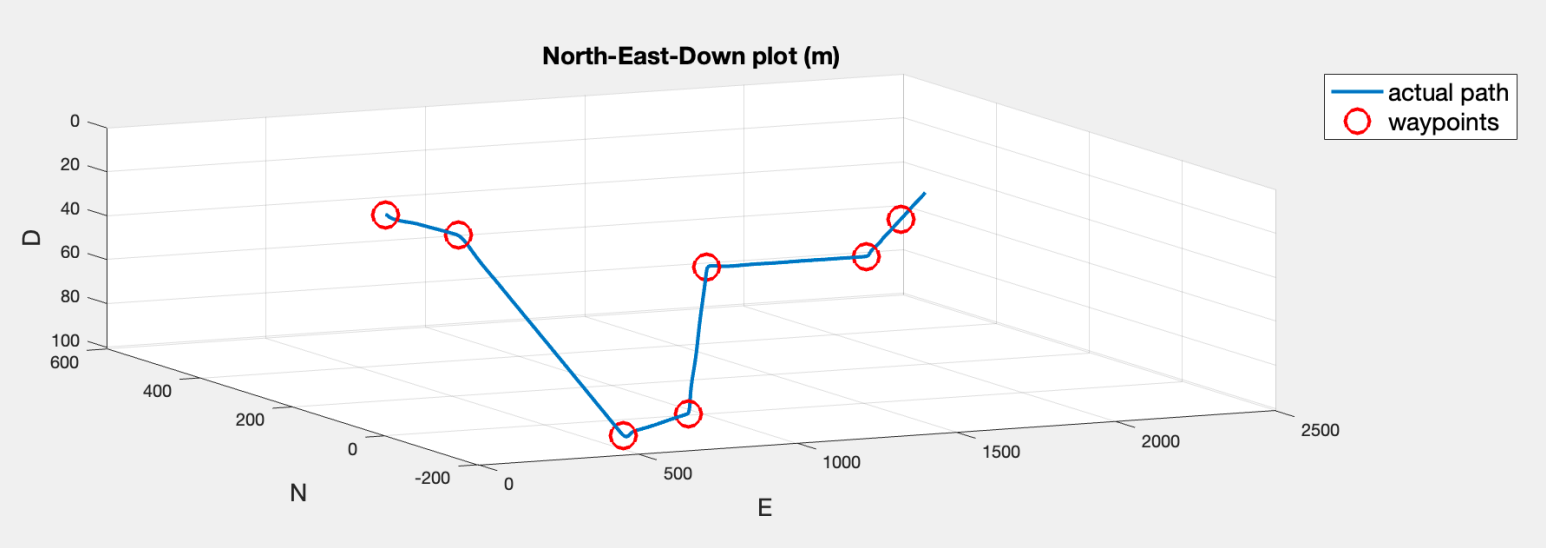
MSS Toolbox: ALOS3D.m

```
function [psi_d, theta_d, y_e, z_e, alpha_c_hat, beta_c_hat] = ...
    ALOS3D(x,y,z,Delta_h,Delta_v,gamma_h,gamma_v,M_theta,h,R_switch,wpt,U,K_f)
```

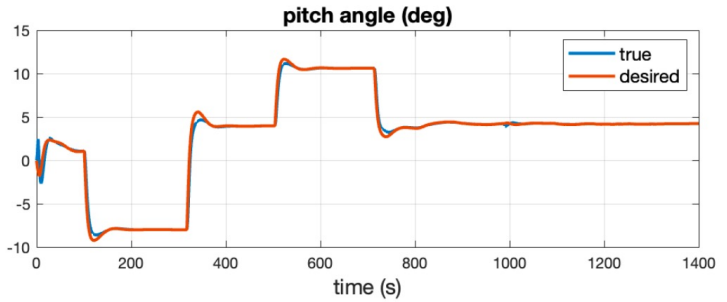
```
% [psi_d, theta_d, y_e, z_e, alpha_c_hat, beta_c_hat] = ...
% ALOS3D(x,y,z,Delta_h,Delta_v,gamma_h,gamma_v,M_theta,h,R_switch,wpt)
% computes the desired heading angle psi_d and pitch angle theta_d when
% the path is a straight line segment going through the waypoints
% (wpt.pos.x,wpt.pos.y,wpt.pos.z). The desired heading and pitch angles are
% computed using the adaptive LOS (ALOS) guidance law by Fossen and Aguiar
% (2023) where
%
%   theta_ref = pi_v + alpha_hat + atan( z_e / Delta_v )
%   psi_ref   = pi_h - beta_hat  - atan( y_e / Delta_h )
%
%   d/dt alpha_hat = gamma_v * Delta_v/sqrt(Delta_v^2+z_e^2) * Proj(alpha_hat,z_e)
%   d/dt beta_hat  = gamma_h * Delta_h/sqrt(Delta_h^2+y_e^2) * Proj(beta_hat,y_e)
%
% The theta_ref and psi_ref commands are filtered using the observer
%
%  psi_f = psi_f + h * (r_d + K_f * ssa( psi_ref - psi_f ) );
%  theta_f = theta_f + h * (q_d + K_f * ssa( theta_ref - theta_f ) );
%
% where K_f > 0. The NED tracking errors are rotated an azimuth angle
% pi_h about the z axis and an elevation angle pi_v about the resulting
% y axis from the first rotation using
%
%   e = Ry' * Rz' * [x-xk, y-yk, z-zk]'
```

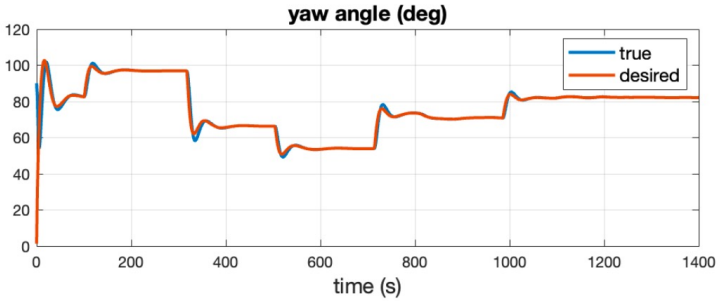
MSS Toolbox: SIMRemus100ALOS.m

Contents

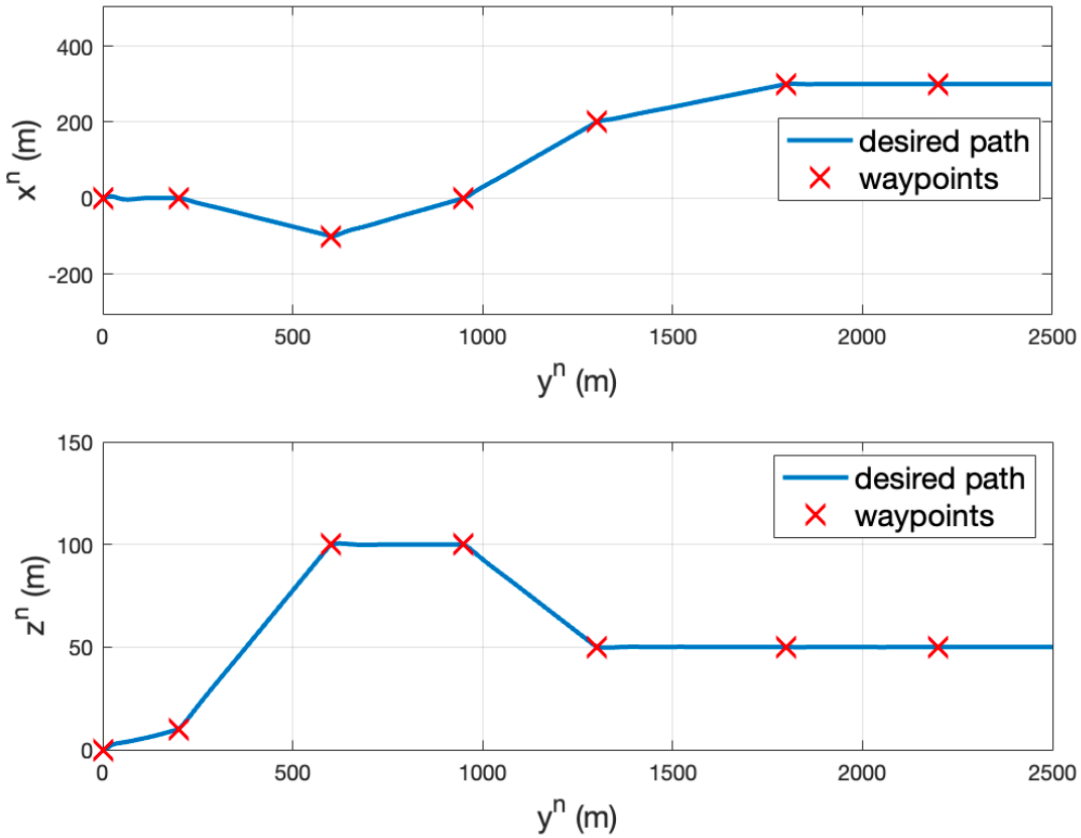



```
% SIMremus100ALOS
% User editable script for 3-D path-following of the Remus 100 AUV
% (remus100.m). The heading and pitch angles are controlled using two
% autopilots designed as PID controllers. The path is represented by
% straigh lines going through waypoints. The heading and pitch commands
% are computed using the 3-D adaptive line-of-sight (ALOS) algorithm by
%
% T. I. Fossen and A. P. Aguiar (2023). A Uniform Semiglobal Exponential
% Stable Adaptive Line-of-Sight (ALOS) Guidance Law for 3-D Path Following.
% Automatica (submitted).
%
% Calls:      remus100.m     Remus 100 vehicle dynamics
%             ALOS3D.m       3-D adaptive LOS guidance law
%
% Author:     Thor I. Fossen
% Date:       2023-10-07
```

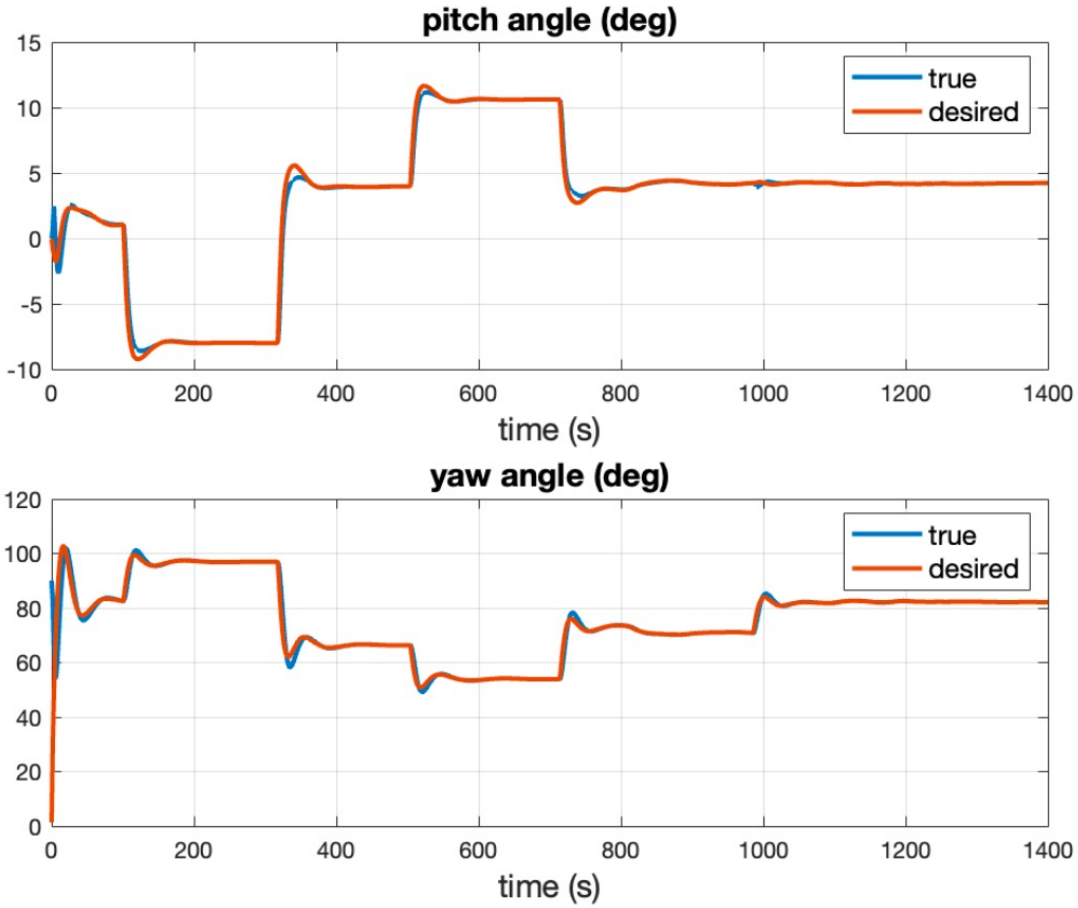




AUV Case Study, cont.

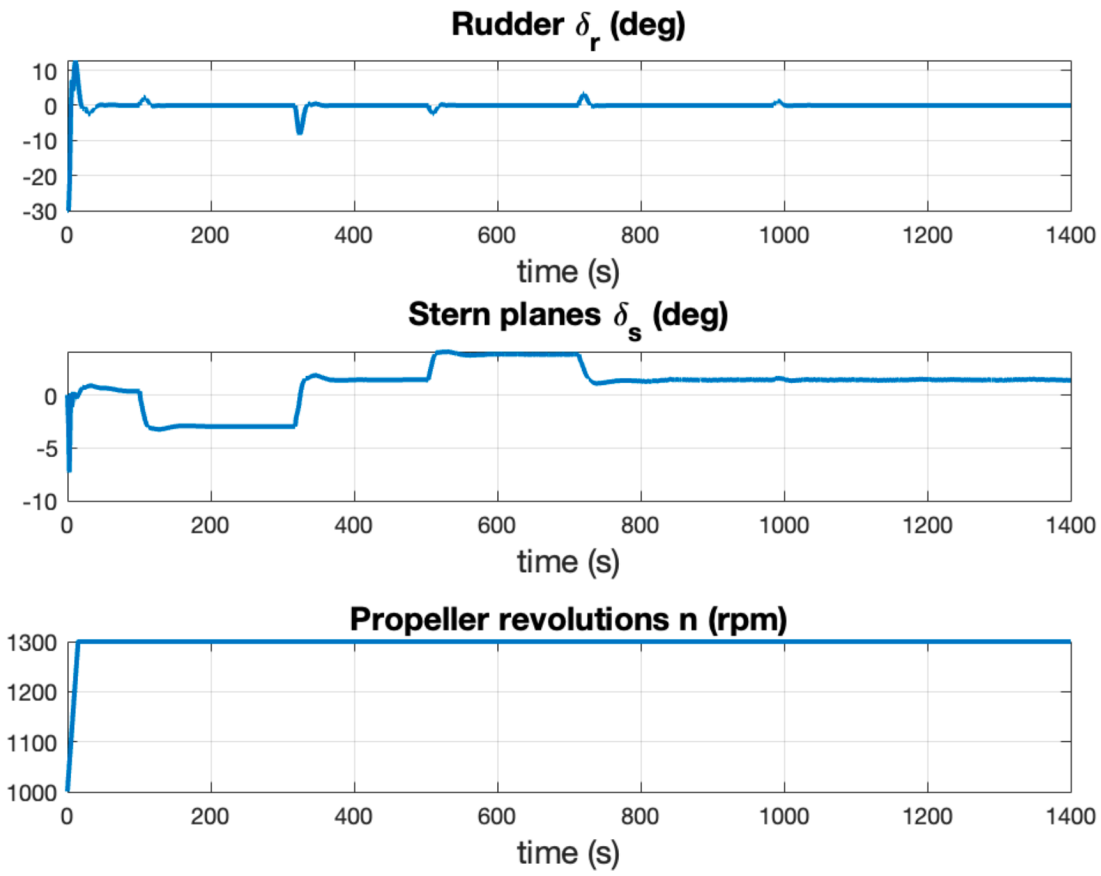


Desired path and waypoints in the horizontal and vertical planes. The maximum depth of the vehicle is 100 m.

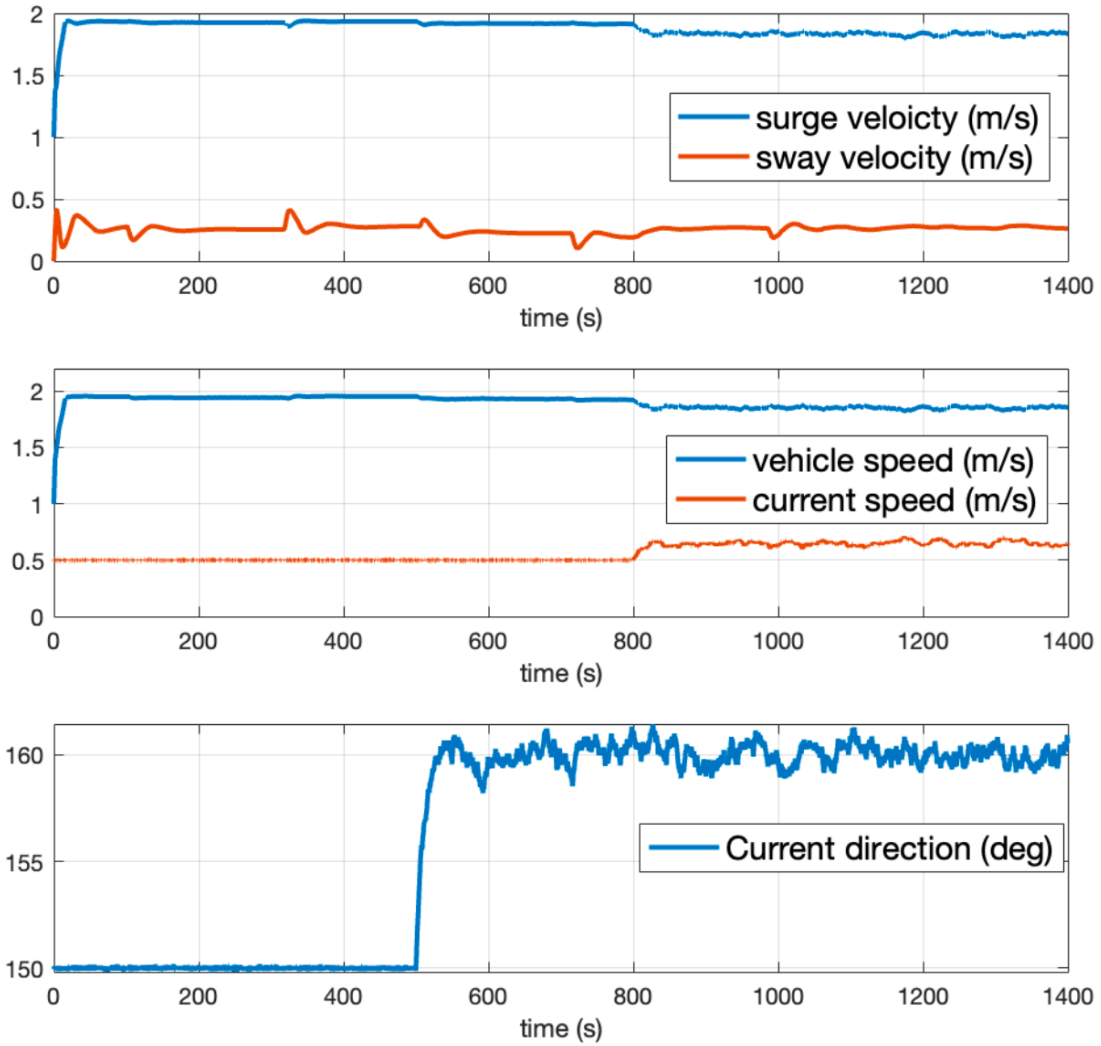


ALOS desired pitch and yaw angles together with their true values obtained by using the depth and heading autopilots.

AUV Case Study, cont.



Rudder angle, stern angle, and propeller revolutions versus time.



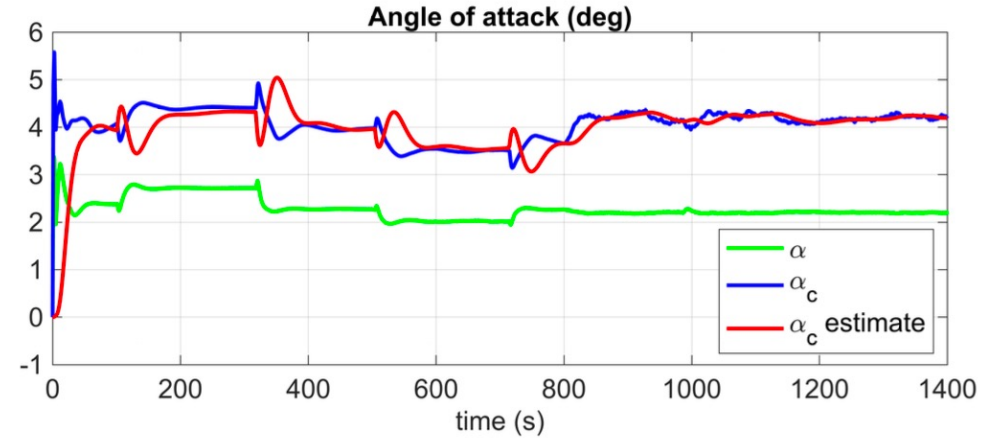
Vehicle surge and sway velocities, speed and ocean current speed/direction in the horizontal plane versus time.

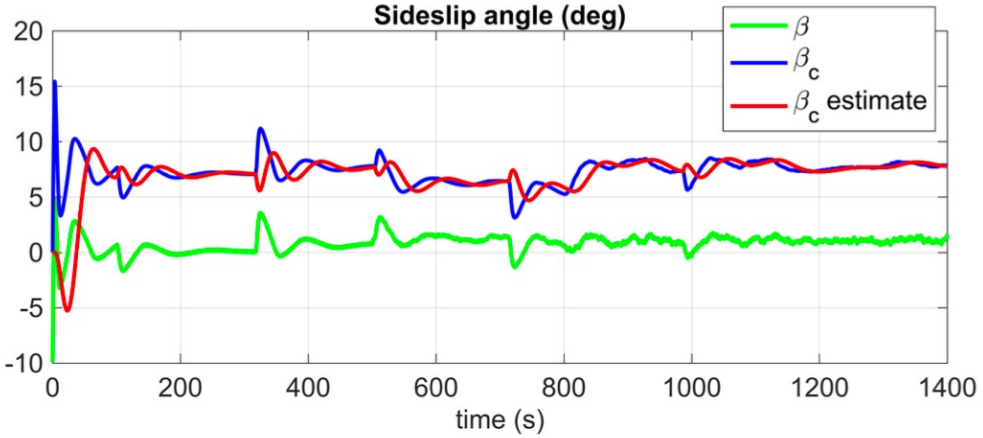
AUV Case Study, cont.

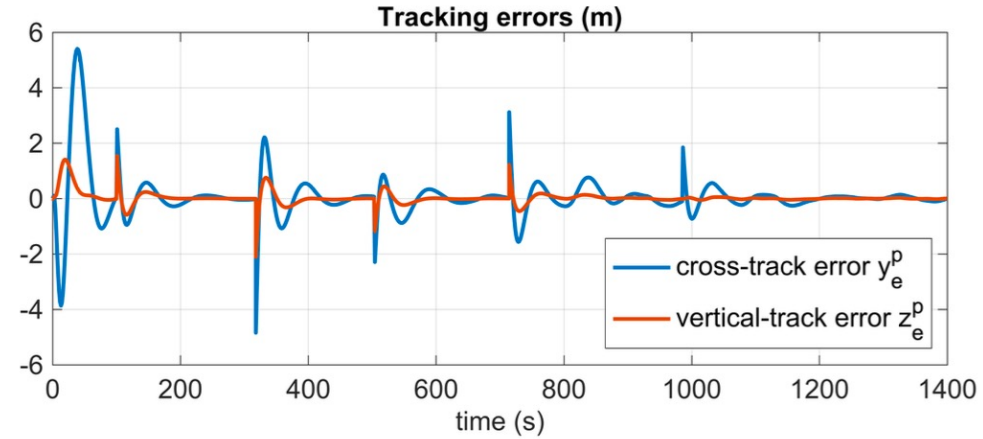
The parameter estimates $\hat{\alpha}_c$ and $\hat{\beta}_c$ track the true parameters α_c and β_c with excellent accuracy, also when the current speed and direction change rapidly.

Note the angle of attack α and β are different from α_c and β_c, and that the vehicle will sideslip with an angle $\beta_c \approx 8.0$ degrees after 1400 s.

The accuracy of the ALOS guidance law is demonstrated by showing the convergence of y_e^p and z_e^p to zero. The initial transient is caused by a 90 degrees initial heading error, while the jumps in the tracking errors are due to waypoint switching.





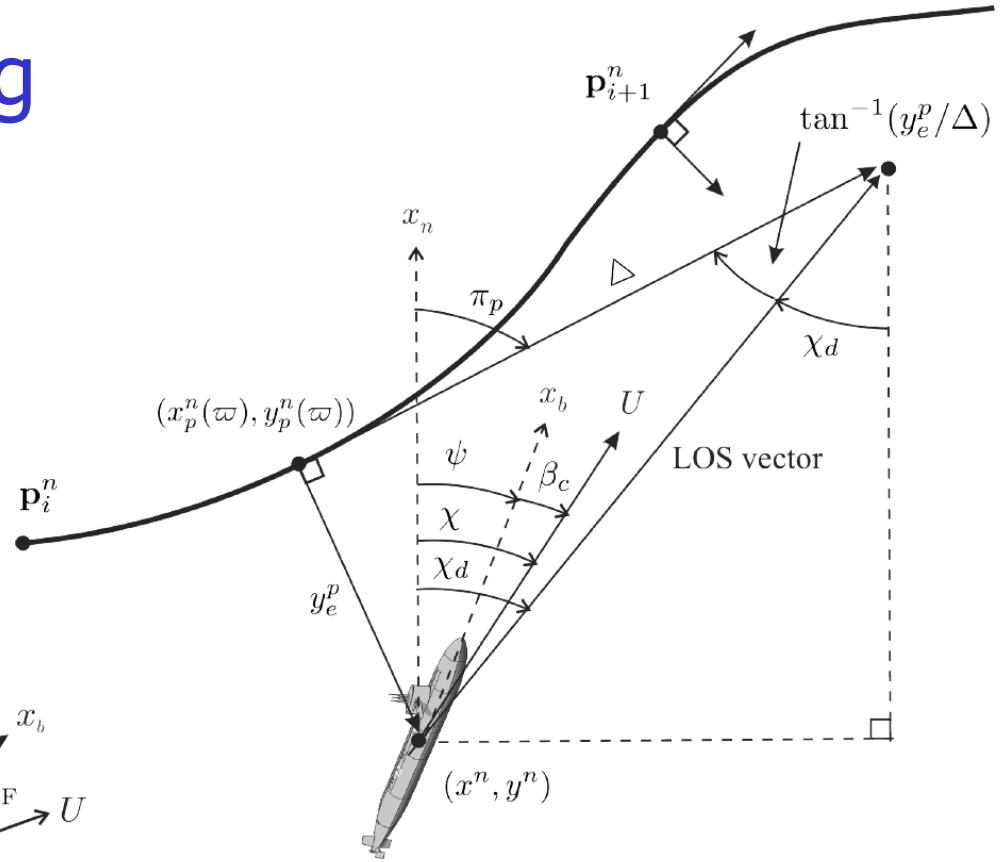


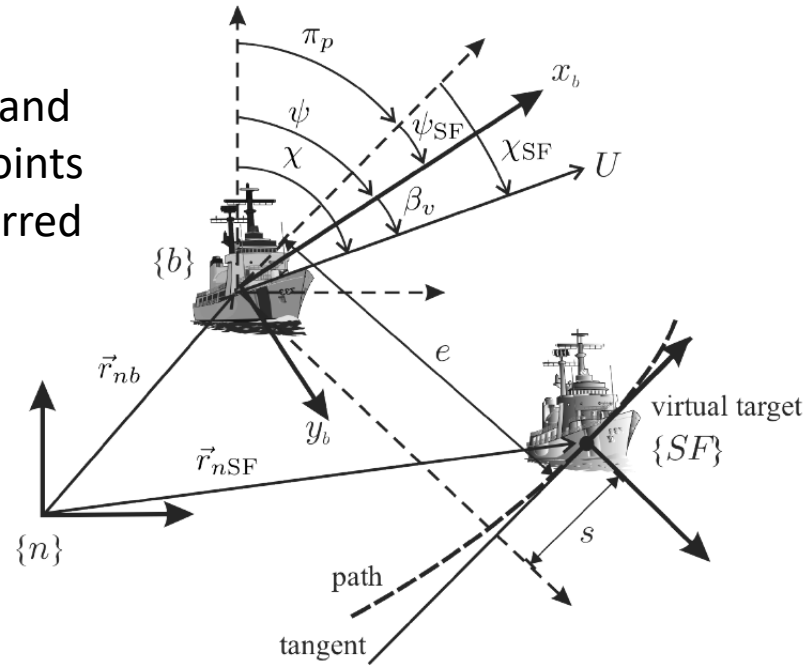
12.6 Curved-Path Path Following

A guidance systems can make use of a predefined parametrized path for which the curvature is known. Methods for this are described by Fossen (2021, Ch. 12.6):

12.6.1 Path generation using interpolation methods
12.6.2 Proportional LOS guidance law for curved paths
12.6.3 Path-following using Serret-Frenet coordinates
12.6.4 Case study: Path-following control using *Serret-Frenet* coordinates

In many cases this is not practical and a simpler path consisting of waypoints and straight lines will be the preferred solution, see Sections 12.3-12.5.





Chapter Goals – Revisited

Guidance systems:

- Be able to design open-loop guidance systems using reference models (low-pass filters) for position and velocity control.
- Understand how to design a closed-loop guidance system using feedback control theory.

Guidance Laws:

- Be able to design guidance laws for target tracking, that is a moving object for which no future motion information, nor the path is known in advance.
- Be able to design a state-of-the-art linear path-following controller. This is the classical design method where the path is generated using waypoints to connect straight lines between the waypoints.
- Be able to design LOS guidance laws for path following using course autopilots.
- Be able to design LOS guidance laws for path following using heading autopilots, with extension to ILOS to compensate for sideslip and unmodelled dynamics.