# 10

# Guidance Systems

This chapter describes methods for the design of *guidance systems* for marine craft (Siouris, 2004, Yanushevsky, 2008). Guidance can be defined as (Shneydor, 1998): "*The process for guiding the path of an object towards a given point, which in general may be moving*." Draper (1971) states: "*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.*" Thus, guidance represents a basic methodology concerned with the transient motion behavior associated with the achievement of motion control objectives (see Breivik and Fossen, 2009).

In its simplest form, open-loop guidance systems for marine craft are used to generate a reference trajectory for time-varying *trajectory tracking* or, alternatively, a path for time-invariant *path following* (see Section 9.2). A motion control system will work in close interaction with the guidance system.

In the control literature, the different motion control scenarios are typically classified according to:

- *Setpoint regulation (point stabilization)* is a special case where the desired position and attitude are chosen to be constant.
- *Trajectory tracking,* where the objective is to force the system output $\boldsymbol{y}(t) \in \mathbb{R}^m$ to track a desired output $\boldsymbol{y}_d(t) \in \mathbb{R}^m$. The desired trajectory can be computed using reference models generated by low-pass filters, optimization methods or by simply simulating the marine craft motion using an adequate model of the craft. Feasible trajectories can be generated in the presence of both *spatial* and *temporal constraints.*
- *Path following* is following a predefined path independent of time. 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.

Tracking control systems can also be designed for target tracking and path tracking. For instance, a target-tracking system tracks the motion of a target that is either stationary (analogous to point stabilization) or that moves such that only its instantaneous motion is known; that is no information about the future target motion is available (Breivik and Fossen, 2009).

As shown in Figure 10.1, the guidance system can use joystick or keyboard inputs, external inputs (weather data, for instance measured wind, wave and current speeds and directions), Earth topological information (digital chart, radar and sonar data), obstacle and collision avoidance data, and finally the state vector, which is available as output from the navigation and sensor systems. The required data are further processed to generate a feasible trajectory for motion control. This can be done using ad hoc techniques or sophisticated methods such as interpolation techniques, dynamic optimization or filtering

*Handbook of Marine Craft Hydrodynamics and Motion Control,* First Edition. Thor I. Fossen.
 ISBN: 978-1-119-99149-6

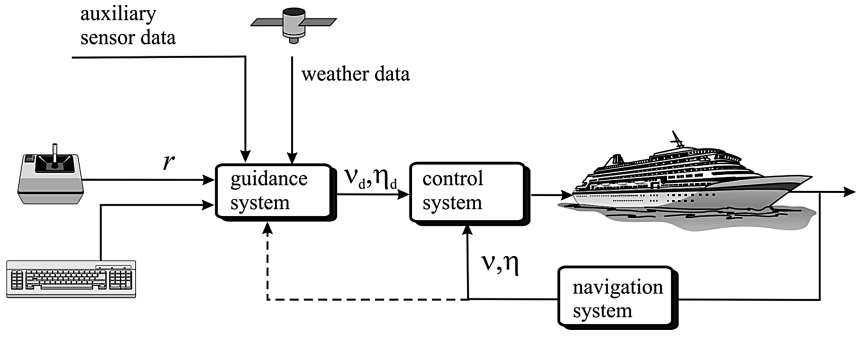


**Figure 10.1** In closed-loop guidance (dotted line) the states are fed back to the guidance system while open-loop guidance only uses sensor and reference signal inputs.

techniques. A feasible trajectory means one that is consistent with the craft dynamics. For a linear system, this implies that the eigenvalues of the desired states must be chosen such that the reference model is slower than the craft dynamics.

For a ship or an underwater vehicle, the guidance and control system usually consists of the following subsystems:

- *Attitude control system*
- *Path-following control system*

In its simplest form, the attitude control system is a heading autopilot, while roll and pitch are regulated to zero or left uncontrolled. The main function of the attitude feedback control system is to maintain the craft in a desired attitude on the ordered path by controlling the craft in roll, pitch and yaw. The task of the path-following controller is to keep the craft on the predescribed path with some predefined dynamics, for instance a speed control system by generating orders to the attitude control system. For surface vessels it is common to use a heading controller in combination with a speed controller while aircraft and underwater vehicles also need a height/depth controller. The principles and definitions of *guidance, navigation* and *control* are further outlined in Section 9.2.

## 10.1 Target Tracking

Sometimes no information about the path is known in advance and there is no trajectory to track. Hence, if the goal is to track a moving object, for which no future motion information is available, target-tracking methods can be applied. 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 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 (Skejic *et al.*, 2009).

For surface vessels, the 2-D position of the target is denoted by $\boldsymbol{p}_t^n = [N_t, E_t]^\top$. The control objective of a target-tracking scenario can be formulated as (Breivik and Fossen, 2009)

$$\lim_{t\to\infty} [\boldsymbol{p}^n(t) - \boldsymbol{p}_t^n(t)] = \boldsymbol{0} \tag{10.1}$$

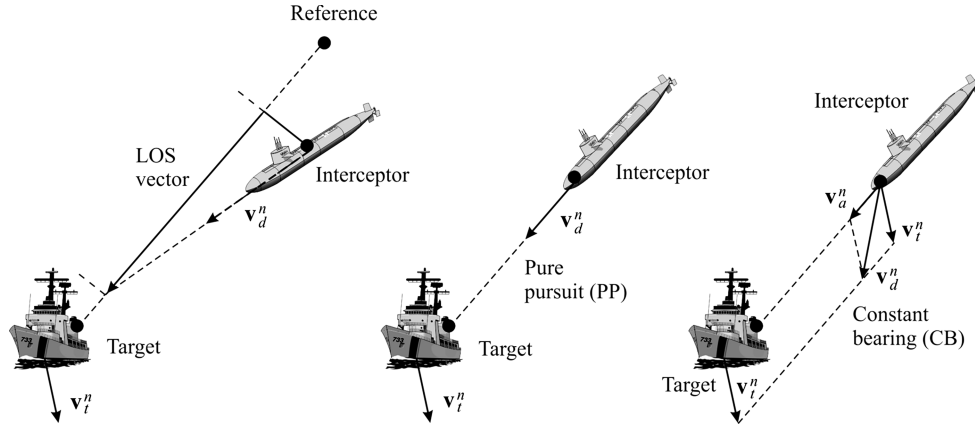


**Figure 10.2** Desired interceptor approach speed $U_d = \|\boldsymbol{v}_d^n\|$ for the classical guidance principles: line-of-sight LOS, pure pursuit (PP) and constant bearing (CB). The target speed is $U_t = \|\boldsymbol{v}_t^n\|$.

where $\boldsymbol{p}^n \in \mathbb{R}^2$ is the craft position. The target velocity is $\boldsymbol{v}_t^n = \dot{\boldsymbol{p}}_t^n \in \mathbb{R}^2$. In the missile guidance community an object that is supposed to destroy another object is referred to as a missile, an interceptor or a pursuer. Conversely, the threatened object is typically called a target or an evader. In the following, the designations *interceptor* and *target* will be used.

An interceptor typically undergoes three phases during its operation:

1. Launch phase
2. Midcourse phase
3. Terminal phase

The greatest accuracy demand is associated with the terminal phase, where the interceptor guidance system must compensate for the accumulated errors from the previous phases to achieve a smallest possible final miss distance to the target. The remainder of this section is based on Breivik and Fossen (2009) and Breivik (2010). The discussion is limited to three terminal guidance strategies, *line-of-sight*, *pure pursuit* and *constant bearing*, which are illustrated in Figure 10.2.

Note that while the main objective of a guided missile is to hit and destroy a physical target in finite time. the analogy is to hit or converge to a virtual target asymptotically. This is also referred to as asymptotic interception given by (10.1).

### *10.1.1 Line-of-Sight Guidance*

*Line-of-sight (LOS) guidance* is classified as a three-point guidance scheme since it involves a typically stationary reference point in addition to the interceptor and the target. The LOS denotation stems from the fact that the interceptor is supposed to achieve an intercept by constraining its motion along the LOS vector between the reference point and the target. LOS guidance has typically been employed for surface-to-air missiles, often mechanized by a ground station, which illuminates the target with a beam that the guided missile is supposed to ride, also known as beam-rider guidance. The LOS guidance principle is illustrated in Figure 10.2, where the interceptor velocity $\boldsymbol{v}_a^n$ is pointed to LOS vector to obtain the desired velocity $\boldsymbol{v}_d^n$. LOS guidance will be applied to track straight-line paths in section 10.3 while curved paths are discussed in Section 10.4.

### 10.1.2 Pure Pursuit Guidance

*Pure pursuit (PP) guidance* belongs to the two-point guidance schemes, where only the interceptor and the target are considered in the engagement geometry. The interceptor aligns its velocity $\boldsymbol{v}_a^n$ along the LOS vector between the interceptor and the target by choosing the desired velocity as

$$\boldsymbol{v}_d^n = -\kappa \frac{\tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|} \tag{10.2}$$

where $\kappa > 0$. This strategy is equivalent to a predator chasing a prey in the animal world, and very often results in a tail chase. PP guidance has typically been employed for air-to-surface missiles. The PP guidance principle is represented in Figure 10.2 by a vector pointing directly at the target.

Deviated pursuit guidance is a variant of PP guidance, where the velocity of the interceptor is supposed to lead the interceptor–target line of sight by a constant angle in the direction of the target movement. An equivalent term is fixed-lead navigation.

### 10.1.3 Constant Bearing Guidance

*Constant bearing (CB) guidance* is also a two-point guidance scheme, with the same engagement geometry as PP guidance. However, in a CB engagement, the interceptor is supposed to align the interceptor–target velocity $\boldsymbol{v}_a^n$ along the LOS vector between the interceptor and the target. This goal is equivalent to reducing the LOS rotation rate to zero such that the interceptor perceives the target at a constant bearing, closing in on a direct collision course. CB guidance is often referred to as *parallel navigation* and has typically been employed for air-to-air missiles. Also, the CB rule has been used for centuries by mariners to avoid collisions at sea, steering away from a situation where another craft approaches at a constant bearing. Thus, guidance principles can just as well be applied to avoid collisions as to achieve them. The CB guidance principle is indicated in Figure 10.2.

The most common method of implementing CB guidance is to make the rotation rate of the interceptor velocity directly proportional to the rotation rate of the interceptor–target LOS, which is widely known as *proportional navigation* (PN). However, CB guidance can also be implemented through the direct velocity assignment as proposed by Breivik *et al.* (2006); see Breivik (2010) for details.

The CB desired velocity is given by

$$\boldsymbol{v}_d^n = \boldsymbol{v}_t^n + \boldsymbol{v}_a^n \tag{10.3}$$

$$\boldsymbol{v}_a^n = -\kappa \frac{\tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|} \tag{10.4}$$

where $\boldsymbol{v}_a^n = [\dot{N}_a, \dot{E}_a]^\top$ is the approach velocity vector specified such that the desired approach speed $U_a = \left\|\boldsymbol{v}_a^n\right\|$ is tangential to the LOS vector as shown in Figure 10.2 and

$$\tilde{\boldsymbol{p}}^n := \boldsymbol{p}^n - \boldsymbol{p}_t^n \tag{10.5}$$

is the LOS vector between the interceptor and the target, $\|\tilde{\boldsymbol{p}}^n\| \geq 0$ is the Euclidean length of this vector and

$$\kappa = U_{\mathrm{a,max}} \frac{\|\tilde{\boldsymbol{p}}^n\|}{\sqrt{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n + \triangle_{\tilde{\mathrm{p}}}^2}} \tag{10.6}$$

where $U_{\mathrm{a,max}}$ specifies the maximum approach speed toward the target and $\triangle_{\tilde{\mathrm{p}}} > 0$ affects the transient interceptor–target rendezvous behavior. The CB guidance law (10.3) computes the velocity commands needed to track the target.

Note that CB guidance becomes equal to PP guidance for a stationary target; that is the basic difference between the two guidance schemes is whether the target velocity is used as a kinematic feedforward or not.

**Convergence and Stability Analyses**

The convergence properties of (10.3)–(10.4) and (10.6) can be investigated by considering a Lyapunov function candidate (LFC):

$$V = \frac{1}{2}(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n > 0, \quad \forall \tilde{\boldsymbol{p}}^n \neq \boldsymbol{0} \tag{10.7}$$

Time differentiation of $V$ along the trajectories of $\tilde{\boldsymbol{p}}^n$ gives

$$\begin{aligned} \dot{V} &= (\tilde{\boldsymbol{p}}^n)^\top \, \tilde{\boldsymbol{v}}^n \\ &= -\kappa \frac{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n}{\|\tilde{\boldsymbol{p}}^n\|} \\ &= -U_{a,\max} \frac{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n}{\sqrt{(\tilde{\boldsymbol{p}}^n)^\top \tilde{\boldsymbol{p}}^n + \triangle_{\tilde{\mathrm{p}}}^2}} \\ &< 0, \quad \forall \tilde{\boldsymbol{p}}^n \neq \boldsymbol{0} \end{aligned} \tag{10.8}$$

The LFC (10.7) is positive definite and radially unbounded, while its derivative with respect to time (10.8) is negative definite when adhering to $U \geq U_{a,\max} > 0$. Hence, by standard Lyapunov arguments the origin $\tilde{\boldsymbol{p}}^n = \boldsymbol{0}$ is UGAS (see Appendix A.1). Finally, the Jacobian of the error dynamics $\tilde{\boldsymbol{p}}^n$ at the origin $\tilde{\boldsymbol{p}}^n = \boldsymbol{0}$ has strictly negative eigenvalues, which proves ULES.

***Example 10.1 (UNREP Operation)***
*For ships equipped with a rudder and a main propeller, the rudder can be used to obtain a parallel course with the guide ship (lateral alignment). We will consider a guide ship moving on a straight line and the goal for the second ship is to approach this ship to carry out a UNREP operation. In Skejic* et al. *(2009) the lookahead-based steering law of Breivik and Fossen (2009) in Section 10.3 is used to ensure that the approach ship is able to assume a parallel course with the guide ship by adhering to the desired course angle*

$$\chi_d = \chi_t + \chi_r \tag{10.9}$$

*where $\chi_t$ is the course angle of the guide ship. The heading of the approach ship is adjusted using the steering law $\chi_r$ such that the lateral distance is adjusted as desired. The lateral distance and cross-track error $(s, e)$ are obtained by the following transformation:*

$$\begin{bmatrix} s \\ e \end{bmatrix} = \begin{bmatrix} s_d \\ e_d \end{bmatrix} + \boldsymbol{R}_p(\chi_t)^\top (\boldsymbol{p}^n - \boldsymbol{p}_t^n) \tag{10.10}$$

*where $s_d = 0$ (interceptor on parallel course), $e_d =$ constant (distance between interceptor and target) and*

$$\boldsymbol{R}_p(\chi_t) = \begin{bmatrix} \cos(\chi_t) & -\sin(\chi_t) \\ \sin(\chi_t) & \cos(\chi_t) \end{bmatrix} \in SO(2) \tag{10.11}$$

*Hence, the LOS steering law can be chosen as*

$$\chi_r = \arctan(-e/\Delta_e) \tag{10.12}$$

*where the steering law tuning parameter* $\Delta_e > 0$ *represents the lookahead distance. This parameter is given in meters and usually takes values between* 1.5 *and* 2.5 *of a ship length* $L_{pp}$*. Finally, the desired heading angle for the approaching ship is input to the ship autopilot, suggesting that*

$$\psi_d = \chi_d - \beta \tag{10.13}$$

*where the sideslip (drift) angle is*

$$\beta = \arcsin\left(\frac{v}{U}\right) \tag{10.14}$$

*The speed command* $U_d = \sqrt{u_d^2 + v_d^2} \approx u_d$ *(assuming that* $u_d \gg v_d$*) is computed according to (10.3), (10.4) and (10.6) such that*

$$U_d = U_t - \kappa \frac{s}{\sqrt{s^2 + \Delta_s^2}} \tag{10.15}$$

*where* $\kappa = U_{a,\max}$ *and*

$$U_t = \sqrt{u_t^2 + v_t^2} \tag{10.16}$$

*The speed tuning parameter* $\Delta_s > 0$ *specifies the rendezvous behavior towards the projection of the guide ship on to the parallel course defined by* $e_d$*, ensuring that the approach ship smoothly ramps down its total speed to* $U_t$ *as the along-course distance goes to zero.*

## 10.2 Trajectory Tracking

Guidance systems designed for tracking a smooth time-varying trajectory $\boldsymbol{y}_d(t) \in \mathbb{R}^m$ are useful in many applications. The desired speed and acceleration are obtained from time-differentiation of $\boldsymbol{y}_d(t)$ one and two times, respectively. This means that the signal $\boldsymbol{y}_d(t)$ defines the desired position/attitude, velocity and acceleration as a function of time $t$ for a moving craft in 6 DOF. We will make use of the following definition in the forthcoming:

***Definition 10.1 (Trajectory Tracking)***
*A control system that forces the system output* $\boldsymbol{y}(t) \in \mathbb{R}^m$ *to track a desired output* $\boldsymbol{y}_d(t) \in \mathbb{R}^m$ *solves a trajectory tracking problem.*

This definition is consistent with Athans and Falb (1966) and later with Hauser and Hindmann (1995), Ortega *et al.* (1998), Encarnacao and Pascoal (2001) and Skjetne *et al.* (2002, 2004). In this section, methods for computation of the desired trajectory corresponding to a desired virtual target will be presented. The following methods are discussed:

- Low-pass filters for the generation of position, velocity and acceleration (PVA) trajectories
- Time-domain simulation using an adequate model of the craft
- Optimization methods

Within this framework, it is possible to generate feasible trajectories incorporating both *spatial constraints* (obstacle avoidance and maximum velocity/acceleration) and *temporal constraints* (minimum time, on time and maximum time problems)

**Trajectory-Tracking Control**

Trajectory-tracking control laws are classified according to the number of available actuators. This can be illustrated by considering a marine craft in *surge, sway* and *yaw*. Tracking of a *time-varying reference trajectory* $\boldsymbol{\eta}_d(t) = [N_d(t), E_d(t), \psi_d(t)]^\top$ is achieved by minimizing the tracking error, $\boldsymbol{e}(t) := \boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(t)$. Moreover,

$$\boldsymbol{e}(t) := \begin{bmatrix} N(t) - N_d(t) \\ E(t) - E_d(t) \\ \psi(t) - \psi_d(t) \end{bmatrix} \tag{10.17}$$

Based on this interpretation, the following considerations can be made (see Section 9.3):

- **Three or more controls:** This is referred to as a *fully actuated* dynamic positioning (DP) system and typical applications are crab wise motions (low-speed maneuvering) and stationkeeping, where the goal is to drive $\boldsymbol{e}(t) \in \mathbb{R}^2 \times S \to \mathbf{0}$. This is the standard configuration for offshore DP vessels. Feedback control laws for fully actuated vessels are discussed in Chapter 12.
- **Two controls and trajectory-tracking control:** Trajectory tracking in 3 DOF, $\boldsymbol{e}(t) \in \mathbb{R}^2 \times S$, 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. However, since all marine craft operate in a uniform force field due to mean wind, waves and ocean currents, it is possible to steer the craft along a path with a constant sideslip angle (given by the mean environmental force field) using only two controls, that is turning and forward speed control. This is the classical approach used in path-following control (see Section 12.2.8).
- **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 positions. In this approach the heading angle is allowed to vary automatically with the mean environmental forces (Pinkster, 1971, Pinkster and Nienhuis, 1986, Fossen and Strand, 2001) (see Section 13.3.10).
- **Two controls and path-following control:** 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 an LOS path-following controller; see Sections 10.3–10.4 and 12.2.8–12.2.9. 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 DOF for a marine craft using only one control.

For underwater vehicles operating in 6 DOF it is also important to control the heave and sometimes the pitch-roll motions in addition to the surge, sway and yaw motions. However, roll and pitch can be left uncontrolled for metacentrically stable vehicles. For operation in 6 DOF, a fully actuated vehicle must have six or more actuators producing independent forces and moments in all directions in order to track a 6 DOF time-varying reference trajectory.

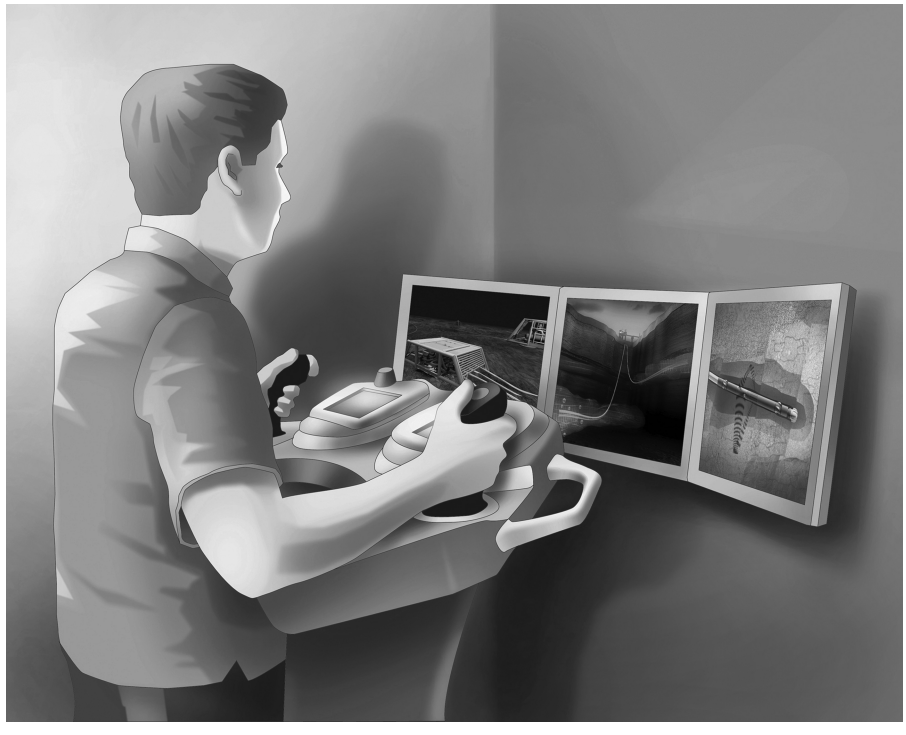

**Figure 10.3** Joystick control system used to generate reference signals. Illustration by Bjarne Stenberg/SINTEF.

### *10.2.1 Reference Models for Trajectory Generation*

In a practical system, it is highly advantageous to keep the software as simple as possible. As a result of this, many industrial systems are designed using linear reference models for trajectory generation. This corresponds to *open-loop guidance* as described in Section 9.2 since no feedback from the states is required. The simplest form of a reference model is obtained by using a low-pass (LP) filter structure:

$$\frac{x_d}{r}(s) = h_{lp}(s) \tag{10.18}$$

where $x_d$ is the desired state and $r$ denotes the reference signal usually specified by an operator (see Figure 10.3). The choice of filter should reflect the dynamics of the craft such that a feasible trajectory is constructed. For instance, it is important to take into account physical speed and acceleration limitations of the craft as well as input saturation. This is a nontrivial task so a compromise between performance and accurate tracking must be made by tuning the bandwidth of the reference model. It is important that the bandwidth of the reference model is chosen lower than the bandwidth of the motion control system in order to obtain satisfactory tracking performance and stability.

A frequently used method to generate a smooth reference trajectory $\boldsymbol{x}_d \in \mathbb{R}^n$ for tracking control is to use a physically motivated model. For marine craft it is convenient to use reference models

motivated by the dynamics of *mass–damper–spring systems* to generate the desired state trajectories, for instance

$$h_{lp}(s) = \frac{\omega_{n_i}^2}{s^2 + 2\zeta_i \omega_{n_i} s + \omega_{n_i}^2} \tag{10.19}$$

where $\zeta_i$ $(i = 1, \ldots, n)$ are the *relative damping ratios* and $\omega_{n_i}$ $(i = 1, \ldots, n)$ are the *natural frequencies*. For a 6 DOF system, the desired states can be expressed by a MIMO mass–damper–spring system:

$$\boldsymbol{M}_d \ddot{\boldsymbol{\eta}}_d + \boldsymbol{D}_d \dot{\boldsymbol{\eta}}_d + \boldsymbol{G}_d \boldsymbol{\eta}_d = \boldsymbol{G}_d \boldsymbol{r} \tag{10.20}$$

where $\boldsymbol{M}_d$, $\boldsymbol{D}_d$ and $\boldsymbol{G}_d$ are positive design matrices specifying the desired dynamics of the system. The model (10.20) can also be represented as a linear time invariant (LTI) system:

$$\dot{\boldsymbol{x}}_d = \boldsymbol{A}_d \boldsymbol{x}_d + \boldsymbol{B}_d \boldsymbol{r} \tag{10.21}$$

where $\boldsymbol{x}_d := [\boldsymbol{\eta}_d^\top, \dot{\boldsymbol{\eta}}_d^\top]^\top \in \mathbb{R}^{2n}$ is the desired state vector, $\boldsymbol{r} \in \mathbb{R}^r$ $(r \leq n)$ is a bounded reference vector usually generated by a joystick or a keyboard. The state and input matrices are recognized as

$$\boldsymbol{A}_d = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{M}_d^{-1}\boldsymbol{G}_d & -\boldsymbol{M}_d^{-1}\boldsymbol{D}_d \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}_d^{-1}\boldsymbol{G}_d \end{bmatrix} \tag{10.22}$$

**Velocity Reference Model**

The velocity reference model should at least be of order two so as to obtain smooth reference signals for the desired velocity $\boldsymbol{v}_d$ and acceleration $\dot{\boldsymbol{v}}_d$. Let $\boldsymbol{r}^b$ denote the operator input expressed in $\{b\}$. The second-order LP filter (10.19) can be used for this purpose. Let

$$\ddot{\boldsymbol{v}}_d + 2\boldsymbol{\Delta}\boldsymbol{\Omega}\dot{\boldsymbol{v}}_d + \boldsymbol{\Omega}^2 \boldsymbol{v}_d = \boldsymbol{\Omega}^2 \boldsymbol{r}^b \tag{10.23}$$

where $\boldsymbol{v}_d$ is the desired velocity, $\dot{\boldsymbol{v}}_d$ is the desired acceleration and $\ddot{\boldsymbol{v}}_d$ is interpreted as the desired "jerk". For this model, $\boldsymbol{\Delta} > 0$ and $\boldsymbol{\Omega} > 0$ are diagonal design matrices of *relative damping ratios* and *natural frequencies:*

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

The state space representation is

$$\boldsymbol{A}_d = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{\Omega}^2 & -2\boldsymbol{\Delta}\boldsymbol{\Omega} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\Omega}^2 \end{bmatrix} \tag{10.24}$$

Note that a step in the command $\boldsymbol{r}^b$ will give a step in $\ddot{\boldsymbol{v}}_d$ while $\dot{\boldsymbol{v}}_d$ and $\boldsymbol{v}_d$ will be low-pass filtered and therefore smooth signals in a tracking control system. We also notice that the steady-state velocity for a constant reference signal $\boldsymbol{r}^b$ is

$$\lim_{t \to \infty} \boldsymbol{v}_d(t) = \boldsymbol{r}^b \tag{10.25}$$

### Position and Attitude Reference Models

The position and attitude reference model $\boldsymbol{\eta}_d$ is typically chosen to be of third order for filtering the steps in $\boldsymbol{r}^n$. This suggests that a first-order LP filter should be cascaded with a mass–damper–spring system. Moreover, consider the transfer function:

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^2}{(1+T_i s)\left(s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2\right)} \quad (i = 1, \ldots, n) \tag{10.26}$$

where a first-order LP filter with time constant $T_i = 1/\omega_{n_i} > 0$ has been added. This can also be written

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^3}{s^3 + (2\zeta_i+1)\omega_{n_i}s^2 + (2\zeta_i+1)\omega_{n_i}^2 s + \omega_{n_i}^3}, \quad (i = 1, \ldots, n) \tag{10.27}$$

or in a vectorial setting as

$$\boldsymbol{\eta}_d^{(3)} + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\boldsymbol{r}^n \tag{10.28}$$

The state-space representation is

$$\boldsymbol{A}_d = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{\Omega}^3 & -(2\boldsymbol{\Delta}+\boldsymbol{I})\boldsymbol{\Omega}^2 & -(2\boldsymbol{\Delta}+\boldsymbol{I})\boldsymbol{\Omega} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{\Omega}^3 \end{bmatrix} \tag{10.29}$$

In the case of $n$ critically damped systems, $\zeta_i = 1$ $(i = 1, \ldots, n)$, we have $\boldsymbol{\Delta} = \boldsymbol{I}$. Consequently,

$$\boldsymbol{\eta}_d^{(3)} + 3\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + 3\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\boldsymbol{r}^n \tag{10.30}$$

$$\Updownarrow$$

$$\left(s + \omega_{n_i}\right)^3 \eta_{d_i} = \omega_{n_i}^3 r_i^n \quad (i = 1, \ldots, n) \tag{10.31}$$

These reference models models also satisfy

$$\lim_{t\to\infty} \boldsymbol{\eta}_d(t) = \boldsymbol{r}^n \tag{10.32}$$

if $\boldsymbol{r}^n = \text{constant}$.

### Saturating Elements

One drawback with a linear reference model is that the time constants in the model often yield a satisfactory response for one operating point of the system while the response for other amplitudes of the operator input $r_i$ results in completely different behavior. This is due to the exponential convergence of the signals in a linear system. One way to circumvent this problem is to use amplitude gain scheduling so that the reference model design parameters $(\zeta_i, \omega_i)$ are scheduled with respect to the magnitude of the input signal $r_i$.

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

$$\text{sat}(x) = \begin{cases} \text{sgn}(x)x_{\max} & \text{if } |x| \geq x_{\max} \\ x & \text{else} \end{cases} \tag{10.33}$$

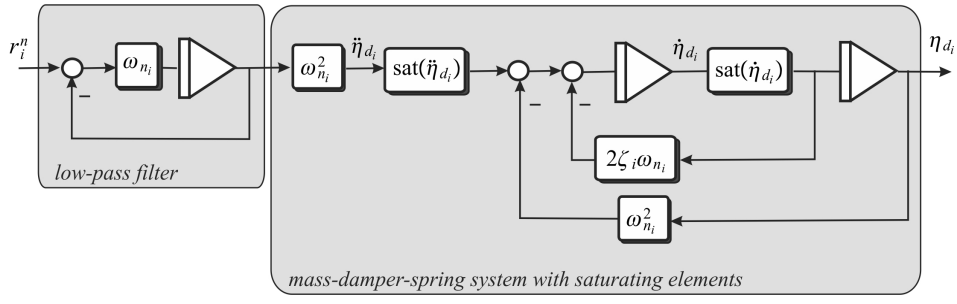


**Figure 10.4** Reference model including saturating elements.

Hence, the saturation limits

$$\nu_i \leq \nu_i^{\max}, \quad \dot{\nu}_i \leq \dot{\nu}_i^{\max} \tag{10.34}$$

should reflect the physical limitations of the craft as illustrated in Example 10.2.

These techniques have been used in model reference adaptive control (MRAC) by Van Amerongen (1982, 1984) and adaptive control of underwater vehicles by Fjellstad *et al.* (1992). The position and attitude reference model should therefore be modified as shown in Figure 10.4.

**Nonlinear Damping**

Nonlinear damping can also be included in the reference model to reduce the velocity for large amplitudes or step inputs $r_i$. This suggests the modified model:

$$\boldsymbol{\eta}_d^{(3)} + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{d}(\dot{\boldsymbol{\eta}}_d) + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\boldsymbol{r}^n \tag{10.35}$$

where the nonlinear function $\boldsymbol{d}(\dot{\boldsymbol{\eta}}_d) = [d_1(\dot{\eta}_{d_1}), \ldots, d_n(\dot{\eta}_{d_n})]^\top$ could be chosen as

$$d_i(\dot{\eta}_{d_i}) = \sum_j \delta_{ij} \left|\dot{\eta}_{d_i}\right|^{p_j} \dot{\eta}_{d_i} \quad (i = 1, \ldots, n) \tag{10.36}$$

where $\delta_{ij} > 0$ are design parameters and $p_j > 0$ are some integers. The effect of nonlinear damping is demonstrated in Example 10.2.

***Example 10.2 (Reference Model)***
*Consider the mass–damper–spring reference model:*

$$\dot{\eta}_d = \nu_d \tag{10.37}$$

$$\dot{\nu}_d + 2\zeta\omega_n\nu_d + \delta\,|\nu_d|\,\nu_d + \omega_n^2\eta_d = \omega_n^2 r \tag{10.38}$$

*where $\zeta = \omega_n = 1$. Figure 10.5 shows a comparison of responses using $\delta = 0$, $\delta = 1$ and a saturating element, $\nu_{\max} = 1$ for an operator step input $r = 10$. The Matlab example file ExRefMod.m in the MSS toolbox was used to generate the plots.*

## *10.2.2 Trajectory Generation using a Marine Craft Simulator*

The reference models in Section 10.2.1 are attractive due to their simplicity. The cutoff frequency of the reference model must never exceed the closed-loop bandwidth of the system in order to guarantee that

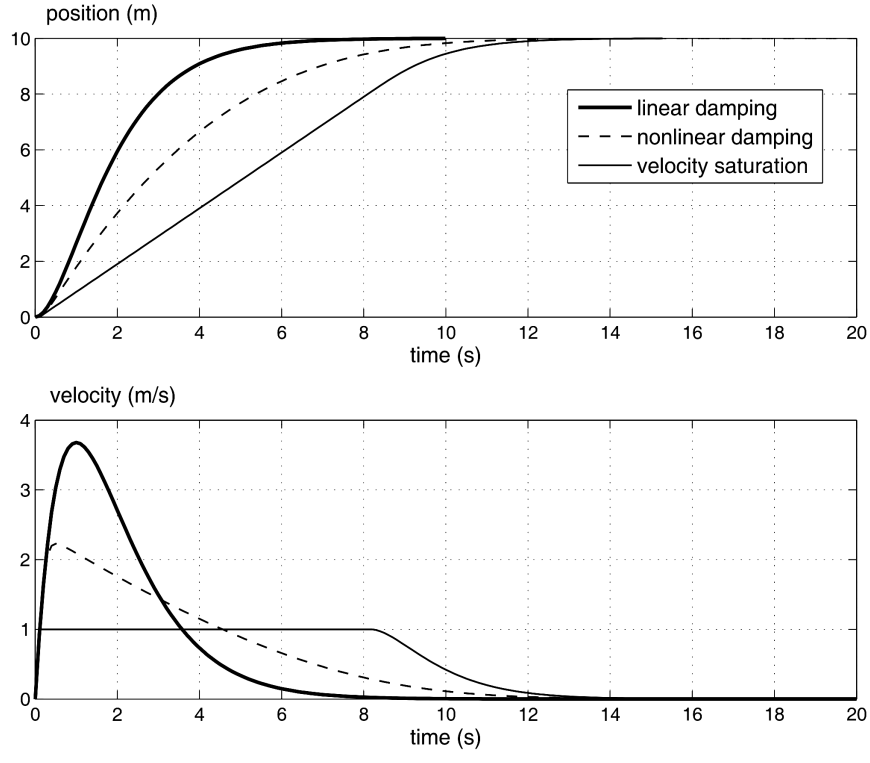


**Figure 10.5** Desired position and velocity for a step input $r = 10$.

the craft is able to track the desired states. This is difficult to verify in a practical system due to factors such as nonlinearities, saturating elements and time delays. An alternative approach could be to 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. For instance, the dynamic model can be chosen as

$$\dot{\boldsymbol{\eta}}_d = \boldsymbol{J}_\Theta(\boldsymbol{\eta}_d)\boldsymbol{\nu}_d \tag{10.39}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_d + \boldsymbol{N}\boldsymbol{\nu}_d + \boldsymbol{g}(\boldsymbol{\eta}_d) = \boldsymbol{\tau} \tag{10.40}$$

where the damping matrix is modeled as a diagonal matrix:

$$\boldsymbol{N} = \text{diag}\{n_1, \ldots, n_6\} > 0 \tag{10.41}$$

The system inertia matrix $\boldsymbol{M}$ is included in the model to guarantee proper scaling of the control inputs $\boldsymbol{\tau}$. Smooth reference trajectories ($\boldsymbol{\eta}_d(t)$, $\boldsymbol{\nu}_d(t)$) are then obtained by simulating the model under closed-loop control, for instance by using a nonlinear PD controller (see Section 12.2):

$$\boldsymbol{\tau} = \boldsymbol{g}(\boldsymbol{\eta}_d) - \boldsymbol{J}_{\Theta}^{\top}(\boldsymbol{\eta}_d)\left[\boldsymbol{K}_p(\boldsymbol{\eta}_d - \boldsymbol{\eta}_{\text{ref}}) + \boldsymbol{K}_d\dot{\boldsymbol{\eta}}_d\right] \tag{10.42}$$

where $\boldsymbol{\eta}_{\text{ref}}$ is the setpoint and $(\boldsymbol{\eta}_d, \boldsymbol{v}_d)$ represents the desired states. The control law (10.42) is in fact a *guidance controller* since it is applied to the reference model. In addition to this, it is useful to include saturation elements for velocity and acceleration to keep these quantities within their physical limits.

***Example 10.3 (Generation of Reference Trajectories using a Marine Craft Simulator)***
*Consider a marine craft moving at forward speed $U \gg 0$ such that $u \approx U$ and $v \approx 0$. The desired reference trajectories can be modeled as*

$$\dot{x}_d = u_d \cos(\psi_d) \tag{10.43}$$

$$\dot{y}_d = u_d \sin(\psi_d) \tag{10.44}$$

*with the surge velocity given by*

$$(m - X_{\dot{u}})\dot{u}_d + \frac{1}{2}\rho C_d A|u_d|u_d = \tau \tag{10.45}$$

*where $u_d \gg 0$ is the desired velocity, $\rho$ is the density of water, $C_d$ is the drag coefficient, $A$ is the projected cross-sectional area of the submerged hull in the $x$ direction and $(m - X_{\dot{u}})$ is the mass including the hydrodynamic added mass. Notice that the ship is moving so fast that quadratic drag dominates and linear damping due to skin friction can be neglected. The yaw dynamics is chosen as a first-order Nomoto model:*

$$\dot{\psi}_d = r_d \tag{10.46}$$

$$T\dot{r}_d + r_d = K\delta \tag{10.47}$$

*where $K$ and $T$ are the design parameters. The guidance system has two inputs, thrust $\tau$ and rudder angle $\delta$. The guidance controllers for speed and yaw angle can be chosen of PI and PID types, respectively:*

$$\tau = -K_{p\tau}(u_d - u_{\text{ref}}) - K_{i\tau}\int_0^t (u_d - u_{\text{ref}})\mathrm{d}\tau \tag{10.48}$$

$$\delta = -K_{p\delta}(\psi_d - \psi_{\text{ref}}) - K_{i\delta}\int_0^t (\psi_d - \psi_{\text{ref}})\mathrm{d}\tau - K_{d\delta}r_d \tag{10.49}$$

*where $\psi_{\text{ref}}$ is generated using an LOS algorithm (see Section 10.3):*

$$\psi_{\text{ref}} = \text{atan2}\left(y_{\text{los}} - y_d(t), x_{\text{los}} - x_d(t)\right) \tag{10.50}$$

*Numerical integration of (10.43)–(10.47) with feedback (10.48)–(10.49) yields a smooth reference trajectory $(x_d, y_d, \psi_d)$ and speed assignment $U_d$.*

### *10.2.3 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 challenge is that an optimization problem must be solved online in order to generate a feasible time-varying trajectory. Implementation and solution of

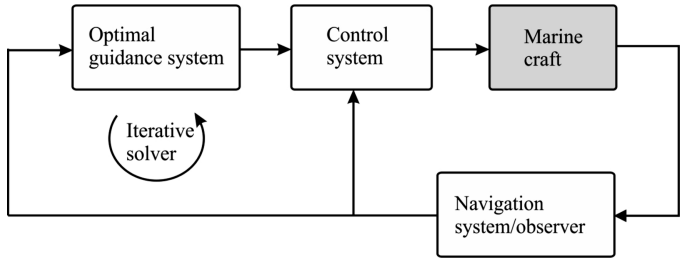


**Figure 10.6** Optimal trajectory generation using an iterative solver to solve a minimum time or minimum power optimization problem.

optimization problems can be done using linear programming (LP), quadratic programming (QP) and nonlinear methods. All these methods require you to have a solver that can be implemented in your program; see Figure 10.6. For testing and development, the different algorithms can be implemented using the optimization toolbox in Matlab. The optimization problem can be formulated as minimum power or minimum time, for instance

$$J = \min_{\eta_d, \nu_d} \{\text{power,time}\} \tag{10.51}$$

subject to

$$|U| \leq U_{\max} \text{ (maximum speed)}$$
$$|r| \leq r_{\max} \text{ (maximum turning rate)}$$
$$|u_i| \leq u_{i,\max} \text{ (saturating limit of control } u_i)$$
$$|\dot{u}_i| \leq \dot{u}_{i,\max} \text{ (saturating limit of rate } \dot{u}_i)$$

which represents the constraints imposed by the vehicle dynamics. It is also possible to add constraints for obstacle avoidance and minimum fuel consumption.

## 10.3 Path Following for Straight-Line Paths

A trajectory describes the motion of a moving object through space as a function of time. The object might be a craft, projectile or a satellite, for example. A trajectory can be described mathematically either by the geometry of the path (see Section 10.4) or as the position of the object over time. *Path following* is the task of following a predefined path independent of time; that is there are no temporal constraints. This means that no restrictions are placed on the temporal propagation along the path. Spatial constraints, however, can be added to represent obstacles and other positional constraints.

A frequently used method for path following is *line-of-sight (LOS) guidance*. A LOS vector from the craft to the next waypoint or a point on the path between two waypoints can be used for both course and heading control. If the craft is equipped with a heading autopilot the angle between the LOS vector and the predescribed path can be used as a setpoint for the heading autopilot. This will force the craft to track the path. Guidance laws composed of *speed* and *LOS steering laws*, which can be combined in various ways to achieve different motion control objectives, are presented in the forthcoming section, which is adapted from Breivik and Fossen (2004b, 2005b, 2009) and Breivik *et al.* (2008).

### 10.3.1 Path Generation based on Waypoints

Systems for waypoint guidance 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. Both trajectory and path-following control systems can be designed for this purpose. Sophisticated features such as weather routing, obstacle avoidance and mission planning can be incorporated in the design of waypoint guidance systems. Some of these features will be discussed in the forthcoming section.

**Waypoint Representation**

The route of a ship or an underwater vehicle is usually specified in terms of waypoints. Each waypoint is defined using Cartesian coordinates $(x_k, y_k, z_k)$ for $k = 1, \ldots, n$. The waypoint *database* therefore consists of

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

For surface craft, only two coordinates $(x_k, y_k)$ are used. Additionally, other waypoint properties such as speed and heading can be defined, that is

$$\text{wpt.speed} = \{U_0, U_1, \ldots, U_n\}$$

$$\text{wpt.heading} = \{\psi_0, \psi_1, \ldots, \psi_n\}$$

For surface craft this means that the craft should pass through waypoint $(x_i, y_i)$ at forward speed $U_i$ with heading angle $\psi_i$. The three states $(x_i, y_i, \psi_i)$ are also called the *pose*. The heading angle is usually unspecified during cross-tracking, whereas it is more important during a crab wise maneuver close to offshore installations (dynamic positioning).

The waypoint database can be generated using many criteria. These are usually based on:

- **Mission:** The craft should move from some starting point $(x_0, y_0, z_0)$ 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 ocean currents can be used for energy optimal routing (or avoidance of bad weather for safety reasons).
- **Geographical data:** Information about shallow waters and islands should be included.
- **Obstacles:** Floating constructions and other obstacles must be avoided.
- **Collision avoidance:** Avoiding moving craft 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 the maximum speed and turning rate.

Online replanning can be used to update the waypoint database in case of time-varying conditions such as changing weather or moving craft (collision avoidance). Optimality with regard to weather is discussed in Section 10.4.1. This is referred to as weather routing.

**Path Generation using Straight Lines and Circular Arcs**

In practice it is common to represent the desired path using straight lines and circle arcs to connect the waypoints, as shown in Figure 10.7. This is related to the famous result of Dubins (1957), which can be summarized as:

> *The shortest path (minimum time) between two configurations $(x, y, \psi)$ of a craft moving at constant speed $U$ is a path formed by straight lines and circular arc segments.*

Since a craft and not a point mass is considered, the start and end configurations of the craft are specified in terms of the positions $(x, y)$, heading angle $\psi$ and speed $U$. In addition, it is assumed that there are bounds

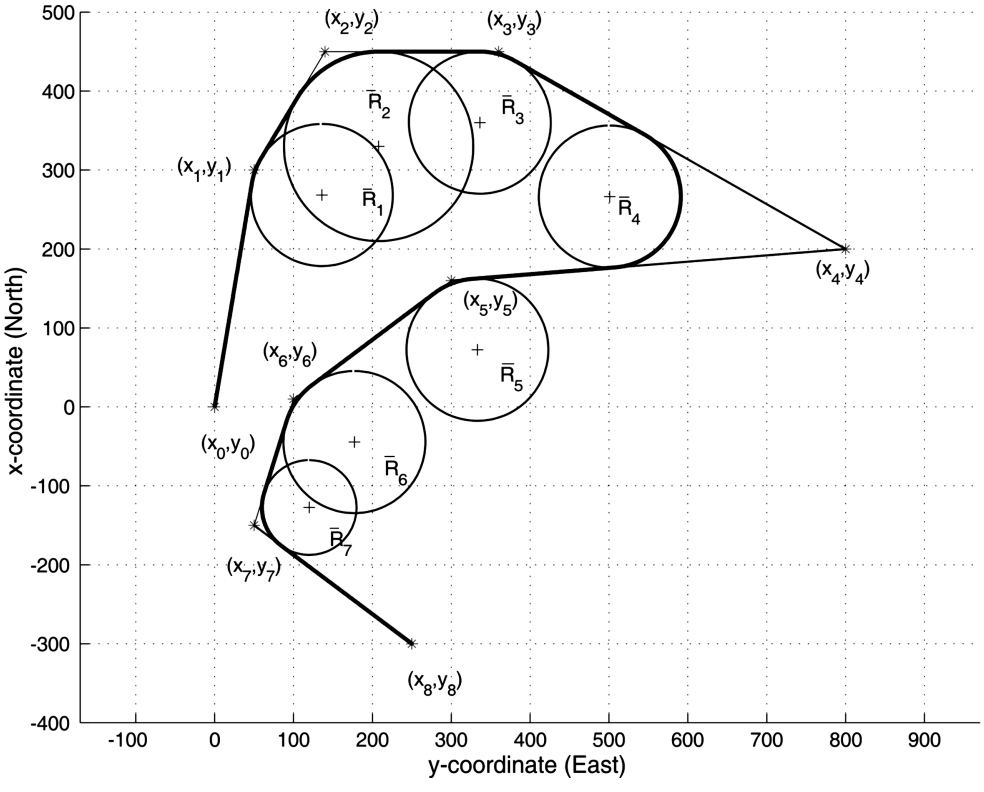


**Figure 10.7** Straight lines and inscribed circles used for waypoint guidance.

on the turning rate $r$ or the radius. The so-called Dubins path can also be proven by using *Pontryagin's maximum principle*. Generation of Dubins paths including obstacle avoidance are discussed by Tsourdos *et al.* (2010). Extensions to the case with turn rate and acceleration limits (convected Dubins path) are made by Kostov and Degtiariova-Kostova (1993) and Scheuer and Laugier (1998). Path generation for the case of uniform currents are discussed by McGee *et al.* (2006) and Techy and Woolsey (2009, 2010). In the case of time-varying speed, a dynamic optimization problem including the marine craft surge dynamics must be solved.

In this section, the discussion is limited to Dubins paths formed by straight lines and circles as shown in Figure 10.7, where the inscribed circle between two straight lines describes the desired turn. The radius of the inscribed circle is denoted $\bar{R}_i$ $(i = 1, \ldots, n)$.

The drawback of this strategy, in comparison with a cubic interpolation strategy, for instance, is that a jump in the desired yaw rate $r_d$ is experienced. This is due to the fact that 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. This produces a small offset during cross-tracking. If a smooth reference trajectory, for instance generated by interpolation, is used, these drawbacks are overcome. However, it is convenient to use straight lines and circle arcs due to their simplicity. Another consideration is that the human operator can specify a circle with radius $R_i$ about each waypoint (see Figure 10.7). These values are stored in the database as

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

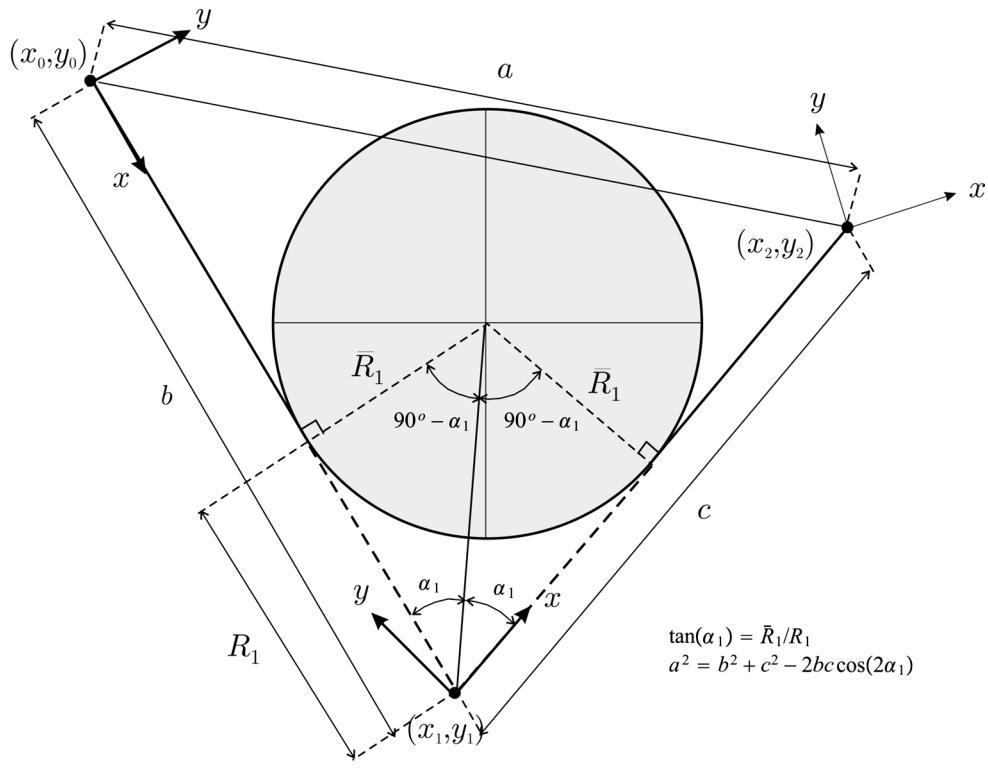


**Figure 10.8** Circle with radius $\bar{R}_1$ inscribed between the points $(x_0, y_0)$, $(x_1, y_2)$ and $(x_2, y_2)$.

The point where the circle arc intersects the straight line represents the turning point of the ship. Hence, the radius of the inscribed circle can be computed from $R_i$ as

$$\bar{R}_i = R_i \tan(\alpha_i) \quad (i = 1, \ldots, n) \tag{10.52}$$

where $\alpha_i$ is defined in Figure 10.8.

### *10.3.2 LOS Steering Laws*

This section is based on Breivik and Fossen (2009) and Breivik (2010). For 2-D horizontal plane motions, the speed of the craft is defined as

$$U(t) := \|\boldsymbol{v}(t)\| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \geq 0 \tag{10.53}$$

while steering is related to the angle

$$\chi(t) := \text{atan2}\,(\dot{y}(t), \dot{x}(t)) \in \mathbb{S} := [-\pi, \pi] \tag{10.54}$$

where atan2 $(y, x)$ is the four-quadrant version of $\arctan(y/x) \in [-\pi/2, \pi/2]$. Path following is ensured by proper assignments to $\chi(t)$ (steering control) as long as $U(t) > 0$ (positive speed) since the scenario only involves a spatial constraint.

Consider a straight-line path implicitly defined by two waypoints $\boldsymbol{p}_{\mathrm{k}}^n = [x_{\mathrm{k}}, y_{\mathrm{k}}]^\top \in \mathbb{R}^2$ and $\boldsymbol{p}_{\mathrm{k+1}}^n = [x_{\mathrm{k+1}}, y_{\mathrm{k+1}}]^\top \in \mathbb{R}^2$, respectively. Also, consider a path-fixed reference frame with origin in $\boldsymbol{p}_{\mathrm{k}}^n$ whose $x$ axis has been rotated by a positive angle:

$$\alpha_{\mathrm{k}} := \mathrm{atan2}\,(y_{\mathrm{k+1}} - y_{\mathrm{k}}, x_{\mathrm{k+1}} - x_{\mathrm{k}}) \in \mathbb{S} \tag{10.55}$$

relative to the $x$ axis. Hence, the coordinates of the craft in the path-fixed reference frame can be computed by

$$\boldsymbol{\varepsilon}(t) = \boldsymbol{R}_p(\alpha_{\mathrm{k}})^\top (\boldsymbol{p}^n(t) - \boldsymbol{p}_{\mathrm{k}}^n) \tag{10.56}$$

where

$$\boldsymbol{R}_p(\alpha_{\mathrm{k}}) := \begin{bmatrix} \cos(\alpha_{\mathrm{k}}) & -\sin(\alpha_{\mathrm{k}}) \\ \sin(\alpha_{\mathrm{k}}) & \cos(\alpha_{\mathrm{k}}) \end{bmatrix} \in SO(2) \tag{10.57}$$

and $\boldsymbol{\varepsilon}(t) = [s(t), e(t)]^\top \in \mathbb{R}^2$ with

$$s(t) = \text{along-track distance (tangential to path)}$$

$$e(t) = \text{cross-track error (normal to path)}$$

For path-following purposes, only the cross-track error is relevant since $e(t) = 0$ means that the craft has converged to the straight line. Expanding (10.56), the along-track distance and cross-track error can be explicitly stated by

$$s(t) = \; [x(t) - x_{\mathrm{k}}] \cos(\alpha_{\mathrm{k}}) + [y(t) - y_{\mathrm{k}}] \sin(\alpha_{\mathrm{k}}) \tag{10.58}$$

$$e(t) = -[x(t) - x_{\mathrm{k}}] \sin(\alpha_{\mathrm{k}}) + [y(t) - y_{\mathrm{k}}] \cos(\alpha_{\mathrm{k}}) \tag{10.59}$$

and the associated control objective for straight-line path following becomes

$$\lim_{t \to \infty} e(t) = 0 \tag{10.60}$$

In order to ensure that $e(t) \to 0$, both course and heading angle commands can be used.

Two different guidance principles can be used to steer along the LOS vector (Breivik and Fossen, 2009):

- *Enclosure-based steering*
- *Lookahead-based steering*

and at the same time stabilize $e(t)$ to the origin. 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.

### Enclosure-Based Steering

Consider a circle with radius $R > 0$ enclosing $\boldsymbol{p}^n = [x, y]^\top$. If the circle radius is chosen sufficiently large, the circle will intersect the straight line at two points. The enclosure-based strategy for driving $e(t)$ to zero is then to direct the velocity toward the intersection point $\boldsymbol{p}_{\mathrm{los}}^n = [x_{\mathrm{los}}, y_{\mathrm{los}}]^\top$ that corresponds to

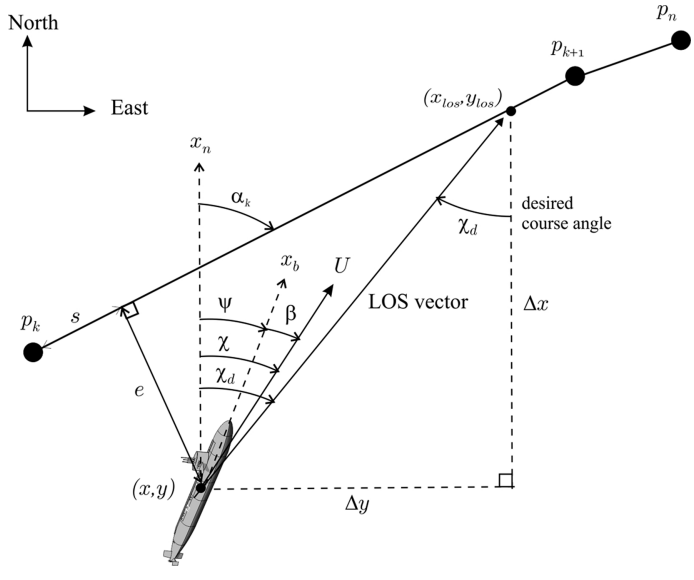


**Figure 10.9** LOS guidance where the desired course angle $\chi_d$ (angle between $x_n$ and the desired velocity vector) is chosen to point toward the LOS intersection point $(x_{los}, y_{los})$.

the desired direction of travel, which is implicitly defined by the sequence in which the waypoints are ordered. Such a solution involves directly assigning $\chi_d$ as shown in Figure 10.9. Since

$$\tan\left(\chi_d(t)\right) = \frac{\Delta y(t)}{\Delta x(t)} = \frac{y_{\text{los}} - y(t)}{x_{\text{los}} - x(t)} \tag{10.61}$$

the desired course angle can be computed as

$$\chi_d(t) = \text{atan2}\left(y_{\text{los}} - y(t), x_{\text{los}} - x(t)\right) \tag{10.62}$$

In order to calculate the two unknowns in $\boldsymbol{p}_{\text{los}}^n = \left[x_{\text{los}}, y_{\text{los}}\right]^\top$, the following two equations must be solved:

$$[x_{\text{los}} - x(t)]^2 + [y_{\text{los}} - y(t)]^2 = R^2 \tag{10.63}$$

$$\begin{aligned} \tan(\alpha_{\text{k}}) &= \frac{y_{\text{k+1}} - y_{\text{k}}}{x_{\text{k+1}} - x_{\text{k}}} \\ &= \frac{y_{\text{los}} - y_{\text{k}}}{x_{\text{los}} - x_{\text{k}}} = \text{constant} \end{aligned} \tag{10.64}$$

where (10.63) represents the *Pythagoras theorem*, while (10.64) states that the slope of the line between the two waypoints is constant. LOS guidance has been applied to surface ships by McGookin *et al.*

(2000b) and Fossen *et al.* (2003b). These equations are solved in the following, temporarily dropping the time dependence of the variables for notational convenience.

Denote the difference between the $x$ and $y$ positions of the two waypoints as $\Delta x := x_{k+1} - x_k$ and $\Delta y := y_{k+1} - y_k$, respectively. The equations are first solved analytically assuming that $|\Delta x| > 0$ and, second, for the case $\Delta x = 0$.

**Case 1:** For $|\Delta x| > 0$, Equation (10.64) results in

$$y_{los} = \left(\frac{\Delta y}{\Delta x}\right)(x_{los} - x_k) + y_k \tag{10.65}$$

when choosing to solve for $y_{los}$. For simplicity and brevity in the calculations that follow, denote

$$d := \left(\frac{\Delta y}{\Delta x}\right), \qquad e := x_k, \qquad f := y_k$$

Expanding (10.63) yields

$$x_{los}^2 - 2xx_{los} + x^2 + y_{los}^2 - 2yy_{los} + y^2 = R^2 \tag{10.66}$$

where

$$\begin{aligned} y_{los}^2 &= \left[\left(\frac{\Delta y}{\Delta x}\right)(x_{los} - x_k) + y_k\right]^2 \\ &= [dx_{los} + (f - de)]^2 \\ &= (dx_{los} + g)^2 \\ &= d^2x_{los}^2 + 2dgx_{los} + g^2 \end{aligned} \tag{10.67}$$

where

$$g := f - de = y_k - \left(\frac{\Delta y}{\Delta x}\right)x_k$$

has been used. Subsequently, consider

$$2yy_{los} = 2y(dx_{los} + g) = 2dyx_{los} + 2gy \tag{10.68}$$

such that (10.67) and (10.68) inserted into (10.66) gives:

$$(1 + d^2)x_{los}^2 + 2(dg - dy - x)x_{los} + (x^2 + y^2 + g^2 - 2gy - R^2) = 0 \tag{10.69}$$

which is a standard, analytically solvable second-order equation. Then, denote

$$\begin{aligned} a &:= 1 + d^2 \\ b &:= 2(dg - dy - x) \\ c &:= x^2 + y^2 + g^2 - 2gy - R^2 \end{aligned}$$

from which the solution of (10.69) becomes

$$x_{los} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{10.70}$$

where if $\Delta x > 0$, then $x_{\text{los}} = -b + \sqrt{b^2 - 4ac}/2a$, and if $\Delta x < 0$, then $x_{\text{los}} = -b - \sqrt{b^2 - 4ac}/2a$.

Having calculated $x_{\text{los}}$, $y_{\text{los}}$ is easily obtained from (10.65). Note that when $\Delta y = 0$, $y_{\text{los}} = y_{\text{k}}$ $(= y_{\text{k}+1})$.

**Case 2:** If $\Delta x = 0$, only (10.63) is valid, which means that

$$y_{\text{los}} = y \pm \sqrt{r^2 - (x_{\text{los}} - x)^2} \tag{10.71}$$

where $x_{\text{los}} = x_{\text{k}}$ $(= x_{\text{k}+1})$. If $\Delta y > 0$, then $y_{\text{los}} = y + \sqrt{R^2 - (x_{\text{los}} - x)^2}$, and if $\Delta y < 0$, then $y_{\text{los}} = y - \sqrt{R^2 - (x_{\text{los}} - x)^2}$. When $\Delta x = 0$, $\Delta y = 0$ is not an option.

**Lookahead-Based Steering**

For lookahead-based steering, the course angle assignment is separated into two parts:

$$\chi_d(e) = \chi_{\text{p}} + \chi_{\text{r}}(e) \tag{10.72}$$

where

$$\chi_{\text{p}} = \alpha_{\text{k}} \tag{10.73}$$

is the *path-tangential angle* (see Figure 10.9), while

$$\chi_{\text{r}}(e) := \arctan\left(\frac{-e}{\triangle}\right) \tag{10.74}$$

is a *velocity-path relative angle*, which ensures that the velocity is directed toward a point on the path that is located a *lookahead distance* $\triangle(t) > 0$ ahead of the direct projection of $\boldsymbol{p}^n(t)$ on to the path (Papoulias, 1991).

As can be immediately noticed, this 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 $R \geq |e(t)|$. Furthermore, Figure 10.10 shows that

$$e(t)^2 + \triangle(t)^2 = R^2 \tag{10.75}$$

which means that the enclosure-based approach corresponds to a lookahead-based scheme with a time variation

$$\triangle(t) = \sqrt{R^2 - e(t)^2} \tag{10.76}$$

varying between 0 and $R$ for $|e(t)| = R$ and $|e(t)| = 0$, respectively.

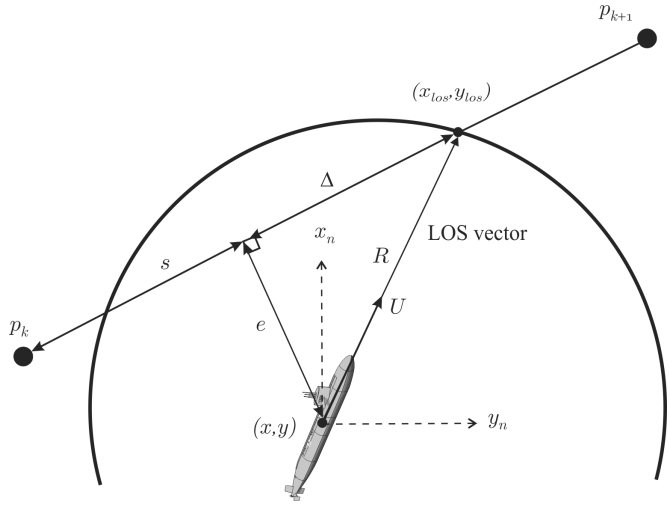


**Figure 10.10** Circle of acceptance with constant radius $R$ illustrating the geometric relationship $e(t)^2 + \Delta(t)^2 = R^2$.

The steering law (10.74) can also be interpreted as a saturating control law:

$$\chi_{\mathrm{r}}(e) = \arctan\left(-K_{\mathrm{p}} e\right) \tag{10.77}$$

where $K_{\mathrm{p}}(t) = 1/\Delta(t) > 0$. Notice that the lookahead-based steering law is equivalent to a saturated proportional control law, effectively mapping $e \in \mathbb{R}$ into $\chi_{\mathrm{r}}(e) \in [-\pi/2, \pi/2]$.

As can be inferred from the geometry of Figure 10.10, a small lookahead distance implies aggressive steering, which intuitively is confirmed by a correspondingly large proportional gain in the saturated control interpretation. This interpretation also suggests the possibility of introducing integral action into the steering law (10.74), such that

$$\chi_{\mathrm{r}}(e) = \arctan\left(-K_{\mathrm{p}} e - K_{\mathrm{i}} \int_0^t e(\tau)\mathrm{d}\tau\right) \tag{10.78}$$

where $K_{\mathrm{i}} > 0$ represents the integral gain. Integral action can be particularly useful for underactuated craft that can only steer by attitude information, enabling them to follow straight-line paths while under the influence of ocean currents and nonzero sideslip angles $\beta$, even without having access to velocity information. Thus, considering horizontal path following along straight lines, the desired yaw angle can be computed by

$$\chi_{\mathrm{d}}(e) = \alpha_{\mathrm{k}} + \chi_{\mathrm{r}}(e) \tag{10.79}$$

with $\chi_{\mathrm{r}}(e)$ as in (10.78). In practice, to avoid overshoot and windup effects, care must be taken when using integral action in the steering law. Specifically, the integral term should only be used when a steady-state off-track condition is detected.

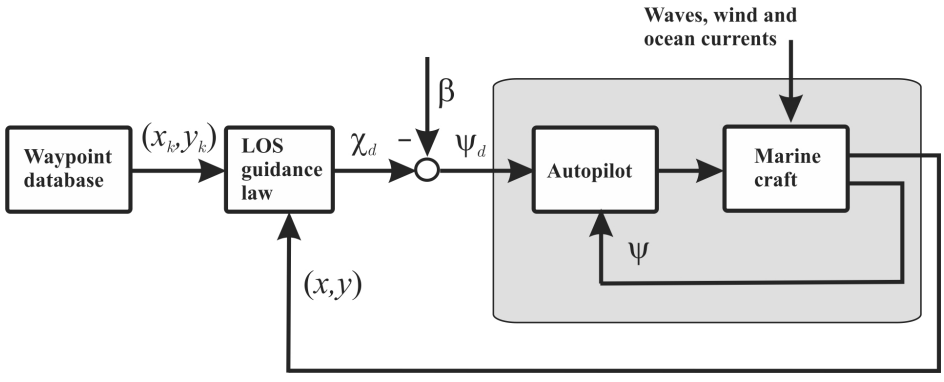


**Figure 10.11** LOS guidance principle where the sideslip angle $\beta$ either can be chosen as zero and compensated for by using integral action or nonzero by using velocity measurements.

### Path-Following Controllers

Consider the LOS intersection point $\boldsymbol{p}^n_{\text{los}}$ in Figure 10.9. Different principles for path following can be applied depending on whether you have access to velocity measurements or not:

**Method A (Body *x* axis and LOS vector aligned):** Assume that the velocity is unknown and compute the desired heading angle according to the *enclosure-based steering law* (10.62):

$$\psi_d(t) = \text{atan2}\left(y_{\text{los}} - y(t), x_{\text{los}} - x(t)\right) \tag{10.80}$$

such that the body $x$ axis of the craft points in the direction of the LOS intersection point $\boldsymbol{p}^n_{\text{los}}$. In this approach, the sideslip angle $\beta$ is assumed to be unknown and the control objective is $\psi \to \psi_d$ (see Figure 10.11). Consequently, a *heading autopilot* of PID type is

$$\tau = -K_p\tilde{\psi} - K_d\dot{\tilde{\psi}} - K_i\int_0^t \tilde{\psi}(\tau)\mathrm{d}\tau \tag{10.81}$$

where $\tilde{\psi} = \psi - \psi_d$ can be used. The price to be paid is that the craft will behave like an object hanging in a rope and the craft's lateral distance to the path will depend on the magnitude of the environmental forces and thus the sideslip angle $\beta$. This is due to the fact that $\psi = \chi$ only if $\beta = 0$. If such deviations cannot be accepted, the speed and LOS vectors should be aligned using Method B (see Figure 10.12).

**Method B (Velocity and LOS vectors aligned):** Compute the desired course angle $\chi_d$ such that the velocity vector is along the path (LOS vector) using the *lookahead-based steering law*:

$$\begin{aligned}\chi_d(e) &= \chi_{\text{p}} + \chi_{\text{r}}(e) \\ &= \alpha_{\text{k}} + \arctan\left(-K_{\text{p}}e\right)\end{aligned} \tag{10.82}$$

The control objective $\chi \to \chi_d$ is satisfied by transforming the course angle command $\chi_d$ to a heading angle command $\psi_d$ by using (2.96). This requires knowledge of $\beta$ since (see Figure 10.11)

$$\psi_d = \chi_d - \beta \tag{10.83}$$

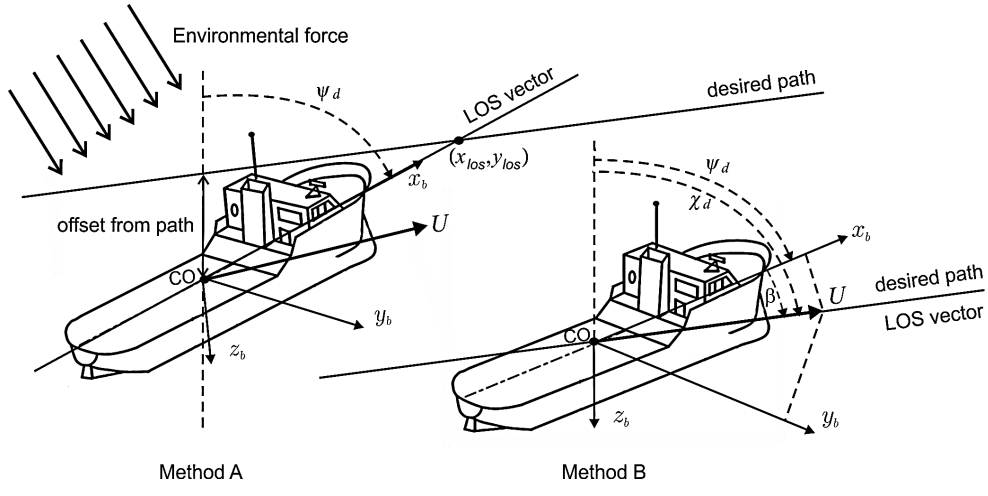


**Figure 10.12** Body $x$-axis aligned with the LOS vector (Method A) or alternatively velocity and LOS vectors aligned (Method B). Notice that Method A gives a lateral offset to the path.

Hence, the velocity and LOS vectors can be aligned using the heading controller (12.178) with the following error signal:

$$\begin{aligned} \tilde{\psi} &= \psi - \psi_d \\ &= \psi - \chi_d + \beta \end{aligned} \tag{10.84}$$

as illustrated in Figure 10.11. If the velocities of the craft are measured, the sideslip angle can be computed by

$$\beta = \arcsin\left(\frac{v}{U}\right) \tag{10.85}$$

Guidance laws of PI type, for instance (10.78), avoid velocity measurements by treating $\beta$ as an unknown slowly varying disturbance satisfying $\dot{\beta} \approx 0$.

### Circle of Acceptance for Surface Vessels

When moving along a piece wise linear path made up of $n$ straight-line segments connected by $n + 1$ waypoints, a switching mechanism for selecting the next waypoint is needed. Waypoint $(x_{k+1}, y_{k+1})$ can be selected on the basis of whether or not the craft lies within a *circle of acceptance* with radius $R_{k+1}$ around $(x_{k+1}, y_{k+1})$. Moreover, if the craft positions $(x, y)$ at time $t$ satisfy

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq R_{k+1}^2 \tag{10.86}$$

the next waypoint $(x_{k+1}, y_{k+1})$ should be selected. A guideline could be to choose $R_{k+1}$ equal to two ship lengths, that is $R_{k+1} = 2L_{pp}$.

Note that for the enclosure-based approach, such a switching criterion entails the additional (conservative) requirement $r \geq R_{k+1}$. A perhaps more suitable switching criterion solely involves the along-track

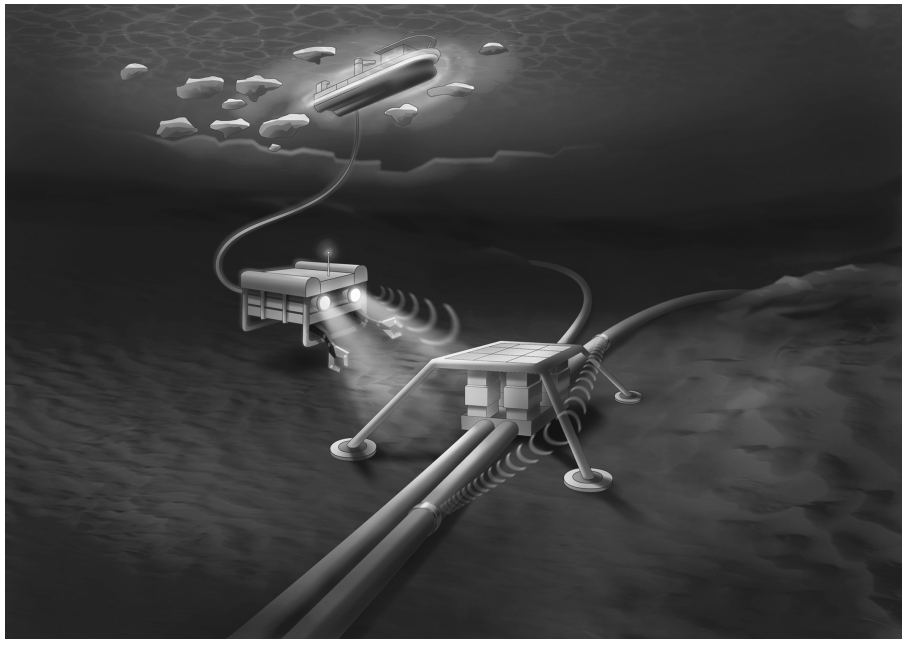

**Figure 10.13** Remotely operated vehicle (ROV) performing offshore inspection and maintenance. Illustration by Bjarne Stenberg/SINTEF.

distance $s$, such that if the total along-track distance between waypoints $\boldsymbol{p}_{\mathrm{k}}^n$ and $\boldsymbol{p}_{\mathrm{k}+1}^n$ is denoted $s_{\mathrm{k}+1}$, a switch is made when

$$s_{\mathrm{k}+1} - s(t) \leq R_{\mathrm{k}+1} \tag{10.87}$$

which is similar to (12.189) but has the advantage that $\boldsymbol{p}^n(t)$ does not need to enter the waypoint enclosing circle for a switch to occur; that is no restrictions are put on the cross-track error. Thus, if no intrinsic value is associated with visiting the waypoints and their only purpose is to implicitly define a piece-wise linear path, there is no reason to apply the circle-of-acceptance switching criterion (12.189).

**Extension to 3-D LOS Guidance for Underwater Vehicles**

It is straightforward to generalize the concepts of LOS guidance to 3-D maneuvering. Also for this case, the desired course angle $\chi_d$ can be chosen as (10.62) with the LOS intersection point given by (10.63) and (10.64) under the assumption that the vehicle performs slow maneuvers in the vertical plane such that a depth controller can easily achieve $z = z_d$. This works quite well for vehicles moving at low speed since it is not necessary to pitch the vehicle in order to move vertically; see Figure 10.13. A typical example is a working ROV with a broad, flattened front (bluff body) moving vertically using a vertical thruster. The circle of acceptance must, however, be replaced by a *sphere of acceptance* (Healey and Lienard, 1993):

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 + [z_{k+1} - z(t)]^2 \leq R_{k+1}^2 \tag{10.88}$$

A more sophisticated approach will be to compute the azimuth and elevation angles needed to move in 3-D to the next waypoint (Breivik and Fossen, 2009). This approach is used for " flying" vehicles equipped with fins for diving and depth control. These vehicles move at a higher speed in order to produce lifting forces (no vertical thrusters) and consequently they behave like an aircraft, where it is possible to control the coupled surge, heave and pitch motions (longitudinal motions).

## 10.4 Path-Following for Curved Paths

This section relaxes the condition that the path consists of straight lines between waypoints. Instead, it is assumed that the guidance systems can make use of a predefined parametrized path. The path-following controller is a *kinematic controller* that generates the desired states for the motion control system using the parametrization of the path. The drawback is that the path must be parametrized and known in advance. In many cases this is not practical and a simpler path consisting of waypoints and straight lines must be used. The solution for this is presented in Section 10.3. Section 10.4.1 discusses path generation while a path-following controller for parametrized paths is derived in Section 10.4.2.

For a parametrized path, the following definitions are adopted from Skjetne *et al.* (2004):

***Definition 10.2 (Parametrized Path)***
*A parametrized path is defined as a geometric curve* $\boldsymbol{\eta}_d(\varpi) \in \mathbb{R}^q$ *with* $q \geq 1$ *parametrized by a continuous path variable* $\varpi$.

For marine craft it is common to use a 3-D representation:

$$\boldsymbol{p}_d^n(\varpi) = [x_d(\varpi),\, y_d(\varpi),\, z_d(\varpi)]^\top \in \mathbb{R}^3 \tag{10.89}$$

where the first two coordinates describe the position in the horizontal plane and the last coordinate is the depth. For surface vessels only $x_d$ and $y_d$ are needed while underwater vehicles use all three coordinates. The first- and second-order derivatives of $\boldsymbol{p}^n(\varpi)$ with respect to $\varpi$ are denoted as $\boldsymbol{p}'$ and $\boldsymbol{p}''$, respectively.

A frequently used solution of the path-following problem is to solve it as the geometric task of a *maneuvering problem*, given by the following definition:

***Definition 10.3 (Maneuvering Problem)***
*The maneuvering problem involves solving two tasks:*

1. *Geometric Task: Force the position* $\boldsymbol{p}^n(t)$ *to converge to a desired path* $\boldsymbol{p}_d^n(\varpi(t))$,

$$\lim_{t\to\infty} \left[\boldsymbol{p}^n(t) - \boldsymbol{p}_d^n(\varpi(t))\right] = \mathbf{0} \tag{10.90}$$

   *for any continuous function* $\varpi(t)$.
2. *Dynamic Task: Force the speed* $\dot{\varpi}$ *to converge to a desired speed* $U_d$ *according to*

$$\lim_{t\to\infty} \left[\dot{\varpi}(t) - \frac{U_d(\varpi(t))}{\sqrt{(x_d')^2 + (y_d')^2}}\right] = 0 \tag{10.91}$$

The dynamic task follows from

$$U_d(t) = \sqrt{\dot{u}_d^2(t) + \dot{v}_d^2(t)} = \sqrt{x_d'(\varpi)^2 + y_d'(\varpi)^2}\,\dot{\varpi}(t) \tag{10.92}$$

Definition 10.3 implies that the dynamics $\varpi(t)$ along the path can be specified independently of the error dynamics. A special case of the maneuvering problem is

$$\dot{\varpi}(t) = 1, \quad \varpi(0) = 0 \tag{10.93}$$

which is recognized as the tracking problem since the solution of (10.93) is $\varpi = t$. A solution to the maneuvering problem for fully actuated craft is found in Skjetne *et al.* (2004).

### *10.4.1 Path Generation using Interpolation Methods*

The path can be generated using spline or polynomial interpolation methods to generate a curve $(x_d(\varpi), y_d(\varpi))$ through a set of $N$ predefined waypoints. Notice that a trajectory $(x_d(t), y_d(t))$ is obtained by choosing $\dot{\varpi} = k$ such that $\varpi = kt$ where $k \in \mathbb{R}$.

#### Cubic Spline and Hermite Interpolation

In Matlab, several methods for interpolation are available.

**Matlab**
The different methods for interpolation are found by typing

```
help polyfun
```

Two useful methods for path generation are the cubic spline interpolant (`spline.m`) and the piecewise cubic Hermite interpolating polynomial (`pchip.m`).

The main difference between Hermite and cubic spline and interpolation is how the slopes at the end points are handled. For simplicity let us consider the problem of trajectory generation. The cubic Hermite interpolant ensures that the first-order derivatives $(\dot{x}_d(t), \dot{y}_d(t))$ are continuous. In addition, the slopes at each endpoint are chosen in such a way that $(x_d(t), y_d(t))$ are shape preserving and respect monotonicity.

Cubic spline interpolation is usually done by requiring that the second-order derivatives $(\ddot{x}_d(t), \ddot{y}_d(t))$ at the endpoints of the polynomials are equal, which gives a smooth spline. Consequently, the cubic spline will be more accurate than the Hermite interpolating polynomial if the data values are of a smooth function. The cubic Hermite interpolant, on the contrary, has less oscillations if the data are nonsmooth.

The results of interpolating a set of predefined waypoints to a trajectory $(x_d(\varpi), y_d(\varpi))$ using the cubic Hermite interpolant and cubic spline interpolation methods are shown in Figure 10.14. It is seen that different behaviors are obtained due to the conditions on the first- and second-order derivatives at the endpoints.

#### Polynomial Interpolation

Instead of using the Matlab functions `pchip.m` and `spline.m` a cubic spline can be interpolated through a set of waypoints by considering the *cubic polynomials*

$$x_d(\varpi) = a_3\varpi^3 + a_2\varpi^2 + a_1\varpi + a_0 \tag{10.94}$$

$$y_d(\varpi) = b_3\varpi^3 + b_2\varpi^2 + b_1\varpi + b_0 \tag{10.95}$$

where $(x_d(\varpi), y_d(\varpi))$ are the position of the craft and where $\varpi$ is a path variable given by

$$\dot{\varpi} = f(\varpi, t) \tag{10.96}$$

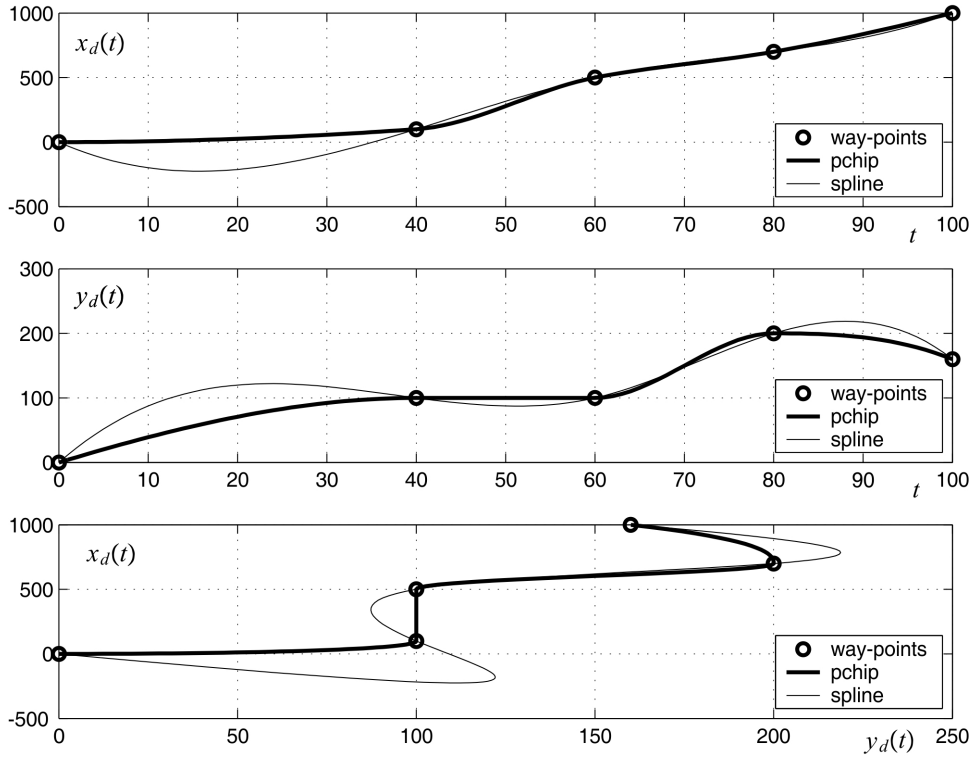


**Figure 10.14** Methods for waypoint interpolation; see ExSpline.m in the MSS toolbox.

The partial derivatives of $x_d(\varpi)$ and $y_d(\varpi)$ with respect to $\varpi$ are

$$x'_d(\varpi) = \frac{\mathrm{d}x_d(\varpi)}{\mathrm{d}\varpi} = 3a_3\varpi^2 + 2a_2\varpi + a_1 \tag{10.97}$$

$$y'_d(\varpi) = \frac{\mathrm{d}y_d(\varpi)}{\mathrm{d}\varpi} = 3b_3\varpi^2 + 2b_2\varpi + b_1 \tag{10.98}$$

Hence, the desired speed $U_d(t)$ of the craft can be computed as

$$\dot{x}_d(t) = \frac{\mathrm{d}x_d(\varpi)}{\mathrm{d}\varpi}\dot{\varpi}(t) \tag{10.99}$$

$$\dot{y}_d(t) = \frac{\mathrm{d}y_d(\varpi)}{\mathrm{d}\varpi}\dot{\varpi}(t) \tag{10.100}$$

resulting in

$$\begin{aligned} U_d(t) &= \sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \\ &= \sqrt{x'_d(\varpi)^2 + y'_d(\varpi)^2}\ \ \dot{\varpi}(t) \end{aligned} \tag{10.101}$$

Similarly, an expression for the acceleration $\dot{U}_d(t)$ can be found.

**Matlab**
The script `ExSpline.m` generates the plots in Figure 10.14:

```
% ExSpline - Cubic Hermite and spline interpolation of waypoints

wpt.pos.x = [0 100 500 700 1000];
wpt.pos.y = [0 100 100 200 160];
wpt.time  = [0 40 60 80 100];

t = 0:1:max(wpt.time);              % time
x_p = pchip(wpt.time,wpt.pos.x,t);  % cubic Hermite interpolation
y_p = pchip(wpt.time,wpt.pos.y,t);
x_s = spline(wpt.time,wpt.pos.x,t); % spline interpolation
y_s = spline(wpt.time,wpt.pos.y,t);

subplot(311), plot(wpt.time,wpt.pos.x,'o',t,[x_p; x_s])
subplot(312), plot(wpt.time,wpt.pos.y,'o',t,[y_p; y_s])
subplot(313), plot(wpt.pos.y,wpt.pos.x,'o',y_p,x_p,y_s,x_s)
```

The unknown parameters $a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3$ in (10.94) and (10.95) can also be computed using a cubic spline algorithm, as shown below.

### Cubic Spline Algorithm for Path Generation

The path through the waypoints $(x_{k-1}, y_{k-1})$ and $(x_k, y_k)$ must satisfy

$$x_d(\varpi_{k-1}) = x_{k-1}, \quad x_d(\varpi_k) = x_k \tag{10.102}$$

$$y_d(\varpi_{k-1}) = y_{k-1}, \quad y_d(\varpi_k) = y_k \tag{10.103}$$

where $k = 1, \ldots, n$. In addition, smoothness is obtained by requiring that

$$\lim_{\varpi \to \varpi_k^-} x_d'(\varpi_k) = \lim_{\varpi \to \varpi_k^+} x_d'(\varpi_k) \tag{10.104}$$

$$\lim_{\varpi \to \varpi_k^-} x_d''(\varpi_k) = \lim_{\varpi \to \varpi_k^+} x_d''(\varpi_k) \tag{10.105}$$

For this problem, it is possible to add only two boundary conditions (velocity or acceleration) for the $x$ and $y$ equations, respectively. Hence,

$$x_d'(\varpi_0) = x_0', \quad x_d'(\varpi_n) = x_n' \tag{10.106}$$

$$y_d'(\varpi_0) = y_0', \quad y_d'(\varpi_n) = y_n' \tag{10.107}$$

or

$$x_d''(\varpi_0) = x_0'', \quad x_d''(\varpi_n) = x_n'' \tag{10.108}$$

$$y_d''(\varpi_0) = y_0'', \quad y_d''(\varpi_n) = y_n'' \tag{10.109}$$

The polynomial $x_d(\varpi_k)$ is given by the parameters $\boldsymbol{a}_k = [a_{3k}, a_{2k}, a_{1k}, a_{0k}]^\top$, resulting in $4(n-1)$ unknown parameters. The number of constraints are also $4(n-1)$ if only velocity or acceleration constraints are chosen at the end points. The unknown parameters for $n$ waypoints are collected into a vector:

$$\boldsymbol{x} = [\boldsymbol{a}_k^\top, \ldots, \boldsymbol{a}_{n-1}^\top]^\top \tag{10.110}$$

Hence, the cubic interpolation problem can be written as a linear equation:

$$\boldsymbol{y} = \boldsymbol{A}(\varpi_{k-1}, \ldots, \varpi_k)\boldsymbol{x}, \qquad k = 1, 2, \ldots, n \tag{10.111}$$

where

$$\boldsymbol{y} = [x_{\text{start}}, x_0, x_1, x_1, 0, 0, x_2, x_2, 0, 0, \ldots, x_n, x_{\text{final}}]^\mathsf{T} \tag{10.112}$$

The start and end points can be specified in terms of velocity or acceleration constraints $x_{\text{start}} \in \{x_0', x_0''\}$ and $x_{\text{final}} \in \{x_n', x_n''\}$, respectively. This gives

$$\boldsymbol{A}(\varpi_{k-1}, \ldots, \varpi_k) = \left[\begin{array}{ccccc}
\boldsymbol{c}_{\text{start}} & \boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & \cdots & \boldsymbol{0}_{1\times4} \\
\boldsymbol{p}(\varpi_0) & \boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
\hline
\boldsymbol{p}(\varpi_1) & \boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
0 & \boldsymbol{p}(\varpi_1) & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
-\boldsymbol{v}(\varpi_1) & \boldsymbol{v}(\varpi_1) & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
-\boldsymbol{a}(\varpi_1) & \boldsymbol{a}(\varpi_1) & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
\hline
\boldsymbol{0}_{1\times4} & \boldsymbol{p}(\varpi_2) & \boldsymbol{0}_{1\times4} & & \boldsymbol{0}_{1\times4} \\
\boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & \boldsymbol{p}(\varpi_2) & & \boldsymbol{0}_{1\times4} \\
\boldsymbol{0}_{1\times4} & -\boldsymbol{v}(\varpi_2) & \boldsymbol{v}(\varpi_2) & & \boldsymbol{0}_{1\times4} \\
\boldsymbol{0}_{1\times4} & -\boldsymbol{a}(\varpi_2) & \boldsymbol{a}(\varpi_2) & & \boldsymbol{0}_{1\times4} \\
\hline
\vdots & & & \ddots & \\
\hline
\boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & & \boldsymbol{p}(\varpi_n) \\
\boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & \boldsymbol{0}_{1\times4} & \cdots & \boldsymbol{c}_{\text{final}}
\end{array}\right] \tag{10.113}$$

where $\boldsymbol{c}_{\text{start}} \in \{x_d'(\varpi_0), x_d''(\varpi_0)\}$, $\boldsymbol{c}_{\text{final}} \in \{x_d'(\varpi_n), x_d''(\varpi_n)\}$ and

$$\boldsymbol{p}(\varpi_k) = [\varpi_k^3,\ \varpi_k^2,\ \varpi_k,\ 1] \tag{10.114}$$

$$\boldsymbol{v}(\varpi_k) = \boldsymbol{p}'(\varpi_k) = [3\varpi_k^2,\ 2\varpi_k,\ 1,\ 0] \tag{10.115}$$

$$\boldsymbol{a}(\varpi_k) = \boldsymbol{p}''(\varpi_k) = [6\varpi_k,\ 2,\ 0,\ 0] \tag{10.116}$$

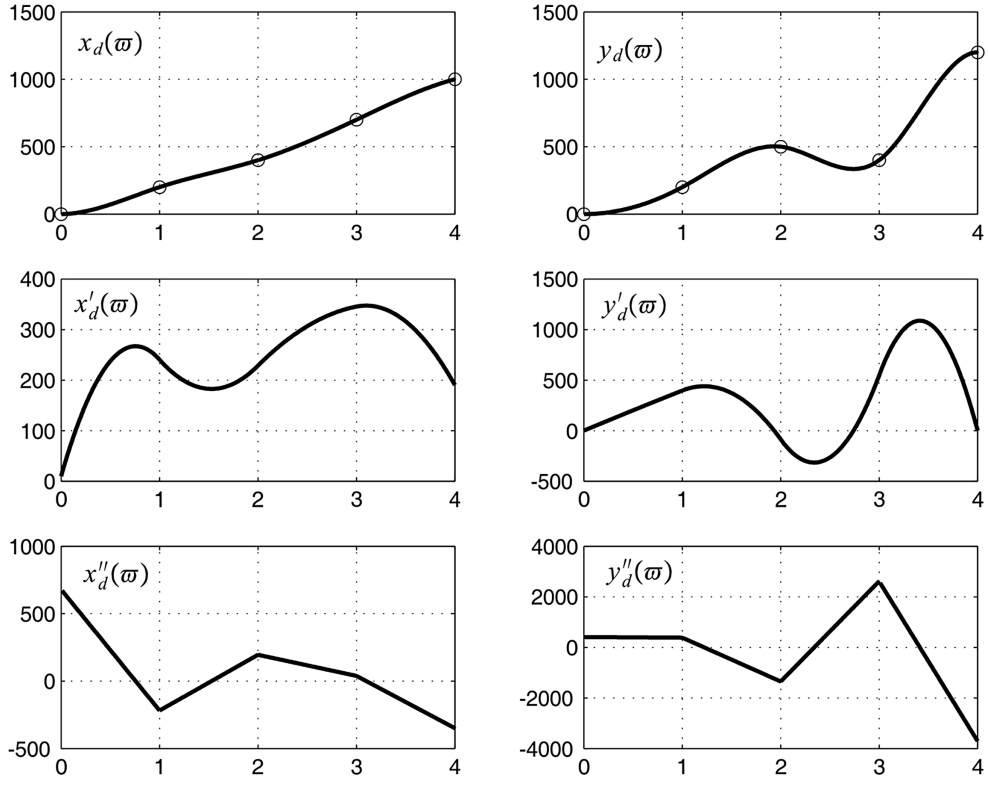


**Figure 10.15** Polynomials $x_d(\varpi)$ and $y_d(\varpi)$ and their first- and second-order derivatives.

Equation (10.111) can be solved for $\varpi_k = 0, 1, \ldots, n$ according to

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y} \tag{10.117}$$

The formulae for $\boldsymbol{b}_k = [b_{3k}, b_{2k}, b_{1k}, b_{0k}]^\top$ are obtained in a similar manner.

**Matlab**
Formula (10.117) has been implemented in the script `ExPathGen.m` and `pva.m`. The results are for the following set of waypoints:

```
wpt.pos.x = [0 200 400 700 1000]
wpt.pos.y = [0 200 500 400 1200]
```

where $\varpi = 0, \ldots, 4$ are shown in Figures 10.15 and 10.16.

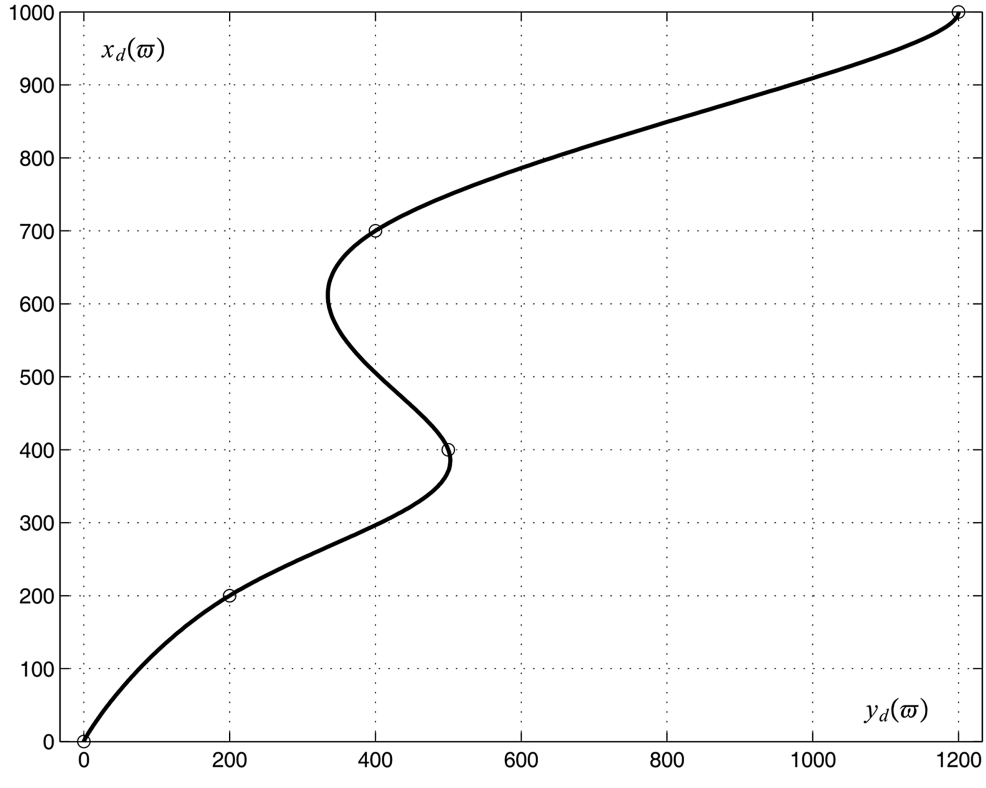


**Figure 10.16** The $xy$ plot based on a cubic spline.

**Transformation of Path to Reference Trajectories using Desired Speed Profiles**

In Figure 10.16 it is seen that the solution between two successive waypoints

$$x_d(\varpi) = a_3\varpi^3 + a_2\varpi^2 + a_1\varpi + a_0 \tag{10.118}$$

$$y_d(\varpi) = b_3\varpi^3 + b_2\varpi^2 + b_1\varpi + b_0 \tag{10.119}$$

is indeed a *time-independent* path when $x_d(\varpi)$ is plotted against $y_d(\varpi)$ for increasing $\varpi$ values.

The path can be transformed to a *time-varying* trajectory by defining a *speed profile.* The speed profile assigns dynamics to $\varpi(t)$ such that the desired path transforms to a time-dependent reference trajectory at the same time as the desired speed and acceleration profiles are preserved. From (10.101) it is seen that

$$\dot{\varpi}(t) = \frac{U_d(t)}{\sqrt{x_d'(\varpi)^2 + y_d'(\varpi)^2}}, \quad \varpi(t_k) = k \tag{10.120}$$

where $\varpi(t_k) = k$ is the initial condition of the differential equation and $U_d(t)$ is the desired speed profile. Let $U_{\text{ref}}$ be the input to a first-order system:

$$T\dot{U}_d(t) + U_d(t) = U_{\text{ref}}, \quad T > 0 \tag{10.121}$$

A smooth transition from the desired speed $U_d(t_k)$ at waypoint $k$ to the next waypoint $k+1$ can be made by using

$$U_{\text{ref}} = U_d(t_{k+1}) \tag{10.122}$$

This is illustrated in the following example.

***Example 10.4 (Transformation of Path to Reference Trajectories)***
*Consider the first two waypoints in the example file* `ExPathGen.m`:

$$(x_0, y_0) = (0, 0)$$
$$(x_1, y_1) = (200, 200)$$

*The cubic polynomials satisfying (10.117) are*

$$x_d(\varpi) = -29.89\,\varpi^3 + 135.63\,\varpi^2 + 94.25\,\varpi$$
$$y_d(\varpi) = 108.05\,\varpi^3 - 2.30\,\varpi^2 + 94.25\,\varpi$$

*for* $\varpi \in [0, 1]$. *Let the speed dynamics time constant be* $T = 10$ s. *Assume that the craft is initially at rest* ($U_d(t_0) = 0$) *and that the desired speed of waypoint number 1 is* $U_{\text{ref}} = U_d(t_1) = 5.0$ m/s. *The numerical solutions of*

$$\dot{\varpi}(t) = \frac{U_d(t)}{\sqrt{x_d'(\varpi)^2 + y_d'(\varpi)^2}} \tag{10.123}$$

$$T\dot{U}_d(t) + U_d(t) = U_{\text{ref}} \tag{10.124}$$

*for waypoints* 0 *and* 1 *corresponding to* $\varpi_0(t_0) = 0$ *and* $\varpi_1(t_1) = 1$ *with* $t_0 = 0$ *and* $t_1$ *unknown, is shown in Figure 10.17; see* `ExPathGen.m`. *It is seen that the desired speed of* 5.0 m/s *is reached in approximately* 67 s. *Hence, the terminal time must be chosen as* $t_1 \geq 67$ s (*corresponding to* $\varpi(t_1) = 1$) *in order to satisfy the desired speed dynamics. If* $t_1 < 67$ s *there is not enough time to reach the desired speed of waypoint* 1 *unless the time constant* $T$ *is reduced. The time constant should reflect what is physically possible for the craft. Notice that the path* ($x_d(\varpi)$, $y_d(\varpi)$) *has been transformed to a time-varying reference trajectory* ($x_d(t)$, $y_d(t)$) *by assigning a speed profile (10.123) to be solved numerically with the path planner (10.117). This gives design flexibility since the path can be generated off-line using a waypoint database while speed is assigned to the path when the dynamics of the actual craft is considered.*

**Nonlinear Constrained Optimization**

Another solution to trajectory and path generation is to use nonlinear constrained optimization techniques. These methods allow an object function to be specified where minimum time and energy are design goals. In addition, the speed and acceleration constraints of the craft can be added. The drawback is that nonlinear constraint optimization problems are much harder to solve numerically than the methods described

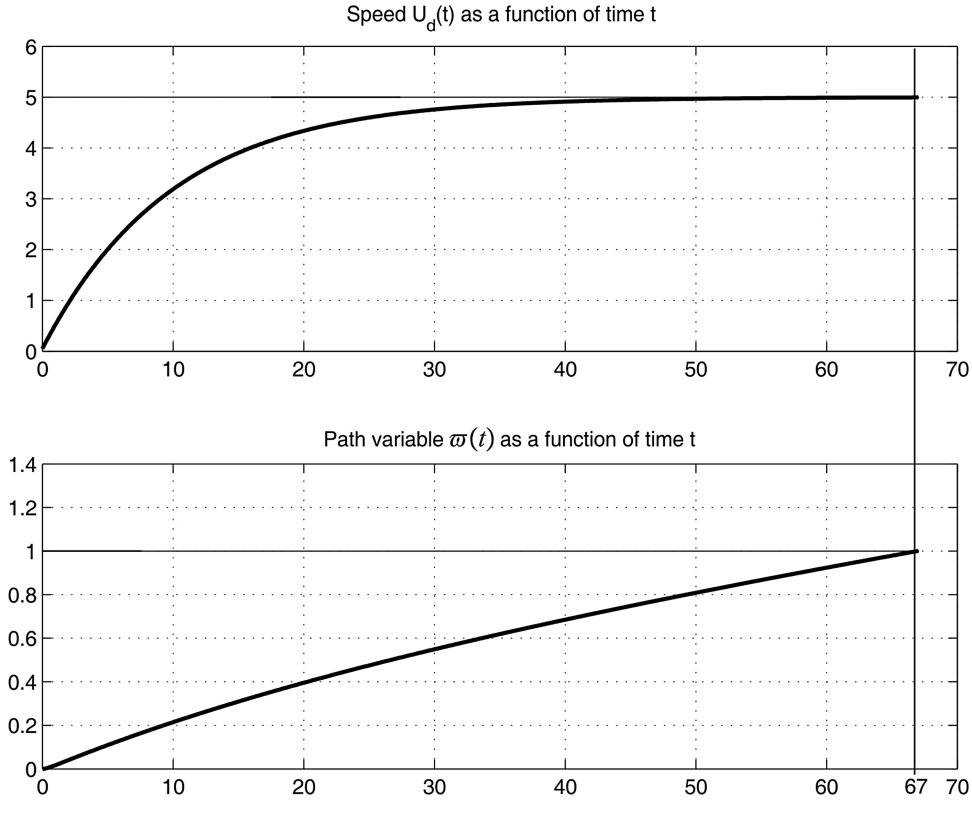


**Figure 10.17** Upper plot shows that the speed $U_d(t)$ reaches the desired value of 5.0 m/s in approximately 67 s. The lower plot shows that the path variable $\varpi(t)$ is incremented from 0 to 1 during the speed transition.

in the previous sections. The Matlab optimization toolbox will be used to demonstrate how this can be done.

In general, trajectory-tracking and path-planning problems can be formulated as

$$J = \min_{\boldsymbol{x}} \{f(\boldsymbol{x})\} \tag{10.125}$$

$$\begin{aligned} \text{subject to} \qquad & g_k(\boldsymbol{x}) \leq 0 \qquad (k = 1, \ldots, n_g) \\ & h_j(\boldsymbol{x}) = 0 \qquad (j = 1, \ldots, n_h) \\ & x_{i,\min} \leq x_i \leq x_{i,\max} \qquad (i = 1, \ldots, n_x) \end{aligned}$$

where $f(\boldsymbol{x})$ should be minimized with respect to the parameter vector $\boldsymbol{x}$ with $g_i(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ as nonlinear inequality and equality constraints, respectively. An attractive simplification is to use quadratic

programming. Consequently,

$$J = \min_{\boldsymbol{x}} \left\{ \frac{1}{2} \boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x} \right\} \tag{10.126}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

$$x_{i,\min} \leq x_i \leq x_{i,\max} \quad (i = 1, \ldots, n_x)$$

For simplicity, consider two waypoints $(x_k, y_k)$ and $(x_{k+1}, y_{k+1})$ satisfying

$$x(t_k) = x_k, \qquad y(t_k) = y_k \tag{10.127}$$

$$x(t_{k+1}) = x_{k+1}, \qquad y(t_{k+1}) = y_{k+1} \tag{10.128}$$

Choosing the speed constraints as

$$\dot{x}_d(t) = U_d(t) \cos(\psi_d(t)) \tag{10.129}$$

$$\dot{y}_d(t) = U_d(t) \sin(\psi_d(t)) \tag{10.130}$$

where the angle $\psi_d(t)$ is computed as $\psi_d(t_k) = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k)$, that is with direction toward the next waypoint. Hence,

$$\dot{x}_d(t_k) = U_k \cos(\psi_k) \tag{10.131}$$

$$\dot{y}_d(t_k) = U_k \sin(\psi_k) \tag{10.132}$$

For two waypoints this results in

$$\boldsymbol{y} = \boldsymbol{A}(t_k, t_{k+1})\boldsymbol{x} \tag{10.133}$$

where

$$\boldsymbol{y} = [x_k, x_{k+1}, y_k, y_{k+1}, U_k \cos(\psi_k), U_k \sin(\psi_k), U_{k+1} \cos(\psi_{k+1}), U_{k+1} \sin(\psi_{k+1})]^\top \tag{10.134}$$

and

$$\boldsymbol{A}(t_k, t_{k+1}) = \begin{bmatrix} t_k^3 & t_k^2 & t_k & 1 & 0 & 0 & 0 & 0 \\ t_{k+1}^3 & t_{k+1}^2 & t_{k+1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_k^3 & t_k^2 & t_k & 1 \\ 0 & 0 & 0 & 0 & t_{k+1}^3 & t_{k+1}^2 & t_{k+1} & 1 \\ 3t_k^2 & 2t_k & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3t_k^2 & 2t_k & 1 & 0 \\ 3t_{k+1}^2 & 2t_{k+1} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3t_{k+1}^2 & 2t_{k+1} & 0 & 0 \end{bmatrix} \tag{10.135}$$

The criterion to minimize is

$$J = \min_{\boldsymbol{x}} \left\{ [\boldsymbol{A}(t_k, t_{k+1})\boldsymbol{x} - \boldsymbol{y}]^\top [\boldsymbol{A}(t_k, t_{k+1})\boldsymbol{x} - \boldsymbol{y}] \right\} \tag{10.136}$$

for given pairs $(t_k, t_{k+1})$ of time. Expanding this expression yields

$$\bar{J} = \frac{1}{2}\left(J - \boldsymbol{y}^\top \boldsymbol{y}\right) = \min_{\boldsymbol{x}} \left\{ \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}^\top(t_k, t_{k+1})\boldsymbol{A}(t_k, t_{k+1})\boldsymbol{x} - \boldsymbol{y}^\top \boldsymbol{A}(t_k, t_{k+1})\boldsymbol{x} \right\} \tag{10.137}$$

implying that

$$\boldsymbol{H} = \boldsymbol{A}^\top(t_k, t_{k+1})\boldsymbol{A}(t_k, t_{k+1}) \tag{10.138}$$

$$\boldsymbol{f} = -\boldsymbol{y}^\top \boldsymbol{A}(t_k, t_{k+1}) \tag{10.139}$$

In this expression, the starting time $t_k$ is given while the arrival time $t_{k+1}$ is unknown. The cubic polynomials (10.94)–(10.95) imply that there are eight additional unknown parameters to optimize:

$$\boldsymbol{x} = [a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0]^\top \tag{10.140}$$

giving a total of nine unknown parameters. In addition, linear constraints $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ can be added. The reference trajectory can be found using quadratic programming.

**Matlab**
Trajectory generation using the optimization toolbox is demonstrated in the following example:

***Example 10.5 (Trajectory Generation using Quadratic Programming)***
*Consider two waypoints:*

$$(x_0, y_0) = (10, 10)$$

$$(x_1, y_1) = (200, 100)$$

*with the speed constraint*

$$U_d(t) \leq 10\,\text{m/s}$$

*in the MSS toolbox script*

`ExQuadProg`

*The desired waypoint speeds are* $U_0(t_0) = 0$ m/s *and* $U_1(t_1) = 5$ m/s *with* $t_0 = 0$ s. *The arrival time* $t_1$ *is computed in a loop by solving the quadratic optimization problem (10.126) for each time* $t_1 = t_0 +$ *dt where dt is incremented by* 1.0 s *each time. This process is terminated when the first solution* $U_d(t) \leq 10$ m/s *is reached (this can be easily changed if other requirements are more important). The optimal solution:*

$$x_d(t) = -0.0102\,t^3 + 0.5219\,t^2 - 4.28 \times 10^{-12}\,t + 10.0$$

$$y_d(t) = -0.0048\,t^3 + 0.2472\,t^2 - 1.04 \times 10^{-12}\,t + 10.0$$

*for* $t \in [t_0, t_1]$ *is obtained after* 29 *loops* ($t_1 = 29$ s) *using* `quadprog.m` *in the Matlab optimization toolbox. The results are shown in Figure 10.18.*

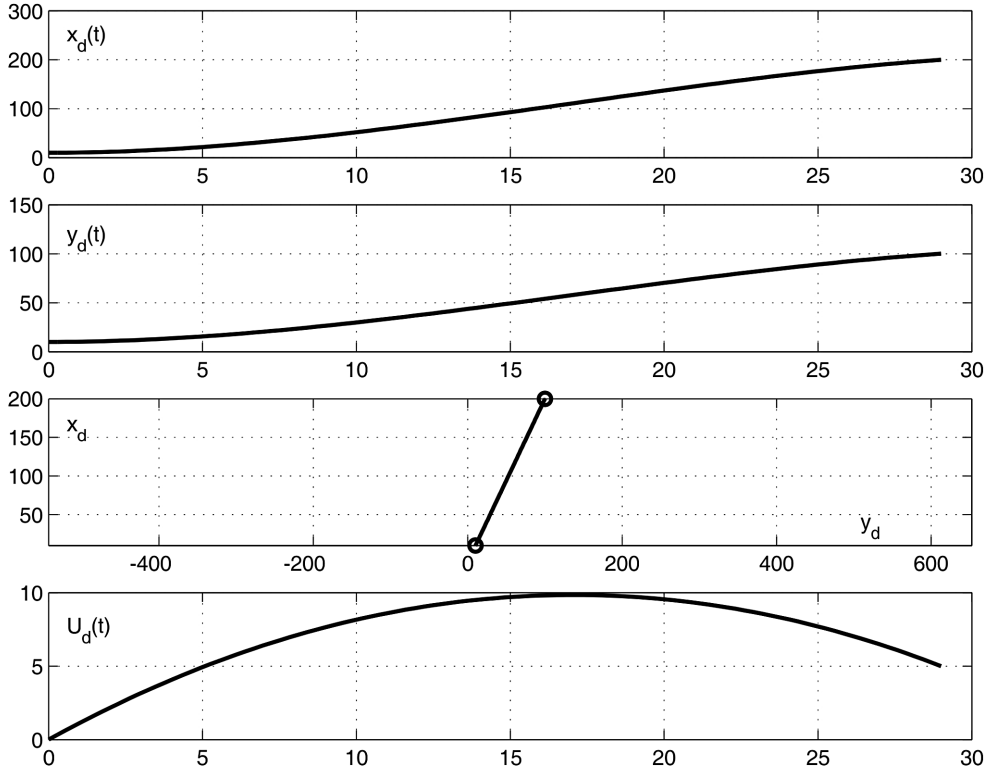


**Figure 10.18** The two upper plots show the cubic polynomials $x_d(t)$ and $y_d(t)$. In the third plot $y_d(t)$ is plotted against $x_d(t)$ while the lower plot is speed $U_d(t)$.

### Weather Routing

A weather routing or voyage planning system (VPS) computes the most efficient route using meteorological and oceanographic data, information about the craft's hull and propulsion system and shipping economics to ensure that the craft reaches port on time. The data from this analysis can be waypoints with optimal speed and heading information. The routing software of a modern weather routing system includes features such as:

- Surface analysis and forecast models
- Sea state and wind wave models
- Upper air models
- Formation description of low-pressure systems
- Hurricanes and tropical weather models
- Ocean current models
- Vessel performance models
- Cargo condition, trim, draft and deck load
- Link to Internet sources for weather data
- Interface to a satellite system transmitting weather data

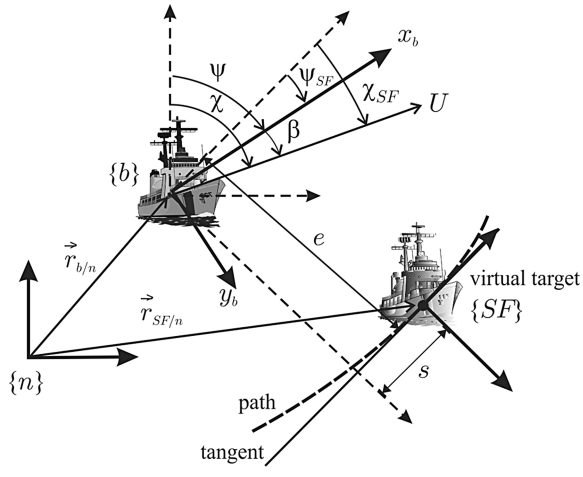


**Figure 10.19** Kinematic description of the Serret–Frenet frame.

- Optimization of routes based on a fixed estimated time of arrival (ETA)
- Routing of vessels around hazardous weather conditions

An optimal route is computed using a numerical optimization offline. This can be done by a computer onboard the craft or by a company onshore transmitting the results to the craft electronically on a 24-hour basis. Several companies offer continuous voyage monitoring with status reports and performance evaluations. This allows for replanning during changing weather conditions. Global weather information is available from several forecast centers.

Some useful references for weather routing of ships are Calvert (1989), Hagiwara (1989), Padadakis and Perakis (1990), Lo (1991), Barbier *et al.* (1994), Lo and McCord (1995), McCord and Smith (1995) and Lo and McCord (1998).

## *10.4.2 Path-Following Kinematic Controller*

The path-following controller is a *kinematic controller* that generates the desired states for the motion control system. For a parametrized path in 2-D,

$$\boldsymbol{p}_d^n(\varpi) = \begin{bmatrix} x_d(\varpi) \\ y_d(\varpi) \end{bmatrix} \in \mathbb{R}^2 \tag{10.141}$$

the kinematic controller can be designed using a dynamic model of the craft by specifying a reference frame that moves along the path; see Figure 10.19. This reference frame is usually chosen as the *Serret–Frenet frame* (see Frenet, 1847, Serret, 1851). During path following, the craft speed is denoted $U$ and the kinematic controller is designed to: (i) regulate the distance $e$ between the vehicle and the path to zero and (ii) regulate the angle $\chi_{\mathrm{SF}}$ between the craft speed vector and the tangent to the path to zero (see Samson, 1992, Micaelli and Samson, 1993).

***Definition 10.4 (Serret-Frenet Frame)***
*The virtual target defined by the projection of an actual craft on to a path-tangential reference frame (Serret–Frenet frame* {SF}*) evolves according to (Lapierre and Soetanto, 2007)*

$$\dot{s} = U\cos(\chi_{\text{SF}}) - (1 - \kappa e)\dot{s}_a \tag{10.142}$$

$$\dot{e} = U\sin(\chi_{\text{SF}}) - \kappa s\dot{s}_a \tag{10.143}$$

$$\dot{\chi}_{\text{SF}} = r + \dot{\beta} - \kappa\dot{s}_a \tag{10.144}$$

*where $U$ is the speed of the craft and $(e, s)$ is the location on the path of* {SF} *relative to* {$b$}. *If $s = 0$, the variable $e$ represents the closest distance between the actual craft and the origin of* {SF} *tangential to the path. Hence, $s$ can be viewed as an extra controller design parameter for evolution along the path. The arc length that the target has moved along the path is denoted $s_a$ while $\chi_{\text{SF}}$ is the angle between the $x$ axis of* {SF} *and the speed vector; see Figure 10.19. Finally, $\kappa$ is the path curvature.*

**Proof:** From Figure 10.19, it is seen that the distance vectors between {$n$}, {$b$} and {SF} satisfies

$$\vec{r}_{b/n} = \vec{r}_{\text{SF}/n} + \vec{r}_{b/\text{SF}} \tag{10.145}$$

Hence, the time differentiation of $\vec{r}_{b/\text{SF}}$ with {$b$} as the moving reference frame gives

$$\frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_{b/\text{SF}} = \frac{^b\mathrm{d}}{\mathrm{d}t}\vec{r}_{b/\text{SF}} + \vec{\omega}_{b/i} \times \vec{r}_{b/\text{SF}} \tag{10.146}$$

such that

$$\vec{v}_{b/n} = \vec{v}_{\text{SF}/n} + \left(\frac{^b\mathrm{d}}{\mathrm{d}t}\vec{r}_{b/\text{SF}} + \vec{\omega}_{\text{SF}/n} \times \vec{r}_{b/\text{SF}}\right) \tag{10.147}$$

Expressing this in {SF} gives

$$\boldsymbol{v}^{\text{SF}}_{b/n} = \boldsymbol{v}^{\text{SF}}_{\text{SF}/n} + \left(\frac{^b\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}^{\text{SF}}_{b/\text{SF}} + \boldsymbol{\omega}^{\text{SF}}_{\text{SF}/n} \times \boldsymbol{r}^{\text{SF}}_{b/\text{SF}}\right) \tag{10.148}$$

where $\boldsymbol{r}^{\text{SF}}_{b/\text{SF}} = [s, e, 0]^\top$ and $\boldsymbol{v}^{\text{SF}}_{b/n} = \boldsymbol{R}_{z,\chi_{\text{SF}}}[U, 0, 0]^\top$ is the velocity of the vehicle expressed in $\{SF\}$. From this it follows that

$$\begin{aligned}\boldsymbol{R}_{z,\chi_{\text{SF}}}\begin{bmatrix} U \\ 0 \\ 0 \end{bmatrix} &= \boldsymbol{v}^{\text{SF}}_{\text{SF}/n} + \left(\frac{^b\mathrm{d}}{\mathrm{d}t}(\boldsymbol{r}^{\text{SF}}_{b/\text{SF}}) + \boldsymbol{\omega}^{\text{SF}}_{\text{SF}/n} \times \boldsymbol{r}^{\text{SF}}_{b/\text{SF}}\right) \\ &= \begin{bmatrix} \dot{s}_a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{s} \\ \dot{e} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \kappa\dot{s}_a \end{bmatrix} \times \begin{bmatrix} s \\ e \\ 0 \end{bmatrix}\end{aligned} \tag{10.149}$$

Expanding this expression yields

$$U\cos(\chi_{\text{SF}}) = \dot{s} + (1 - \kappa e)\dot{s}_a \tag{10.150}$$

$$U\sin(\chi_{\text{SF}}) = \dot{e} + \kappa s\dot{s}_a \tag{10.151}$$

which proves (10.142) and (10.143). The rotation rate of the angle $\psi - \psi_{\text{SF}}$ between $\{n\}$ and {SF} (see Figure 10.19) is $(\dot{\psi} - \dot{\psi}_{\text{SF}}) = \kappa \dot{s}_a$. Since $\chi_{\text{SF}} = \psi_{\text{SF}} + \beta$, it follows that

$$\begin{aligned} \dot{\chi}_{\text{SF}} &= \dot{\psi} - \kappa \dot{s}_a + \dot{\beta} \\ &= r + \dot{\beta} - \kappa \dot{s}_a \end{aligned} \tag{10.152}$$

which proves (10.144).

***Remark 10.1***
*If* $\dot{s} = s = 0$*, the* {SF} *equations become*

$$\dot{s}_a = \frac{U \cos(\chi_{\text{SF}})}{1 - \kappa e} \tag{10.153}$$

$$\dot{e} = U \sin(\chi_{\text{SF}}) \tag{10.154}$$

*where the term* $1 - \kappa e$ *in the denominator creates a singularity. Hence, the control law requires that the initial position of the craft must be restricted to a tube around the path with radius less than* $1/\kappa_{\max}$*. A discussion on the limitation of this approach is found in Breivik and Fossen (2004a). The constraint* $1 - \kappa e \neq 0$ *is, however, removed by using (10.142)–(10.143) where an additional controller parameter* $s$ *allows the origin of the* {SF} *frame to evolve along the path (Lapierre and Soetanto, 2007).*

***Remark 10.2***
*In Encarnacao* et al. *(2000), the ocean current velocities are included in the kinematic equations of motion together with a state estimator to obtain the optimal sideslip angle during path following. This section presents a different approach where the current velocities are modeled as physical forces, with moments in the expression for* $\dot{\beta}$ *representing the equation of motion in sway. Furthermore, the ocean currents are compensated for by using integral action in the kinematic controller to reduce sensitivity to model parameters.*

**Marine Craft Model for Kinematic Controller**

For a conventional marine craft with no actuation in the transverse direction, the sway force can be approximated by a maneuvering model (see Section 7.1.1):

$$(m - Y_{\dot{v}})\dot{v} + mur = Y(u_r, v_r, r) \tag{10.155}$$

where the hydrodynamic force is due to added mass and linear damping:

$$Y(u_r, v_r, r) = X_{\dot{u}} u_r r + Y_v v_r + Y_r r \tag{10.156}$$

The relative velocities satisfy (see Section 2.4.2)

$$u_r = u - u_c \tag{10.157}$$

$$v_r = v - v_c \tag{10.158}$$

Since ocean currents are slowly varying and the craft speed is constant, the sway acceleration can be approximated by time differentiation of

$$v = U \sin(\beta) \tag{10.159}$$

under the assumptions that $\dot{U} = 0$. This gives

$$\dot{v} = U \cos(\beta)\dot{\beta} \tag{10.160}$$

Combining (10.155)–(10.156) and (10.160) gives

$$\begin{aligned}\dot{\beta} &= \frac{1}{U\cos(\beta)}\dot{v} \\ &= \frac{1}{(m - Y_{\dot{v}})U\cos(\beta)}\left[X_{\dot{u}}u_r r + Y_v v_r + Y_r r - mur\right]\end{aligned} \tag{10.161}$$

Consequently,

$$\dot{\beta} = \frac{1}{(m - Y_{\dot{v}})U\cos(\beta)}\left[(Y_r - (m - X_{\dot{u}})U\cos(\beta) - X_{\dot{u}}u_c))\, r + Y_v U\sin(\beta) - Y_v v_c\right] \tag{10.162}$$

**Kinematic Controller**

The {SF} frame plays the role of the virtual target body axes and tracks the real craft. The error coordinates for control design purposes become $s, e$ and $\tilde{\chi}_{\rm SF} = \chi_{\rm SF} - \chi_d$ which all should be driven to zero. The desired approach angle can be chosen as a function of $e$ (Micaelli and Samson, 1993):

$$\chi_d(e) = -\chi_a \frac{e^{2ke-1}}{e^{2ke+1}} \tag{10.163}$$

where $k > 0$ and $0 < \chi_a < \pi/2$ satisfying $e\chi_d(e) \leq 0$ for all $e$.

An alternative approach is motivated by the LOS algorithm (10.74) (see Breivik and Fossen, 2004a, and Børhaug and Pettersen, 2006):

$$\chi_d(e) = \arctan\left(\frac{-e}{\Delta}\right) \tag{10.164}$$

where $\Delta > 0$ is a constant parameter. Again notice that $e\chi_d(e) \leq 0$ for all $e$.

***Theorem 10.1 (Kinematic Path-Following Controller)***
*A feedback linearization controller for (10.144) (Lapierre and Soetanto, 2007)*

$$r = \dot{\chi}_d - \dot{\beta} + \kappa\dot{s}_a - K_1\,\tilde{\chi}_{\rm SF} \tag{10.165}$$

$$\dot{s}_a = U\cos(\chi_{\rm SF}) + K_2 s \tag{10.166}$$

*where the yaw rate $r$ and path tangential speed $U_d = \dot{s}_a$ are used as control variables, renders the equilibrium point $(s, e, \tilde{\chi}_{\rm SF}) = (0, 0, 0)$ UGAS and ULES for $K_1 > 0$ and $K_2 > 0$.*

**Proof.** Convergence and stability can be proven by noticing that the error dynamics forms a cascade of two systems. For the first system:

$$\dot{\tilde{\chi}}_{\rm SF} + K_1\,\tilde{\chi}_{\rm SF} = 0 \tag{10.167}$$

Consequently, the angle $\chi_{\text{SF}} \to \chi_d$. For the second system in the cascade, consider the Lyapunov function candidate:

$$V = \frac{1}{2}(s^2 + e^2) > 0, \qquad s \neq 0, e \neq 0 \tag{10.168}$$

The time derivative of $V$ under the assumption that $\chi_{\text{SF}} = \chi_d$ is

$$\begin{aligned}\dot{V} &= s\left(U\cos(\chi_d) - (1 - \kappa e)\dot{s}_a\right) + e\left(U\sin(\chi_d) - \kappa s\dot{s}_a\right) \\ &= sU\cos(\chi_d) + eU\sin(\chi_d) - s\left(U\cos(\chi_d) + K_2 s\right) \\ &= -K_2 s^2 + eU\sin(\chi_d)\end{aligned}$$

Exploiting the fact that the desired course angle given by (10.164) satisfies

$$\sin(\chi_d) = \frac{-e}{\sqrt{e^2 + \Delta^2}} \tag{10.169}$$

finally gives

$$\begin{aligned}\dot{V} &= -K_2 s^2 - \frac{U}{\sqrt{e^2 + \Delta^2}} e^2 \\ &< 0, \qquad s \neq 0, e \neq 0\end{aligned} \tag{10.170}$$

for $\Delta > 0$ and $U > 0$. Since the LFC is positive definite and radially unbounded, while its derivative with respect to time is negative, standard Lyapunov arguments for cascaded systems proves that the equilibrium point $(s, e, \tilde{\chi}_{\text{SF}}) = (0, 0, 0)$ is UGAS. In addition, the Jacobian of the error dynamics about the equilibrium point has strictly negative eigenvalues, which proves ULES.

***Remark 10.3***
*A differential equation for the path variable $\varpi$ can be derived by considering the path curvature $\kappa(\varpi)$ given by*

$$\kappa(\varpi) = \frac{|x_d' y_d'' - y_d' x_d''|}{\sqrt{\left(x_d'\right)^2 + \left(y_d'\right)^2}} \tag{10.171}$$

*where $x_d = x_d(\varpi)$ and $y_d = y_d(\varpi)$. The arc length $s_a$ satisfies*

$$\mathrm{d}s_a^2 = \mathrm{d}x^2 + \mathrm{d}y^2 \tag{10.172}$$

*and by dividing by $\mathrm{d}\varpi^2$, this can be rewritten as*

$$\mathrm{d}\varpi = \frac{1}{\sqrt{x_d'(\varpi)^2 + y_d'(\varpi)^2}} \mathrm{d}s_a \tag{10.173}$$

*Hence, from (10.166) it follows that*

$$\dot{\varpi} = \frac{U\cos(\chi_{\rm SF}) + K_2 s}{\sqrt{x_d'(\varpi)^2 + y_d'(\varpi)^2}} \tag{10.174}$$

**Implementation Aspects**

When implementing the kinematic controller (10.165)–(10.166), an expression for $\dot{\beta}$ must be computed from the sway dynamics (10.162). This expression depends on the model parameters. Consider the expression

$$\begin{aligned} r + \dot{\beta} &= r + \frac{1}{(m - Y_{\dot{v}})U\cos(\beta)}\left[(Y_r - (m - X_{\dot{u}})U\cos(\beta) - X_{\dot{u}}u_c))\, r + Y_v U\sin(\beta) - Y_v v_c\right] \\ &= \left(1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})} + \frac{Y_r - X_{\dot{u}}u_c}{(m - Y_{\dot{v}})U\cos(\beta)}\right) r + \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right) \\ &\approx \left(1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})}\right) r + \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right) \\ &= \dot{\chi}_d + \kappa \dot{s}_a - K_1\,\tilde{\chi}_{\rm SF} \end{aligned} \tag{10.175}$$

where the physical property

$$(m - Y_{\dot{v}})U\cos(\beta) \gg Y_r - X_{\dot{u}}u_c \tag{10.176}$$

has been exploited. Solving for $r = r_d$ gives the kinematic controller

$$r_d = \left(1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})}\right)^{-1}\left[\dot{\chi}_d + \kappa U_d - K_1\,\tilde{\chi}_{\rm SF} - \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right)\right] \tag{10.177}$$

$$U_d = U\cos(\chi_{\rm SF}) + K_2 s \tag{10.178}$$

where the desired yaw rate is denoted $r_d$ and the desired speed $U_d = \dot{s}_a$ is the path-tangential speed. The sideslip angle

$$\beta = \arcsin\left(\frac{v}{U}\right) \tag{10.179}$$

and current velocity $v_c$ must be measured or estimated in a state observer. Alternatively, $\beta$ and $v_c$ can be treated as slowly varying parameters, which can be compensated for by adding integral action. This suggests that

$$r_d = \left[1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})}\right]^{-1}\left[\dot{\chi}_d + \kappa U_d - 2\lambda\,\tilde{\chi}_{\rm SF} - \lambda^2 \int_0^t \tilde{\chi}_{\rm SF}(\tau)\mathrm{d}\tau\right] \tag{10.180}$$

where $\lambda > 0$ is a constant parameter used to tune the bandwidth of the error system:

$$\dot{\tilde{\chi}}_{\mathrm{SF}} + 2\lambda\,\tilde{\chi}_{\mathrm{SF}} + \lambda^2 \int_0^t \tilde{\chi}_{\mathrm{SF}}(\tau)\mathrm{d}\tau = \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right) \tag{10.181}$$

For a marine craft at constant course, the integral term will balance the forcing term in the steady state such that

$$\lambda^2 \int_0^t \tilde{\chi}_{\mathrm{SF}}(\tau)\mathrm{d}\tau = \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right) \tag{10.182}$$

and $\tilde{\chi}_{\mathrm{SF}} \to 0$.