Chapter 10

Path Following
Control Architecture

- Path planner
  - Waypoints
  - Path definition
  - Airspeed, altitude, heading, commands
  - Servo commands
  - Wind

- Path manager
  - Path following
    - Unmanned aircraft
  - Autopilot
    - State estimator
      - On-board sensors
  - Map
    - Status
      - Tracking error
        - Position error
          - Destination, obstacles

Path Following

• For small UAVs, a major issue is wind
  – Always present to some degree
  – Usually significant with respect to commanded airspeed

• Wind makes traditional trajectory tracking approaches difficult, if not infeasible
  – Have to know the wind precisely at every instant to determine desired airspeed

• Better approach: path following
• Rather than “follow this trajectory”, we control UAV to “stay on this path”
Path Types

• We will focus on two types of paths to follow:
  – Straight lines between two points in 3-D
    • Inclination of path within climb capabilities of UAV
  – Circular orbits or arcs in the horizontal plane

• Paths for common applications can be built up from these path primitives
  – Methods for following other types of paths found in literature
Straight Line Path Description

\[ \mathbf{p} = \mathbf{r} - \mathbf{e}_p \]

- \( \mathbf{r} \): vector defining initiation of path
- \( \mathbf{q} \): unit vector defining direction of path
- \( \mathbf{p} \): vector defining location of MAV
- \( \chi_q \): course direction of path
- \( e_{py} \): lateral tracking error

Lateral Tracking Problem

Path error:

$$e_p = e_{px} \triangleq \mathcal{R}_i^P (p^i - r^i)$$

where the transformation from inertial frame to path frame is

$$\mathcal{R}_i^P \triangleq \begin{pmatrix} \cos \chi_q & \sin \chi_q & 0 \\ -\sin \chi_q & \cos \chi_q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Lateral Tracking Problem

Relative error dynamics in path frame:

\[
\begin{pmatrix}
\dot{e}_{px} \\
\dot{e}_{py}
\end{pmatrix}
= \begin{pmatrix}
\cos \chi_q & \sin \chi_q \\
-sin \chi_q & \cos \chi_q
\end{pmatrix}
\begin{pmatrix}
V_g \cos \chi \\
V_g \sin \chi
\end{pmatrix}
= V_g \begin{pmatrix}
\cos(\chi - \chi_q) \\
\sin(\chi - \chi_q)
\end{pmatrix}
\]

Regulate the cross-track error \(e_{py}\) to zero by commanding the course angle:

\[
\dot{e}_{py} = V_g \sin(\chi - \chi_q) \\
\ddot{x} = b\ddot{x}(\dot{x}^c - \dot{x}) + b\chi(\chi^c - \chi)
\]

Select \(\chi^c\) so that \(e_{py} \rightarrow 0\)
Longitudinal Tracking Problem

$e_p = p - r$

$e_p = \begin{pmatrix} e_{pn} \\ e_{pe} \\ e_{pd} \end{pmatrix} \Delta p^i - r^i = \begin{pmatrix} p_n - r_n \\ p_e - r_e \\ p_d - r_d \end{pmatrix}$

$n = \frac{q \times k^i}{\|q \times k^i\|}$

$s^i = \begin{pmatrix} s_n \\ s_e \\ s_d \end{pmatrix}$

$a = e_p^i - (e_p^i \cdot n)n$

Longitudinal Tracking Problem

By similar triangles

\[
\frac{h_d + r_d}{\sqrt{s_n^2 + s_e^2}} = \frac{-q_d}{\sqrt{q_n^2 + q_e^2}}
\]

Desired altitude based on current location

\[
h_d(r, p, q) = -r_d - \sqrt{s_n^2 + s_e^2} \left( \frac{q_d}{\sqrt{q_n^2 + q_e^2}} \right)
\]

Select \(h^c\) so that \(h \to h_d(r, p, q)\)
Longitudinal Guidance Strategy

Use altitude state machine from Ch. 6.
Closed-loop altitude dynamics:

\[
\frac{h}{h^c} = \frac{b_h s + b_h}{s^2 + b_h s + b_h}.
\]

Altitude error:

\[
e_h \triangleq h - h_d(r, p, q) = h - h^c
\]

Error dynamics:

\[
\frac{e_h}{h^c} = 1 - \frac{h}{h^c} = \frac{s^2}{s^2 + b_h s + b_h}
\]

Applying FVT:

\[
e_{h,ss} = \lim_{s \to 0} s \frac{s^2}{s^2 + b_h s + b_h} h^c = 0, \quad \text{for } h^c = \frac{H_0}{s}, \frac{H_0}{s^2}
\]
Lateral Tracking - Vector Field Concept

Desired course based on cross-track error:

\[ \chi_d(e_{py}) = -\chi \infty \frac{2}{\pi} \tan^{-1}(k_{path} e_{py}) \]
Vector Field Tuning

$k_{\text{path}}$ is a positive constant that affects the rate of transition of the desired course

- $k_{\text{path}}$ large $\rightarrow$ short, abrupt transition
- $k_{\text{path}}$ small $\rightarrow$ long, gradual transition
Lyapunov’s 2\textsuperscript{nd} Method

For a system having a state vector \( x \), consider an energy-like function \( V(x) : \mathbb{R}^n \mapsto \mathbb{R} \) such that

\[
V(x) \geq 0 \text{ (positive definite)} \\
V(x) = 0 \text{ for } x = 0 \\
\text{and} \\
\dot{V}(x) \leq 0 \text{ (negative definite)} \\
\dot{V}(x) = 0 \text{ for } x = 0.
\]

If such a function \( V(x) \) can be defined, then \( x \) goes to zero asymptotically and the system is stable.
Lateral Tracking Stability Analysis

Define the Lyapunov function $W(e_{py}) = \frac{1}{2}e_{py}^2$

Assume that course controller works and $\chi = \chi_q + \chi^d(e_{py})$

Since

$$\dot{W} = e_{py}\dot{e}_{py}$$

$$= -V_a e_{py} \sin \left( \chi \frac{2}{\pi} \tan^{-1}(k_{path} e_{py}) \right)$$

$$< 0$$

for $e_{py} \neq 0$, then $e_{py} \to 0$ asymptotically
Smallest Angle Turn Logic

\[
\chi_q = \arctan2(q_e, q_n) + 2\pi m
\]

\[m \in \mathcal{N}\] is selected so that \(-\pi \leq \chi_q - \chi \leq \pi\)

---

**Algorithm 3** Straight-line Following: \([h^c, \chi^c] = \text{followStraightLine}(r, q, p, \chi)\)

- **Input:** Path definition \(r = (r_n, r_e, r_d)^T\) and \(q = (q_n, q_e, q_d)^T\), MAV position \(p=(p_n, p_e, p_d)^T\), course \(\chi\), gains \(\chi_\infty, k_{\text{path}}\), sample rate \(T_s\).
- 1: Compute commanded altitude using equation (10.5).
- 2: \(\chi_q \leftarrow \arctan2(q_e, q_n)\)
- 3: while \(\chi_q - \chi < -\pi\) do
- 4: \(\chi_q \leftarrow \chi_q + 2\pi\)
- 5: end while
- 6: while \(\chi_q - \chi > \pi\) do
- 7: \(\chi_q \leftarrow \chi_q - 2\pi\)
- 8: end while
- 9: \(e_{py} \leftarrow -\sin \chi_q (p_n - r_n) + \cos \chi_q (p_e - r_e)\)
- 10: Compute commanded course angle using equation (10.8).
- 11: return \(h^c, \chi^c\)

---

Orbit Following

Orbit definition:

\[ \mathcal{P}_{\text{orbit}}(c, \rho, \lambda) = \left\{ r \in \mathbb{R}^3 : r = c + \lambda \rho \begin{pmatrix} \cos \varphi, & \sin \varphi, & 0 \end{pmatrix}^\top, \varphi \in [0, 2\pi) \right\} \]

Center: \( c \in \mathbb{R}^3 \)
Radius: \( \rho \in \mathbb{R} \)
Direction: \( \lambda = 1 \) (CW) or \( \lambda = -1 \) (CCW)

In polar coordinates, the position of MAV given by:
\( d \): radial distance from orbit center
\( \varphi \): phase angle of relative position

**Orbit Following**

Easiest to analyze in polar coordinates. Using

\[
\begin{pmatrix}
\dot{p}_n \\
\dot{p}_e
\end{pmatrix}
= \begin{pmatrix}
V_g \cos \chi \\
V_g \sin \chi
\end{pmatrix}
\]

and converting to polar coordinates gives

\[
\begin{pmatrix}
\dot{d} \\
\dot{d} \phi
\end{pmatrix}
= \begin{pmatrix}
\cos \varphi & \sin \varphi \\
-\sin \varphi & \cos \varphi
\end{pmatrix}
\begin{pmatrix}
\dot{p}_n \\
\dot{p}_e
\end{pmatrix}
= \begin{pmatrix}
\cos \varphi & \sin \varphi \\
-\sin \varphi & \cos \varphi
\end{pmatrix}
\begin{pmatrix}
V_g \cos \chi \\
V_g \sin \chi
\end{pmatrix}
= \begin{pmatrix}
V_g \cos(\chi - \varphi) \\
V_g \sin(\chi - \varphi)
\end{pmatrix}
\]

The result is

\[
\begin{align*}
\dot{d} &= V_g \cos(\chi - \varphi) \\
\dot{d} \phi &= \frac{V_g}{d} \sin(\chi - \varphi) \\
\ddot{\chi} &= -b \dot{\chi} \dot{\chi} + b \chi (\chi^c - \chi)
\end{align*}
\]
Orbit Following

Define

\[ \chi^o = \varphi + \lambda \frac{\pi}{2} \]

When \( d \gg \rho \) \( \rightarrow \chi_d \approx \chi^o + \lambda \frac{\pi}{2} \).

When \( d = \rho \) \( \rightarrow \chi_d = \chi^o \).

Therefore, let the desired course angle be

\[ \chi_d(d-\rho, \lambda) = \chi^o + \lambda \tan^{-1}\left(k_{\text{orbit}} \left( \frac{d - \rho}{\rho} \right) \right) \]
Orbit Tracking Stability Analysis

Define the Lyapunov function $W = \frac{1}{2}(d - \rho)^2$

Assume that course controller works and $\chi = \chi^d(d - \rho, \lambda)$

Since

$$\dot{W} = (d - \rho)\dot{d}$$
$$= (d - \rho)(V_g \cos(\chi - \varphi))$$
$$= -V_g(d - \rho)\sin\left(\tan^{-1}\left(k_{\text{orbit}}\left(\frac{d - \rho}{\rho}\right)\right)\right)$$
$$< 0$$

for $d - \rho \neq 0$, then $d - \rho \to 0$ asymptotically
Orbit Following

The commanded course is

\[ \chi^c(t) = \varphi + \lambda \left[ \frac{\pi}{2} + \tan^{-1} \left( k_{\text{orbit}} \left( \frac{d - \rho}{\rho} \right) \right) \right] \]

The orbit angle must be wrapped:

\[ \varphi = \text{atan2}(p_e - c_e, p_n - c_n) + 2\pi m \]

Algorithm 4 Circular Orbit Following: \([h^c, \chi^c] = \text{followOrbit}(c, \rho, \lambda, p, \chi)\)

<table>
<thead>
<tr>
<th>Input: Orbit center (c = (c_n, c_e, c_d)^\top), radius (\rho), and direction (\lambda), MAV position (p = (p_n, p_e, p_d)^\top), course (\chi), gains (k_{\text{orbit}}), sample rate (T_s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: (h^c \leftarrow -c_d)</td>
</tr>
<tr>
<td>2: (d \leftarrow \sqrt{(p_n - c_n)^2 + (p_e - c_e)^2})</td>
</tr>
<tr>
<td>3: (\varphi \leftarrow \text{atan2}(p_e - c_e, p_n - c_n))</td>
</tr>
<tr>
<td>4: while (\varphi - \chi &lt; -\pi) do</td>
</tr>
<tr>
<td>5: (\varphi \leftarrow \varphi + 2\pi)</td>
</tr>
<tr>
<td>6: end while</td>
</tr>
<tr>
<td>7: while (\varphi - \chi &gt; \pi) do</td>
</tr>
<tr>
<td>8: (\varphi \leftarrow \varphi - 2\pi)</td>
</tr>
<tr>
<td>9: end while</td>
</tr>
<tr>
<td>10: Compute commanded course angle using equation (10.13).</td>
</tr>
<tr>
<td>11: return (h^c, \chi^c)</td>
</tr>
</tbody>
</table>
Roll Feedforward: no wind

For orbit following:

\[ \chi^c(t) = \varphi + \lambda \left[ \frac{\pi}{2} + \tan^{-1} \left( k_{\text{orbit}} \left( \frac{d - \rho}{\rho} \right) \right) \right]. \]

Note:

\[ d - \rho = 0 \quad \Rightarrow \quad \chi^c = 0 \]
\[ \Rightarrow \quad \varphi^c = 0 \]
\[ \Rightarrow \quad \text{UAV will immediately deviate from orbit.} \]

Problem can be fixed by commanded a roll feedforward for when aircraft is on the orbit.

If on the orbit and no wind, then

\[ \dot{\psi}^d = \lambda \frac{V_a}{R}. \]

Coordinated turn condition:

\[ \dot{\psi} = \frac{g}{V_a} \tan \phi. \]

Equating and solving for roll gives

\[ \phi_{ff} = \lambda \tan^{-1} \left( \frac{V_a^2}{gR} \right). \quad (1) \]
Roll Feedforward: wind

When wind is present we have

\[ \dot{\chi}^d(t) = \lambda \frac{V_g(t)}{R}, \]

where \( V_g \) is the time varying ground speed. The coordinated turn condition in wind is

\[ \dot{\chi} = \frac{g}{V_g} \tan \phi \cos(\chi - \psi). \]

Equating these expressions and solving for \( \phi \) gives

\[ \phi_{ff} = \lambda \tan^{-1} \left( \frac{V_g^2}{gR \cos(\chi - \psi)} \right). \]

Or equivalently

\[ \phi_{ff} = \lambda \tan^{-1} \left( \frac{(w_n \cos \chi + w_e \sin \chi) + \sqrt{V_a^2 - (w_n \sin \chi - w_e \cos \chi)^2 - w_d^2}}{gR \sqrt{\frac{V_a^2 - (w_n \sin \chi - w_e \cos \chi)^2 - w_d^2}{V_a^2 - w_d^2}}} \right), \]
Dubins Airplane Model


Dubins Airplane model:

\[
\begin{align*}
\dot{r}_n &= V \cos \psi \cos \gamma^c \\
\dot{r}_e &= V \sin \psi \cos \gamma^c \\
\dot{r}_d &= -V \sin \gamma^c \\
\dot{\psi} &= \frac{g}{V} \tan \phi^c
\end{align*}
\]

Where the commanded flight path angle \( \gamma^c \) and the commanded roll angle \( \phi^c \) are constrained by

\[
\begin{align*}
|\phi^c| &\leq \bar{\phi} \\
|\gamma^c| &\leq \bar{\gamma}.
\end{align*}
\]
3D Vector Field Path Following


The path is specified as the intersection of two 2D manifolds given by

\[ \alpha_1(r) = 0 \]
\[ \alpha_2(r) = 0 \]

\( r \in \mathbb{R}^3 \). Define the composite function

\[ W(r) = \frac{1}{2} \alpha_1^2(r) + \frac{1}{2} \alpha_2^2(r), \]

Note that the gradient

\[ \frac{\partial W}{\partial r} = \alpha_1(r) \frac{\partial \alpha_1}{\partial r}(r) + \alpha_2(r) \frac{\partial \alpha_2}{\partial r}(r). \]

points away from the path.
3D Vector Field Path Following

The desired velocity vector can be chosen as

\[
\mathbf{u}' = -K_1 \frac{\partial W}{\partial \mathbf{r}} + K_2 \frac{\partial \alpha_1}{\partial \mathbf{r}} \times \frac{\partial \alpha_2}{\partial \mathbf{r}}
\]

velocity directed toward the path \hspace{1cm} velocity directed along the path

where \( K_1 > 0 \) and \( K_2 \) are symmetric tuning matrices, and the definiteness of \( K_2 \) determines the direction of travel along the path.

Since \( \mathbf{u}' \) may not equal \( V_a \), normalize to get

\[
\mathbf{u} = V_a \frac{\mathbf{u}'}{\| \mathbf{u}' \|}.
\]
3D Vector Field Path Following

Setting the NED components of the velocity of the Dubins airplane model to \( \mathbf{u} = (u_1, u_2, u_3)^\top \) gives

\[
\begin{align*}
V \cos \psi^d \cos \gamma^c &= u_1 \\
V \sin \psi^d \cos \gamma^c &= u_2 \\
-V \sin \gamma^c &= u_3.
\end{align*}
\]

Solving for \( \gamma^c \), and \( \psi^d \) results in

\[
\begin{align*}
\gamma^c &= -\text{sat}_{\gamma} \left[ \sin^{-1} \left( \frac{u_3}{V} \right) \right] \\
\psi^d &= \text{atan2}(u_2, u_1).
\end{align*}
\]

Assuming the inner-loop lateral-directional dynamics are accurately modeled by the coordinated-turn equation, the commanded roll angle is

\[
\dot{\phi}^c = \text{sat}_{\phi} \left[ k_\phi (\psi^d - \psi) \right],
\]

where \( k_\phi \) is a positive constant.
3D Vector Field – Straight Line path

The straight line path is given by

\[ \mathcal{P}_{\text{line}}(c_\ell, \psi_\ell, \gamma_\ell) = \left\{ \mathbf{r} \in \mathbb{R}^3 : \mathbf{r} = c_\ell + \sigma \mathbf{q}_\ell, \sigma \in \mathbb{R} \right\}, \]

where

\[ \mathbf{q}_\ell = \begin{pmatrix} q_n \\ q_e \\ q_d \end{pmatrix} = \begin{pmatrix} \cos \psi_\ell \cos \gamma_\ell \\ \sin \psi_\ell \cos \gamma_\ell \\ -\sin \gamma_\ell \end{pmatrix}. \]

Define

\[ \mathbf{n}_{\text{lon}} = \begin{pmatrix} -\sin \psi_\ell \\ \cos \psi_\ell \\ 0 \end{pmatrix}, \]

\[ \mathbf{n}_{\text{lat}} = \mathbf{n}_{\text{lon}} \times \mathbf{q}_\ell = \begin{pmatrix} -\cos \psi_\ell \sin \gamma_\ell \\ -\sin \psi_\ell \sin \gamma_\ell \\ -\cos \gamma_\ell \end{pmatrix}, \]

to get

\[ \alpha_{\text{lon}}(\mathbf{r}) = \mathbf{n}_{\text{lon}}^\top(\mathbf{r} - \mathbf{c}_\ell) = 0 \]
\[ \alpha_{\text{lat}}(\mathbf{r}) = \mathbf{n}_{\text{lat}}^\top(\mathbf{r} - \mathbf{c}_\ell) = 0. \]
3D Vector Field – Helical Path

A helical path is then defined as

\[ \mathcal{P}_{\text{helix}}(c_h, \psi_h, \lambda_h, R_h, \gamma_h) = \{ \mathbf{r} \in \mathbb{R}^3 : \alpha_{\text{cyl}}(\mathbf{r}) = 0 \text{ and } \alpha_{\text{pl}}(\mathbf{r}) = 0 \} . \]

where

\[ \alpha_{\text{cyl}}(\mathbf{r}) = \left( \frac{r_n - c_n}{R_h} \right)^2 + \left( \frac{r_e - c_e}{R_h} \right)^2 - 1 \]

\[ \alpha_{\text{pl}}(\mathbf{r}) = \left( \frac{r_d - c_d}{R_h} \right) + \frac{\tan \gamma_h}{\lambda_h} \left( \tan^{-1} \left( \frac{r_e - c_e}{r_n - c_n} \right) - \psi_h \right) \]

where the initial position along the helix is

\[ \mathbf{r}(0) = c_h + \begin{pmatrix} R_h \cos \psi_h \\ R_h \sin \psi_h \\ 0 \end{pmatrix} . \]

\( c_h \) is the center of the helix, \( R_h \) is the radius, \( \gamma_h \) is the climb angle.