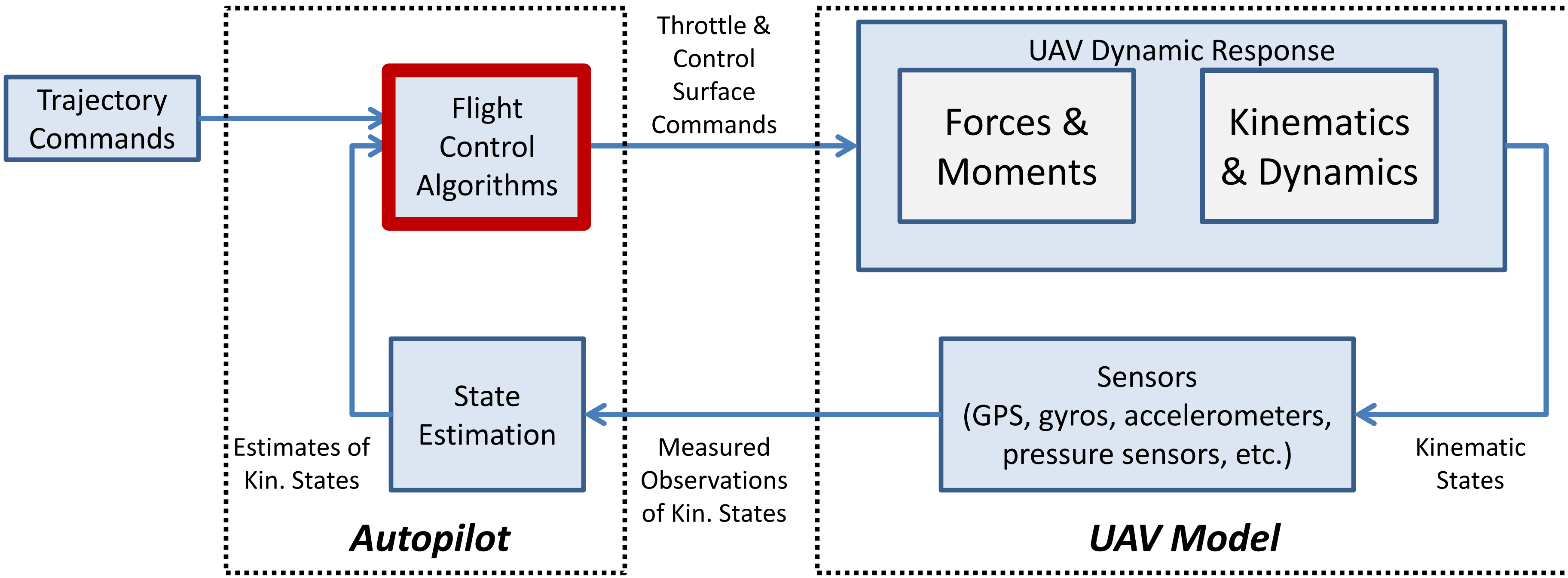
UAV Systems & Control
Lecture 8

Analytically-derived PID gains for:

-Lateral channel

-Longitudinal channel

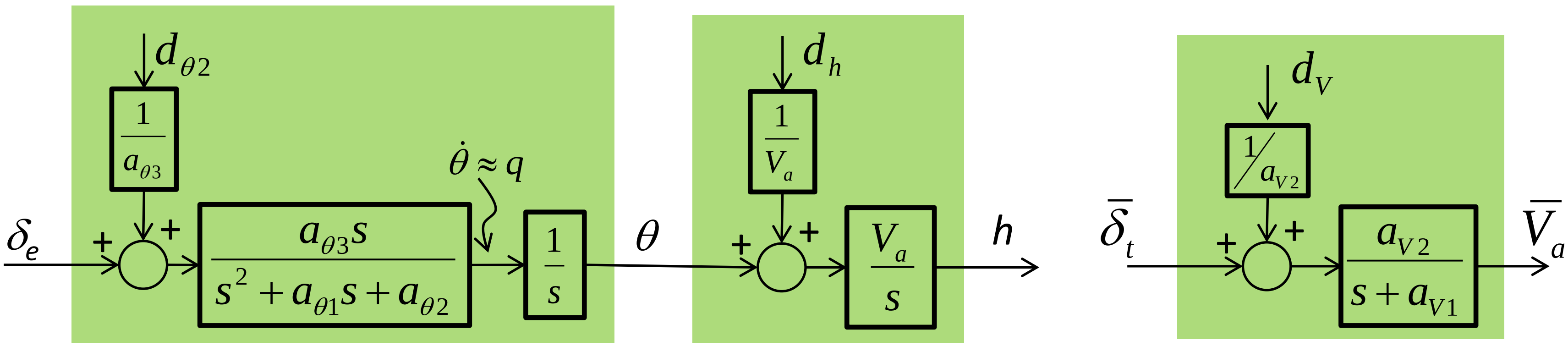
UAV System



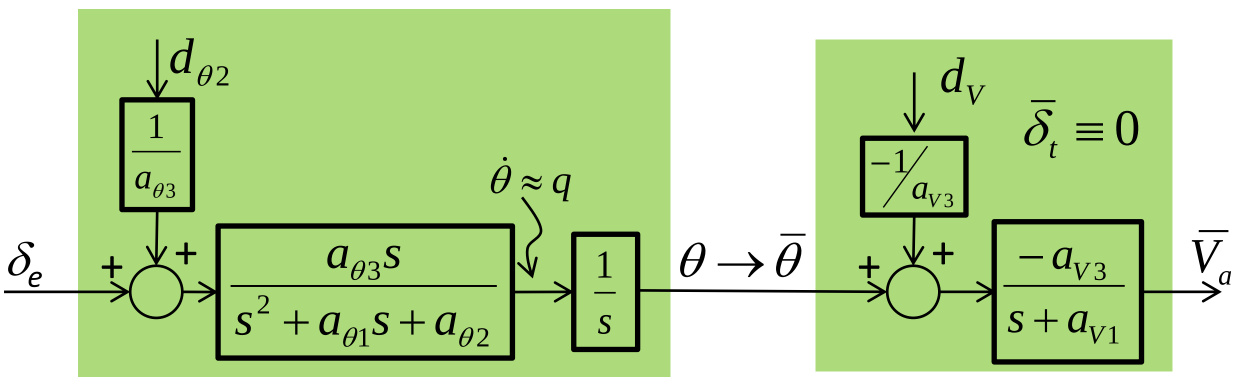
- In the previous lectures we:
  - Developed the simplified models we'll use for a "cookie-cutter" method of autopilot gain selection
  - Used method to develop roll hold autopilot
- In this section we will further develop the "cookie-cutter" autopilot gain selection method develop lateral and longitudinal autopilot gains
  - Autopilot logic will be implemented in "Flight Control Algorithms"

Fixed Wing Longitudinal Linear Models

Level Flight Mode: Pitch controls Altitude, Throttle controls Airspeed



Launch/Climb/Descend Modes: Pitch controls Airspeed, Throttle is fixed



$$a_{\theta 1} = -\frac{\rho V_a^2 S c}{2} \frac{1}{J_y} \frac{c}{2V_a} C_{mq}$$

$$a_{\theta 2} = -\frac{\rho V_a^2 S c}{2} \frac{1}{J_y} C_{m\alpha}$$

$$a_{\theta 3} = +\frac{\rho V_a^2 S c}{2} \frac{1}{J_y} C_{m\delta e}$$

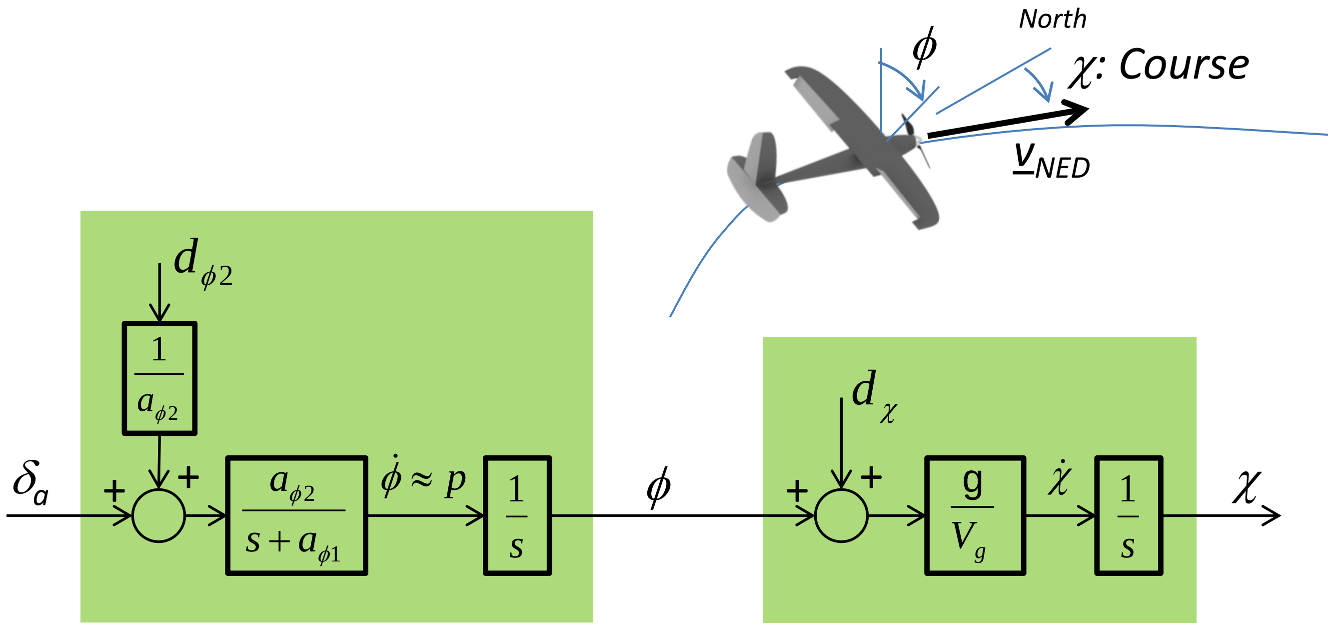
$$a_{V1} = -\frac{\rho C_{prop} S_{prop}}{mass} \left\{ \delta_t^* \left(1 - 2\delta_t^*\right)\left(k_{motor} - V_a^*\right) - \delta_t^* V_a^* \right\} + \frac{\rho V_a^* S}{mass}\left(C_{Do} + C_{D\alpha}\alpha^* + C_{D\delta e}\delta_e^*\right)$$

$$a_{V2} = \frac{\rho C_{prop} S_{prop}}{mass}\left(k_{motor} - V_a^*\right)\left(V_a^* + 2\delta_t^*\left(k_{motor} - V_a^*\right)\right)$$

$$a_{V3} = g\cos\left(\theta^* - \alpha^*\right)$$

*x** values are "trim" values. Response models are a function of nominal flight condition.

Fixed Wing Lateral Linear Models



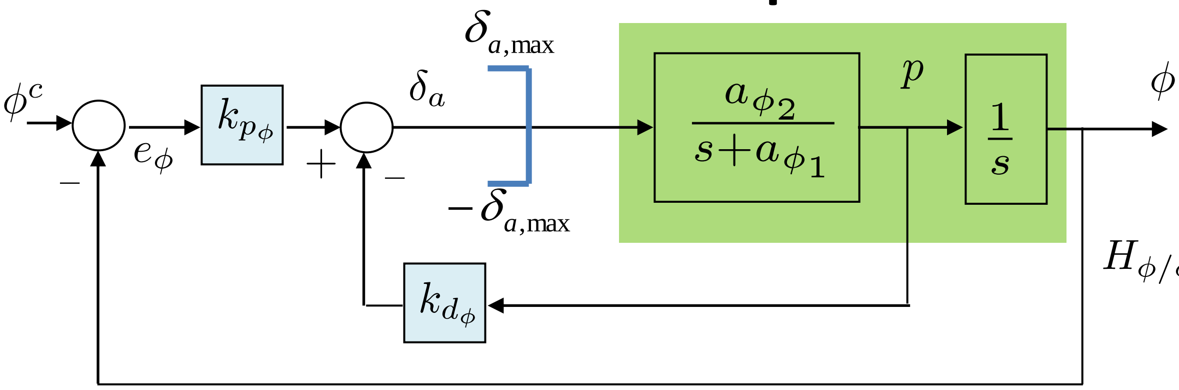
$$a_{\phi 1} = -\frac{\rho V_a^2 S b}{2} \frac{J_z C_{lp} + J_{xz} C_{np}}{\Gamma} \frac{b}{2V_a}$$

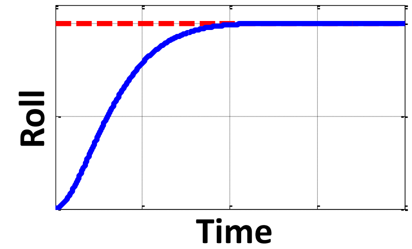
$$\Gamma = J_x J_z - J_{xz}^2$$

$$a_{\phi 2} = +\frac{\rho V_a^2 S b}{2} \frac{J_z C_{l\delta a} + J_{xz} C_{n\delta a}}{\Gamma}$$

$$V_g \approx V_a$$

Roll Control Autopilot





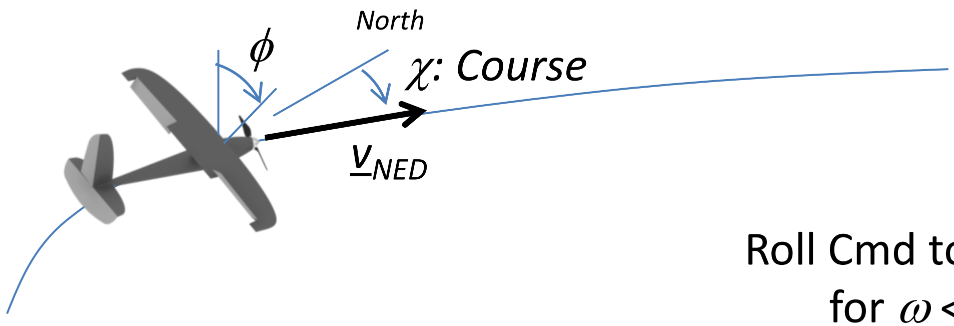
$$H_{\phi/\phi^c}(s) = \frac{k_{p_\phi} a_{\phi_2}}{s^2 + \underbrace{(a_{\phi_1} + a_{\phi_2} k_{d_\phi})}_{2\zeta_\phi \omega_{n,\phi}} s + \underbrace{k_{p_\phi} a_{\phi_2}}_{\omega_{n,\phi}^2}}$$

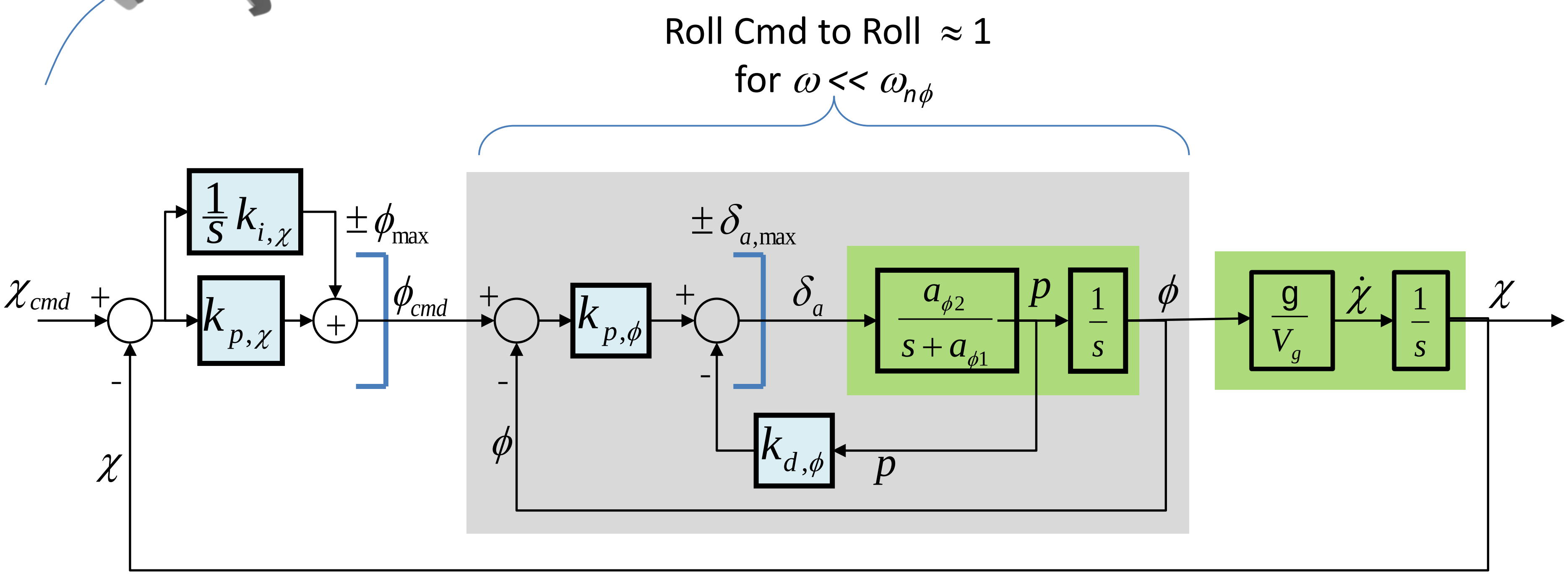
- Using a simplified linear design model, we've developed a method to analytically choose autopilot gains for roll control
  - Choose design parameters:
    - $e_{\phi,max}$: Amount of roll error which will cause aileron saturation (smaller value means faster response, but more saturation)
    - ζ_ϕ : Use to choose overshoot
  - Then:

$$k_{p,\phi} = \frac{\delta_{a,\max}}{e_{\phi,\max}} sign(a_{\phi 2}) \qquad \omega_{n,\phi} = \sqrt{k_{p,\phi}\, a_{\phi 2}} \qquad k_{d,\phi} = \frac{2\zeta_\phi \omega_{n,\phi} - a_{\phi 1}}{a_{\phi 2}}$$

  - Note: Book describes using an integral gain to remove steady-state error due to disturbances. We won't. Integrators add delay and instability, which isn't desired on inner loops. An integrator on the course loop will correct any steady state errors. (Per advice on book website.)

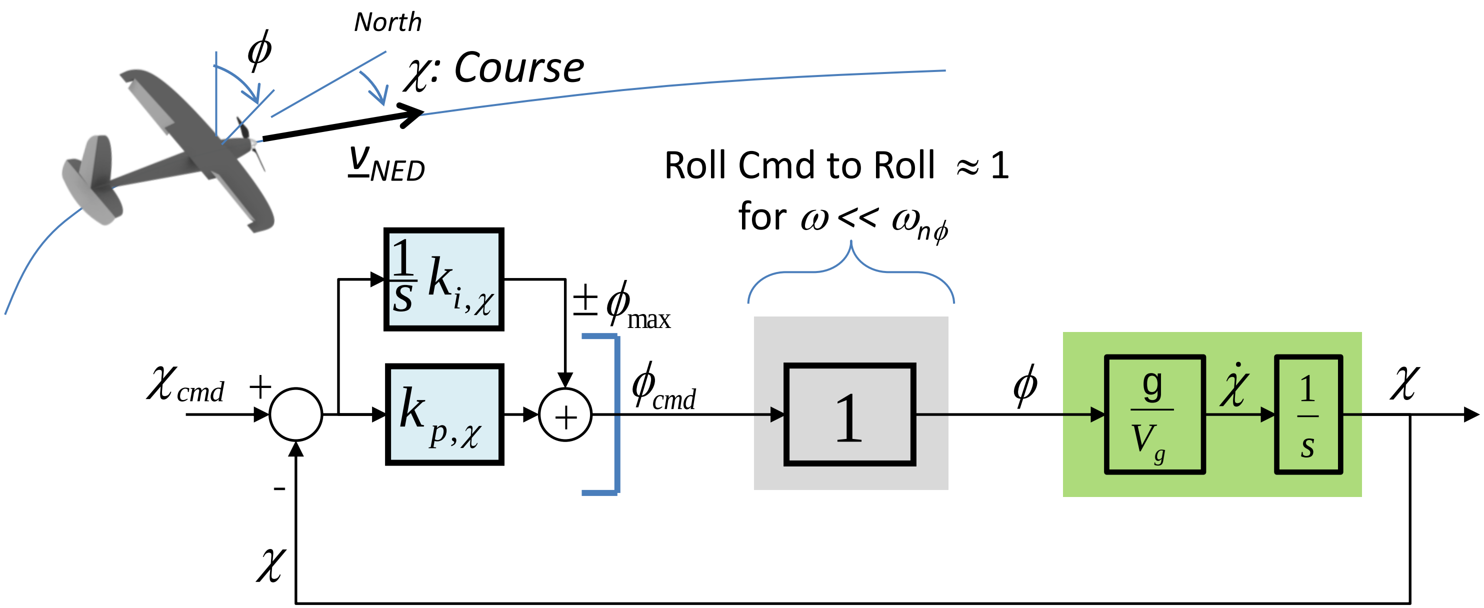
Course Control Autopilot





- A bank-to-turn aircraft changes course by rolling into a turn
- We will use a PI controller to control our course angle (Roll Cmd is a function of course error)
- We can simplify our course controller design by assuming that the roll response is *quick*
  - i.e. We will *slow down* the course control enough to assume [Roll Cmd to Roll] ≈ 1

Course Control Autopilot



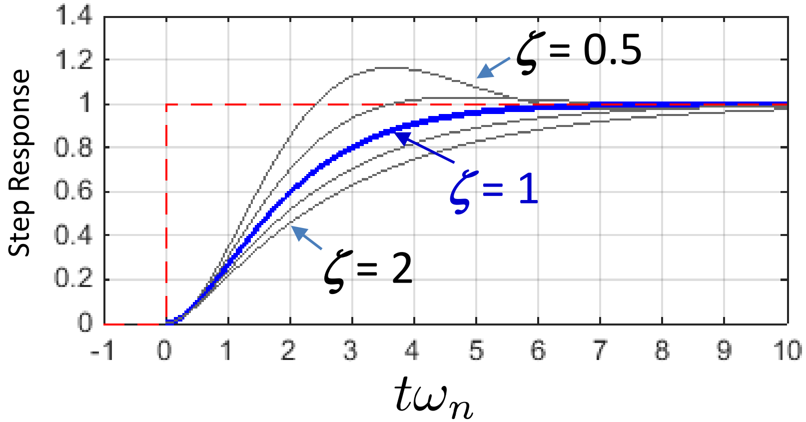
$$H_{\chi/\chi_C} = \frac{k_{p,\chi} g/V_g s + k_{i,\chi} g/V_g}{s^2 + k_{p,\chi} g/V_g s + k_{i,\chi} g/V_g} = \frac{2\zeta_\chi \omega_{n,\chi} s + \omega_{n,\chi}^2}{s^2 + 2\zeta_\chi \omega_{n,\chi} s + \omega_{n,\chi}^2}$$

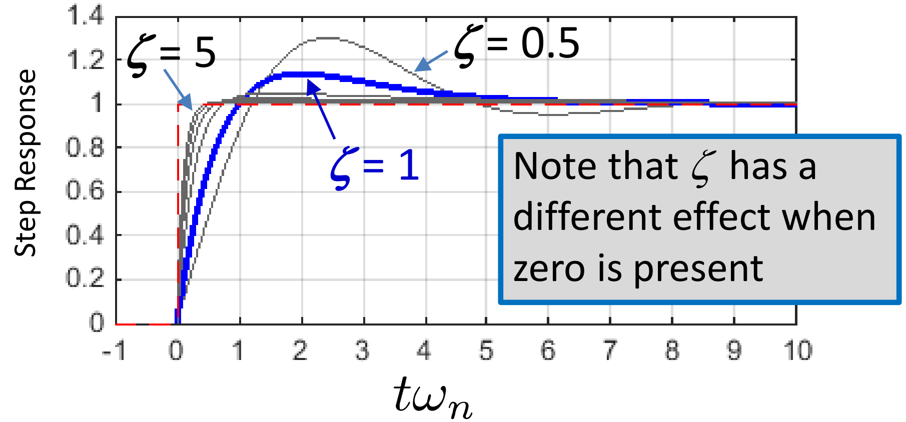
- Treating the “Roll Cmd to Roll” response as unity simplifies the course control loop
  - Valid assumption if course control is notably slower than roll control
- Resulting course response has a 2nd-order denominator, but with a numerator zero
  - We’ll want: $\omega_{n\chi} << \omega_{n\phi}$

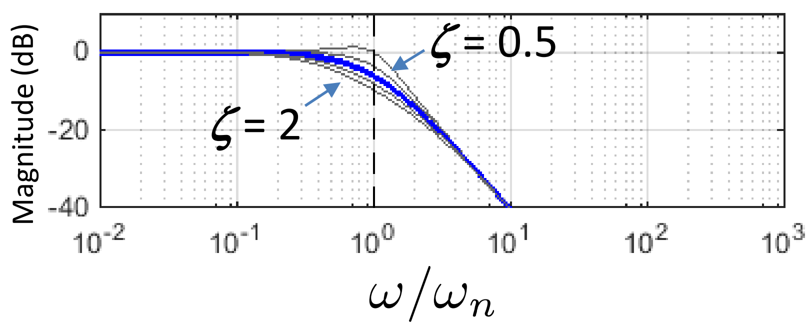
Canonical 2nd Order *vs.* 2nd Order with Zero

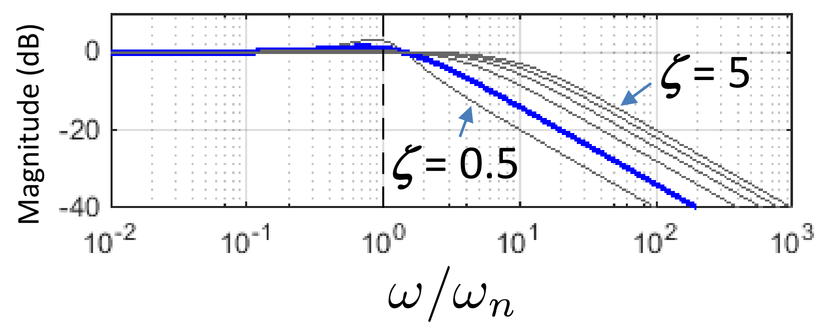
$$H = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$H = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$









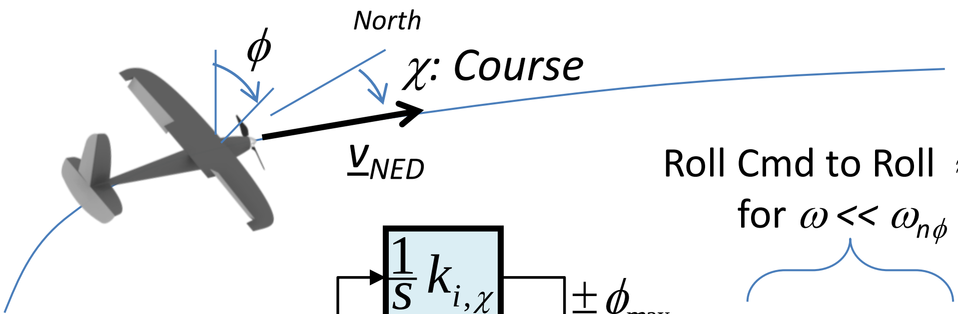
Canonical 2nd Order:

As ζ increases:
- Overshoot decreases
- Rise time increases
- Bandwidth decreases slightly

2nd Order with a zero:

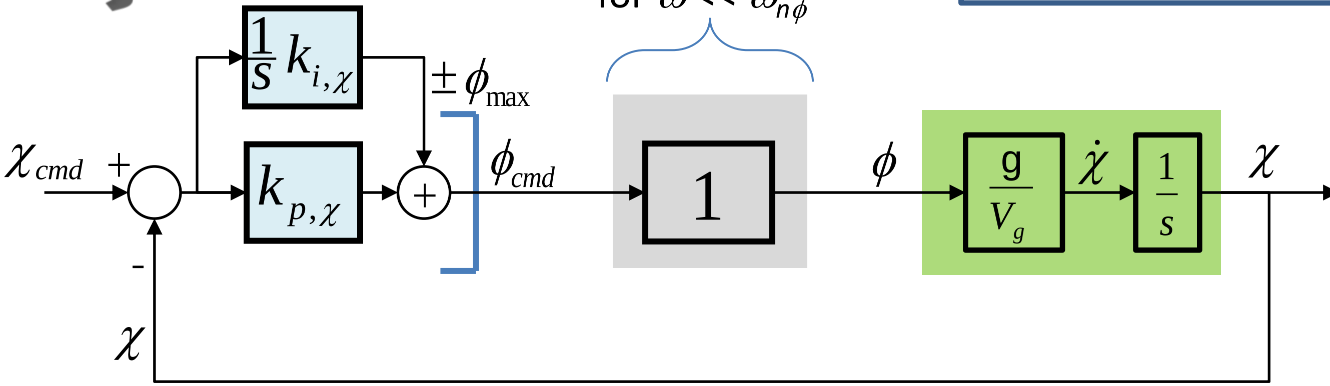
As ζ increases:
- Overshoot decreases
- Rise time *decreases*
- Bandwidth *increases*

Course Control Autopilot



Note: For gain generation, assume no winds. Thus:

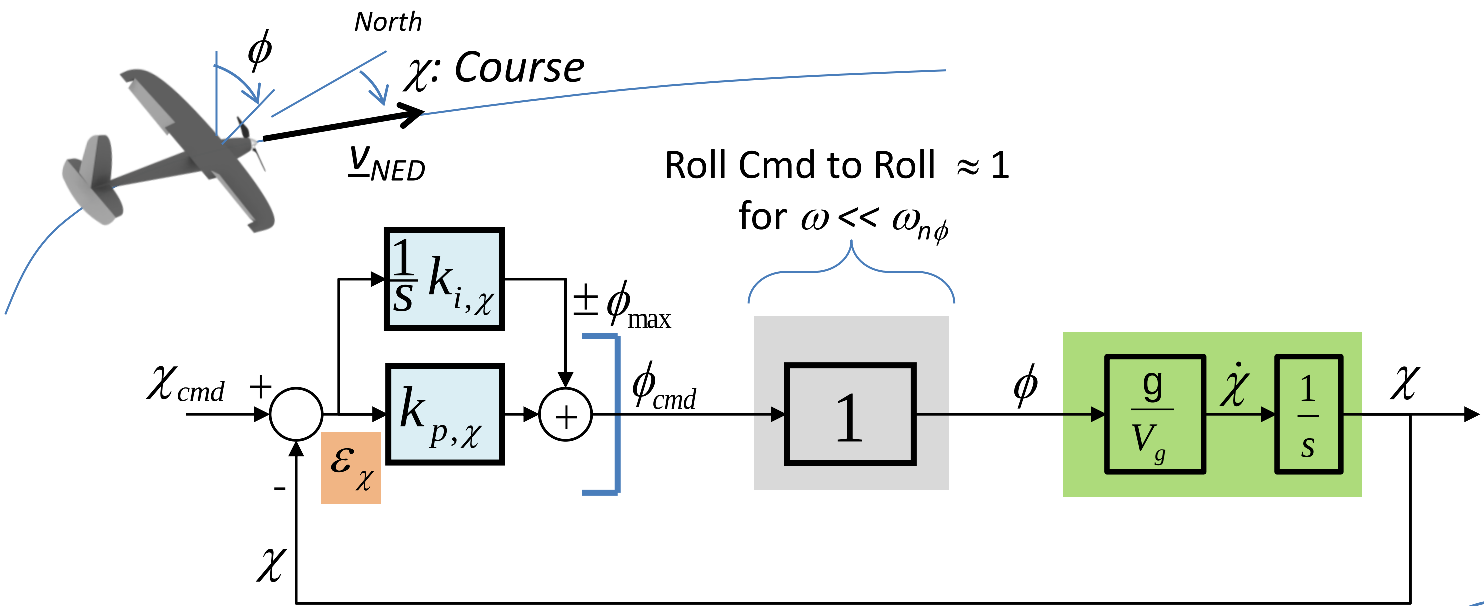
$$V_g = V_a$$

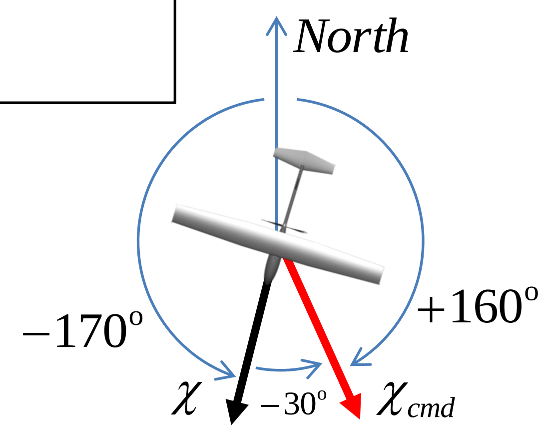


$$H_{\chi/\chi_C} = \frac{k_{p,\chi} g/V_g s + k_{i,\chi} g/V_g}{s^2 + k_{p,\chi} g/V_g s + k_{i,\chi} g/V_g} = \frac{2\zeta_\chi \omega_{n,\chi} s + \omega_{n,\chi}^2}{s^2 + 2\zeta_\chi \omega_{n,\chi} s + \omega_{n,\chi}^2}$$

- Equating coefficients: $k_{p,\chi} = 2\zeta_\chi \omega_{n,\chi} V_g / g \qquad k_{i,\chi} = \omega_{n,\chi}^2 V_g / g$
- Let: $\omega_{n,\chi} = \frac{1}{W_\chi} \omega_{n,\phi}$, $W_\chi >> 1$, e.g. 5-50
- Course Loop Design Parameters:
  - W_χ : Bandwidth separation factor
  - ζ_χ : Damping Ratio (higher due to numerator zero)

Course Control Autopilot





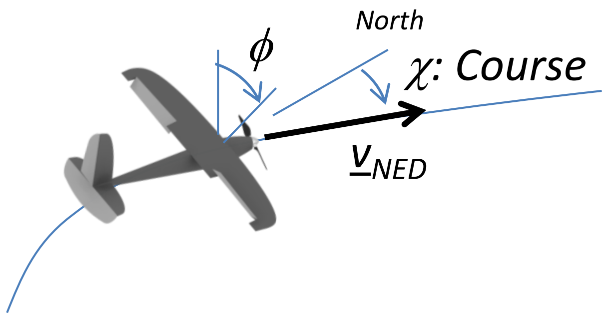
Implementation Note:

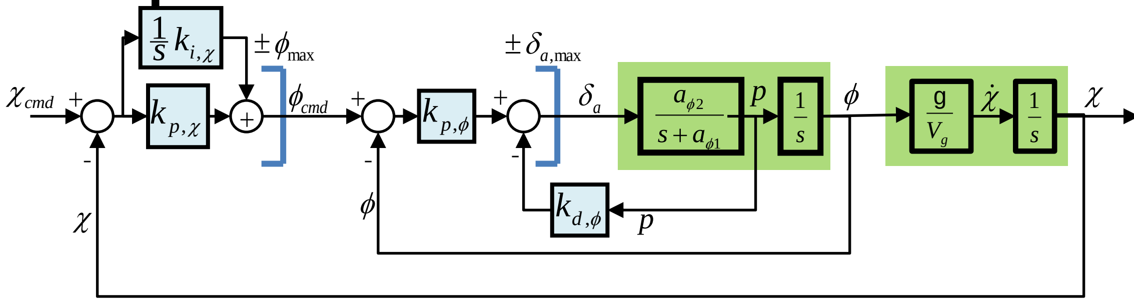
- ε_χ is the error between the course command and the achieved course: $\varepsilon_\chi = \chi_{cmd} - \chi$
- Care must be taken because χ_{cmd} and χ "wrap" every 2π radians (or 360°)
- Want: $-\pi \le \varepsilon_\chi \le \pi \qquad (\pi \text{ rad} = 180^\circ)$
- Use: $\varepsilon_\chi = \mathrm{mod}(\chi_{cmd} - \chi + \pi, 2\pi) - \pi$

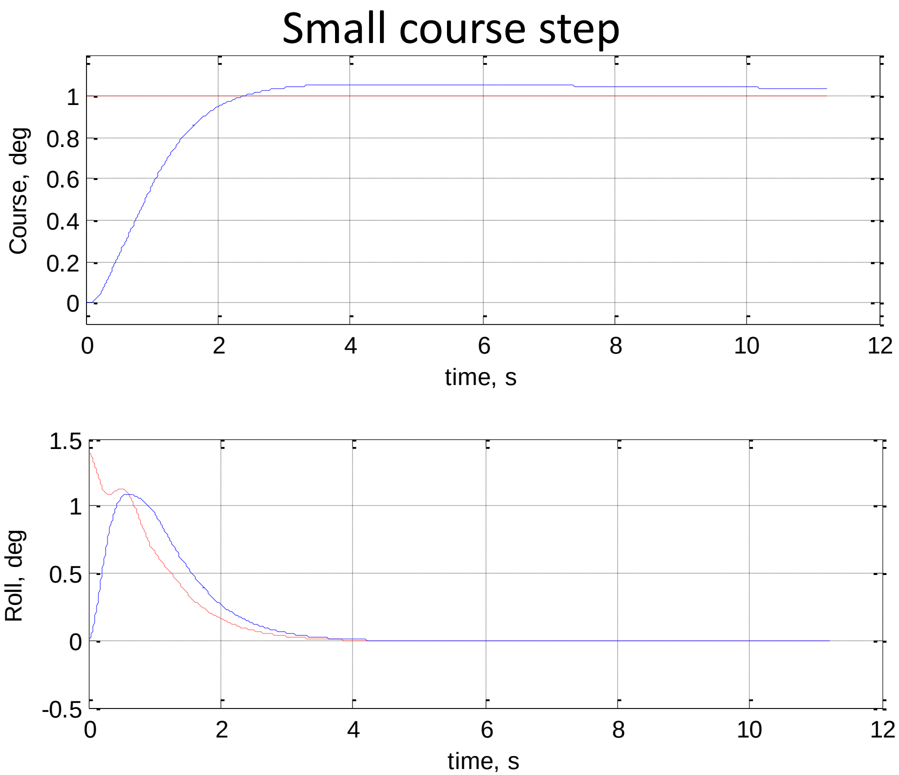
$$\chi_{cmd} - \chi = 160^\circ - (-170^\circ) = +330^\circ$$

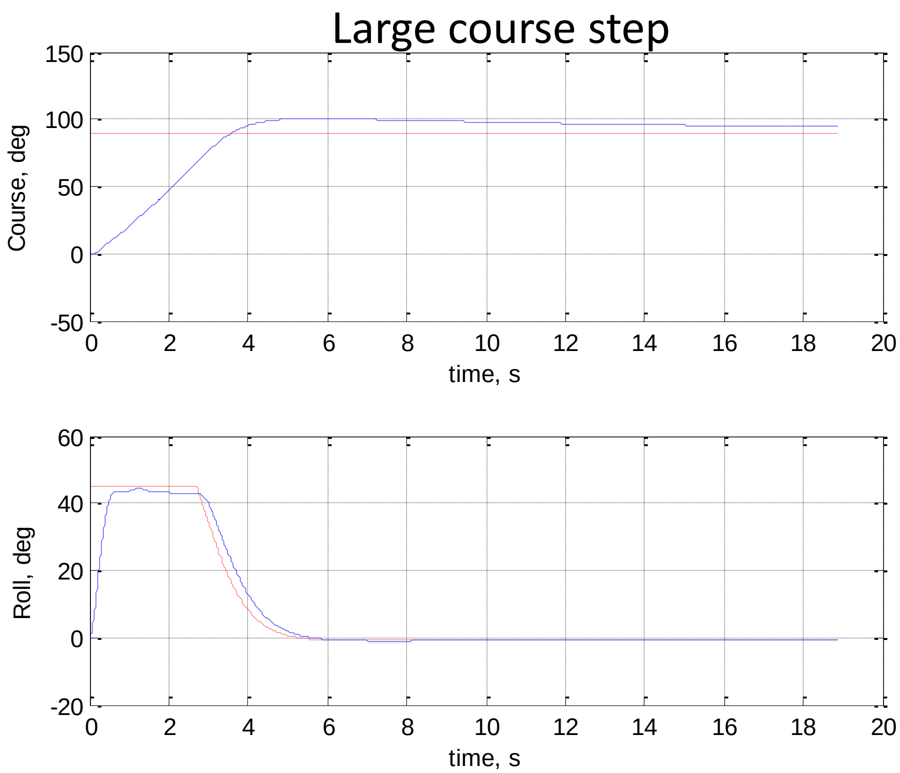
Would cause UAV to turn 330° CW instead of 30° CCW!!

Course Control Example

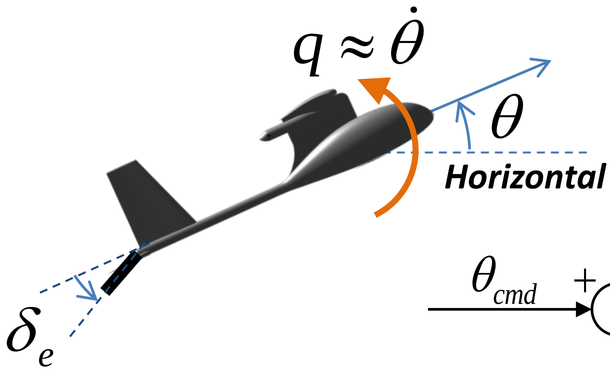


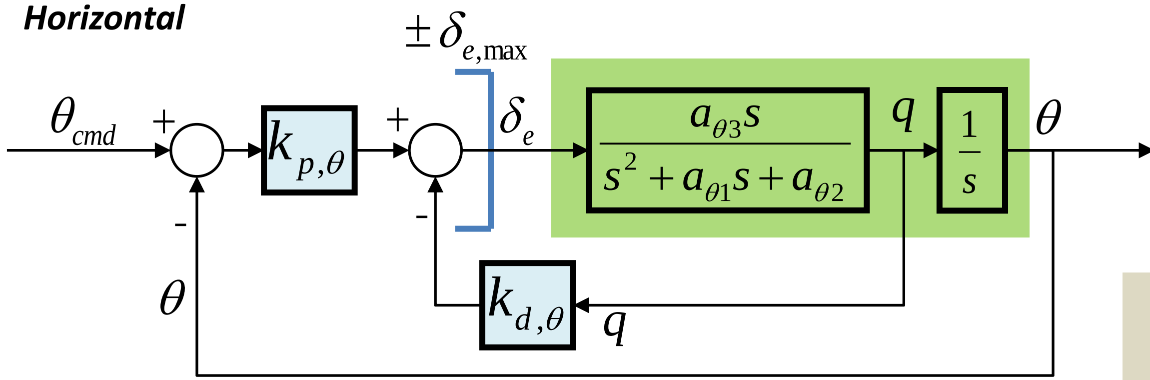






Pitch Control Autopilot





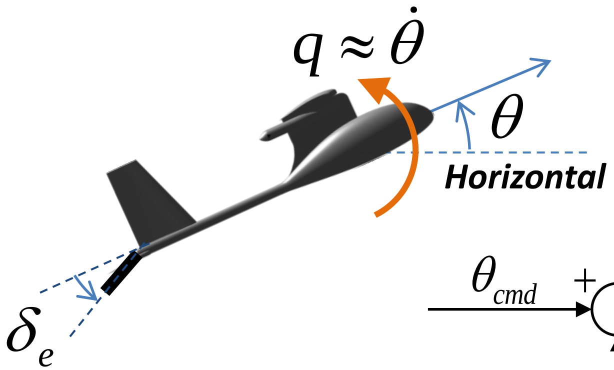
Note:
NOT unity gain

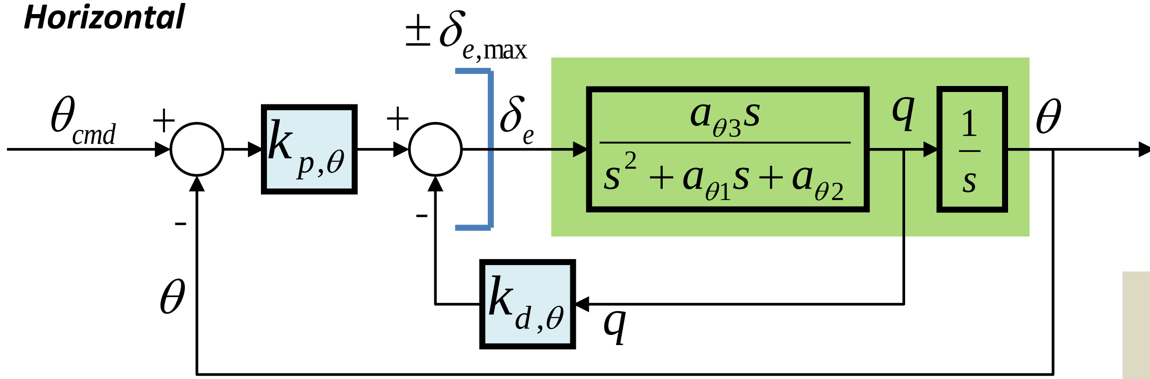
$$H_{\theta/\theta_C} = \frac{k_{p,\theta} a_{\theta 3}}{s^2 + \left(a_{\theta 1} + k_{d,\theta} a_{\theta 3}\right)s + \left(a_{\theta 2} + k_{p,\theta} a_{\theta 3}\right)} = \frac{K_{\theta,DC}\,\omega_{n,\theta}^2}{s^2 + 2\zeta_\theta \omega_{n,\theta} s + \omega_{n,\theta}^2}$$

- Equating coefficients: $2\zeta_\theta \omega_{n,\theta} = a_{\theta 1} + k_{d,\theta} a_{\theta 3}$ $\qquad \omega_{n,\theta}^2 = a_{\theta 2} + k_{p,\theta} a_{\theta 3}$
  - Provides 2 equations and 2 unknowns, but we need to account for $\delta_{e,max}$
- Select $k_{p\theta}$ such that the elevator saturates only when the pitch error exceeds some design threshold, $e_{\theta,max}$.
  - Also, positive pitch error should cause a positive q.

$$k_{p,\theta} = \frac{\delta_{e,\max}}{e_{\theta,\max}} \, sign\left(a_{\theta 3}\right)$$

Pitch Control Autopilot





Note:
NOT unity gain

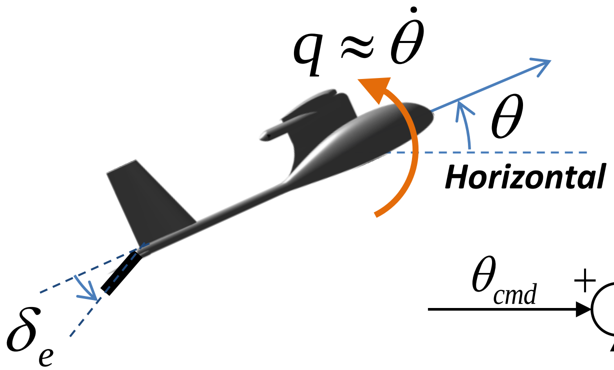
$$H_{\theta/\theta_C} = \frac{k_{p,\theta} a_{\theta 3}}{s^2 + \left(a_{\theta 1} + k_{d,\theta} a_{\theta 3}\right)s + \left(a_{\theta 2} + k_{p,\theta} a_{\theta 3}\right)} = \frac{K_{\theta,DC}\,\omega_{n,\theta}^2}{s^2 + 2\zeta_\theta \omega_{n,\theta} s + \omega_{n,\theta}^2}$$

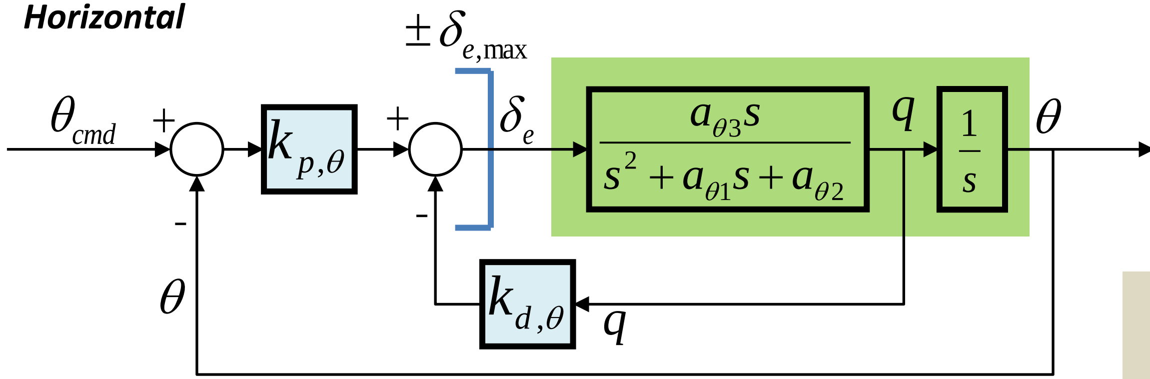
– Choose design parameters:
  - $e_{\theta,max}$: Amount of pitch error which will cause elevator saturation (smaller value means faster response, but more saturation)
  - ζ_θ : Use to choose overshoot

– Then:

$$k_{p,\theta} = \frac{\delta_{e,\max}}{e_{\theta,\max}} sign\left(a_{\theta 3}\right) \qquad \omega_{n,\theta} = \sqrt{a_{\theta 2} + k_{p,\theta} a_{\theta 3}} \qquad k_{d,\theta} = \frac{2\zeta_\theta \omega_{n,\theta} - a_{\theta 1}}{a_{\theta 3}}$$

Pitch Control Autopilot



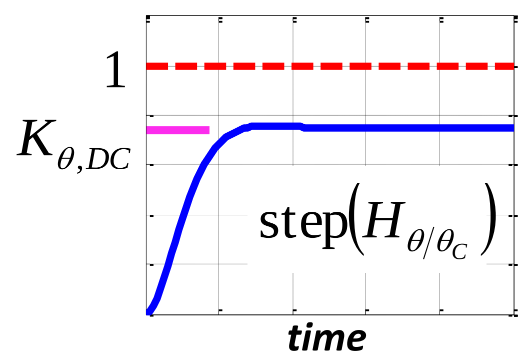


Note:
NOT unity gain

$$H_{\theta/\theta_C} = \frac{k_{p,\theta} a_{\theta 3}}{s^2 + \left(a_{\theta 1} + k_{d,\theta} a_{\theta 3}\right)s + \left(a_{\theta 2} + k_{p,\theta} a_{\theta 3}\right)} = \frac{K_{\theta,DC}\,\omega_{n,\theta}^2}{s^2 + 2\zeta_\theta \omega_{n,\theta} s + \omega_{n,\theta}^2}$$

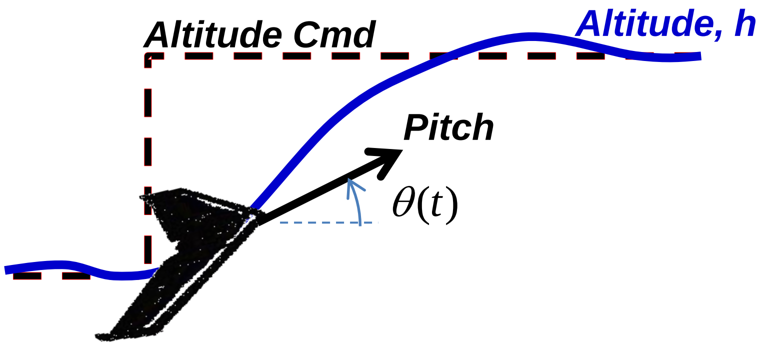
- Pitch loop has 2$^{\text{nd}}$ order response, but it does not have unity gain.
  - Pitch loop response gain:

$$K_{\theta,DC} = \frac{k_{p,\theta} a_{\theta 3}}{a_{\theta 2} + k_{p,\theta} a_{\theta 3}}$$

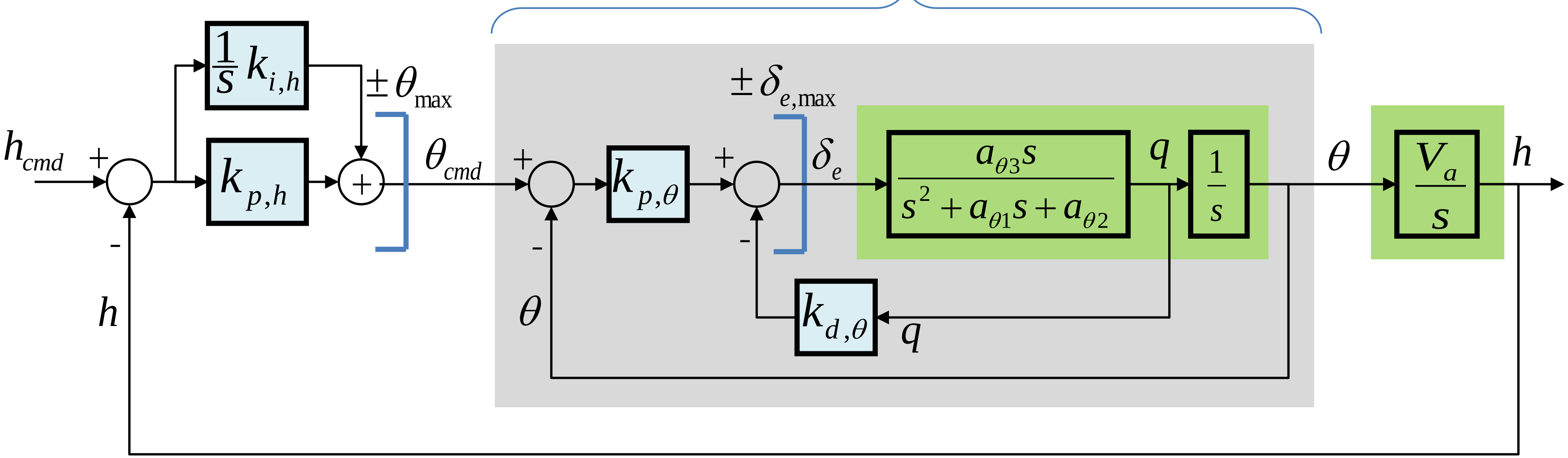


- We could eliminate this steady-state error with an integrator (e.g. $k_{i,\theta}$), but that would slow down the response. Outer loop will compensate instead.

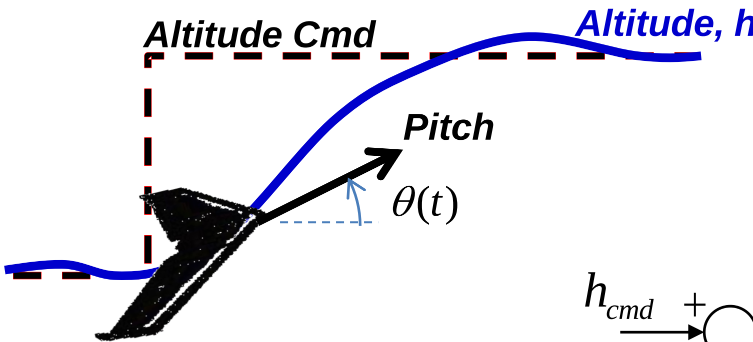
Altitude Control using Pitch



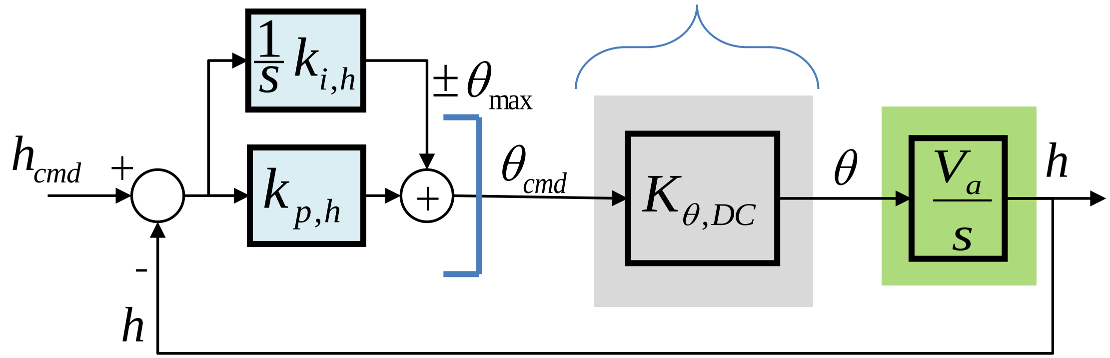
Pitch Cmd to Pitch $\approx K_{\theta,DC}$
for $\omega << \omega_{n\theta}$



Altitude Control using Pitch



Pitch Cmd to Pitch $\approx K_{\theta,DC}$
for $\omega << \omega_{n\theta}$



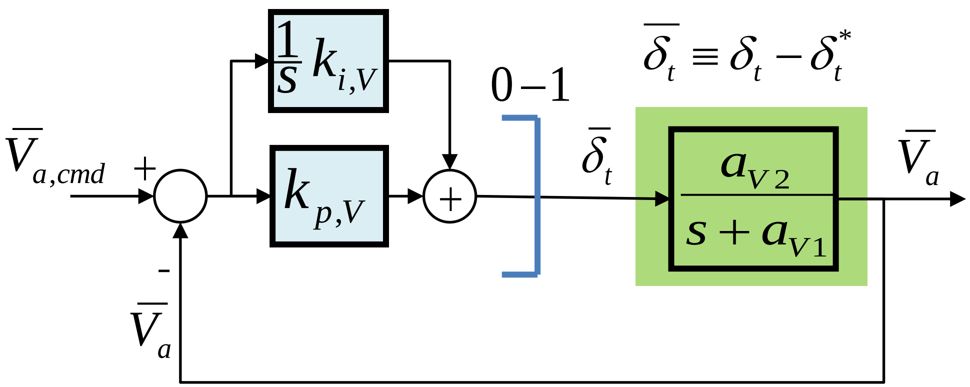
$$H_{h/h_C} = \frac{K_{\theta,DC}V_a k_{p,h}s + K_{\theta,DC}V_a k_{i,h}}{s^2 + K_{\theta,DC}V_a k_{p,h}s + K_{\theta,DC}V_a k_{i,h}} = \frac{2\zeta_h\omega_{n,h}s + \omega_{n,h}^2}{s^2 + 2\zeta_h\omega_{n,h}s + \omega_{n,h}^2}$$

- Equating coefficients: $k_{p,h} = \frac{2\zeta_h\omega_{n,h}}{K_{\theta,DC}V_a} \qquad k_{i,h} = \frac{\omega_{n,h}^2}{K_{\theta,DC}V_a}$
- Let: $\omega_{n,h} = \frac{1}{W_h}\omega_{n,\theta}$, $W_h >> 1$, e.g. 5-100
- Altitude Loop Design Parameters:
  - W_h : Bandwidth separation factor
  - ζ_h : Damping Ratio (higher due to numerator zero)

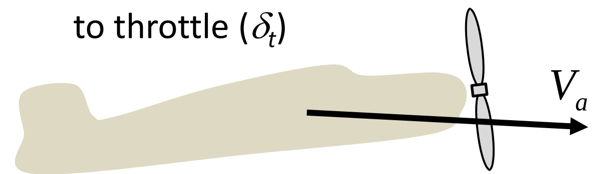
Airspeed Control using Throttle

$$\bar{V}_a \equiv V_a - V_a^*$$

$$\bar{\delta}_t \equiv \delta_t - \delta_t^*$$



Propeller speed is proportional to throttle (δ_t)



Recall: The "throttle-to-airspeed" model was derived as a "variations from trim" model. The autopilot integrator will account for the "trim" values.

$$H_{Va/Va_C} = \frac{a_{V2} k_{p,V} s + a_{V2} k_{i,V}}{s^2 + \left(a_{V1} + a_{V2} k_{p,V}\right) s + a_{V2} k_{i,V}} \approx \frac{2\zeta_V \omega_{n,V} s + \omega_{n,V}^2}{s^2 + 2\zeta_V \omega_{n,V} s + \omega_{n,V}^2}$$

Approximate, because a_{V1} is small

- Equating coefficients: $k_{p,V} = \dfrac{2\zeta_V \omega_{n,V} - a_{V1}}{a_{V2}} \qquad k_{i,V} = \dfrac{\omega_{n,V}^2}{a_{V2}}$

- Airspeed using Throttle Loop Design Parameters:

  $\omega_{n,V}$: Airspeed using throttle natural frequency

  Advice: Use bandwidth separation from pitch loop

  e.g.: $\omega_{n,V} = \frac{1}{W_V} \omega_{n,\theta}, \; W_V >> 1, \text{ e.g. } 10 - 50$

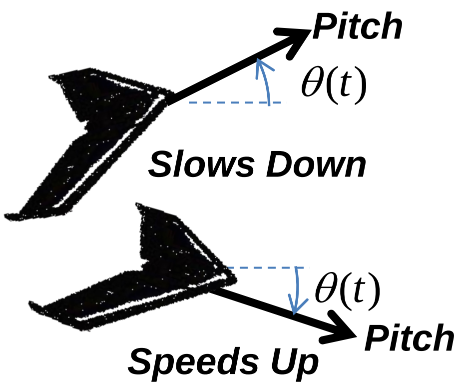
  ζ_V : Damping Ratio (higher due to numerator zero)

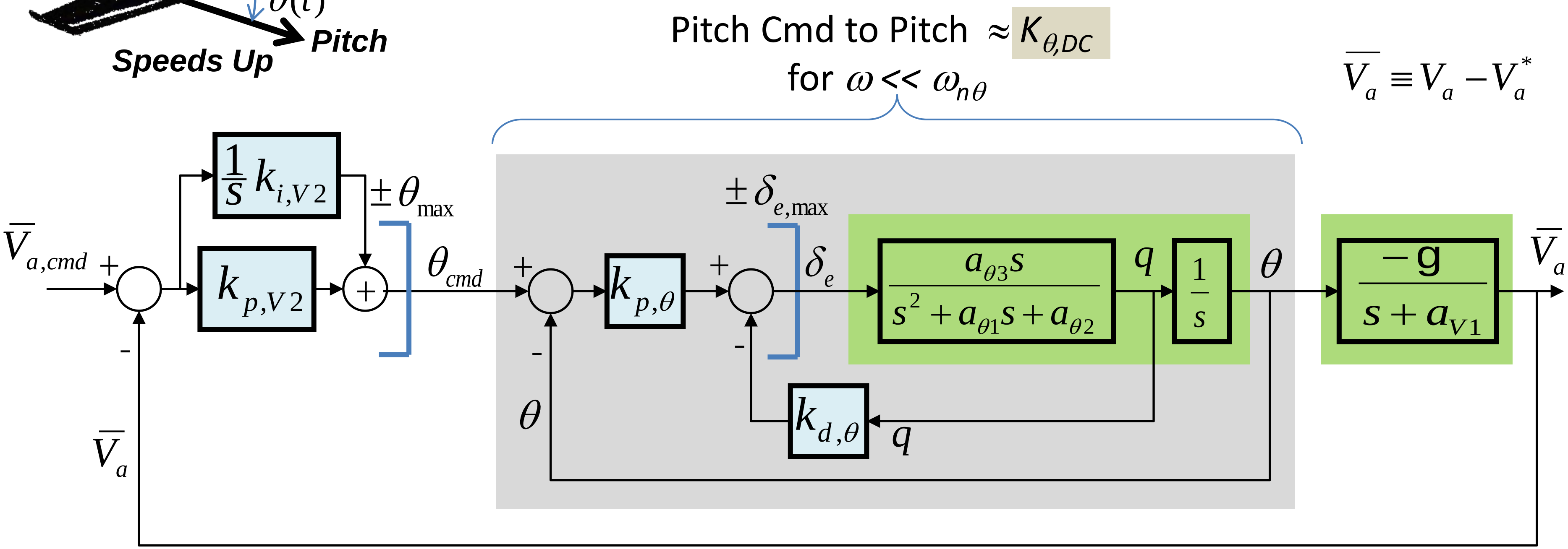
Longitudinal Control Modes

- Altitude Hold Mode:
  Pitch controls altitude, Throttle controls airspeed
  - Tight altitude and speed control for small deviations
  - Undesirable for large altitude steps
    - For climbs, results in large pitch up and potential slow-down
    - For descents, results in large pitch downs and massive speed-ups!!!
- Climb Mode:
  - *Go full throttle, and use pitch to control airspeed*
- Descent Mode:
  - *Go zero throttle, and use pitch to control airspeed*

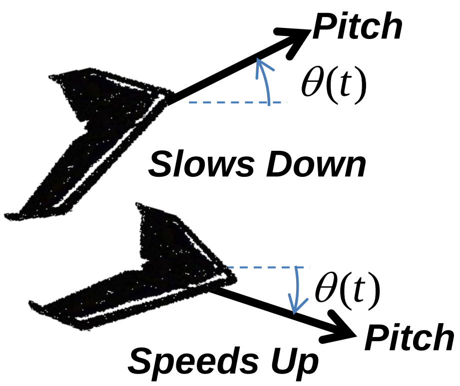
$\Rightarrow$ Need another controller:
Airspeed control using Pitch

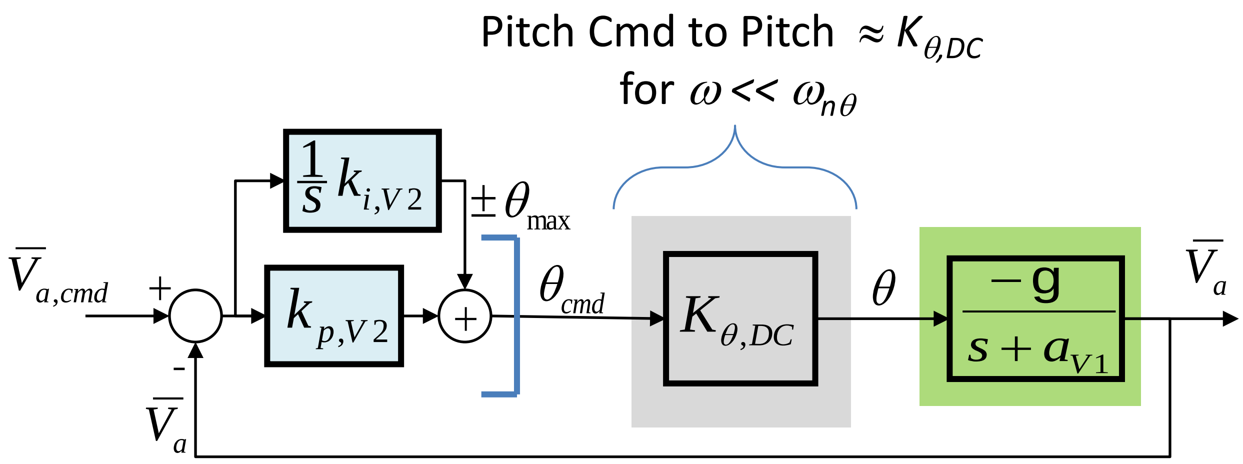
Airspeed Control using Pitch





Airspeed Control using Pitch





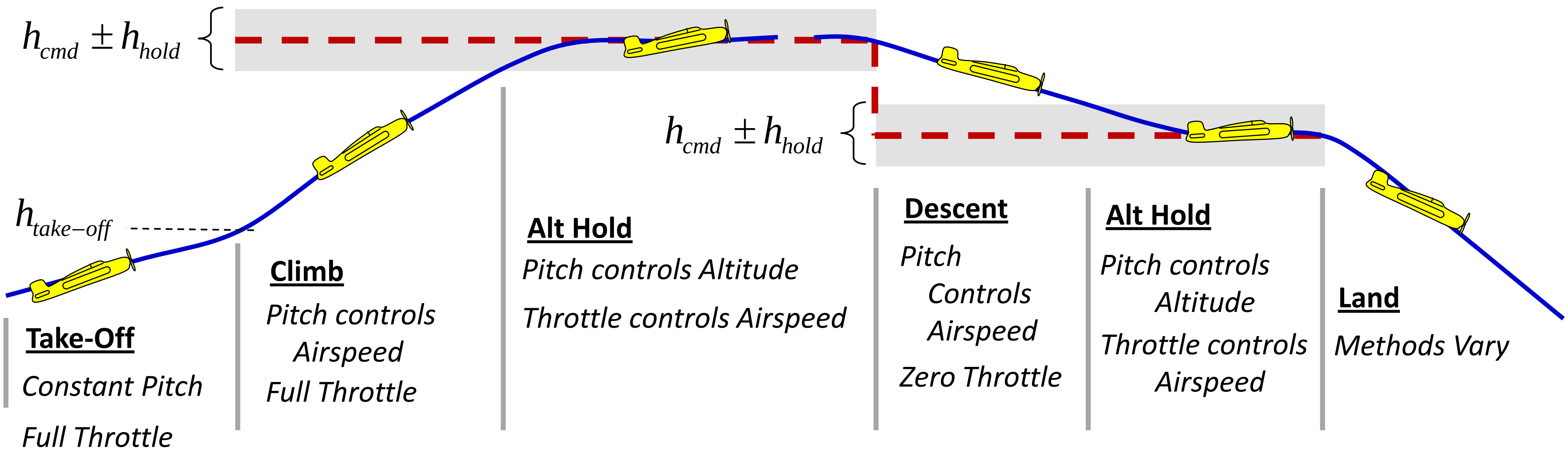
$$H_{Va/Va_C} = \frac{-K_{\theta,DC}\,g\,k_{p,V2}\,s - K_{\theta,DC}\,g\,k_{i,V2}}{s^2 + \left(a_{V1} - K_{\theta,DC}\,g\,k_{p,V2}\right)s - K_{\theta,DC}\,g\,k_{i,V2}} \approx \frac{2\zeta_{V2}\omega_{n,V2}\,s + \omega_{n,V2}^2}{s^2 + 2\zeta_{V2}\omega_{n,V2}\,s + \omega_{n,V2}^2}$$

Approximate, because a_{V1} is small

- Equating coefficients: $k_{p,V2} = \dfrac{a_{V1} - 2\zeta_{V2}\omega_{n,V2}}{K_{\theta,DC}\,g} \qquad k_{i,V2} = \dfrac{-\omega_{n,V2}^2}{K_{\theta,DC}\,g}$
- Let: $\omega_{n,V2} = \dfrac{1}{W_{V2}}\omega_{n,\theta}, \quad W_{V2} >> 1, \ \text{e.g.}\ 5\text{-}50$
- Airspeed with Pitch Loop Design Parameters:
  - W_{V2} : Bandwidth separation factor
  - ζ_{V2} : Damping Ratio (higher due to numerator zero)

Altitude & Airspeed Control Modes

A UAV will generally switch between different Longitudinal Control Modes during flight



Example Mode Logic

Take-Off: Full throttle, and regulate pitch to $\theta_{take\text{-}off}$

- Start at Launch, end upon initially reaching an altitude of $h_{take\text{-}off}$

Climb: Full throttle, and use pitch to regulate airspeed to $V_{a,cmd}$

- Whenever $h < h_{cmd} - h_{hold}$ (and not in *Take-Off* or *Land*)

Alt Hold: Pitch controls altitude to (h_{cmd}), and throttle controls airspeed ($V_{a,cmd}$)

- Whenever $|h - h_{cmd}| < h_{hold}$ (and not in *Take-Off* or *Land*)

Descend: Zero throttle, and use pitch to regulate airspeed to $V_{a,cmd}$

- Whenever $h > h_{cmd} + h_{hold}$ (and not in *Take-Off* or *Land*)

Alternative Descent:
- Zero Throttle
- $\theta_{cmd} = \theta_{descent}$

Land: Methods vary (e.g. fly down a prescribed glideslope, etc.). Initiated by command.

Longitudinal Autopilot - Summary

If model is known, the the design parameters are

Inner Loop (pitch attitude hold)

- $e_\theta^{\max}$ - Error in pitch when elevator just saturates.
- ζ_θ - Damping ratio for pitch attitude loop.

Altitude Hold Outer Loop

- $W_h > 1$ - Bandwidth separation between pitch and altitude loops.
- ζ_h - Damping ratio for altitude hold loop.

Airspeed Hold Outer Loop

- $W_{V_2} > 1$ - Bandwidth separation between pitch and airspeed loops.
- ζ_{V_2} - Damping ratio for airspeed hold loop.

Throttle hold (inner loop)

- ω_{n_V} - Natural frequency for throttle loop.
- ζ_V - Damping ratio for throttle loop.

Lateral Autopilot - Summary

If model is known, the the design parameters are

Inner Loop (roll attitude hold)

- $e_\phi^{\max}$ - Error in roll when aileron just saturates.
- ζ_ϕ - Damping ratio for roll attitude loop.

Outer Loop (course hold)

- $W_\chi > 1$ - Bandwidth separation between roll and course loops.
- ζ_χ - Damping ratio for course hold loop.

General Gain Tuning Procedures

Decide which knobs will you turn

Gains
- e.g. k_p, k_i, k_d
- Pros:
  - Easy to do
- Cons:
  - Not always intuitive
  - May be unstable

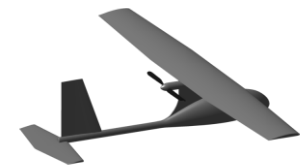
Design Parameters
- e.g. e^{max}, ζ, W
- Pros:
  - More intuitive result
  - Stable for reasonable parameter choices
- Cons:
  - Additional math & effort involved

Decide goodness criteria

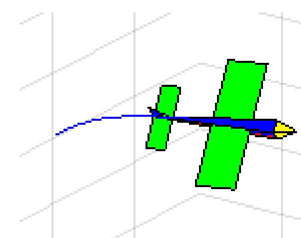
Examples
- Rise time
- Settling time
- Overshoot
- Avoid oscillations
- Stability margins
- Avoiding limits
- etc.

Decide how you will test response

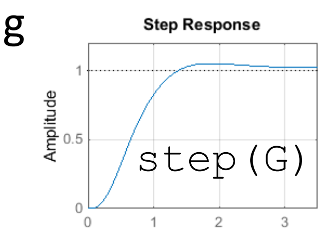
In-flight tuning
- Pros:
  - Less math
- Cons:
  - Requires expertise
  - May be disastrous



In-Simulation tuning
- Pros:
  - Not disastrous
- Cons:
  - Requires a sim
  - Responses may be coupled



Transfer Function tuning
- Pros:
  - Quick
- Cons:
  - Still need to simulate/fly



Useful Transfer Functions for Tuning

Specifically using methods described in this class:

```
%% Lateral Channel

% Open Loop response from aileron to roll
   H(kphi,kda) % Higher fidelity than models.G_da2phi
% Closed Loop response from roll command to roll
   G_phic2phi = PI_rateFeedback_TF(H(kphi,kda),P.roll_kp,P.roll_ki,P.roll_kd)
% Closed Loop response from course command to course
   G_chic2chi = PI_rateFeedback_TF(G_phic2phi*models.G_phi2chi, ...
                          P.course_kp,P.course_ki,P.course_kd)

%% Longitudinal Channel

% Open Loop response from elevator to pitch
   H(ktheta,kde) % Higher fidelity than models.G_de2theta
% Closed Loop response from pitch command to pitch
   G_thetac2theta = PI_rateFeedback_TF(H(ktheta,kde),P.pitch_kp,P.pitch_ki,P.pitch_kd)
% Closed Loop response from alt command to alt
   G_altc2alt = PI_rateFeedback_TF(G_thetac2theta*models.G_theta2h, ...
                           P.altitude_kp,P.altitude_ki,P.altitude_kd)
% Closed Loop response from airspeed commmand to airspeed using pitch
   G_vac2va_pitch = PI_rateFeedback_TF(G_thetac2theta*models.G_theta2Va, ...
                           P.airspeed_pitch_kp,P.airspeed_pitch_ki,P.airspeed_pitch_kd)

% Open Loop response from throttle to airspeed
   models.G_dt2Va  % Note: Can't use H(ku,kdt) because it is dominated by phugoid
% Closed Loop response from airspeed commmand to airspeed using throttle
   G_vac2va_throttle = PI_rateFeedback_TF(models.G_dt2Va, ...
                           P.airspeed_throttle_kp,P.airspeed_throttle_ki,P.airspeed_throttle_kd)
```

Lecture 8 Homework, 1/5

Note: re-run `compute_autopilot_gains` (or `load_uavsim`) each time you modify the gains!

1) **Use design parameters to tune and simulate course control.**
   - **Modify `compute_autopilot_gains.m` to generate the course control gains.**
   - **Use design parameters and transfer functions (G_chic2chi) to achieve design goals:**
     - **<5% overshoot and a 95% rise time of under 3 seconds**
     - **Minimal oscillations**
   - **In `uavsim_control.m`, generate a chi_hat (in radians) using Vn_hat and Ve_hat.**
   - **Augment `uavsim_control.m` to have a PIR_course_hold() routine, and use it to generate the phi_c used in the existing roll controller. Use appropriate limits on the routine output (i.e. P.phi_max). NOTE: Make sure to "wrap" the course error to +/- 180deg!)**
   - **Run simulation to achieve a course command of 140 degrees with no winds.**

   a. **Show derivation of closed-loop course-control transfer function and gains.**
   b. **Turn in code that generates gains, highlighting chosen design parameters.**
   c. **Turn in step response of G_chic2chi**
   d. **Turn in the portion of PIR_course_hold() sufficient to show PIR "set up" and course error "wrapping".**
   e. **Run sim for 10 seconds and turn in plots:**
      Comparing course with course command
      Comparing roll with roll command
      Showing aileron
   f. **Test error wrapping by verifying that a course command of 270deg results in a UAV turning left to fly due west. (i.e. same result as a command of -90deg)**

Enable plotting of commands by modifying uavsim_display: `plot_commands = 1;`

Enable logging of commands by modifying uavsim_logging: `log_commands = 1;`

Lecture 8 Homework, 2/5

2) Use design parameters to tune and simulate pitch control. (Remove manual tuning gains!)
- **Modify `compute_autopilot_gains.m` and `uavsim_control.m` as necessary.**
- **For design parameters, use $e_{\theta,max}$=30deg and ζ_θ=0.9.**
- **Run simulation to achieve a pitch command of 20 degrees with no winds and χ_{cmd}=0.**
- **Note: Without an integrator, achieved pitch will have steady-state error.**

a. Show derivation of closed loop pitch control transfer function and gains.

b. Turn in code that generates gains.

c. Turn in resulting gains, including P.K_theta_DC.

d. Run for 10 seconds and turn in plots:
   1) Comparing θ & θ_{cmd}, and 2) showing elevator

e. Verify that the steady-state pitch value is *approximately* θ_{cmd}*$K_{\theta,,DC}$.

Note: $K_{\theta,,DC}$ was derived from the linear model, so the resulting steady-state pitch may still differ from $\theta_{cmd}*K_{\theta,,DC}$ by a degree or so.

3) Use design parameters to tune altitude control using pitch. (Remove manual gains!)
- **Modify `compute_autopilot_gains.m` and `uavsim_control.m` as necessary.**
- **Modify** PIR_alt_hold_using_pitch() **to account for the known steady-state pitch error:**
  u_lower_limit = -P.theta_max/P.K_theta_DC;
  u_upper_limit = +P.theta_max/P.K_theta_DC;
- **Simulate 50 seconds of no-wind flight with χ_{cmd}=0 and altitude command from:**
  `if mod(time,20)<10, h_c=50; else, h_c=51; end`
- **Adjust "altitude via pitch" control design parameters to achieve "good" response.**

a. Show derivation of closed loop altitude control transfer function and gains

b. Turn in code that generates gains, highlighting chosen design parameters.

c. Turn in resulting gains.

d. Run for 50 seconds and turn in plots:
   1) Comparing altitude with altitude command
   2) Comparing θ & θ_{cmd}, and 3) showing elevator

Student's choice what constitutes a "good" response

Lecture 8 Homework, 3/5

4) Use design parameters to tune airspeed control using throttle.

- **Modify `compute_autopilot_gains.m` and `uavsim_control.m` as necessary.**
- **Create** PIR_airspeed_hold_using_throttle() **and limit throttle output (u) to [0 1].**
- **Simulate 50 seconds of no-wind flight with χ_{cmd}=0, h_cmd=50, and airspeed cmd from:**
  `if mod(time,20)<10, Va_c=13; else, Va_c=16; end`
- **Use the following design parameters for "airspeed via throttle":**
  - $\omega_{n,V} = \omega_{n,\theta} / 40$; **and** $\zeta_V = 1$.

a. **Show derivation of closed loop speed control using throttle transfer function and gains**

b. **Turn in code that generates gains.**

c. **Turn in resulting gains.**

d. **Run for 50 seconds and turn in plots:**
 Comparing airspeed with airspeed command
 Showing throttle

e. **Re-run simulation for 100 seconds with following commands (large altitude steps):**

```
Va_c = 13;
chi_c = 0;
if time<30, h_c=100; else, h_c=50; end; % 100m for time<30, then 50m
```

Is the result desirable? If not, why not?

Lecture 8 Homework, 4/5

5) **Develop airspeed-using-pitch controller, and implement altitude mode control logic.**

- **Create** `PIR_airspeed_hold_using_pitch.` **Limit pitch output to:** ±P.theta_max/P.K_theta_DC.
- **Modify** `uavsim_control.m` **to achieve the provided over-simplified altitude mode logic (We won't worry about launch and land modes):**
- **Use the following design parameters for "airspeed via pitch":** $\omega_{n,V2} = \omega_{n,\theta} / 40$; **and** $\zeta_{V2} = 1$.

a. **Show derivations (transfer function and gains)**

b. **Turn in code that generates gains, and gain values.**

c. **Turn in altitude mode code.**

d. **Run simulation for 100 seconds with following commands (large alt. steps):**

```
Va_c = 13;
chi_c = 0;
if time<30, h_c=100;
else, h_c=50; end;
```

Plot commands (where applicable) and responses for altitude, airspeed, pitch, and throttle. Highlight modes and transitions.

e. **(See question at right…)**

```
if(firstTime)
   % Initialize integrators
   PIR_pitch_hold(0,0,0,firstTime,P);
   PIR_alt_hold_using_pitch(0,0,0,firstTime,P);
   PIR_airspeed_hold_using_throttle(0,0,0,firstTime,P);
   PIR_airspeed_hold_using_pitch(0,0,0,firstTime,P);
end
h_hold = 5; % m, alt threshold
if h_hat < h_c - h_hold
   % Climb Logic
elseif h_hat > h_c + h_hold
   % Descend Logic
else
   % Altitude Hold Logic
end
```

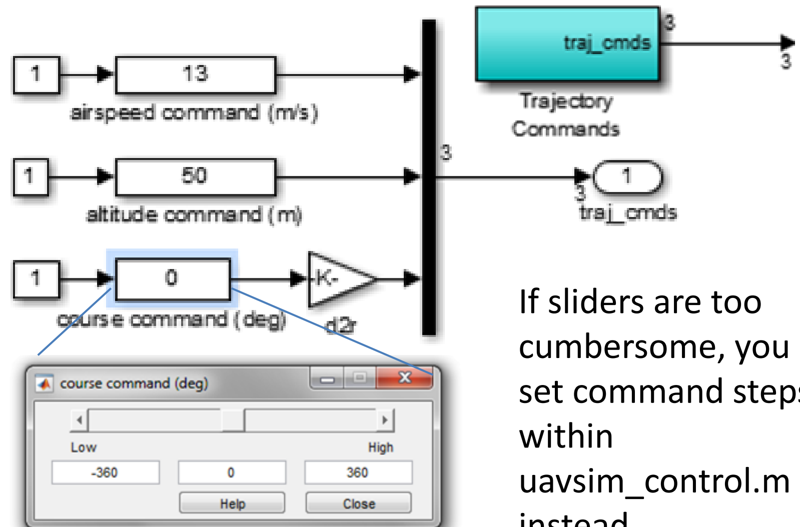
Each should generate throttle and elevator commands

e. **Did the longitudinal performance improve with the altitude moding logic? What would you do to improve it further? (You don't need to implement anything more, just discuss.)**

Lecture 8 Homework, 5/5

6) Demonstrate complete flight control in a gusting environment.

- **Turn on gusting winds (steady winds can remain zero)**
- **Remove any h_c, Va_c, and chi_c overrides in `uavsim_control.m`.**
- **Run simulation for at least 100 seconds and manually perform multiple large step commands in airspeed, altitude and course using the sliders in the "Trajectory Commands" block.**
- **Plot the following longitudinal variables:**
  - **Altitude command and response**
  - **Airspeed command and response**
  - **Pitch command and response**
  - **Elevator**
  - **Throttle**
- **Plot the following lateral variables:**
  - **Course command and response**
  - **Roll command and response**
  - **Aileron**
- **Hopefully, your existing controllers will be robust enough for these commands and disturbances. If not, re-tune using the design parameters until they are.**



If sliders are too cumbersome, you may set command steps within uavsim_control.m instead.

Recommended Reading: Beard & McLain Chapter 6	**Errata:** -	Extra "-" signs in Figures 6.6, 6.7, 6.8, 6.9
	-	Equation on p101: Numerator should be $a_{\phi 2} k_{p,\phi}$
	-	P116 & 117, line number references incorrect (but we're not using the book's exact algorithm anyway)