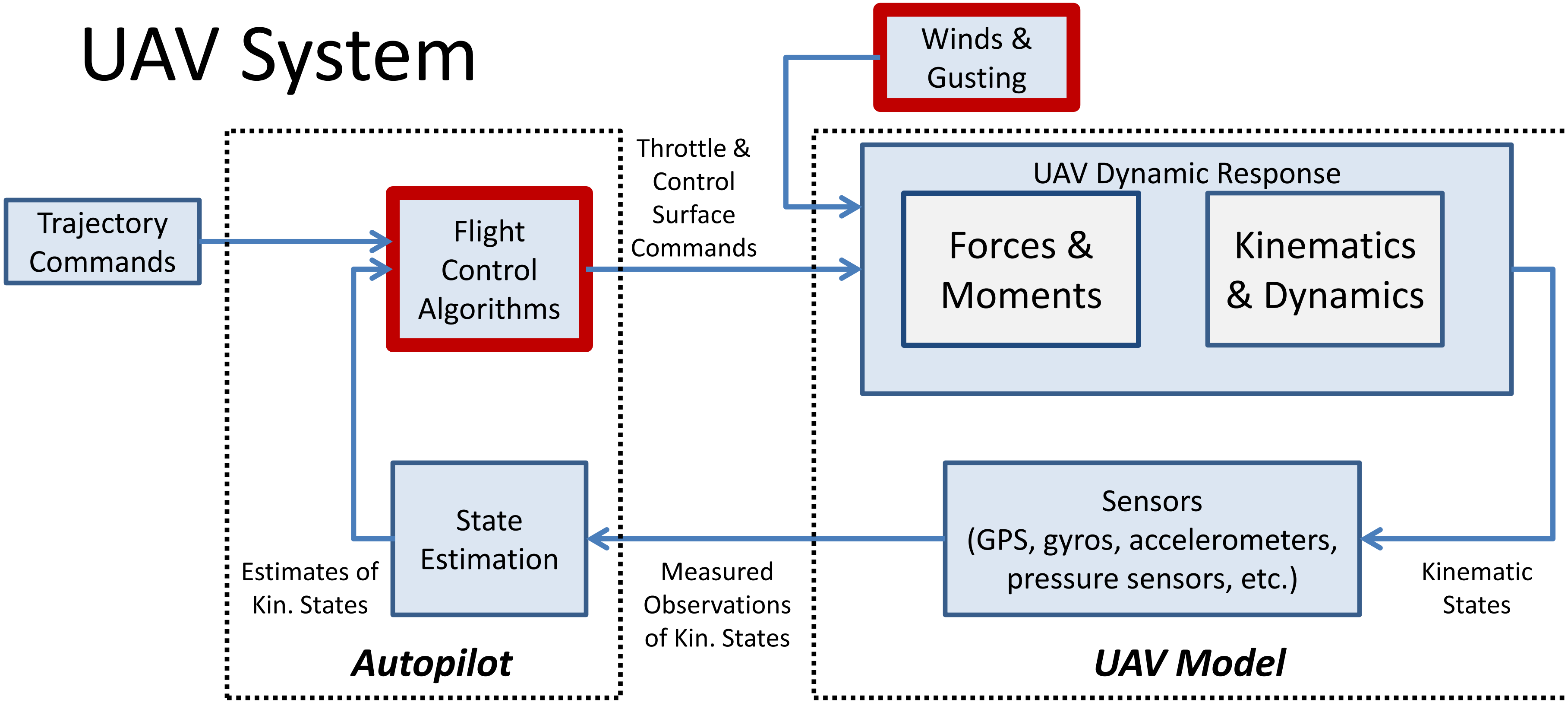
UAV Systems & Control
Lecture 6

Winds & Gusting

Autopilot Control Structures

Manual Autopilot Tuning

UAV System

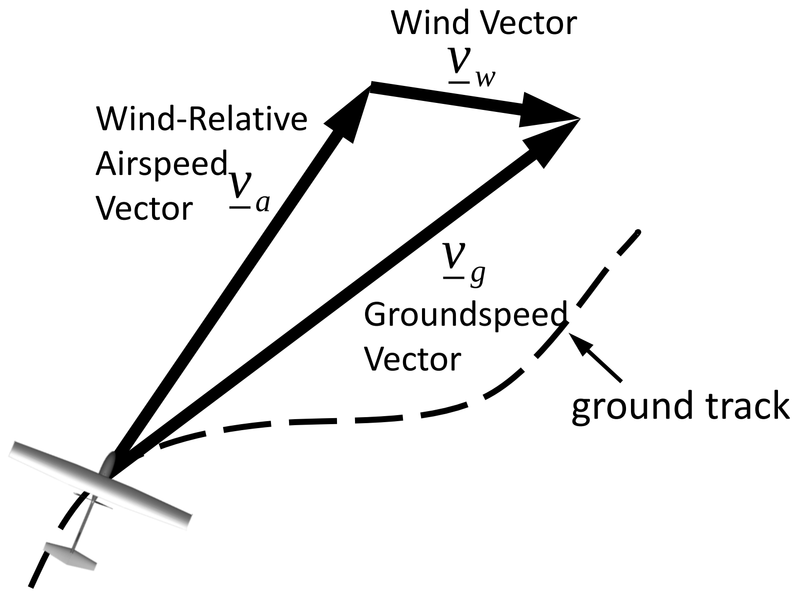


- In the previous lectures:
  - We developed the equations of motion: a complicated set of 12 nonlinear, coupled, differential equations
- In this section we will discuss:
  - Atmospheric disturbances (winds and gusting)
  - Autopilot control structures
  - Manual autopilot tuning

Wind Model

Recall the wind triangle:

$$\underline{v}_g = \underline{v}_a + \underline{v}_w$$



The wind vector can be decomposed into steady-state and gust components:

$$\underline{v}_w = \underline{v}_{ws} + \underline{v}_{wg}$$

Steady wind *Time-varying "gusting"*

The steady, ambient wind is typically expressed in NED components:

$$\underline{v}_{ws}^{ned} = \begin{bmatrix} w_{ns} \\ w_{es} \\ w_{ds} \end{bmatrix} \ {}^{m}\!/_{s}$$

The gusting component is a time-varying stochastic process which will be derived in body frame:

$$\underline{v}_{wg}^{b} = \begin{bmatrix} u_{wg} \\ v_{wg} \\ w_{wg} \end{bmatrix} \ {}^{m}\!/_{s}$$

Gusting (aka Turbulence) Model

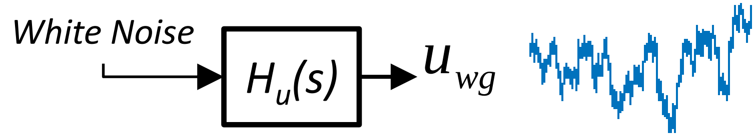
$$\underline{v}_{wg}^{b} = \begin{bmatrix} u_{wg} \\ v_{wg} \\ w_{wg} \end{bmatrix} \; m/s$$

We need a mathematical model emulating time-varying gusts

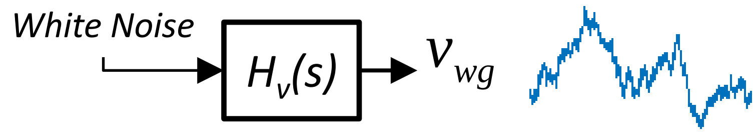
- Common model: <u>Dryden</u>

  - Pass white noise through a linear filter
  - Generated in body coordinates to model higher frequency effects in the forward direction

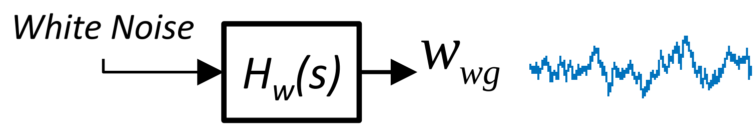
$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{L_u}} \frac{1}{s + \frac{V_a}{L_u}}$$



$$H_v(s) = \sigma_v \sqrt{\frac{3V_a}{L_v}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_v}\right)}{\left(s + \frac{V_a}{L_v}\right)^2}$$



$$H_w(s) = \sigma_w \sqrt{\frac{3V_a}{L_w}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_w}\right)}{\left(s + \frac{V_a}{L_w}\right)^2}$$



L_u, L_v, L_w : Spatial Wavelengths (m)

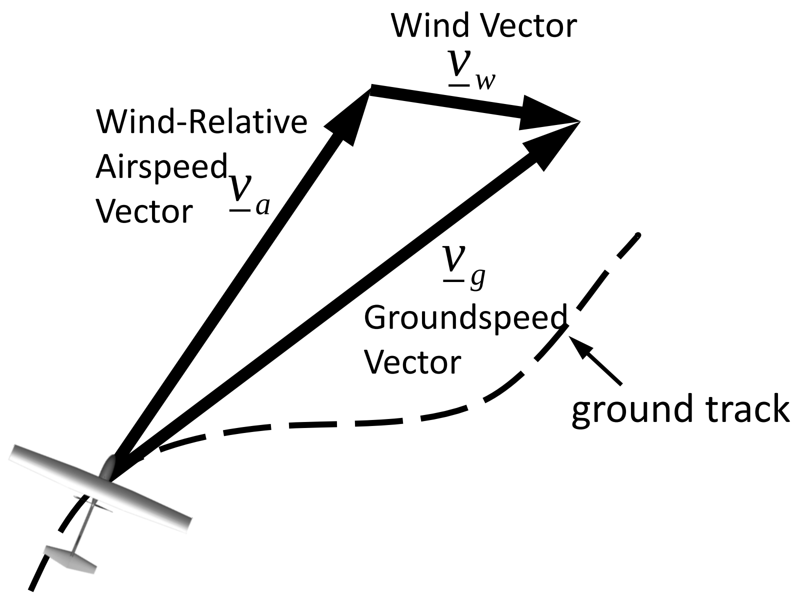
$\sigma_u, \sigma_v, \sigma_w$: Turbulence Intensities (m/s)

Dryden Gust Parameters

gust description	altitude (m)	$L_u = L_v$ (m)	L_w (m)	$\sigma_u = \sigma_v$ (m/s)	σ_w (m/s)
low altitude, light turbulence	50	200	50	1.06	0.7
low altitude, moderate turbulence	50	200	50	2.12	1.4
medium altitude, light turbulence	600	533	533	1.5	1.5
medium altitude, moderate turbulence	600	533	533	3.0	3.0

We'll use

Wind Model



The wind vector can be decomposed into steady-state and gust components:

$$\underline{v}_w = \underline{v}_{ws} + \underline{v}_{wg}$$

Steady wind is provided in NED coordinates:

$$\underline{v}_{ws}^{ned} = \begin{bmatrix} w_{ns} \\ w_{es} \\ w_{ds} \end{bmatrix} \ {}^{m}\!/_{s}$$

Gusting is generated in body coordinates by the Dryden model

$$\underline{v}_{wg}^{b} = \begin{bmatrix} u_{wg} \\ v_{wg} \\ w_{wg} \end{bmatrix} \ {}^{m}\!/_{s}$$

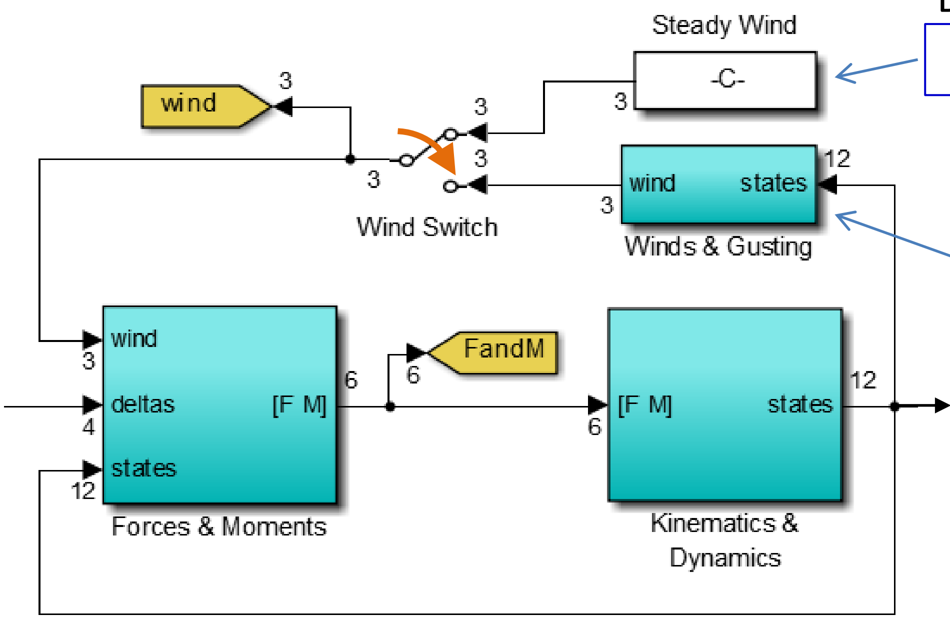
Resolve total wind vector in NED coordinates:

$$\underline{v}_w^{ned} = \underline{v}_{ws}^{ned} + R_b^{ned}\,\underline{v}_{wg}^{b} \ {}^{m}\!/_{s}$$

$$where: \quad R_b^{ned} = \left[R_{ned}^{b}\right]^T$$

Note: Book resolves total wind vector into body coordinates. Our simulation needs it resolved in NED coordinates.

Gusting Implementation in UAVSIM

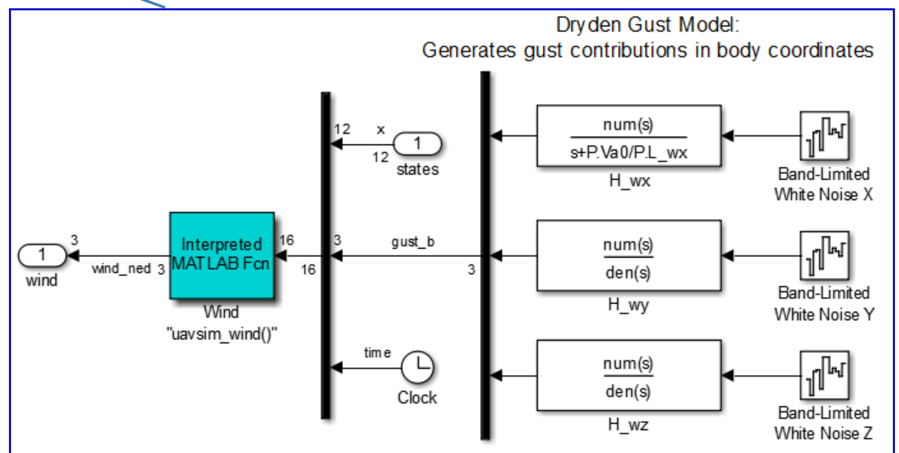


Disturb UAV with constant, steady wind

`[P.wind_n P.wind_e P.wind_d]`

Parameters from input file

Disturb UAV with both gusting and constant, steady wind



$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{L_u}} \frac{1}{s + \frac{V_a}{L_u}}$$

$$H_v(s) = \sigma_v \sqrt{\frac{3V_a}{L_v}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_v}\right)}{\left(s + \frac{V_a}{L_v}\right)^2}$$

$$H_w(s) = \sigma_w \sqrt{\frac{3V_a}{L_w}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_w}\right)}{\left(s + \frac{V_a}{L_w}\right)^2}$$

Parameters from input file

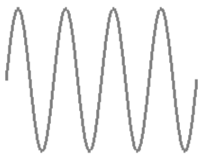
- Dryden Gusting is provided using Laplace Transfer Function blocks and white noise generators
  - No modification necessary
- “uavsim_wind” needs to be modified to combine gusting with steady winds
- Switch provided to choose:
  - steady winds
  - steady + gusting winds

Note: Dryden model (as-is) isn’t appropriate for quadrotor flight

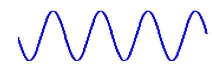
Frequency Response of a System

A property of a linear system (e.g. a Laplace transfer function or a states space model) is:

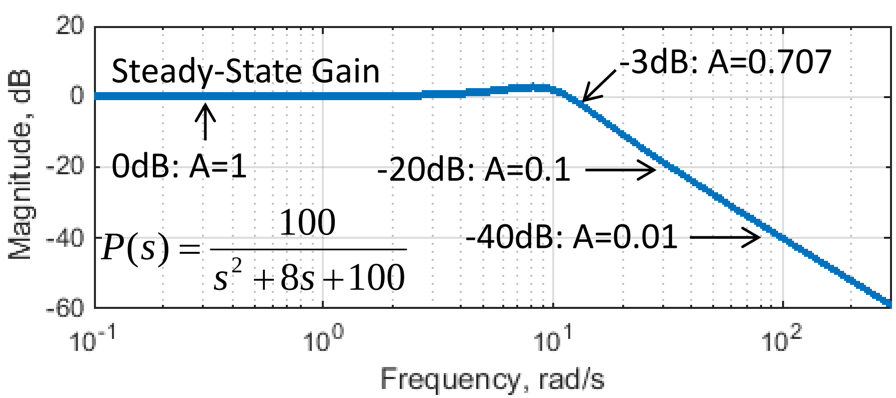
- A sinusoidal input yields a sinusoidal output at the same frequency (but with possible amplification/attenuation and phase shift)



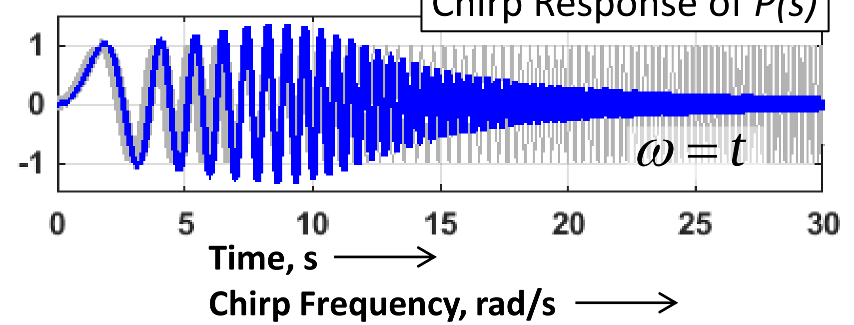
$$\sin(\omega t) \longrightarrow \boxed{P(s)} \longrightarrow A\sin(\omega t + \phi)$$



- Thus, we can also view systems in a frequency domain:

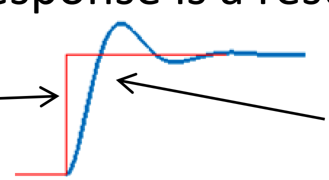


For the example system shown, frequencies above 10 rad/s are attenuated



- The time-domain step response is a result of high frequency attenuation:

Perfect step would require infinitely many frequencies

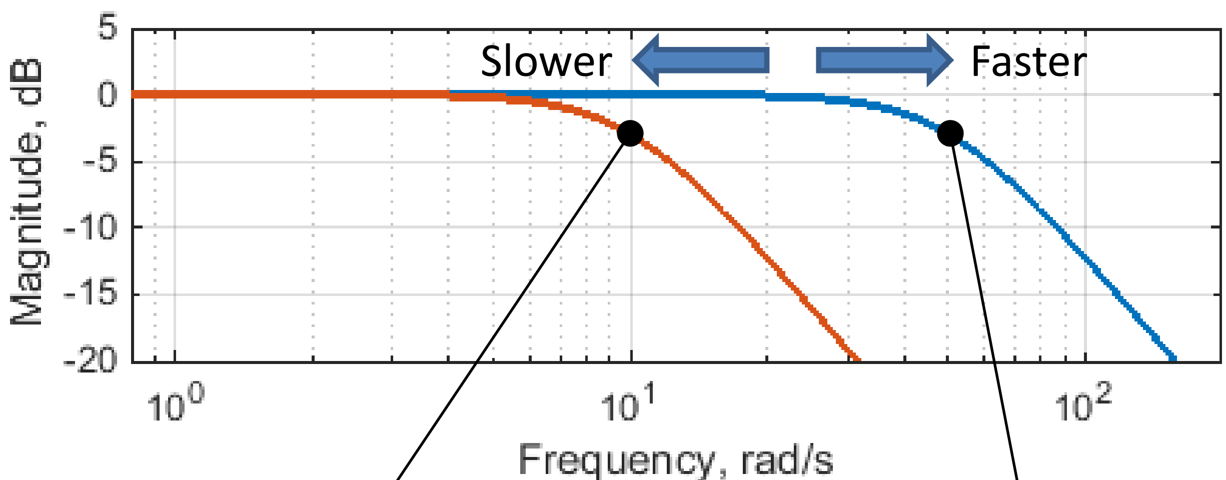


Step response is the result after higher frequencies attenuated

Bandwidth of a System

<u>Bandwidth:</u> The frequency beyond which the system attenuates an input by 3dB or more from its steady-state gain. (-3dB = 0.707)

- The higher the bandwidth, the faster the response



1st Order System: $\frac{b}{s+a}$

$$\omega_{BW} \cong a$$

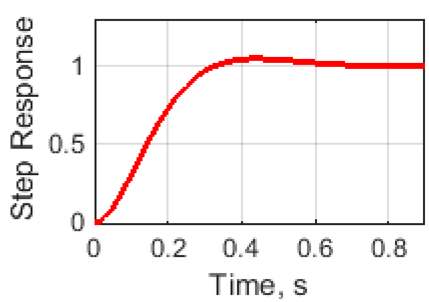
2nd Order System: $\frac{k_{dc}\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$

$$\omega_{BW} \cong \omega_n$$

$$for:\ 0.5<\zeta<1.1$$

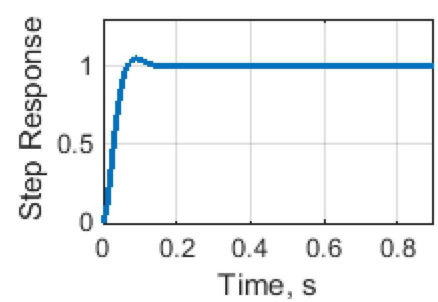
Bandwidth: 10 rad/s
Peak Time: 0.44 s

Bandwidth: 50 rad/s
Peak Time: 0.09 s



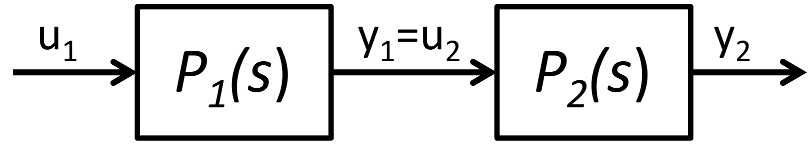
$\omega_{BW,slow} < \omega_{BW,fast}$

$T_{peak,slow} > T_{peak,fast}$



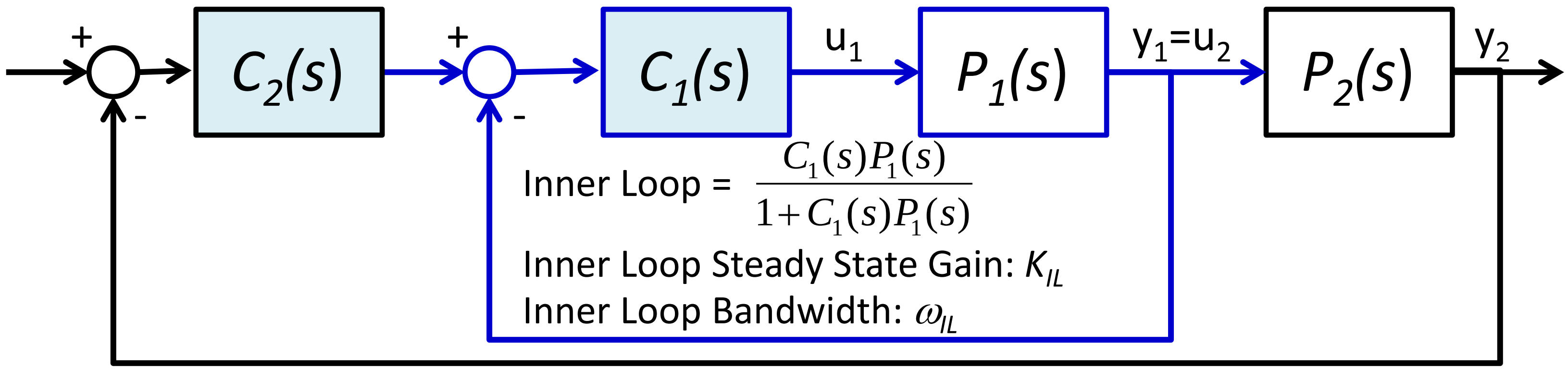
Feedback Control: Successive Loop Closure

Consider a plant model consisting of two cascaded systems, $P_1(s)$ & $P_2(s)$:

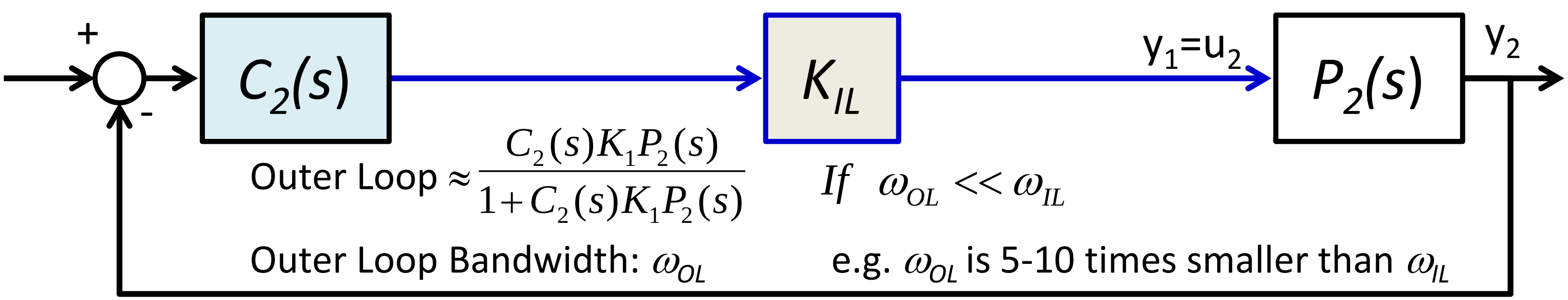


Let $C_1(s)$ be a feedback controller around $P_1(s)$

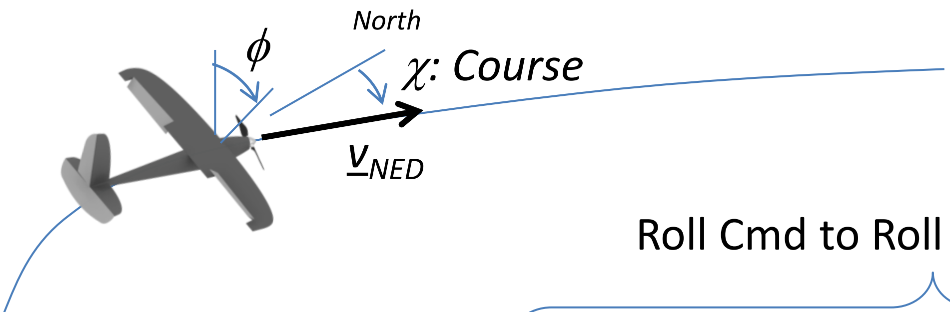
Let $C_2(s)$ be a feedback controller around entire system



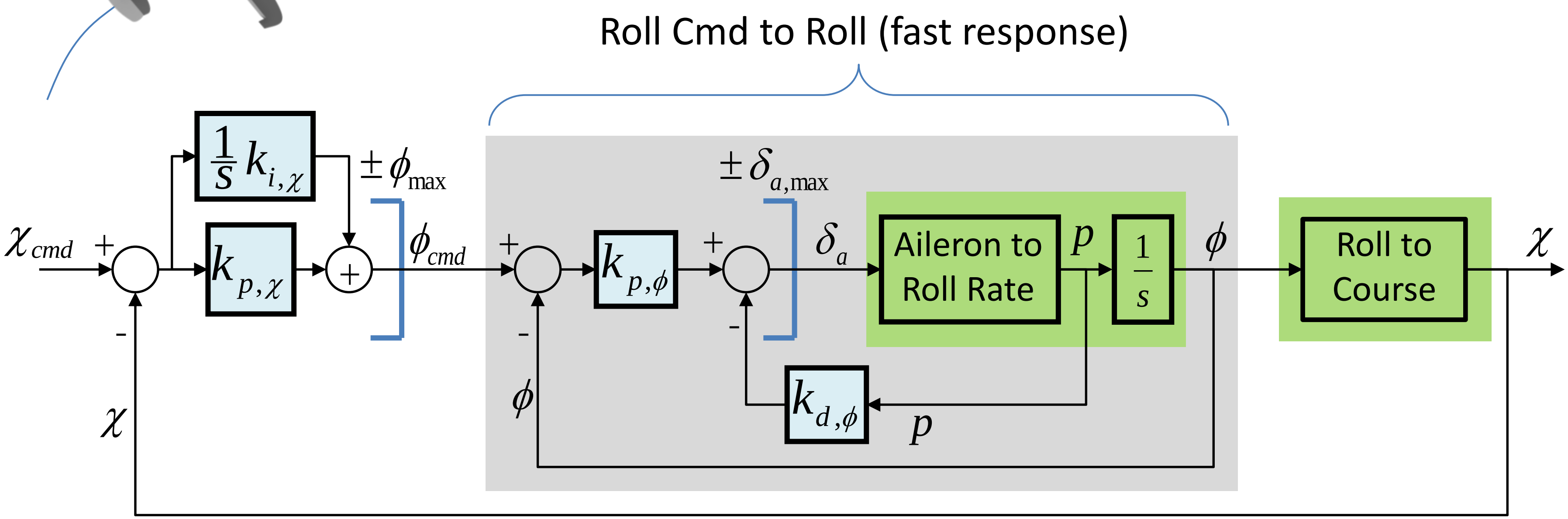
Inner loop can be treated merely as a constant gain (K_{IL}) if outer loop has a bandwidth significantly smaller than ω_{IL} (i.e. outer loop is a lot slower than inner loop)



Course Control Autopilot



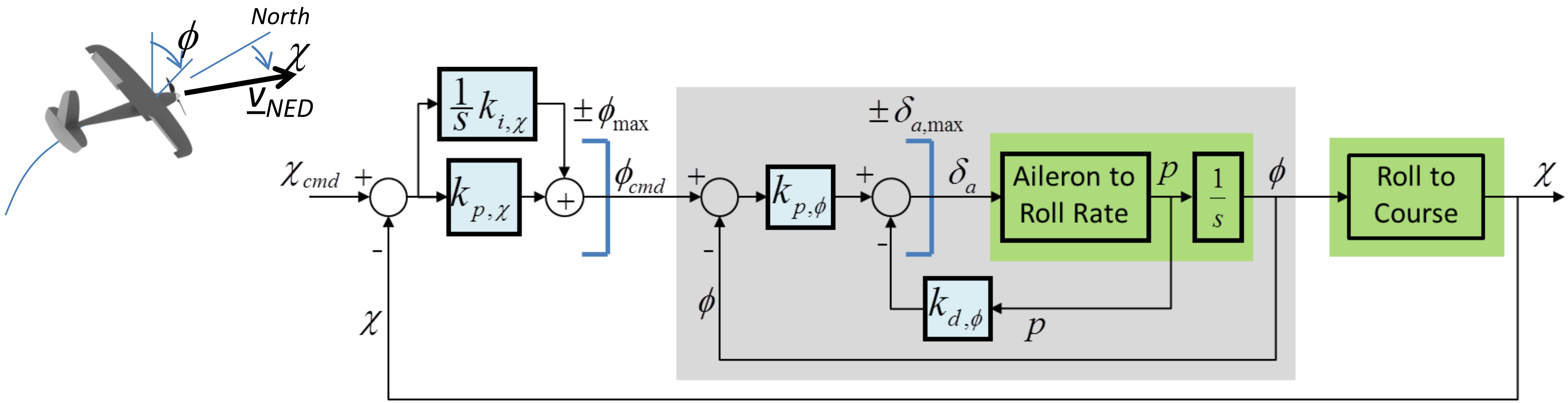
Shown is a common UAV autopilot structure which we'll use in this course. Many other possible structures exist.



- A bank-to-turn aircraft changes course by rolling into a turn
- We will use a P with rate feedback to control Roll
- We will use PI control for our course angle (Roll Cmd is a function of course error)

- *Successive-Loop-Closure*: We'll simplify our course controller design by designing so that our roll response (inner loop) is much quicker than our course response (outer loop)

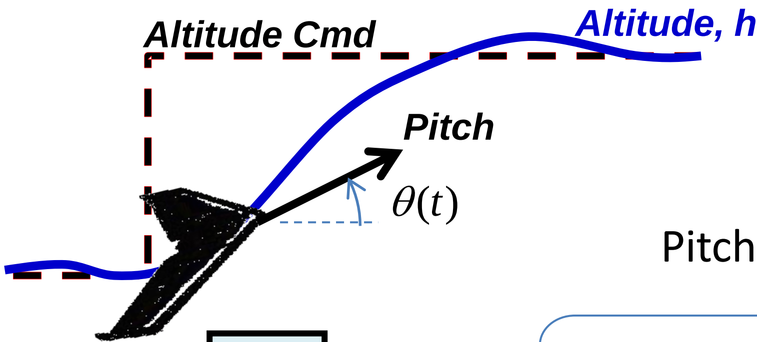
Course Control Autopilot Tuning



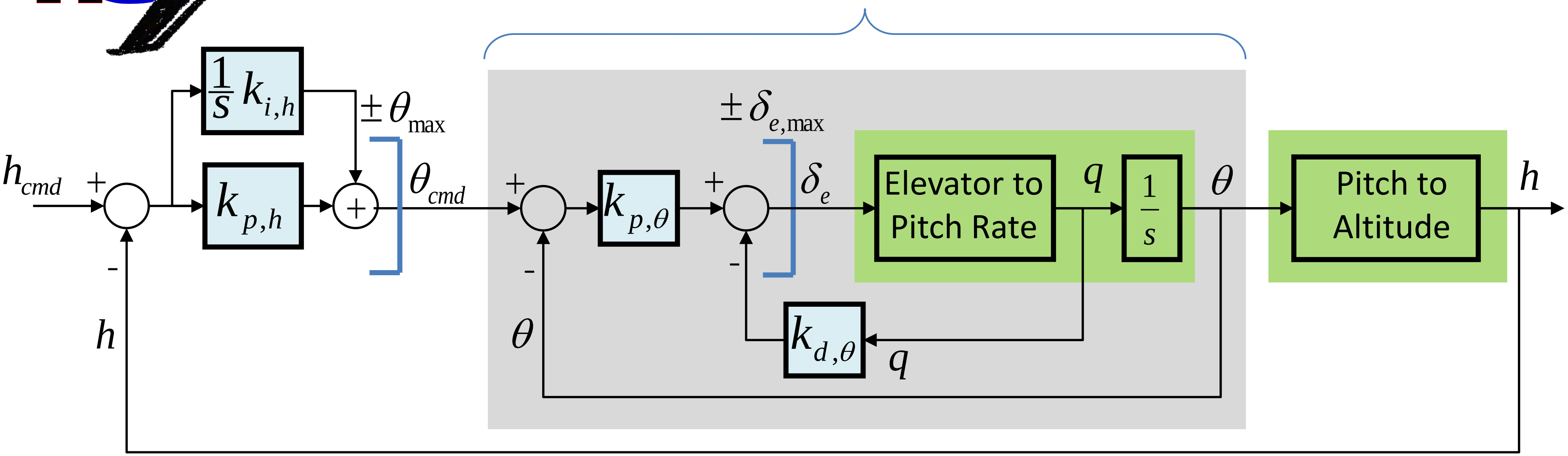
- **If response models are known or estimated:**
  - *Use design parameters to determine four gains*
    - We will derive simplified response models in the next lecture, and develop gain selection logic thereafter
- **If response models are not known, then autopilot must be tuned in-flight**
  - *Beard & McLain suggest the following method, in order:*
    1. Set all gains to zero and manually trim vehicle in-flight
    2. $k_{d,\phi}$: Increase $k_{d,\phi}$ until onset of instability, then back off 20%
    3. $k_{p,\phi}$: Tune $k_{p,\phi}$ to get acceptable roll step response
    4. $k_{p,\chi}$: Tune $k_{p,\chi}$ to get acceptable course step response
    5. $k_{i,\chi}$: Tune $k_{i,\chi}$ to remove steady-state error

Altitude Control using Pitch

Shown is a common UAV autopilot structure which we'll use in this course. Many other possible structures exist.



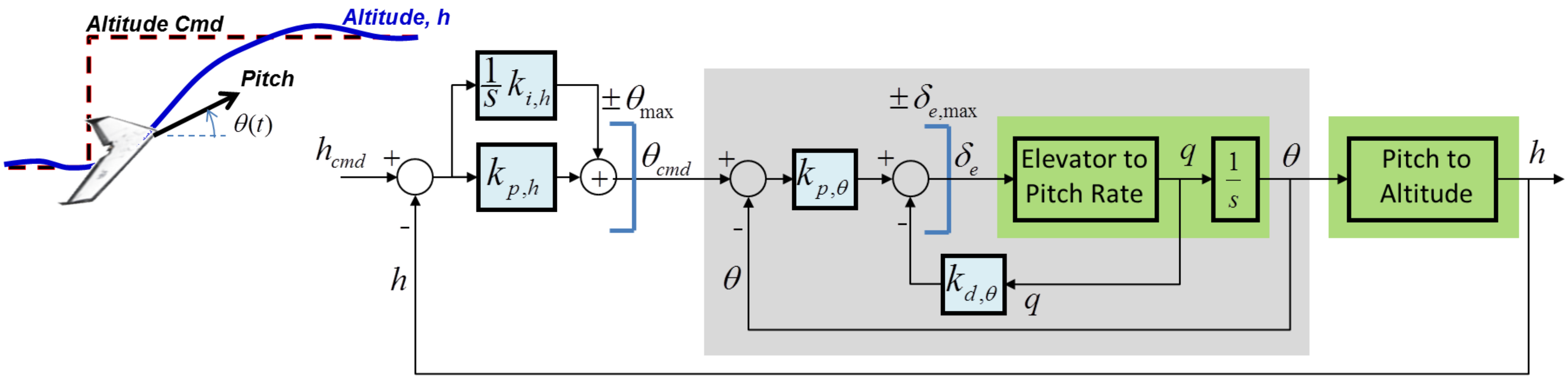
Pitch Cmd to Pitch (fast response)



- Altitude can be controlled using pitch *(or it can be controlled using airspeed, not shown here)*
- We will use a P with rate feedback to control Pitch
- We will use PI control for our altitude (Pitch Cmd is a function of altitude error)

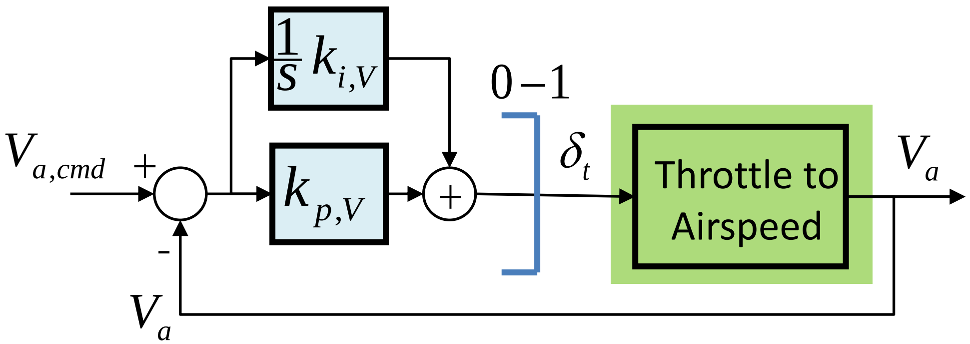
- *Successive-Loop-Closure*: We'll simplify our altitude controller design by designing so that our pitch response (inner loop) is much quicker than our altitude response (outer loop)

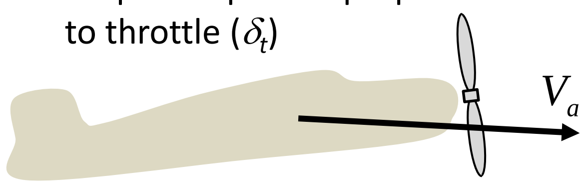
Altitude Control Autopilot Tuning



- **If response models are known or estimated:**
  - *Use design parameters to determine four gains*
    - We will derive simplified response models in the next lecture, and develop gain selection logic thereafter
- **If response models are not known, then autopilot must be tuned in-flight**
  - *Beard & McLain suggest the following method, in order:*
    1. Set all gains to zero and manually trim vehicle in-flight
    2. $k_{d,\theta}$: Increase $k_{d,\theta}$ until onset of instability, then back off 20%
    3. $k_{p,\theta}$: Tune $k_{p,\theta}$ to get acceptable pitch step response
    4. $k_{p,h}$: Tune $k_{p,h}$ to get acceptable altitude step response
    5. $k_{i,h}$: Tune $k_{i,h}$ to remove steady-state error

Airspeed Control Autopilot & Tuning





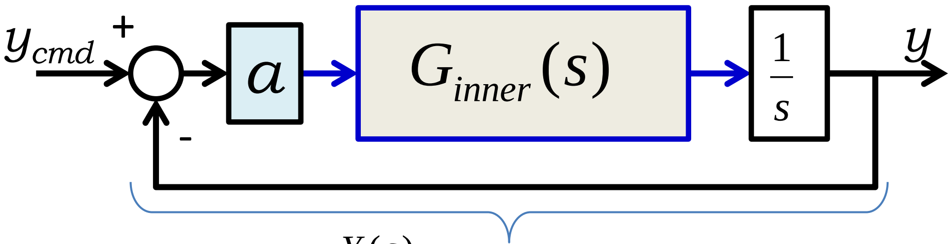
- **Airspeed can be controlled using throttle**
  - or controlled using pitch, not shown here

Shown is a common UAV autopilot structure which we'll use in this course. Many other possible structures exist.

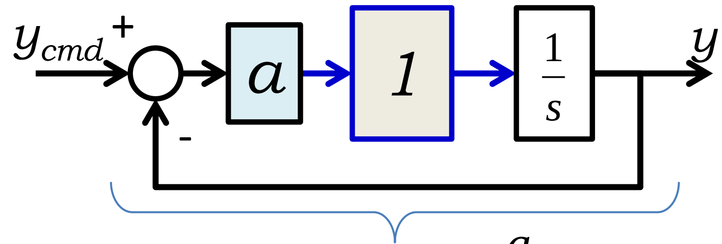
- **If response models are known or estimated:**
  - *Use design parameters to determine two gains*
    - We will derive simplified response models in the next lecture, and develop gain selection logic thereafter
- **If response models are not known, then autopilot must be tuned in-flight**
  - *Beard & McLain suggest the following method, in order:*
    1. $k_{p,V}$: Tune $k_{p,V}$ to get acceptable airspeed step response
    2. $k_{i,V}$: Tune $k_{i,V}$ to remove steady-state error

Lecture 6 Homework, 1/5

Recommended Reading:
Beard & McLain 4.3



$$\frac{Y(s)}{Y_{cmd}(s)} = G_{outer}(s)$$



$$\frac{Y(s)}{Y_{cmd}(s)} \approx G_{approx}(s) = \frac{a}{s+a}$$ (Bandwidth: a rad/s)

1) Consider the shown feedback block diagrams, where $G_{inner}(s)$ represents a stable inner loop with a steady-state value of 1. From the successive loop closure discussion, $G_{outer}(s)$ should be approximately $a/(s+a)$ if the bandwidth of $G_{outer}(s)$ is notably smaller than the bandwidth of $G_{inner}(s)$. (i.e. $G_{outer}(s)$ is much slower than $G_{inner}(s)$.) In this problem, you will vary the bandwidth of the outer loop to explore when the "successive loop closure" approximation is valid.

Let: $$G_{inner}(s) = \frac{\omega_{in}^2}{s^2 + 2\zeta\omega_{in}s + \omega_{in}^2}$$

a) **Symbolically solve for $G_{outer}(s)$ as a function of s, a, ω_{in}, and ζ. Does it look at all similar to $a/(s+a)$?**

b) **Let ω_{in}=50 rad/s and ζ=0.7. Set $a=\omega_{in}/20$, which should yield an outer-loop bandwidth approximately 1/20th of the inner loop bandwidth. Create Matlab transfer functions for $G_{inner}(s)$, $G_{outer}(s)$ and $G_{approx}(s)$.**
Plot a 2-second step response of the 3 TFs: `step( Ginner, Gouter, Gapprox, 2)`
Is $G_{approx}(s)$ a valid approximation of $G_{outer}(s)$?

c) **Repeat for larger values of a: $a=\omega_{in}/10$, $a=\omega_{in}/5$, $a=\omega_{in}/2$. (i.e. increase the outer loop bandwidth.) In your opinion, for which value of a does the approximation no longer hold?**

Lecture 6 Homework, 2/5

2) Manually tune pitch controller. The file “uavsim_control.m” contains a subfunction called “PIR_pitch_hold” which implements PI control with rate feedback. Add the following code to “uavsim_control.m” and manually tune the gains to achieve a reasonable pitch step response for a 20 degree command.

State “estimates” used for feedback (truth for now)

Integrator initialization flag

uavsim parameters

```
theta_c = 20*pi/180;
P.pitch_kp = 0; % kp<0
P.pitch_ki = 0; % ki<=0
P.pitch_kd = 0; % kd<0
delta_e = PIR_pitch_hold(theta_c, theta_hat, q_hat, firstTime, P);
```

Command

Re-run load_uavsim!
(Should start from trim)

Requirements:

stable response, peak time under 2 seconds, settle at 10deg or more.

Notes:

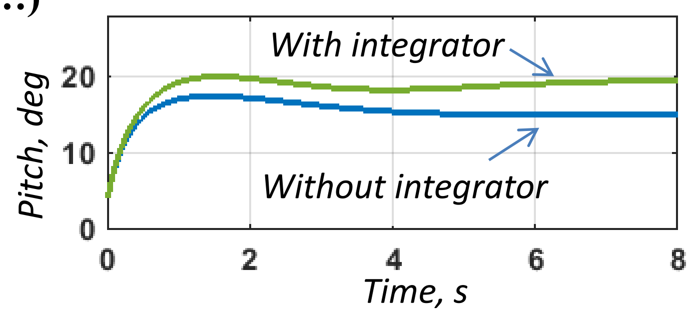
- **May use integrator gain (ki) if you wish (or leave it at zero)**
- **Steady-state pitch error is acceptable (outer altitude loop will remove errors)**
- **All pitch control gains will be negative (because positive elevator is downward)**
- **Start with small gains (e.g. -0.1) and increase in magnitude from there**

May use any method you wish. Examples:

- **Beard & McLain suggested method (described in lecture)**
- **Tuning gains using “PI_rateFeedback_TF()” with G_de2pitch=H(8,1).**
- **Manual tuning using intuition (i.e. playing around…)**

Turn in:

- **Resulting gains (kp, ki & kd)**
- **Brief description of your method/process**
- **Plots of: pitch & elevator deflection**



Lecture 6 Homework, 3/5

3) Manually tune altitude controller. The file "uavsim_control.m" contains a subfunction called "PIR_alt_hold_using_pitch" which implements PI control with rate feedback. Add the following code to "uavsim_control.m" prior to the pitch controller and manually tune the gains to achieve a reasonable altitude step response.

```
if mod(time,20)<10, h_c=50; else, h_c=51; end
P.altitude_kp = 0; % kp>0
P.altitude_ki = 0; % ki>0
P.altitude_kd = 0; % <-- Don't use
theta_c = PIR_alt_hold_using_pitch(h_c, h_hat, 0, firstTime, P);
```

Requirements:

stable response, settle with less than 0.25 m error within 5 seconds

Notes:

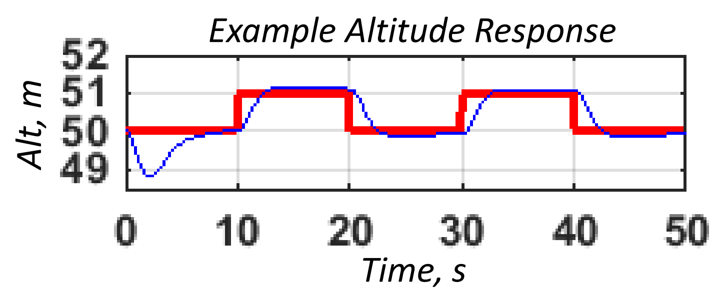
- **Only use kp and ki (our altitude controller won't use rate feedback)**
- **Error may grow initially (see example) due to the integrators winding up to steady values**
  - **Don't worry about it, as long as it settles before 10 seconds)**
- **Make sure to:**
  - **Remove constant 20 degree pitch command (`theta_c=20*pi/180;`) from previous problem**
  - **Change simulation end-time to 50 seconds**
- **Start with small gains (e.g. +0.01) and increase in magnitude from there**

May use any method you wish. Examples:

- **Beard & McLain suggested method (from lecture)**
- **Manual tuning using intuition (i.e. playing around…)**

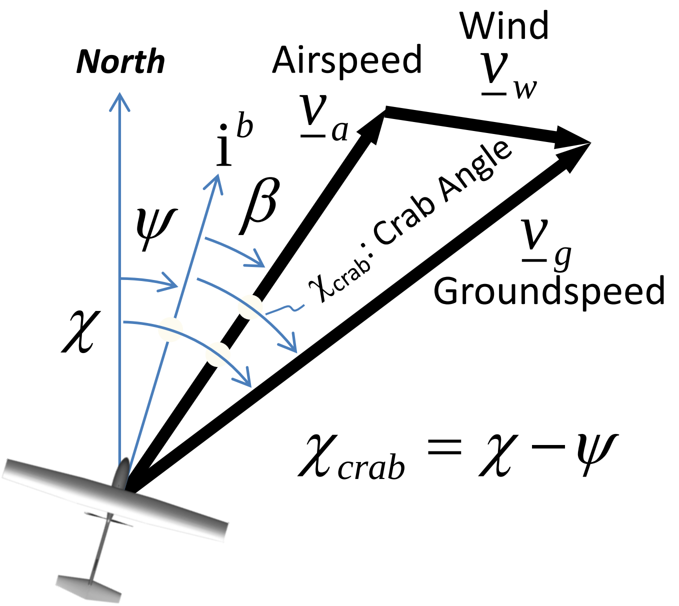
Turn in:

- **Resulting gains (kp & ki)**
- **Brief description of your method/process**
- **Plots of: altitude & pitch**



Lecture 6 Homework, 4/5

If you were successful in making an altitude hold controller, set the altitude command (h_c) to a constant 50 meters. If not, continue with the trimmed elevator: “delta_e=P.delta_e0”.



4) Add a steady cross wind on your UAV. Temporarily (i.e. from the Matlab command window) set P.wind_e to 5 m/s. With the wind switch set to “Steady Wind”, the forces acting on the UAV should be affected by the steady wind (via V_a, α & β). Run the simulation for 20 seconds and:

a) Plot: Course, Yaw, Sideslip & Crab Angle

b) Describe the resulting motion

c) Plot or re-draw a “look-down” view of the 3D UAV display, e.g. after the simulation, click on the 3D display and type “view([0 90])”. On the “look-down” view, draw the wind vector, the groundspeed vector, and the body x-axis unit vector. Highlight χ_{crab}.

5) Add Gusting Winds in addition to the steady wind from #4. In “uavsim_winds.m”:

- Set steady wind vector (ws_ned) using parameters in “init_uavsim_params”

- Combine steady wind (ws_ned) and gusting (gust_b) to output total wind in NED frame

Set the switch to “Winds & Gusting” and run the simulation for 20 seconds.

a) Print the relevant parts of uavsim_winds.m

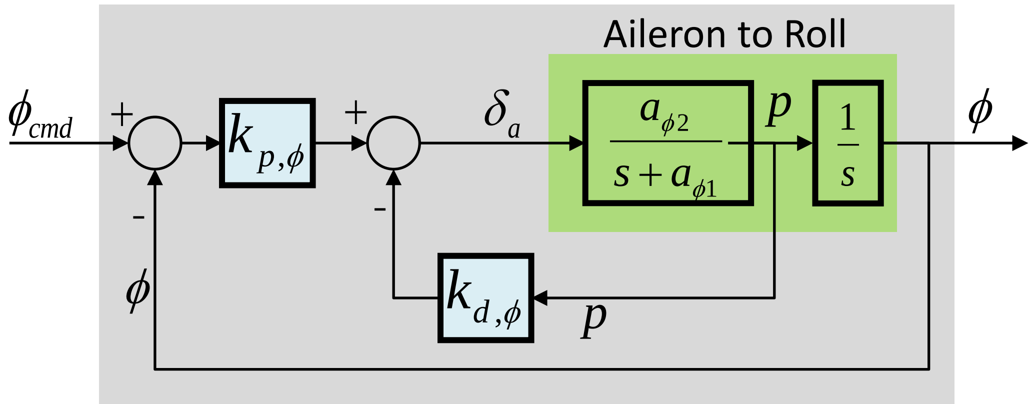
b) Plot the NED components of the wind vector

c) Plot the resulting roll and sideslip angles, and note the effect of the gusting

d) Plot the resulting altitude and the elevator control surface. Does your controller still work? How does the gusting affect the elevator control surface?

Lecture 6 Homework, 5/5

6) **Consider the Roll-Cmd-To-Roll block diagram described in this lecture. During the next lecture we will derive an approximation for the aileron-to-roll response, as shown here. Derive the closed-loop transfer function for the Roll-Cmd-To-Roll response. (Show your work!)**



7) **Consider the Course-Cmd-To-Course block diagram described in this lecture. During the next lecture we will derive an approximation for the roll-to-course response, as shown below. Using this roll-to-course response, and assuming that the inner loop response (Roll-Cmd-to-Roll) is approximately unity, derive the closed-loop transfer function for the Course-Cmd-To-Course response. (Show your work!)**