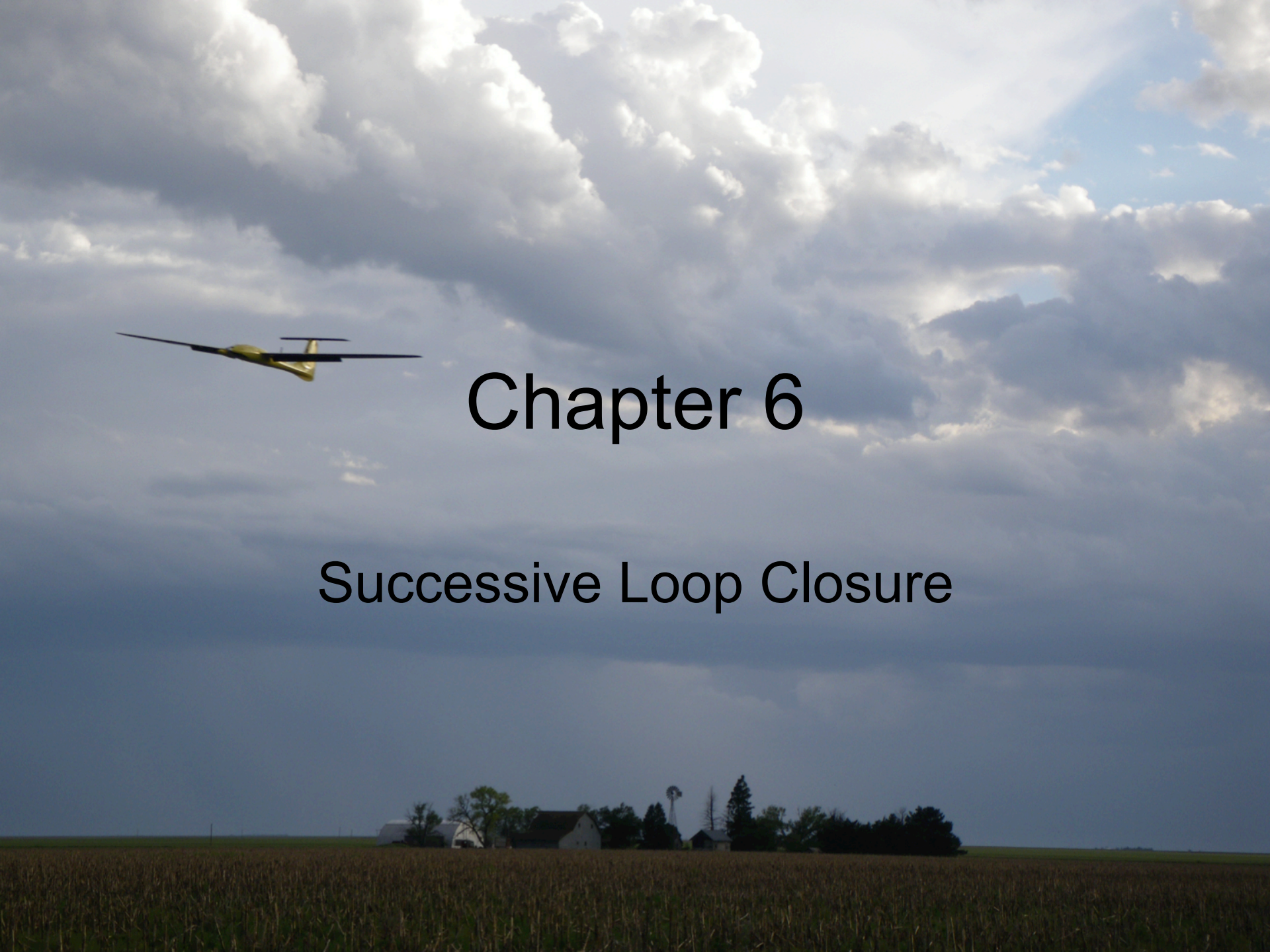
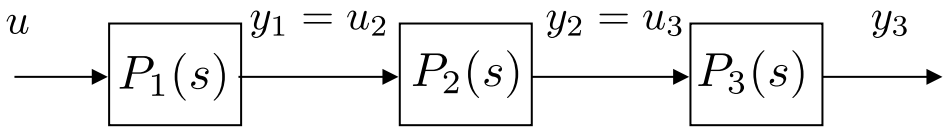


Chapter 6

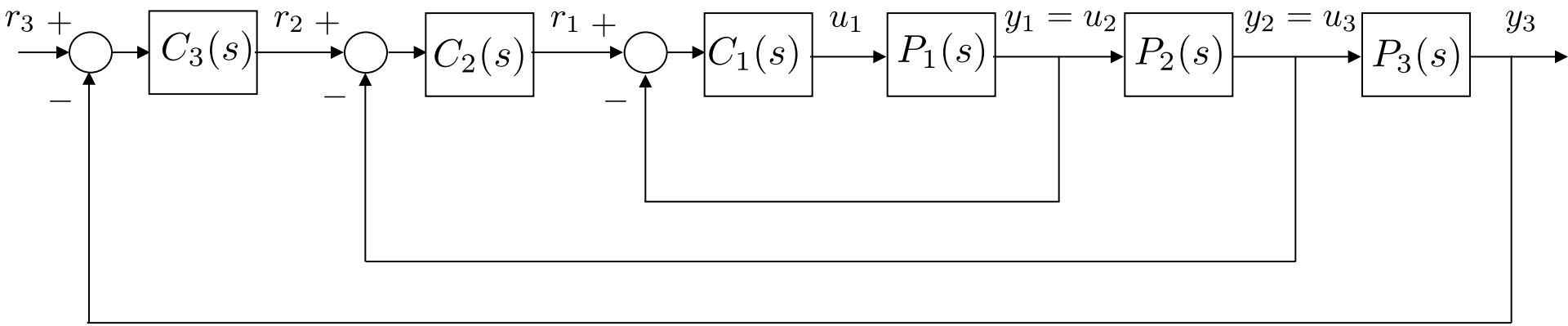
Successive Loop Closure

Successive Loop Closure

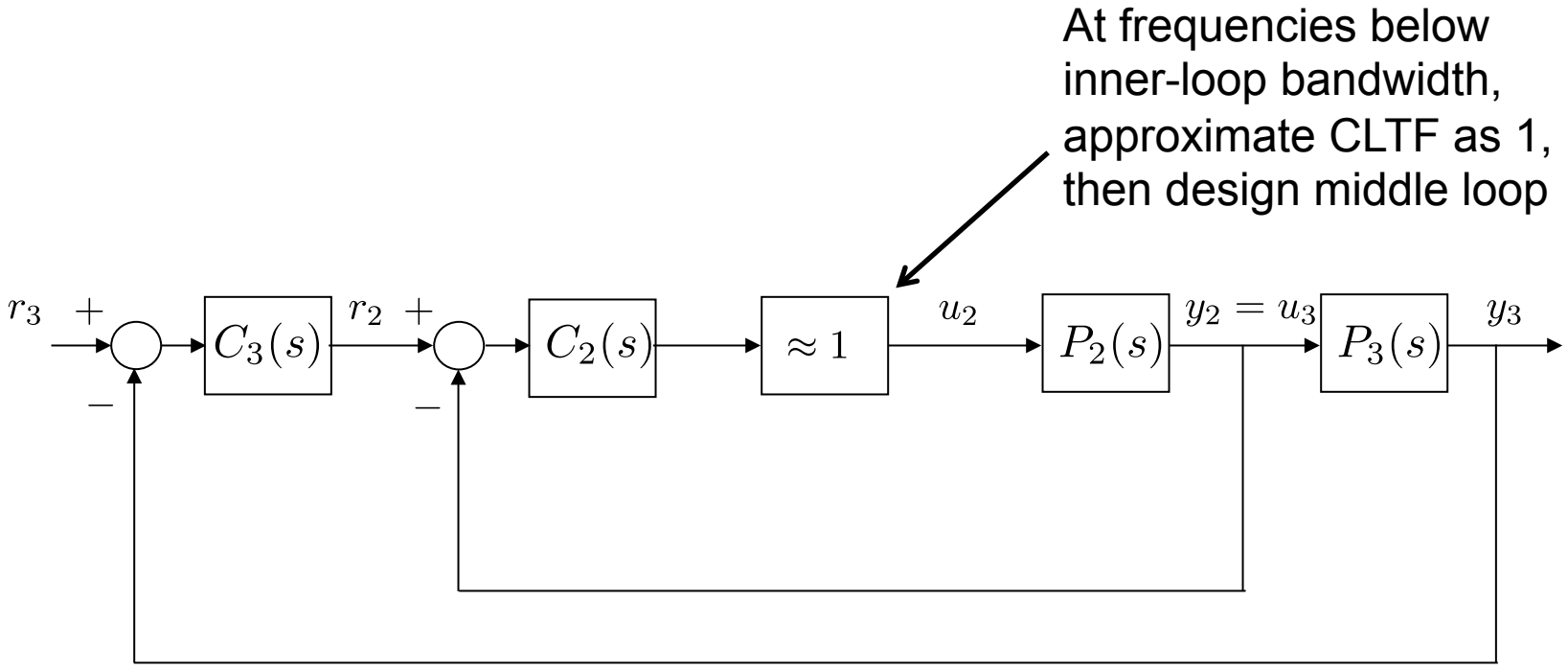
Open-loop system



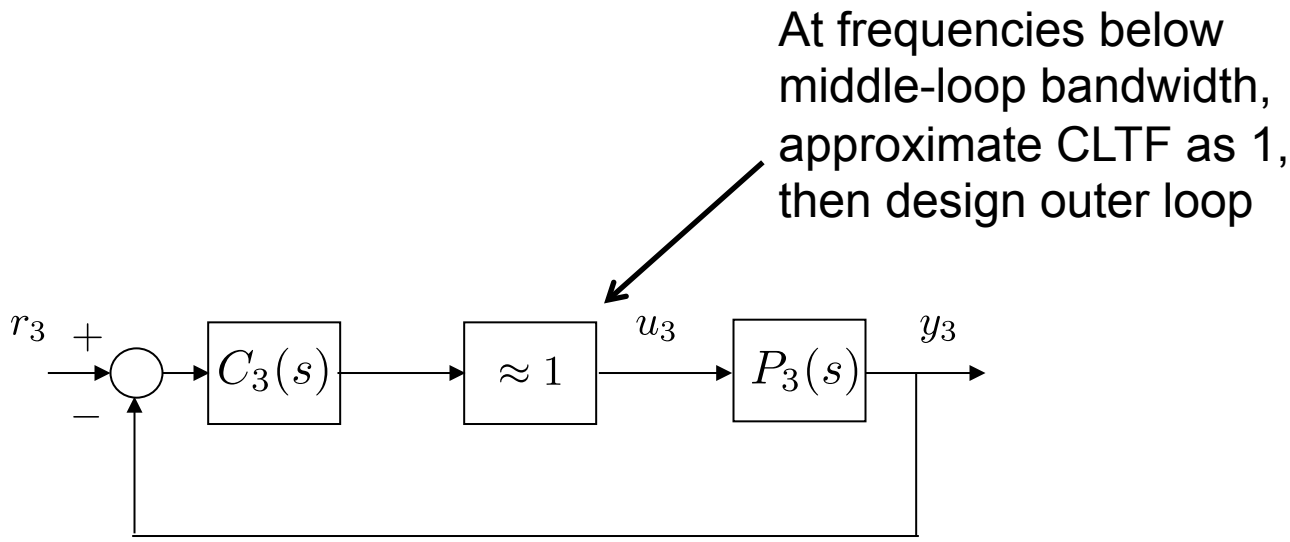
Closed-loop system



SLC: Inner Loop Closed

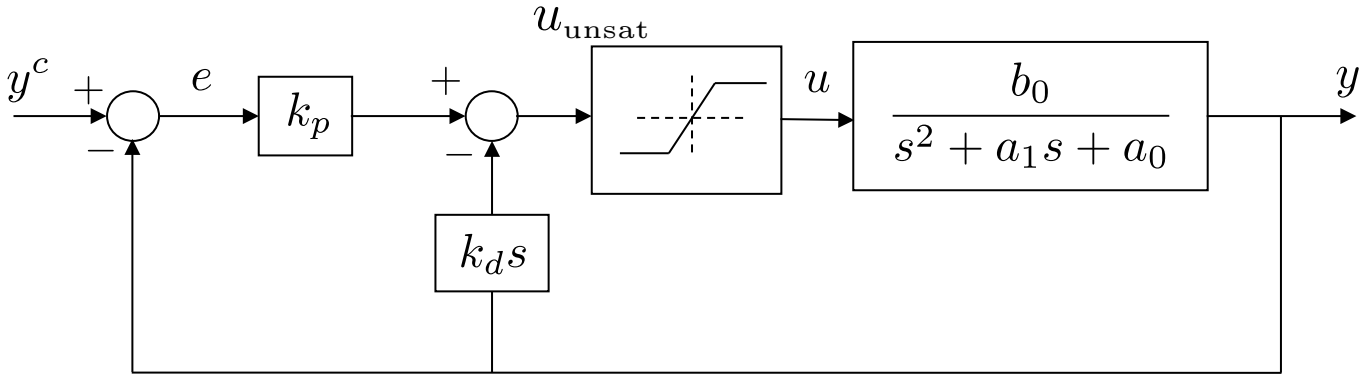


SLC: Two Loops Closed



Key idea: Each successive loop must be lower in bandwidth
--- typically by a factor of 5 to 10

Saturation Limits



The control signal u is largest immediately after a step on y_c, at which point the output of the differentiator is essentially zeros. Therefore $u \approx k_p e$. Let $u^{\max}$ be the input saturation limit, and $e^{\max}$, the largest expected step, then set

$$\boxed{k_p = \frac{u^{\max}}{e^{\max}}.}$$

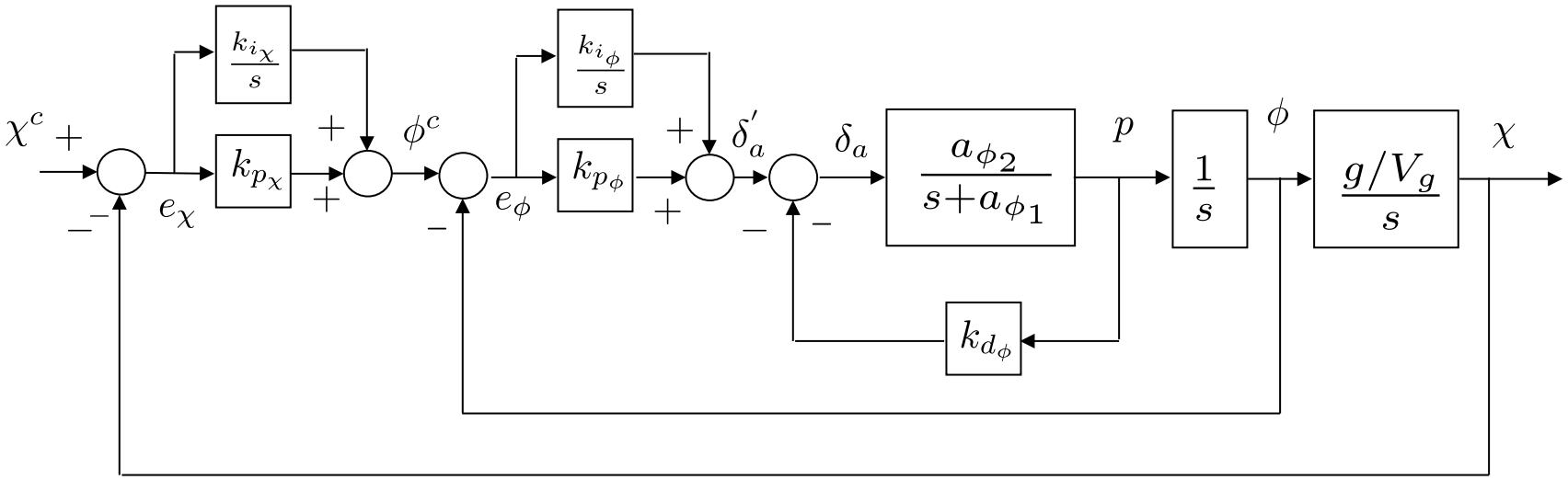
The closed loop transfer function is

$$Y(s) = \frac{b_0 k_p}{s^2 + (a_1 + b_0 k_d)s + (a_0 + b_0 k_p)} Y^c(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} Y^c(s)$$

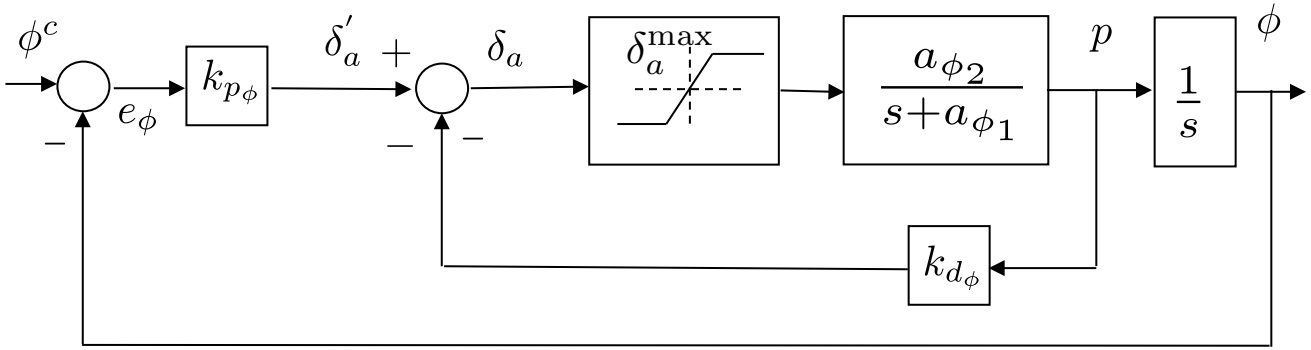
Equating terms gives

$$\omega_n = \sqrt{a_0 + b_0 k_p} \qquad \boxed{k_d = \frac{2\zeta\omega_n - a_1}{b_0}.}$$

Lateral-directional Autopilot



Roll Autopilot



$$H_{\phi/\phi^c}(s) = \underbrace{\frac{k_{p_\phi} a_{\phi 2}}{s^2 + (a_{\phi 1} + a_{\phi 2} k_{d_\phi})s + k_{p_\phi} a_{\phi 2}}}_{\text{Closed Loop TF}} = \underbrace{\frac{\omega_{n_\phi}^2}{s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega_{n_\phi}^2}}_{\text{Canonical } 2^{nd}\text{-order TF}}$$

Design parameters are $e_\phi^{\max}$ and ζ_ϕ

Gains are given by

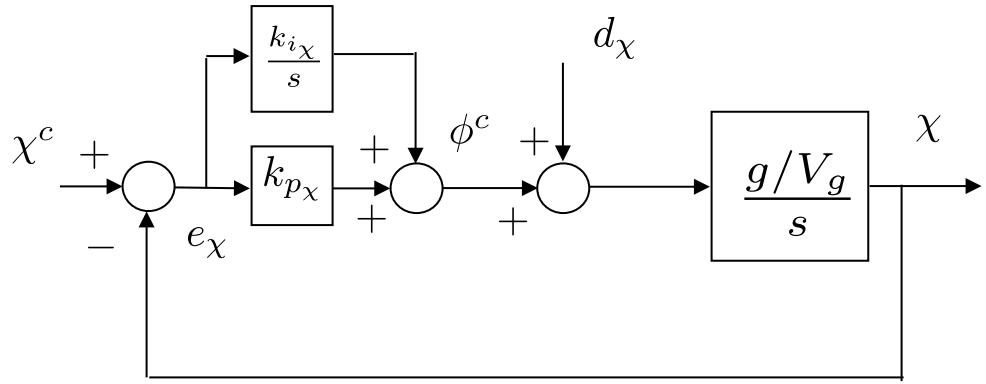
$$k_{p_\phi} = \frac{\delta_a^{\max}}{e_\phi^{\max}}$$

$$k_{d_\phi} = \frac{2\zeta_\phi \omega_{n_\phi} - a_{\phi 1}}{a_{\phi 2}}$$

Roll Autopilot

- The book suggests using an integrator on roll the roll loop to correct for steady state error.
- Our current suggestion is to not have an integrator on inner loops including the roll loop.
    - Integrators add delay and instability -> not a good idea for the inner most loops.
    - An integrator will be used on the course loop to correct for steady state values.

Course Hold Loop



For the course loop, note the presence of the input disturbance.

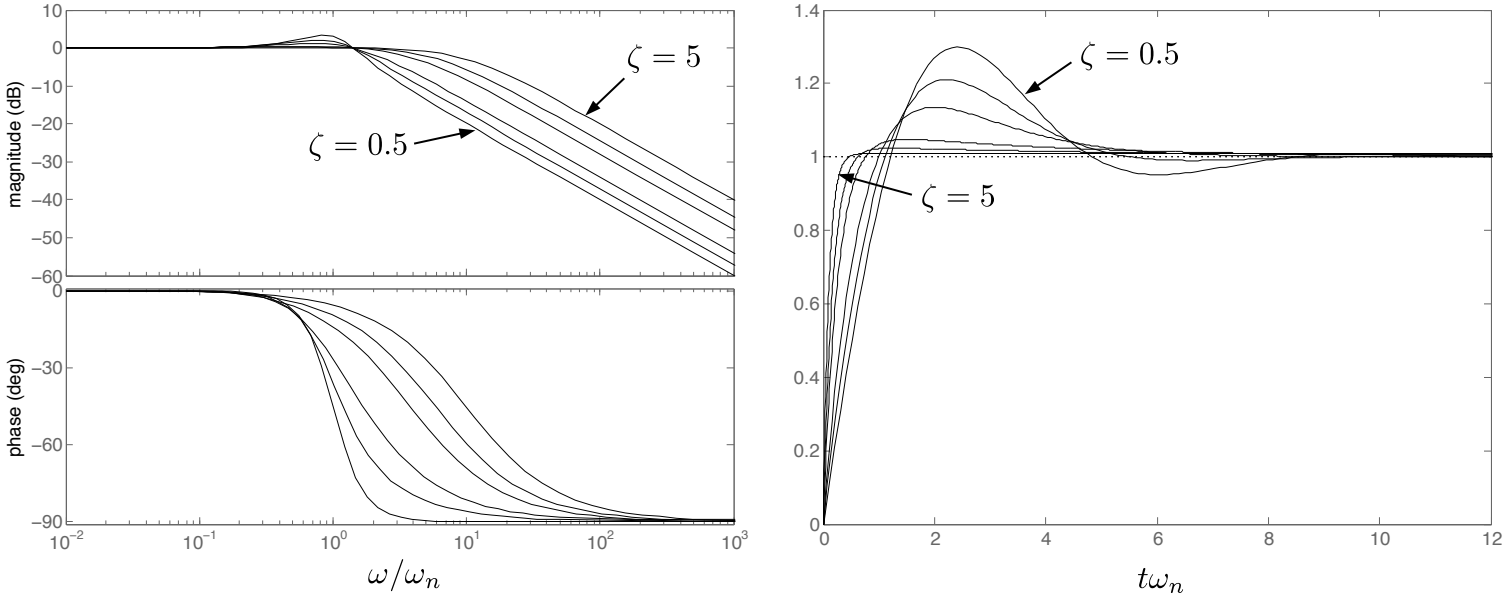
Using a PI controller for course, the response to the course command and disturbance is given by

$$\chi = \frac{k_{p_\chi} g/V_a s + k_{i_\chi} g/V_a}{s^2 + k_{p_\chi} g/V_a s + k_{i_\chi} g/V_a} \chi^c + \frac{g/V_a s}{s^2 + k_{p_\chi} g/V_a s + k_{i_\chi} g/V_a} d_\chi$$

Note:

- There is a zero in the response to the course command χ^c.
- The presence of the zero at the origin ensures rejection of low frequency disturbances.

TF Zero Affects Response

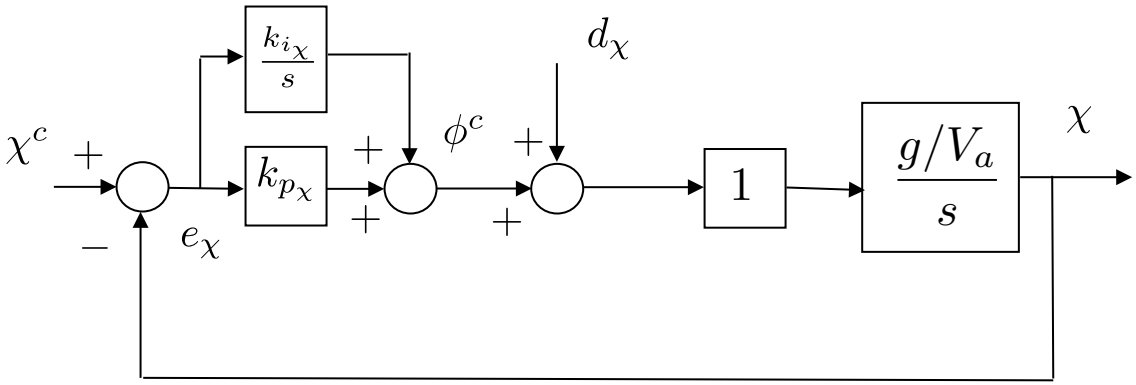


With a zero, the canonical 2^{nd}-order TF is given by

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

Note that ζ has a different effect when the zero is present.

Course Hold Loop



$$\chi = \underbrace{\frac{(k_{p_\chi} g/V_g)s + (k_{i_\chi} g/V_g)}{s^2 + (k_{p_\chi} g/V_g)s + (k_{i_\chi} g/V_g)}\chi^c}_{\text{Response to course command}} + \underbrace{\frac{(g/V_g)s}{s^2 + (k_{p_\chi} g/V_g)s + (k_{i_\chi} g/V_g)}d_\chi}_{\text{Response to disturbance}}$$

Equating coefficients to canonical TF gives:

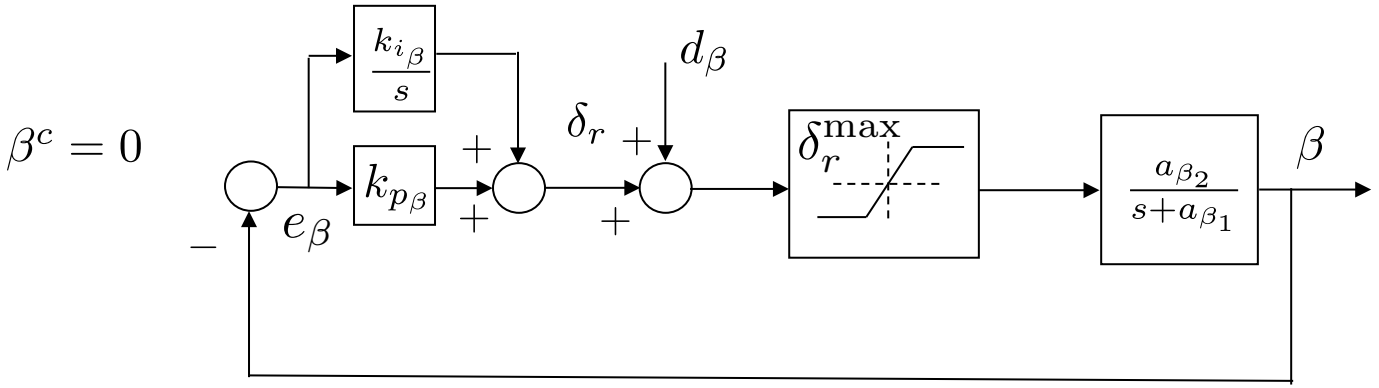
$$\omega_{n_\chi}^2 = k_{i_\chi} g/V_g \qquad \text{and} \qquad 2\zeta_\chi \omega_{n_\chi} = k_{p_\chi} g/V_g$$

or

$$\omega_{n_\chi} = \frac{1}{W_\chi}\omega_{n_\phi} \qquad \boxed{k_{p_\chi} = 2\zeta_\chi \omega_{n_\chi} V_g/g} \qquad \boxed{k_{i_\chi} = \omega_{n_\chi}^2 V_g/g}$$

Design parameters are bandwidth separation W_χ and damping ratio ζ_χ

Sideslip Hold



The transfer function for sideslip hold is

$$H_{\beta/\beta^c}(s) = \frac{a_{\beta_2} k_{p_\beta} s + a_{\beta_2} k_{i_\beta}}{s^2 + (a_{\beta_1} + a_{\beta_2} k_{p_\beta}) s + a_{\beta_2} k_{i_\beta}}$$

Equating coefficients to canonical TF with zero gives

$$\boxed{k_{p_\beta} = \frac{\delta_r^{\max}}{e_\beta^{\max}}} \qquad \omega_{n_\beta} = \frac{a_{\beta_1} + a_{\beta_2} k_{p_\beta}}{2\zeta_\beta} \qquad \boxed{k_{i_\beta} = \frac{\omega_{n_\beta}^2}{a_{\beta_2}}}$$

Design parameters are maximum error $e_\beta^{\max}$ and damping ratio ζ_β

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.

Sideslip hold (if rudder is available)

- $e_\beta^{\max}$ - Error in sideslip when rudder just saturates.
- ζ_β - Damping ratio for sideslip loop.

Lateral Autopilot – In Flight Tuning

If model is not known, and autopilot must be tuned in flight, then the following gains are tuned one at a time, in this specific order:

Inner Loop (roll attitude hold)

- k_{d_ϕ} - Increase k_{d_ϕ} until onset of instability, and then back off by 20%.
- k_{p_ϕ} - Tune k_{p_ϕ} to get acceptable step response.

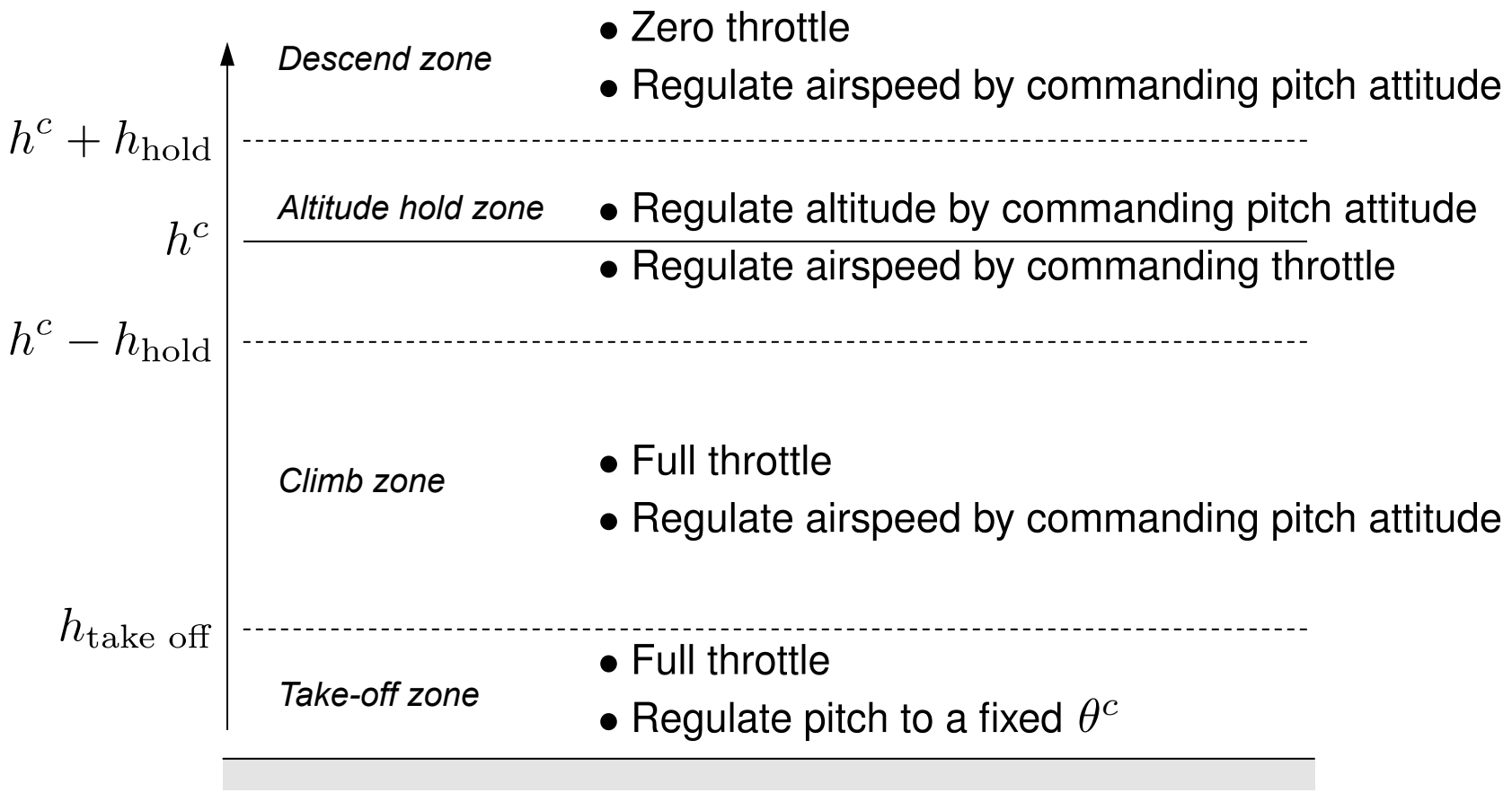
Outer Loop (course hold)

- k_{p_χ} - Tune k_{p_χ} to get acceptable step response.
- k_{i_χ} - Tune k_{i_χ} to remove steady state error.

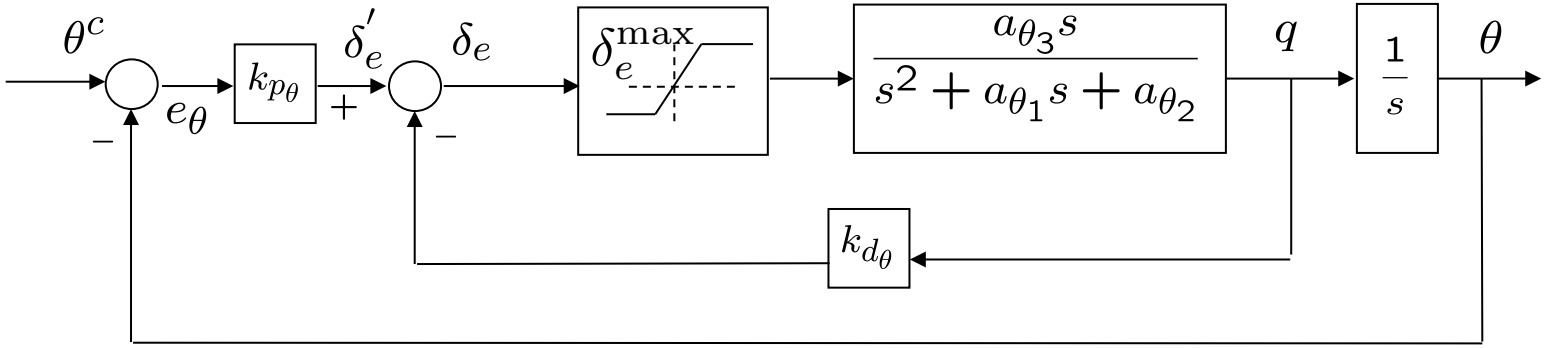
Sideslip hold (if rudder is available)

- k_{p_β} - Tune k_{p_β} to get acceptable step response.
- k_{i_β} - Tune k_{i_β} to remove steady state error.

Longitudinal Flight Regimes



Pitch Attitude Hold



$$H_{\theta/\theta^c}(s) = \underbrace{\frac{k_{p_\theta}a_{\theta_3}}{s^2+(a_{\theta_1}+k_{d_\theta}a_{\theta_3})s+(a_{\theta_2}+k_{p_\theta}a_{\theta_3})}}_{\text{Closed Loop TF}} = \underbrace{\frac{K_{\theta_{\rm DC}}\omega_{n_\theta}^2}{s^2+2\zeta_\theta\omega_{n_\theta}s+\omega_{n_\theta}^2}}_{\text{Note: Non-unity DC Gain}}$$

Equating coefficients, the gains are given by

$$k_{p_\theta} = \frac{\delta_e^{\max}}{e_\theta^{\max}}\text{sign}(a_{\theta_3})$$

$$\omega_{n_\theta} = \sqrt{a_{\theta_2}+k_{p_\theta}a_{\theta_3}}$$

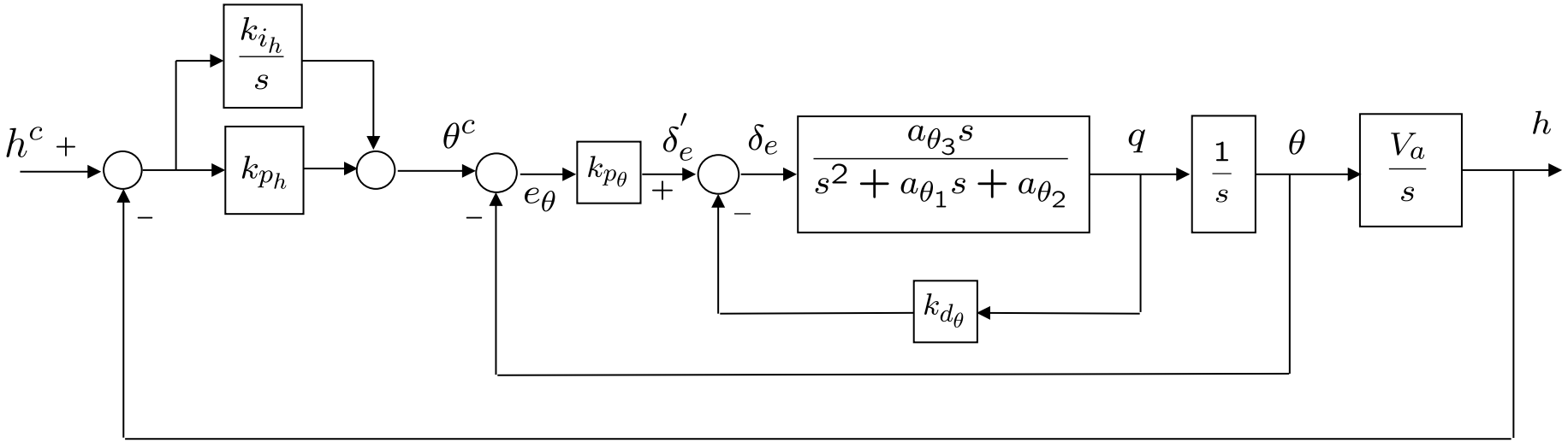
$$k_{d_\phi} = \frac{2\zeta_\theta\omega_{n_\theta}-a_{\theta_1}}{a_{\theta_3}}$$

Design parameters are $e_\theta^{\max}$ and ζ_θ

The DC gain is

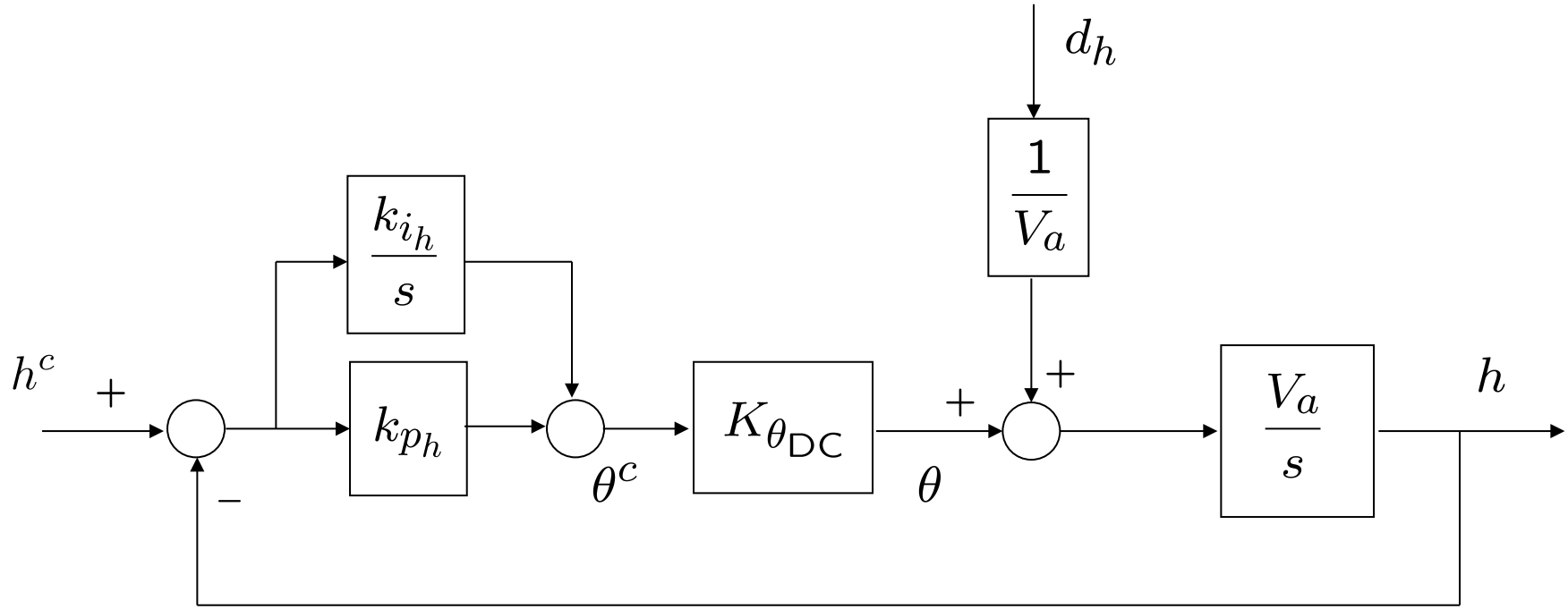
$$K_{\theta_{DC}} = \frac{k_{p_\theta}a_{\theta_3}}{a_{\theta_2}+k_{p_\theta}a_{\theta_3}}$$

Altitude Hold Using Commanded Pitch



Provided pitch loop functions as intended, we can simplify the inner-loop dynamics to $\theta^c/\theta \approx K_{\theta_{\rm DC}}$.

Altitude from Pitch – Simplified



$$h(s) = \left(\frac{K_{\theta_{\text{DC}}} V_a k_{p_h} \left(s + \frac{k_{i_h}}{k_{p_h}} \right)}{s^2 + K_{\theta_{\text{DC}}} V_a k_{p_h} s + K_{\theta_{\text{DC}}} V_a k_{i_h}} \right) h^c(s) + \left(\frac{s}{s^2 + K_{\theta_{\text{DC}}} V_a k_{p_h} s + K_{\theta_{\text{DC}}} V_a k_{i_h}} \right) d_h(s)$$

A PI control on altitude ensures that h tracks constant h^c with zero steady state error, and rejects low frequency disturbances.

Altitude from Pitch Gain Calculations

Equating the transfer functions

$$H_{h/h^c} = \left(\frac{K_{\theta_{\rm DC}} V_a k_{p_h} \left(s + \frac{k_{i_h}}{k_{p_h}} \right)}{s^2 + K_{\theta_{\rm DC}} V_a k_{p_h} s + K_{\theta_{\rm DC}} V_a k_{i_h}} \right) h^c(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

gives the coefficients

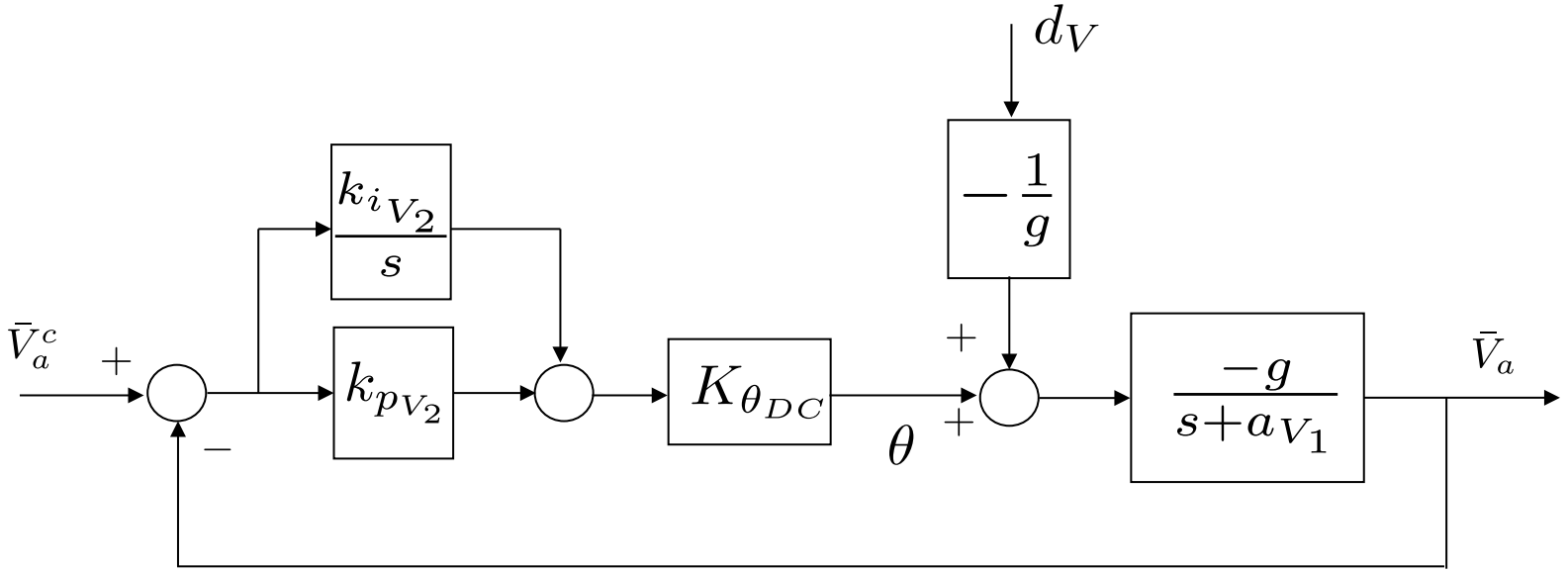
$$\boxed{k_{i_h} = \frac{\omega_{n_h}^2}{K_{\theta_{\rm DC}} V_a}} \qquad \boxed{k_{p_h} = \frac{2\zeta_h \omega_{n_h}}{K_{\theta_{\rm DC}} V_a}}$$

where bandwidth separation is achieved by selecting

$$\omega_{n_h} = \frac{1}{W_h} \omega_{n_\theta}.$$

Design parameters are bandwidth separation W_h and damping ratio ζ_θ

Airspeed Hold Using Commanded Pitch



$$V_a(s) = \left(\frac{-K_{\theta_{\rm DC}} g k_{p_{V_2}} \left(s + \frac{k_{i_{V_2}}}{k_{p_{V_2}}} \right)}{s^2 + (a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}}) s - K_{\theta_{\rm DC}} g k_{i_{V_2}}} \right) V_a^c(s) + \left(\frac{s}{s^2 + (a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}}) s - K_{\theta_{\rm DC}} g k_{i_{V_2}}} \right) d_V(s)$$

A PI control on the pitch to airspeed loop ensures that V_a tracks a constant V_a^c with zero steady state error, and rejects low frequency disturbances.

Airspeed from Pitch Gain Calculations

Equating the transfer functions

$$H_{V_a/V_a^c}(s) = \left(\frac{-K_{\theta_{\rm DC}} g k_{p_{V_2}} \left(s + \frac{k_{i_{V_2}}}{k_{p_{V_2}}} \right)}{s^2 + (a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}}) s - K_{\theta_{\rm DC}} g k_{i_{V_2}}} \right) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

gives the coefficients

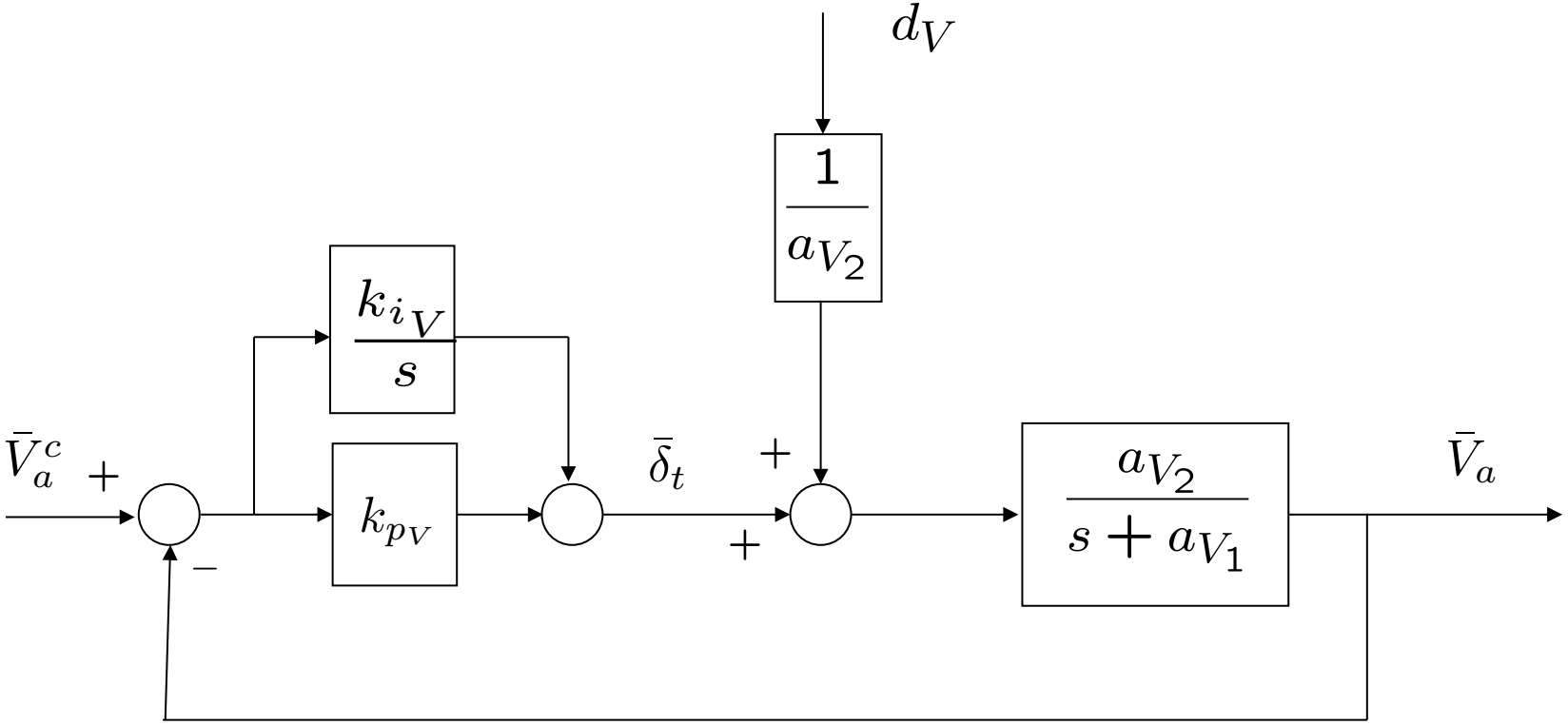
$$\boxed{k_{i_{V_2}} = \frac{\omega_{n_{V_2}}^2}{K_{\theta_{\rm DC}} g}} \qquad \boxed{k_{p_{V_2}} = \frac{a_{V_1} - 2\zeta_{V_2}\omega_{n_{V_2}}}{K_{\theta_{\rm DC}} g}}$$

where bandwidth separation is achieved by selecting

$$\omega_{n_{V_2}} = \frac{1}{W_{V_2}} \omega_{n_\theta}$$

Design parameters are bandwidth separation W_{V_2} and damping ratio ζ_{V_2}

Airspeed Hold Using Throttle



$$V_a = \left(\frac{a_{V_2}(k_{p_V} s + k_{i_V})}{s^2 + (a_{V_1} + a_{V_2} k_{p_V})s + a_{V_2} k_{i_V}} \right) V_a^c + \left(\frac{s}{s^2 + (a_{V_1} + a_{V_2} k_{p_V})s + a_{V_2} k_{i_V}} \right) d_V$$

A PI control on the throttle to airspeed loop ensures that V_a tracks a constant V_a^c with zero steady state error, and rejects low frequency disturbances.

Airspeed Hold Using Throttle

Equating the transfer functions

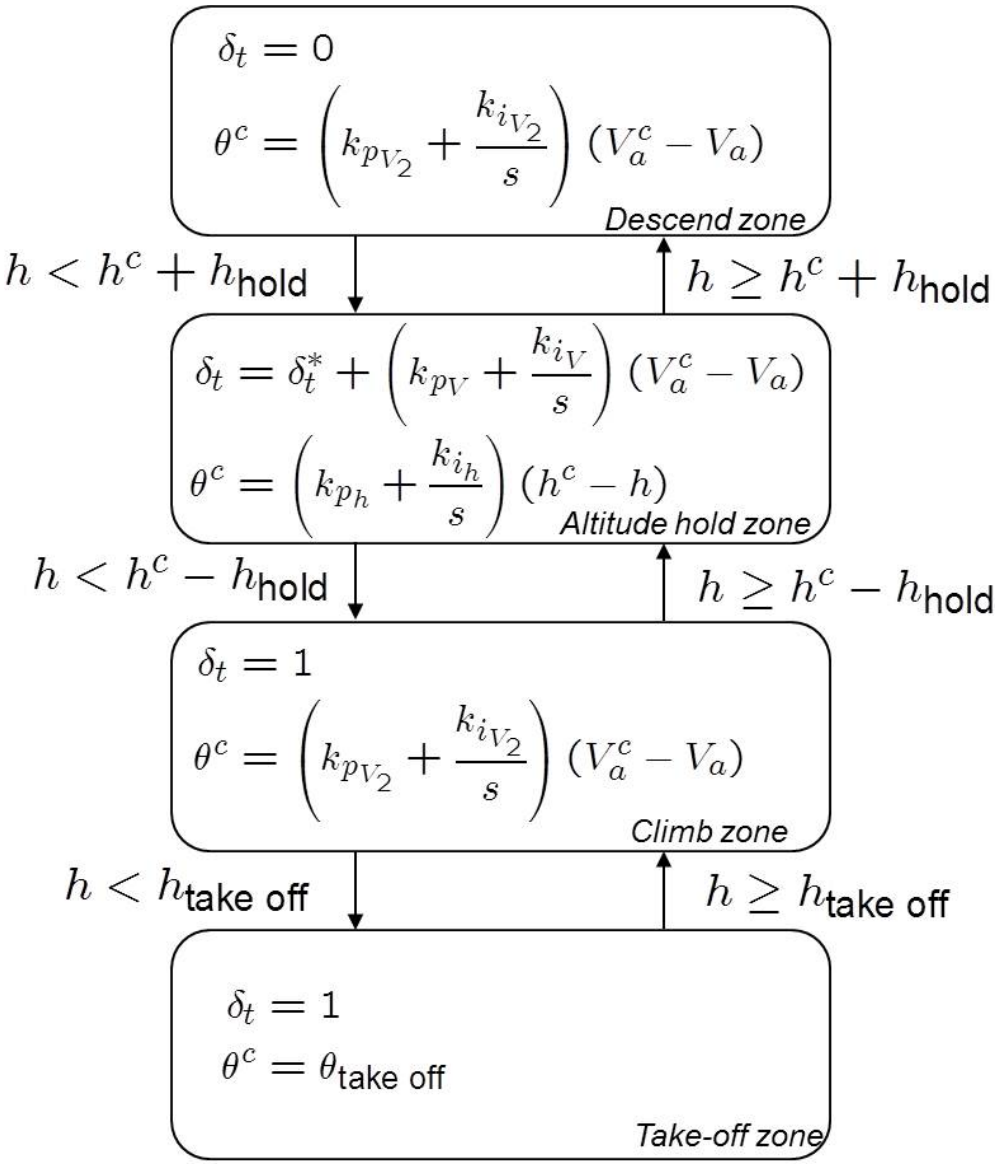
$$H_{V_a/V_a^c}(s) = \left(\frac{a_{V_2}(k_{pV}\, s + k_{iV})}{s^2 + (a_{V_1} + a_{V_2} k_{pV})s + a_{V_2} k_{iV}} \right) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

gives the coefficients

$$\boxed{k_{iV} = \frac{{\omega_{nV}}^2}{a_{V_2}}} \qquad \boxed{k_{pV} = \frac{2\zeta_V \omega_{nV} - a_{V_1}}{a_{V_2}}}$$

Design parameters are natural frequency ω_{nV} and damping ratio ζ_V.

Altitude Control State Machine



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.

Longitudinal Autopilot – In Flight Tuning

If model is not known, and autopilot must be tuned in flight, then the following gains are tuned one at a time, in this specific order:

Inner Loop (pitch attitude hold)

- k_{d_θ} - Increase k_{d_θ} until onset of instability, and then back off by 20%.
- k_{p_θ} - Tune k_{p_θ} to get acceptable step response.

Altitude Hold Outer Loop

- k_{p_h} - Tune k_{p_h} to get acceptable step response.
- k_{i_h} - Tune k_{i_h} to remove steady state error.

Airspeed Hold Outer Loop

- $k_{p_{V_2}}$ - Tune $k_{p_{V_2}}$ to get acceptable step response.
- $k_{i_{V_2}}$ - Tune $k_{i_{V_2}}$ to remove steady state error.

Throttle hold (inner loop)

- k_{p_V} - Tune k_{p_V} to get acceptable step response.
- k_{i_V} - Tune k_{i_V} to remove steady state error.

PID Loop Implementation

PID control in continuous time is given by

$$u(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau) d\tau + k_d \frac{de}{dt}(t)$$

where

$$e(t) = y^c(t) - y(t).$$

Taking the Laplace transform gives

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d s E(s).$$

Use a dirty derivative for causality and to reduce noise:

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d \frac{s}{\tau s + 1} E(s),$$

where $1/\tau$ is the bandwidth of the differentiator.

PID Loop Implementation

To convert to discrete time implementation, use the Tustin (or trapezoidal) rule

$$s \mapsto \frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right).$$

The integrator $I(s) = \frac{1}{s}E(s)$ becomes

$$I(z) = \frac{T_s}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right)E(z).$$

Taking the inverse z-transform gives

$$I[n] = I[n-1] + \frac{T_s}{2}\left(E[n] + E[n-1]\right).$$

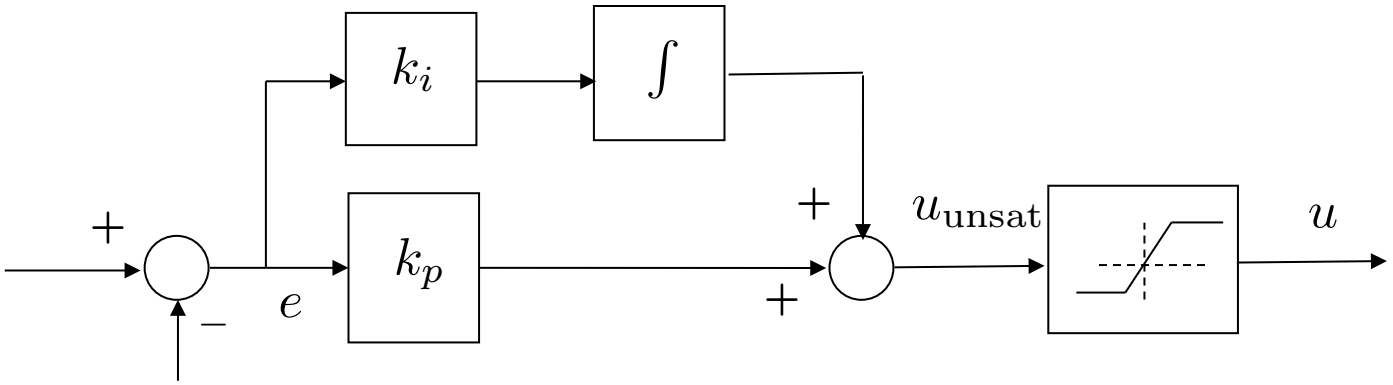
The differentiator $D(s) = \frac{s}{\tau s+1}E(s)$ becomes

$$D(z) = \frac{\frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}{\frac{2\tau}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)+1}E(z) = \frac{\left(\frac{2}{2\tau+T_s}\right)(1-z^{-1})}{1-\left(\frac{2\tau-T_s}{2\tau+T_s}\right)z^{-1}}E(z).$$

Taking the inverse z-transform gives

$$D[n] = \left(\frac{2\tau-T_s}{2\tau+T_s}\right)D[n-1] + \left(\frac{2}{2\tau+T_s}\right)\left(E[n]-E[n-1]\right).$$

Integrator Anti-wind-up



Integrator wind-up happens when the error $e(t)$ persists, causing the integrator to add area, so that u_{unsat} is beyond saturation. When the error changes sign, so that u should also change sign, the positive area under the integrator hold u at the wrong sign until the integrator un-winds.

Anti-wind-up schemes are intended to limit the integrator from winding-up after u is in saturation.

Integrator Anti-wind-up

Let the control before the anti-wind-up update be given by

$$u^-_{\rm unsat} = k_p e + k_d D + k_i I^-$$

and the control after the anti-wind-up update be given by

$$u^+_{\rm unsat} = k_p e + k_d D + k_i I^+.$$

Subtracting the two gives

$$u^+_{\rm unsat} - u^-_{\rm unsat} = k_i (I^+ - I^-).$$

Therefore

$$I^+ = I^- + \frac{1}{k_i}(u^+_{\rm unsat} - u^-_{\rm unsat}),$$

where $u^+_{\rm unsat}$ is selected to be the saturation limit $\bar{u}{\rm sign}(u^-_{\rm unsat})$.

Anti-wind-up is applied when $|u^-_{\rm unsat}| \geq \bar{u}$.

PID Implementation

```
function u = pidloop(y_c, y, flag, kp, ki, kd, limit, Ts, tau)
 persistent integrator;
 persistent differentiator;
 persistent error_d1;
 if flag==1, % reset (initialize) persistent variables
             % when flag==1
   integrator = 0;
   differentiator = 0;
   error_d1  = 0; % _d1 means delayed by one time step
 end
 error = y_c - y; % compute the current error
 integrator = integrator + (Ts/2)*(error + error_d1);
 % update integrator
 differentiator = (2*tau-Ts)/(2*tau+Ts)*differentiator...
   + 2/(2*tau+Ts)*(error - error_d1);
 % update differentiator
 error_d1 = error; % update the error for next time through
                   % the loop
 u = sat(...                 % implement PID control
   kp * error +...           % proportional term
   ki * integrator +...      % integral term
   kd * differentiator,...   % derivative term
   limit...                  % ensure abs(u)<=limit
   );
 % implement integrator anti-windup
 if ki~=0
  u_unsat = kp*error + ki*integrator + kd*differentiator;
  integrator = integrator + Ts/ki * (u - u_unsat);
 end

function out = sat(in, limit)
 if in > limit,    out = limit;
 elseif in < -limit;  out = -limit;
 else         out = in;
 end
```

Simulation Project

- Roll attitude loop. For Aerosonde model use $V_a = 17\ m/s$. Note that $\phi^{\max}$ is a design parameter. Put aircraft in trim, and command steps on roll.
- Course attitude loop. Command steps in χ. Can by-pass simplified simulink files.
- For sideslip, assume no rudder, i.e., set $\delta_r = 0$.
- Pitch attitude loop. Note that $e_\theta^{\max}$ is a design parameter. Don't use simplified simulink file. Command steps in pitch angle.
- Altitude using pitch, airspeed using pitch, and airspeed using throttle: implement directly on full Simulink model.
- Implement full autopilot using state machine for longitudinal control. Simulation should be from take-off to altitude hold.