Integrator windup and PID controller design

by Ania Baetica

11/19/2015

Integrator windup mechanism

- **Windup** = When the controller reaches the actuator limit, then the actuator becomes saturated and the system effectively operates in open loop

- The integral term and the controller output may become really large = large overshoot

- The controller signal remains saturated even if the error begins to increase; hence, very bad transients

- Example: When a car is on a steep hill, the throttle saturates when the cruise control attempts to maintain speed

- We are interested in an anti-windup compensator that prevents the error from building up excessively in the integral term of the controller.

How to avoid integrator windup?

Avoiding Windup

There are many ways to avoid windup. One method is illustrated in Figure 10.8: the system has an extra feedback path that is generated by measuring the actual actuator output, or the output of a mathematical model of the saturating actuator, and forming an error signal (e_s) as the difference between the output of the controller (v) and the actuator output (u). The signal e_s is fed to the input of the integrator through gain k_t. The signal e_s is zero when there is no saturation and the extra feedback loop has no effect on the system. When the actuator saturates, the signal e_s is feedback to the integrator in such a way that e_s goes towards zero. This implies that controller output is kept close to the saturation limit. The controller output will then change as soon as the error changes sign and integral windup is avoided.

The rate at which the controller output is reset is governed by the feedback gain, k_t, a large value of k_t give a short reset time. The parameter k_t can, however, not be too large because measurement error can then cause an undesirable reset. A reasonable compromise is to choose $k_t \approx 1/T_i$ for PI control and as $k_t \approx 1/\sqrt{T_i T_d}$ for PID control. We illustrate how integral windup can be avoided by investigating the cruise control system.

Avoiding windup (1)

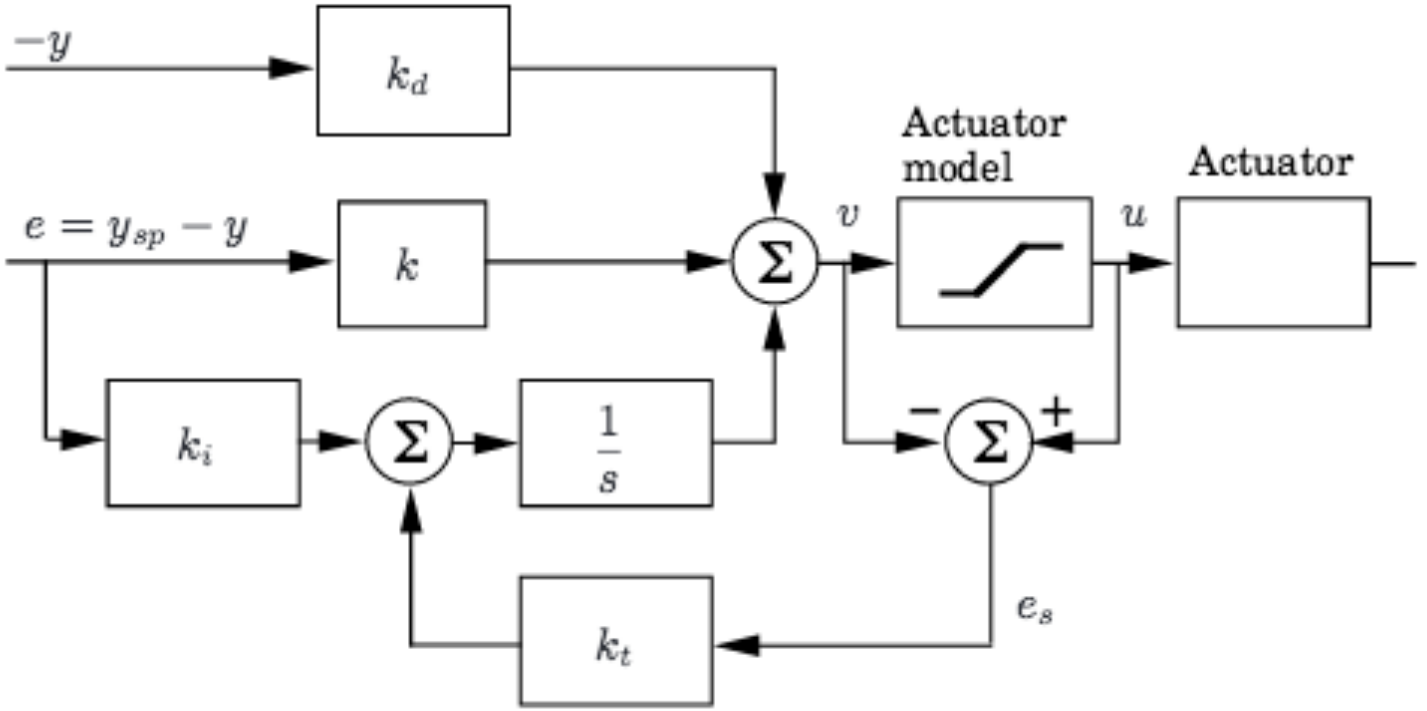


Figure 10.8: PID controller with anti-windup.

Avoiding windup (2)

- Back-calculation: Use feedback of the difference between the desired (v) and the actual control (u) as input for the integral term

- If v is not saturated, then set u = v
- If v is at the saturation value, then set u = saturation value -> the process is in open loop

- The error signal e_s= u – v. It is non-zero only when the actuator is saturated => no effect on normal operation.

- The normal feedback path around process is broken and a new feedback path around the integrator is formed. The integrator input becomes: $e_s/k_t + e*k/k_i$

- The integrator input is 0 at steady state. The transfer function between it and the error is $s/(s+k_t)$.

- The rule of thumb for choosing k_t is that it be smaller than $1/k_i$, so that the integrator resets slower than integration.

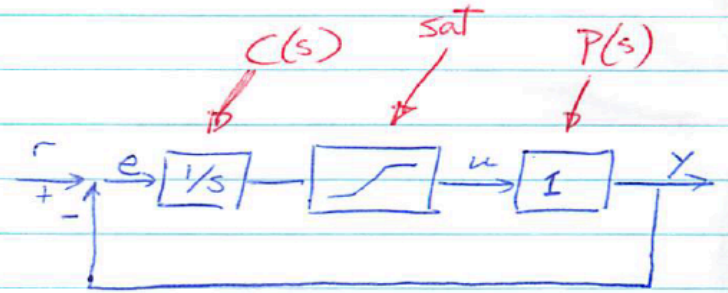
- Back-calculating never allows the input to the actuator to reach its actual saturation level because it forecasts what will actually go into the actuator model beforehand.

Integrator windup example (1)

Windup

Consider $P(s) = 1$

$C(s) = 1/s$



$y = u$

$e = r - y$

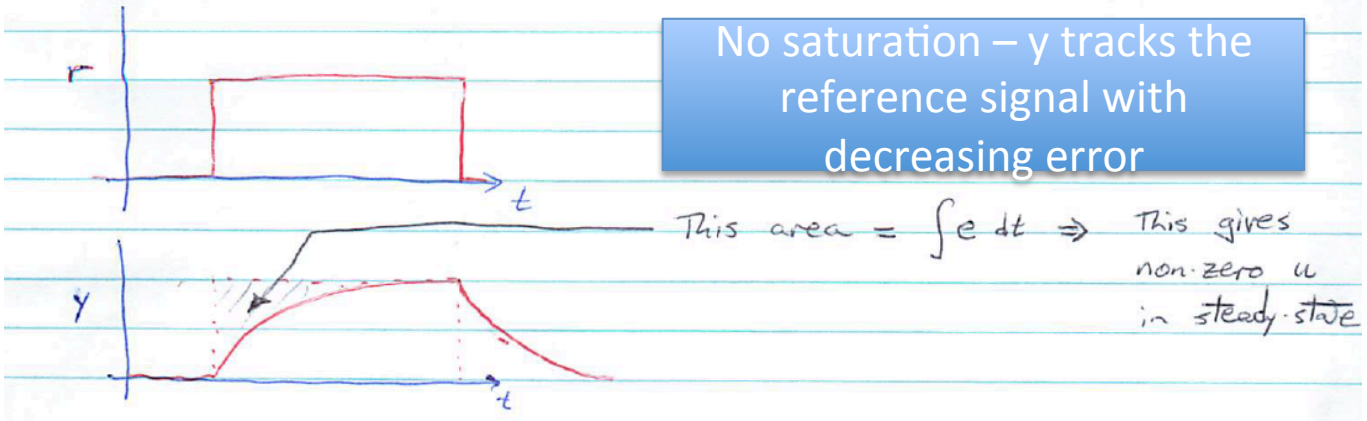
$\dot{x}_c = e$

$u = \min(x_c, 1)$ ← saturation

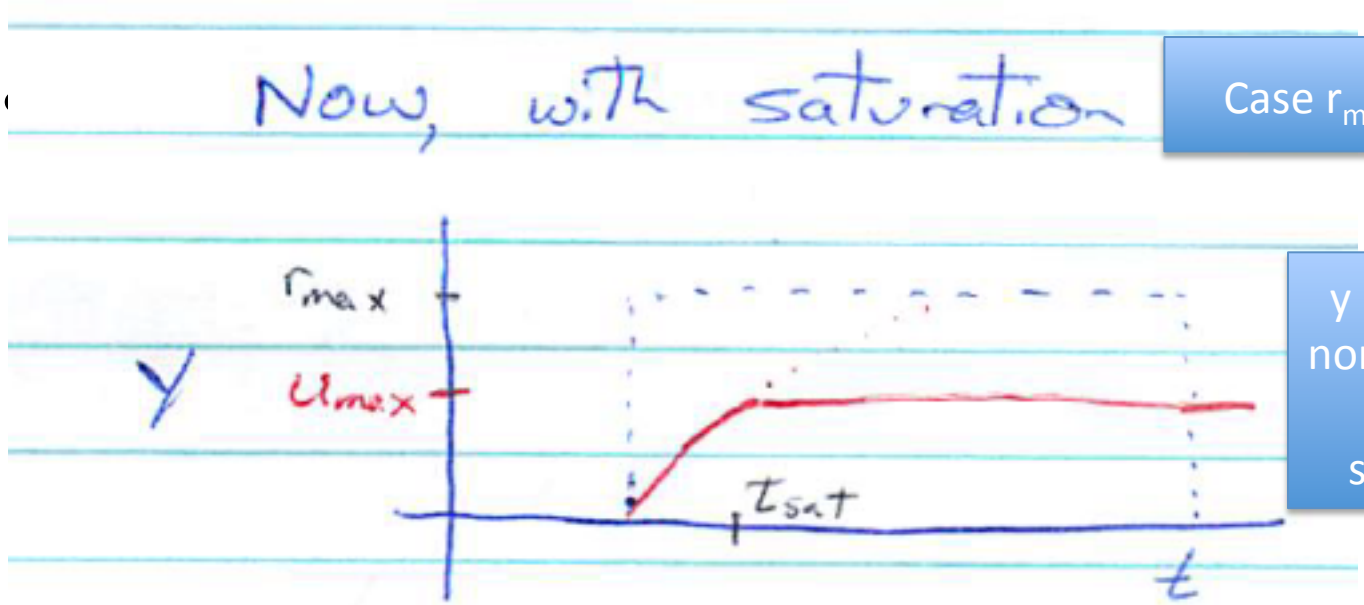
If step response for $r_{max} < 1$ (no saturation) is

$$\dot{x}_c = -x_c + r \Rightarrow x_c(t) = (1 - e^{-t})\, r_{max}$$

$$r = \begin{cases} 0 & t < 0 \\ r_{max} & t \geq 0 \end{cases}$$

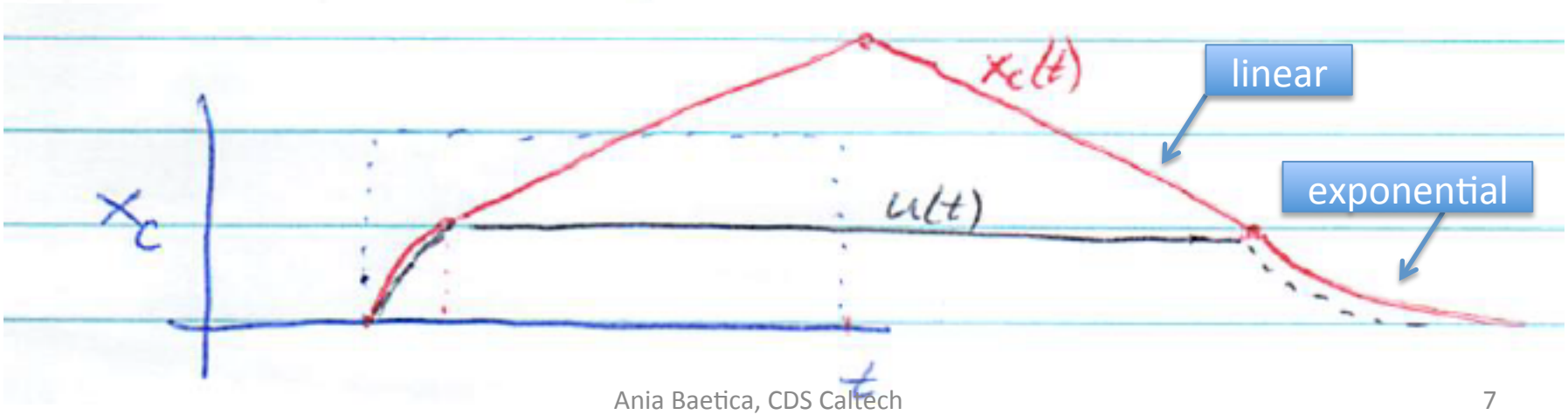


Integrator windup example (2)



Case $r_{max} > 1$

y can't track r_{max} => non-zero steady state error due to saturation effects



linear

exponential

PID specifications

1. The closed loop system is stable <-> the CLS poles have real part < 0 or check the Nyquist plot

2. Less than x% error at frequency 0; it's the same as 3.

3. Less than x% error at steady state

Since $H_{er} = E(s) = \frac{1}{1 + L(s)}$ from the Final Value Thm,

then the condition is |1 + P(0)C(0)| > 100/x; if |L(0)| >> 1, then replace it by |C(0)| > 100 / (x*|P(0)|).

4. At least n degrees of phase margin -> add a derivative term to lift phase; exercise caution!

5. Tracking error of less than x% from 0 to z Hz (remember to convert the Hz to rad/sec, if needed!)

The condition is |C(s)| > 100 / (x*|P(s)|) between frequencies 0 and 2π z rad/sec.

Other possible PID specifications

- Bandwidth (frequency at which the closed loop system is = ½) is at least z ~ the frequency at which |L(s)| ~ 1, so then find s > z such that |C(s)|~ 1/|P(s)|

- Step response specifications on overshoot, settling time, rise time.

PID controller design (1)

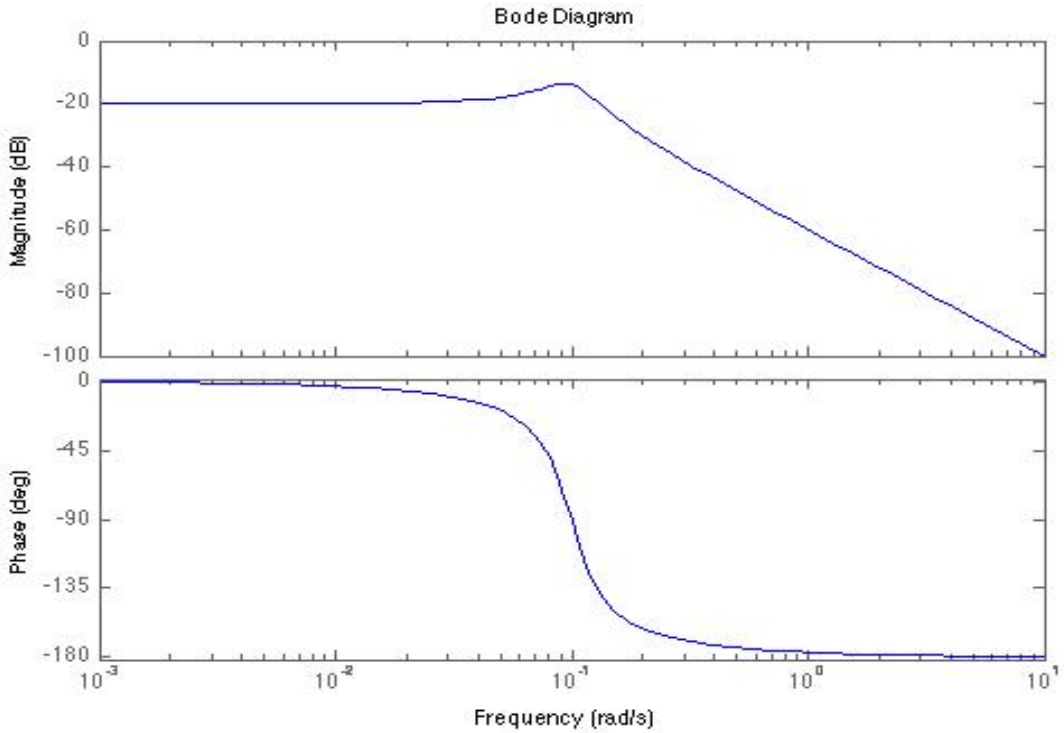
- We have a plant $P(s) = 1 / (ms^2 + bs + c)$, m = 1000, b = 50, c = 10.

- We want to design a controller such that:

1. The steady state error is <= 2%
2. The tracking error is less than 10% between 0 and 1 rad/sec
3. The phase margin is >= 30 degrees

PID controller design (2)

- We first plot the bode plot of the plant and observe that we don't satisfy specifications 1-3.



PID controller design (3)

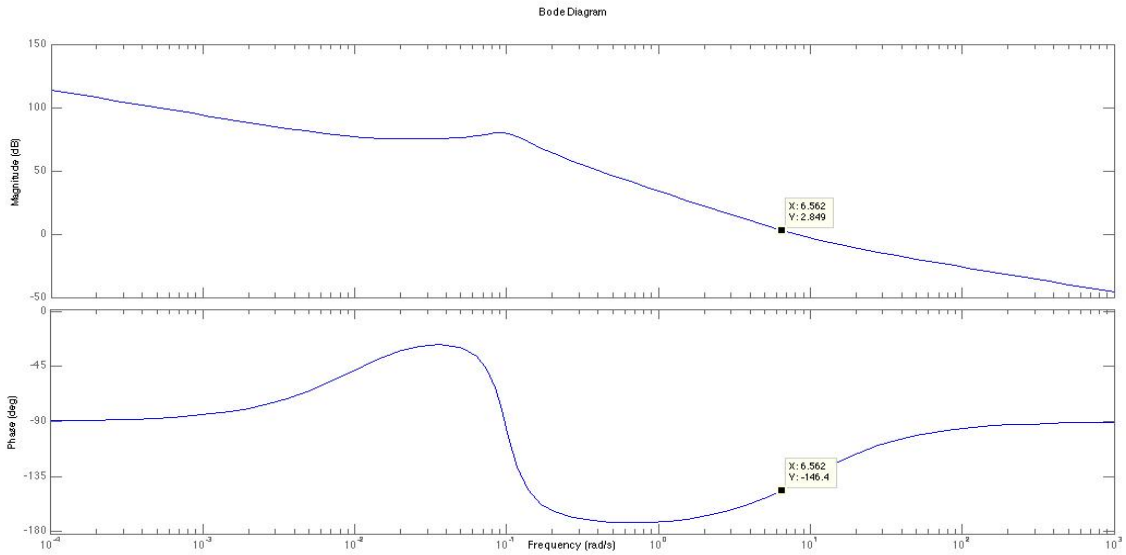
- We first think of adding a proportional controller to lift the graph so that $|L(s)| > 10$ between 0 and 1 rad/sec. We pick gain $k_p = 100$.

- Then we try to set steady state error to 0 (it's the easiest way), so we include an integral term $k_i = 1$.

- Finally, our phase margin is really bad to begin with, so we add a derivative term $k_d = 10$ (after tuning for the right phase margin).

PID controller design (4)

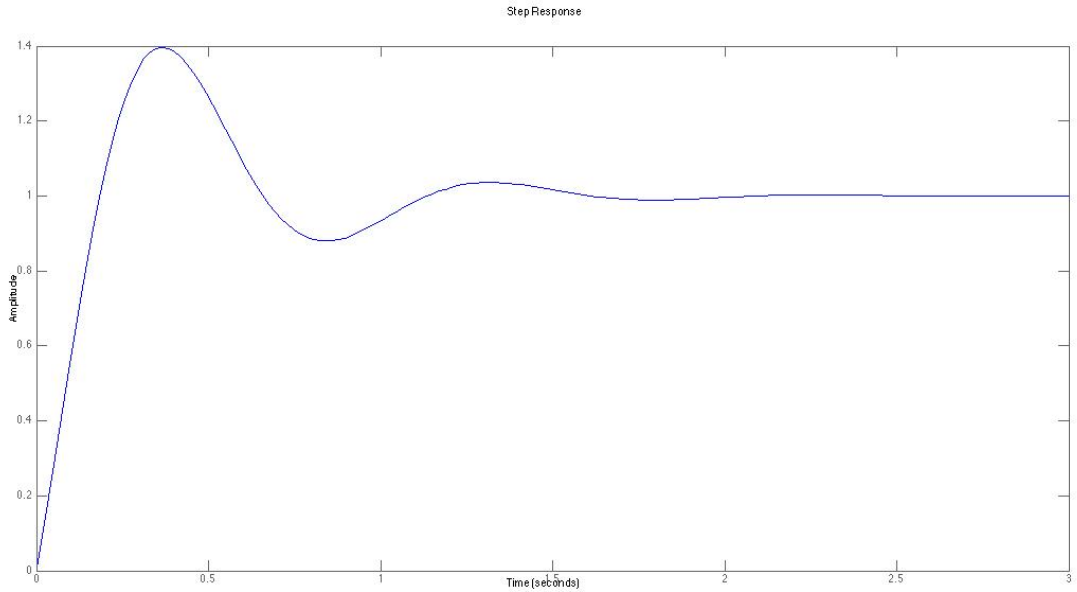
- The controller we have now is C(s) = 100 + 1/s + 10s.
- This is the open loop bode plot:



- We observe that we satisfy specifications 1-3. It has phase margin 34^0.
- Comment: the frequency we compute the phase margin at should be the gain crossover. The freq marked on the plot is close enough.

PID controller design (5)

- Always plot the step response of the closed loop system to see whether you are stable and if the step response behaves reasonably (e.g.: doesn't have large overshoot, poor settling time). Alternatively, you might also be required to satisfy step response specifications, case in which more tuning might be necessary.



PID controller design references

http://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=ControlPID

-> explains how to tune PID under step response specifications

http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID#10

-> check out their website for PID control

Windup references

- https://controls.engin.umich.edu/wiki/index.php/PIDDownsides#Windup (windup)
- https://jagger.berkeley.edu/~pack/me132/Section15.pdf (windup with min and max saturation)
- http://cse.lab.imtlucca.it/~bemporad/teaching/ac/pdf/AC2-09-AntiWindup.pdf (other anti-windup schemes)