ELEC4410

Control System Design

Lecture 6: Affine Parameterisation - Open Loop Unstable Model. Anti-windup Schemes

School of Electrical Engineering and Computer Science
The University of Newcastle

Outline

- Affine Parameterisation - Open Loop Unstable Model.

Outline

- Affine Parameterisation - Open Loop Unstable Model.
- Saturation and Slew Rate Limitations.

Outline

- Affine Parameterisation - Open Loop Unstable Model.
- Saturation and Slew Rate Limitations.
- Anti-windup Schemes.

Outline

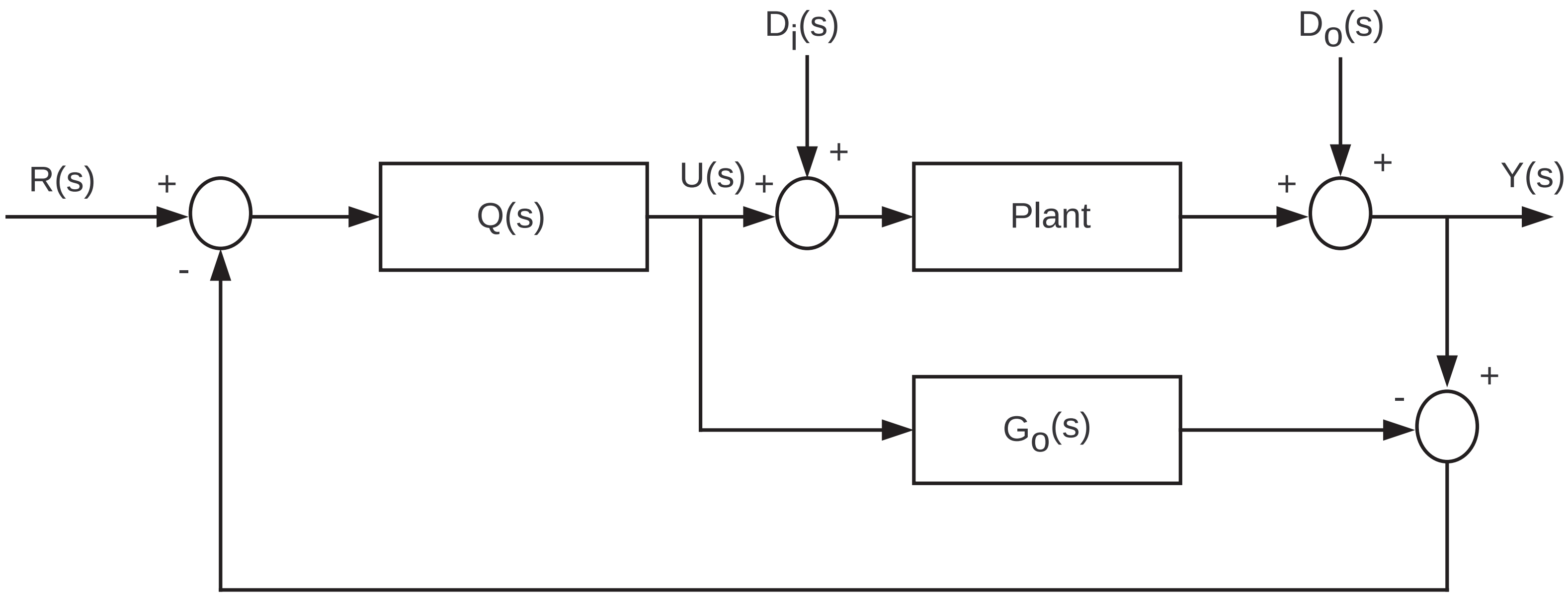
- Affine Parameterisation - Open Loop Unstable Model.
- Saturation and Slew Rate Limitations.
- Anti-windup Schemes.

Reference: Control System Design, Goodwin, Graebe & Salgado.

Affine Parameterisation - Open Loop Unstable Model

- Recall for affine parameterisation for the stable case:

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)}$$



Affine Parameterisation - Open Loop Unstable Model

- If the plant contains unstable poles then its output could increase exponentially.

Affine Parameterisation - Open Loop Unstable Model

- If the plant contains unstable poles then its output could increase exponentially.

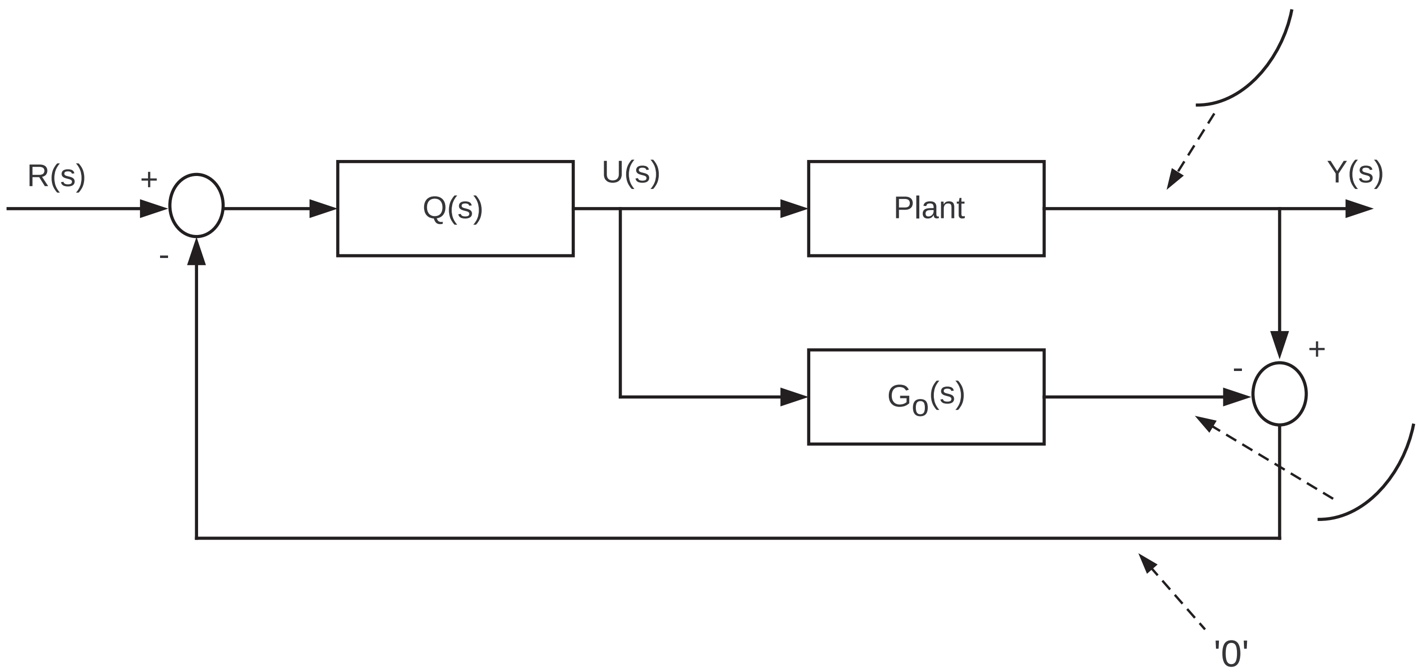
- Likewise the nominal model, $G_o(s)$, will also contain unstable poles, hence its output can also increase exponentially.

Affine Parameterisation - Open Loop Unstable Model

- If the plant contains unstable poles then its output could increase exponentially.
- Likewise the nominal model, $G_o(s)$, will also contain unstable poles, hence its output can also increase exponentially.
- This could result in the difference between the plant and nominal model outputs being zero and hence no control action being taken.

Affine Parameterisation - Open Loop Unstable Model

- If the plant contains unstable poles then its output could increase exponentially.
- Likewise the nominal model, $G_o(s)$, will also contain unstable poles, hence its output can also increase exponentially.
- This could result in the difference between the plant and nominal model outputs being zero and hence no control action being taken.



Affine Parameterisation - Open Loop Unstable Model

- Now in the unstable case,

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)},$$

is still okay! But, we have to make some further points regarding stability of the sensitivity functions.

Affine Parameterisation - Open Loop Unstable Model

- Now in the unstable case,

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)},$$

is still okay! But, we have to make some further points regarding stability of the sensitivity functions.

- Essentially we need to add more interpolation constraints.

Affine Parameterisation - Open Loop Unstable Model

- Now in the unstable case,

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)},$$

  is still okay! But, we have to make some further points regarding stability of the sensitivity functions.

- Essentially we need to add more interpolation constraints.

- To ensure $T_o(s)$, $S_o(s)$, $S_{io}(s)$ and $S_{uo}(s)$ are stable, we still need $Q(s)$ stable and proper. In addition, we add further interpolation constraints:

Affine Parameterisation - Open Loop Unstable Model

- Now in the unstable case,

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)},$$

is still okay! But, we have to make some further points regarding stability of the sensitivity functions.

- Essentially we need to add more interpolation constraints.

- To ensure $T_o(s)$, $S_o(s)$, $S_{io}(s)$ and $S_{uo}(s)$ are stable, we still need $Q(s)$ stable and proper. In addition, we add further interpolation constraints:
  - $Q(s)G_o(s)$ stable.
    [Unstable poles of $G_o(s) \rightarrow$ Zeros of $Q(s)$]

Affine Parameterisation - Open Loop Unstable Model

- Now in the unstable case,

$$C(s) = \frac{Q(s)}{1 - Q(s)G_o(s)},$$

  is still okay! But, we have to make some further points regarding stability of the sensitivity functions.

- Essentially we need to add more interpolation constraints.

- To ensure $T_o(s)$, $S_o(s)$, $S_{io}(s)$ and $S_{uo}(s)$ are stable, we still need $Q(s)$ stable and proper. In addition, we add further interpolation constraints:
  - $Q(s)G_o(s)$ stable.
    [Unstable poles of $G_o(s) \rightarrow$ Zeros of $Q(s)$]
  - $(1 - Q(s)G_o(s))G_o(s)$ stable.
    [Unstable poles of $G_o(s) \rightarrow$ Zeros of $1 - Q(s)G_o(s)$]

Affine Parameterisation - Open Loop Unstable Model

- Recall from last lecture, that unstable pole-zero cancellations should be performed analytically.

Affine Parameterisation - Open Loop Unstable Model

- Recall from last lecture, that unstable pole-zero cancellations should be performed analytically.

- To begin, we express the nominal model in its fractional form

$$G_o(s) = \frac{B_o(s)}{A_o(s)}$$

and assume all the poles of $A_o(s)$ are unstable.

Affine Parameterisation - Open Loop Unstable Model

- Recall from last lecture, that unstable pole-zero cancellations should be performed analytically.

- To begin, we express the nominal model in its fractional form

$$G_o(s) = \frac{B_o(s)}{A_o(s)}$$

and assume all the poles of $A_o(s)$ are unstable.

- Next we choose

$$Q(s) = \frac{\tilde{P}(s)}{\tilde{E}(s)}$$

where $\tilde{E}(s)$ is stable. In particular the zeros of $\tilde{E}(s)$ lie in a desirable region of the complex plane.

Affine Parameterisation - Open Loop Unstable Model

- As we need unstable poles of $G_o(s)$ to be zeros in $Q(s)$, we can write

$$Q(s) = \frac{A_o(s)\bar{P}(s)}{\tilde{E}(s)} \quad ; \quad \tilde{P}(s) = A_o(s)\bar{P}(s)$$

Affine Parameterisation - Open Loop Unstable Model

- As we need unstable poles of $G_o(s)$ to be zeros in $Q(s)$, we can write

$$Q(s) = \frac{A_o(s)\bar{P}(s)}{\tilde{E}(s)} \quad ; \quad \tilde{P}(s) = A_o(s)\bar{P}(s)$$

- Then $Q(s)$ is stable and has zeros to cancel unstable poles of $G_o(s)$.

Affine Parameterisation - Open Loop Unstable Model

- What about $S_{io}(s) = (1 - Q(s)G_o(s))G_o(s)$?

Affine Parameterisation - Open Loop Unstable Model

- What about $S_{io}(s) = (1 - Q(s)G_o(s))G_o(s)$?
- We require the unstable poles of $G_o(s)$ to be zeros in $S_o(s)$.

$$\begin{aligned} 1 - Q(s)G_o(s) &= 1 - \frac{\tilde{P}(s)B_o(s)}{\tilde{E}(s)A_o(s)} \\ &= 1 - \frac{\bar{P}(s)B_o(s)}{\tilde{E}(s)} \\ &= \frac{\tilde{E}(s) - \bar{P}(s)B_o(s)}{\tilde{E}(s)} \end{aligned}$$

Affine Parameterisation - Open Loop Unstable Model

- What about $S_{io}(s) = (1 - Q(s)G_o(s))G_o(s)$?

- We require the unstable poles of $G_o(s)$ to be zeros in $S_o(s)$.

$$
\begin{aligned}
1 - Q(s)G_o(s) &= 1 - \frac{\tilde{P}(s)B_o(s)}{\tilde{E}(s)A_o(s)} \\
&= 1 - \frac{\bar{P}(s)B_o(s)}{\tilde{E}(s)} \\
&= \frac{\tilde{E}(s) - \bar{P}(s)B_o(s)}{\tilde{E}(s)}
\end{aligned}
$$

- We thus require $A_o(s)$ to be a factor of $\tilde{E}(s) - \bar{P}(s)B_o(s)$.

Affine Parameterisation - Open Loop Unstable Model

- Which we can write as

$$\tilde{E}(s) - \bar{P}(s)B_o(s) = \bar{L}(s)A_o(s)$$

or

$$\bar{L}(s)A_o(s) + \bar{P}(s)B_o(s) = \tilde{E}(s) \qquad (1)$$

This is a standard pole assignment problem and choosing a desired $\tilde{E}(s)$ and the orders of $\bar{L}(s)$, $\bar{P}(s)$ will result in a unique solution.

Affine Parameterisation - Open Loop Unstable Model

- Which we can write as

$$\tilde{E}(s) - \bar{P}(s)B_o(s) = \bar{L}(s)A_o(s)$$

  or

$$\bar{L}(s)A_o(s) + \bar{P}(s)B_o(s) = \tilde{E}(s) \tag{2}$$

  This is a standard pole assignment problem and choosing a desired $\tilde{E}(s)$ and the orders of $\bar{L}(s)$, $\bar{P}(s)$ will result in a unique solution.

- Note: We set $\tilde{E}(s) = E(s)F(s)$.

Affine Parameterisation - Open Loop Unstable Model

- Now we have

$$Q(s) = \frac{\tilde{P}(s)}{\tilde{E}(s)}$$

then

$$\begin{aligned} C(s) &= \frac{Q(s)}{1 - Q(s)G_o(s)} \\ &= \frac{\tilde{P}(s)A_o(s)}{\tilde{E}(s)A_o(s) - \tilde{P}(s)B_o(s)} \\ &= \frac{\bar{P}(s)}{\bar{L}(s)} \end{aligned} \tag{3}$$

Affine Parameterisation - Open Loop Unstable Model

- The method to design a controller for the unstable open loop case outlined above is then:

Affine Parameterisation - Open Loop Unstable Model

- The method to design a controller for the unstable open loop case outlined above is then:

  Step 1 Choose $\tilde{E}(s)$. (closed loop poles)

Affine Parameterisation - Open Loop Unstable Model

- The method to design a controller for the unstable open loop case outlined above is then:

Step 1 Choose $\tilde{E}(s)$. (closed loop poles)

Step 2 Given $A_o(s)$, $B_o(s)$ & $\tilde{E}(s)$ solve $\bar{L}(s)A_o(s) + \bar{P}(s)B_o(s) = \tilde{E}(s)$ for a unique $\bar{L}(s)$ and $\bar{P}(s)$ that we denote as $L(s)$ and $P(s)$.

Affine Parameterisation - Open Loop Unstable Model

- The method to design a controller for the unstable open loop case outlined above is then:

Step 1 Choose $\tilde{E}(s)$. (closed loop poles)

Step 2 Given $A_o(s)$, $B_o(s)$ & $\tilde{E}(s)$ solve $\bar{L}(s)A_o(s) + \bar{P}(s)B_o(s) = \tilde{E}(s)$ for a unique $\bar{L}(s)$ and $\bar{P}(s)$ that we denote as $L(s)$ and $P(s)$.

Step 3

$$C(s) = \frac{P(s)}{L(s)}$$

Affine Parameterisation - Open Loop Unstable Model

- Given a solution to equation (1), standard results in algebra state that any other solution can be expressed as

$$\frac{\overline{L}(s)}{E(s)} = \frac{L(s)}{E(s)} - Q_u(s)\frac{B_o(s)}{E(s)} \tag{4}$$

$$\frac{\overline{P}(s)}{E(s)} = \frac{P(s)}{E(s)} + Q_u(s)\frac{A_o(s)}{E(s)} \tag{5}$$

where $Q_u(s)$ is any stable proper transfer function having no undesirable poles.

Affine Parameterisation - Open Loop Unstable Model

- Given a solution to equation (1), standard results in algebra state that any other solution can be expressed as

$$\frac{\bar{L}(s)}{E(s)} = \frac{L(s)}{E(s)} - Q_u(s)\frac{B_o(s)}{E(s)} \tag{6}$$

$$\frac{\bar{P}(s)}{E(s)} = \frac{P(s)}{E(s)} + Q_u(s)\frac{A_o(s)}{E(s)} \tag{7}$$

  where $Q_u(s)$ is any stable proper transfer function having no undesirable poles.

- Substitute (4) and (5) into (3) and we get

$$C(s) = \frac{\dfrac{P(s)}{E(s)} + Q_u(s)\dfrac{A_o(s)}{E(s)}}{\dfrac{L(s)}{E(s)} - Q_u(s)\dfrac{B_o(s)}{E(s)}}$$

Affine Parameterisation - Open Loop Unstable Model

- Now say $A_o(s)$ contains both desirable and undesirable poles,

$$A_o(s) = A_d(s)A_u(s)$$

then we can write $E(s) = A_d(s)\overline{E(s)}$ giving the pole assignment problem of

$$A_d(s)A_u(s)\bar{L}(s) + B_o(s)\bar{P}(s) = A_d(s)\bar{E}(s)F(s)$$

clearly this requires $\bar{P}(s) = \tilde{P}(s)A_d(s)$.

Affine Parameterisation - Open Loop Unstable Model

- Now say $A_o(s)$ contains both desirable and undesirable poles,

$$A_o(s) = A_d(s)A_u(s)$$

then we can write $E(s) = A_d(s)\overline{E(s)}$ giving the pole assignment problem of

$$A_d(s)A_u(s)\bar{L}(s) + B_o(s)\bar{P}(s) = A_d(s)\bar{E}(s)F(s)$$

clearly this requires $\bar{P}(s) = \tilde{P}(s)A_d(s)$.

- Therefore we have cancellations hence,

$$A_u(s)\bar{L}(s) + B_o(s)\tilde{P}(s) = \bar{E}(s)F(s) \tag{9}$$

Affine Parameterisation - Open Loop Unstable Model

- Now say $A_o(s)$ contains both desirable and undesirable poles,

$$A_o(s) = A_d(s)A_u(s)$$

then we can write $E(s) = A_d(s)\bar{E}(s)$ giving the pole assignment problem of

$$A_d(s)A_u(s)\bar{L}(s) + B_o(s)\bar{P}(s) = A_d(s)\bar{E}(s)F(s)$$

clearly this requires $\bar{P}(s) = \tilde{P}(s)A_d(s)$.

- Therefore we have cancellations hence,

$$A_u(s)\bar{L}(s) + B_o(s)\tilde{P}(s) = \bar{E}(s)F(s) \tag{10}$$

- For the unstable open loop case where the plant possesses some desirable poles then the same method as stated above applies except the pole assignment problem becomes equation (8).

Affine Parameterisation - Open Loop Unstable Model

- The nominal complementary sensitivity for the class of stabilising controllers is,

$$
\begin{aligned}
T_o(s) &= \frac{B_o(s)\overline{P}(s)}{A_o(s)\overline{L}(s) + B_o(s)\overline{P}(s)} \\
&= \frac{B_o(s)\left(\frac{P(s)}{E(s)} + Q_u(s)\frac{A_o(s)}{E(s)}\right)}{A_o(s)\left(\frac{L(s)}{E(s)} - Q_u(s)\frac{B_o(s)}{E(s)}\right) + B_o(s)\left(\frac{P(s)}{E(s)} + Q_u(s)\frac{A_o(s)}{E(s)}\right)} \\
&= \frac{\frac{B_o(s)P(s)}{E(s)} + \frac{Q_u(s)B_o(s)A_o(s)}{E(s)}}{\frac{A_o(s)L(s)}{E(s)} + \frac{B_o(s)P(s)}{E(s)}} \\
&= \frac{B_o(s)P(s) + Q_u(s)B_o(s)A_o(s)}{E(s)F(s)}
\end{aligned}
$$

Affine Parameterisation - Open Loop Unstable Model

- Given the controller parameterisation for unstable plants as

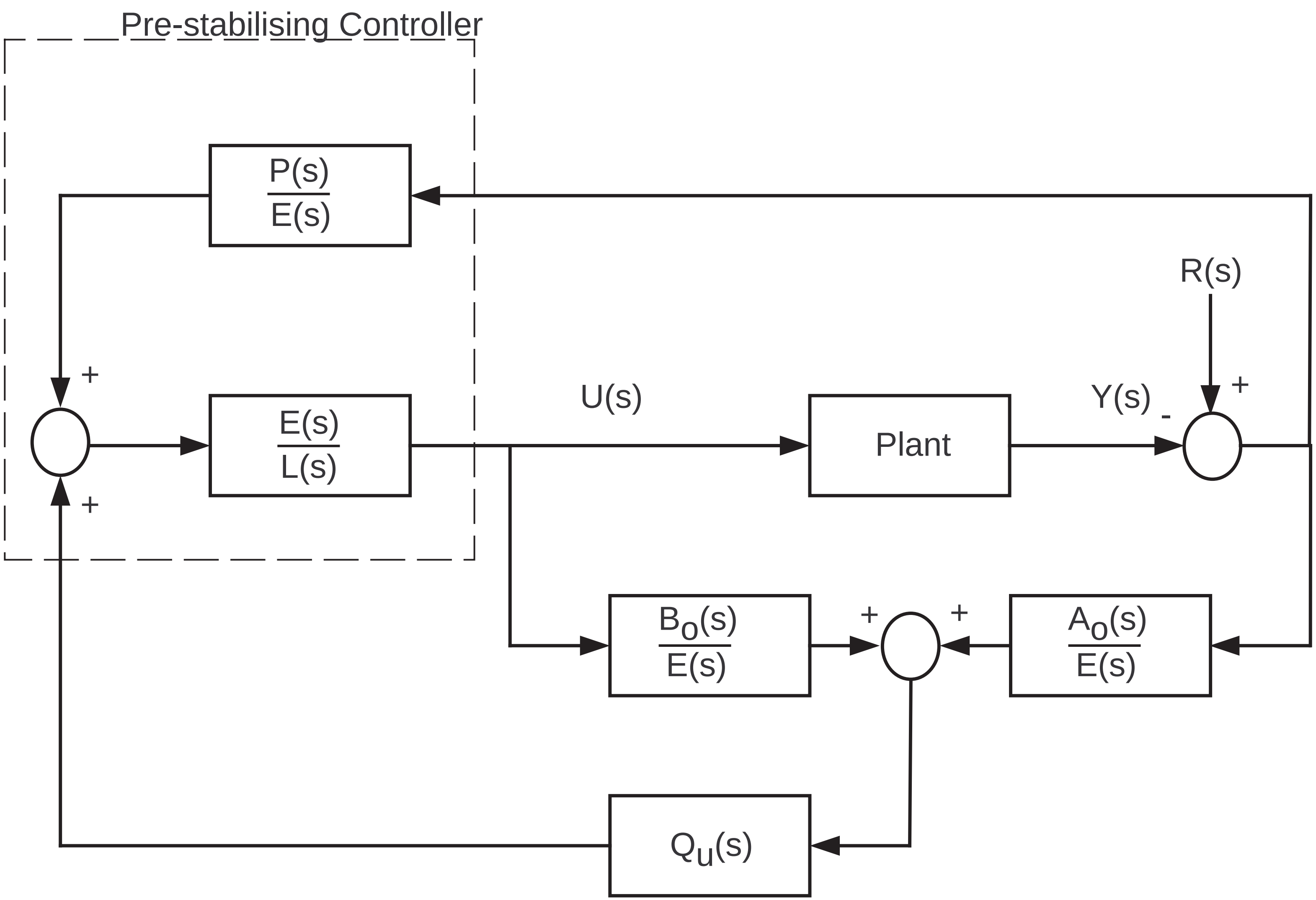
$$C(s) = \frac{\dfrac{P(s)}{E(s)} + Q_u(s)\dfrac{A_o(s)}{E(s)}}{\dfrac{L(s)}{E(s)} - Q_u(s)\dfrac{B_o(s)}{E(s)}}$$

  and that $u = Ce$ where e is the error signal we can write (note the argument s has been dropped)

$$\begin{aligned} u &= Ce \\ &= \left(\frac{\dfrac{P}{E} + Q_u\dfrac{A_o}{E}}{\dfrac{L}{E} - Q_u\dfrac{B_o}{E}} \right) e \\ &= \frac{E}{L}\left[\frac{P}{E}e + Q_u\frac{A_o}{E}e + Q_u\frac{B_o}{E}u \right] \end{aligned}$$

Affine Parameterisation - Open Loop Unstable Model

Affine Parameterisation: Unstable Open Loop Case



Affine Parameterisation - Open Loop Unstable Model

- The parametrisation as discussed above leads to the following parameterised version of the nominal sensitivities:

$$S_o(s) = \frac{A_o(s)L(s)}{E(s)F(s)} - Q_u(s)\frac{B_o(s)A_o(s)}{E(s)F(s)}$$

$$T_o(s) = \frac{B_o(s)P(s)}{E(s)F(s)} + Q_u(s)\frac{B_o(s)A_o(s)}{E(s)F(s)}$$

$$S_{io}(s) = \frac{B_o(s)L(s)}{E(s)F(s)} - Q_u(s)\frac{(B_o(s))^2}{E(s)F(s)}$$

$$S_{uo}(s) = \frac{A_o(s)P(s)}{E(s)F(s)} + Q_u(s)\frac{(A_o(s))^2}{E(s)F(s)}$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- What is Saturation?

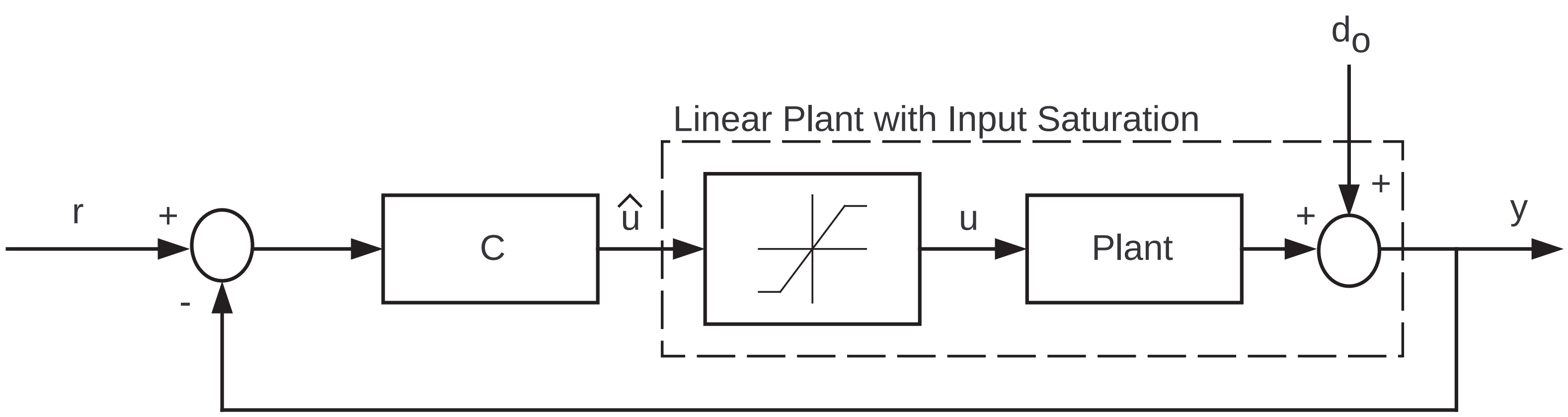
Saturation, Slew Rate Limitations and Anti-windup Schemes

- What is Saturation?
- **FACT:** All actuators saturate!

Saturation, Slew Rate Limitations and Anti-windup Schemes

- What is Saturation?
- **FACT:** All actuators saturate!
- **Objective:** How do we deal with the nonlinear affects of Saturation and Slew Rate Limitation? $\{u(t)$ and $\dot{u}(t)$ are limited$\}$

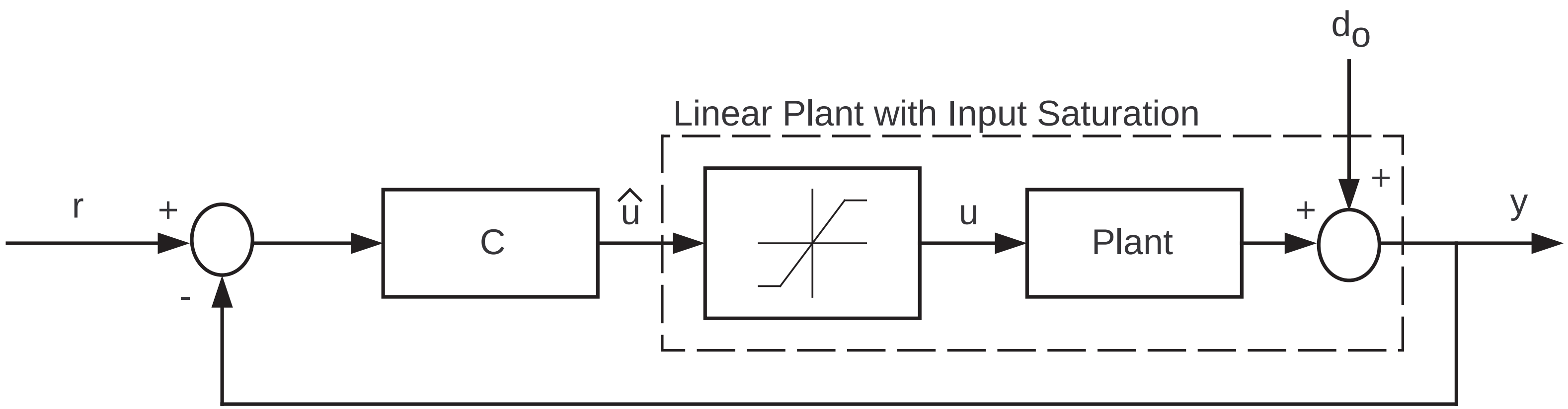
Control System with Saturation on the Plant Input



Saturation, Slew Rate Limitations and Anti-windup Schemes

- Saturation

$$u = sat\{\hat{u}\} = \begin{cases} u_{min} & \text{if } \hat{u} < u_{min}, \\ \hat{u} & \text{if } u_{min} \le \hat{u} \le u_{max}, \\ u_{max} & \text{if } \hat{u} > u_{max}. \end{cases}$$



Saturation, Slew Rate Limitations and Anti-windup Schemes

- Ignoring the presence of saturations can cause long undesirable transients in the closed loop.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Ignoring the presence of saturations can cause long undesirable transients in the closed loop.
- The transients are due to the controller states having ‘wound up’ to large values.

Saturation, Slew Rate Limitations and Anti-windup Schemes

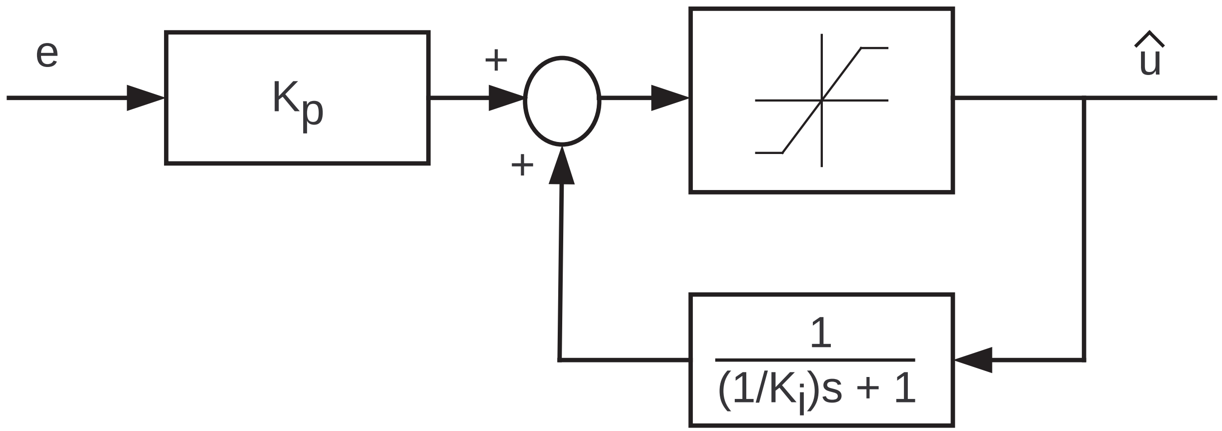
- Ignoring the presence of saturations can cause long undesirable transients in the closed loop.
- The transients are due to the controller states having ‘wound up’ to large values.
- In a PID controller, there is only one state that is subject to wind-up - the integrator state!

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Ignoring the presence of saturations can cause long undesirable transients in the closed loop.
- The transients are due to the controller states having ‘wound up’ to large values.
- In a PID controller, there is only one state that is subject to wind-up - the integrator state!
- Therefore in a PID controller, an anti-windup scheme involves limiting the integrator state in the some way.

Saturation, Slew Rate Limitations and Anti-windup Schemes

Anti-windup scheme for a PI controller



When the controller is in the linear region of the saturation

$$\frac{\hat{u}}{e} = K_p\left(1 + \frac{K_i}{s}\right).$$

Thus the state of the controller is updated only with the actual plant input $\hat{u}$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The key properties of an anti-windup scheme are:

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The key properties of an anti-windup scheme are:
    - The state of the controller should be driven by the actual (i.e. constrained) plant input.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The key properties of an anti-windup scheme are:
  - The state of the controller should be driven by the actual (i.e. constrained) plant input.
  - The states of the controller should have a stable realisation when driven by the actual plant input.

Saturation, Slew Rate Limitations and Anti-windup Schemes

The Problem of Windup in IMC

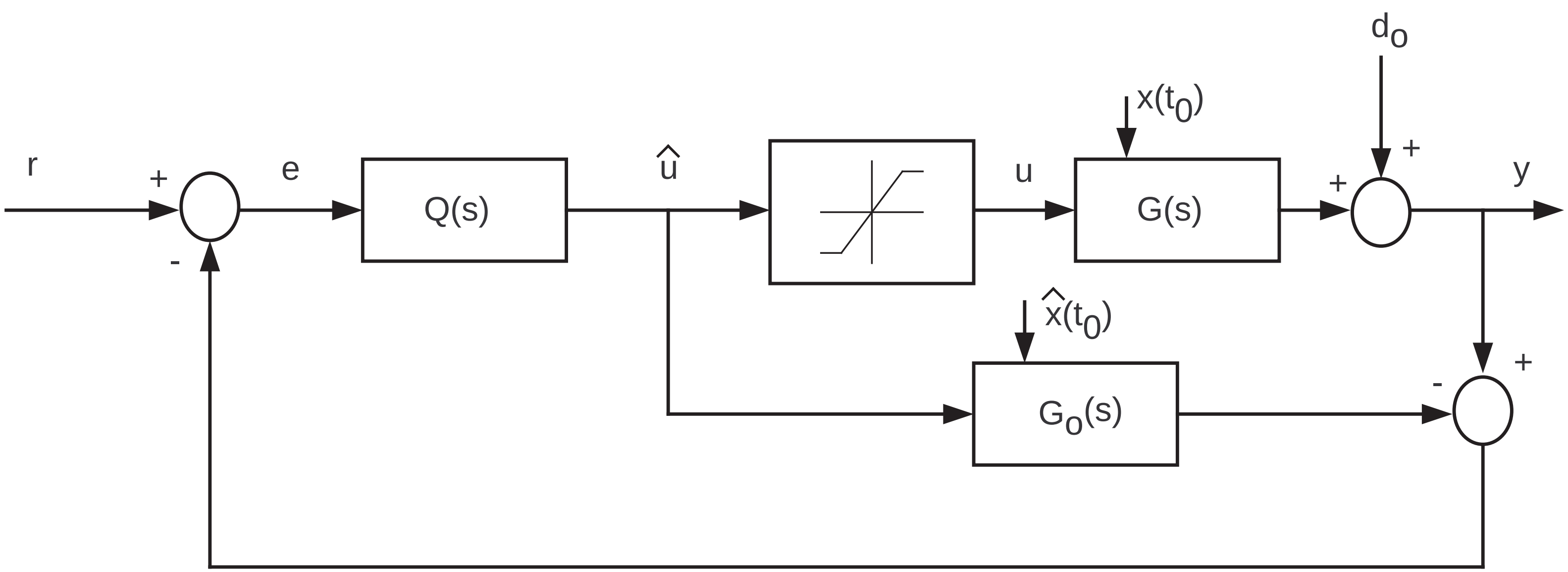
- We consider here open loop stable plants.

Saturation, Slew Rate Limitations and Anti-windup Schemes

The Problem of Windup in IMC

- We consider here open loop stable plants.
- The block diagram shows an IMC structure where saturation is included on the input to the plant. The initial states of the system are depicted as $x(t_0)$ and $\hat{x}(t_0)$. We also assume $G_o(s) = G(s)$.

Internal Model Control with Saturation on Plant Input



Saturation, Slew Rate Limitations and Anti-windup Schemes

- When $\hat{u}$ is in the linear region of the saturation,

$$y = Q(s)G_o(s)r + (1 - Q(s)G_o(s))d_o + IC(x - \hat{x})$$

where IC is a transfer function relating the initial conditions to the output.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- When $\hat{u}$ is in the linear region of the saturation,

$$y = Q(s)G_o(s)r + (1 - Q(s)G_o(s))d_o + IC(x - \hat{x})$$

  where IC is a transfer function relating the initial conditions to the output.

- Once the initial transient has decayed we see that the output is what we would expect, i.e. only a function of the reference and disturbance.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u

- Now, if saturation ends at time t_0,

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u
- Now, if saturation ends at time $t_{\mathbf{0}}$,
  - the loop is once again linear

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u

- Now, if saturation ends at time $t_\mathbf{0}$,
  - the loop is once again linear
  - G and G_o are both driven by $\hat{u}$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u

- Now, if saturation ends at time t_0,
  - the loop is once again linear
  - G and G_o are both driven by $\hat{u}$
  - **But** $x(t_0) \neq \hat{x}(t_0)$ since we had $\hat{u} \neq u$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u

- Now, if saturation ends at time t_0,
  - the loop is once again linear
  - G and G_o are both driven by $\hat{u}$
  - **But** $x(t_0) \neq \hat{x}(t_0)$ since we had $\hat{u} \neq u$

- Hence we will again see a transient in the output.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- During saturation (i.e. $\hat{u} < u_{min}$ or $\hat{u} > u_{max}$)
  - the loop is nonlinear
  - G_o is driven by $\hat{u}$
  - G is driven by u
- Now, if saturation ends at time t_0,
  - the loop is once again linear
  - G and G_o are both driven by $\hat{u}$
  - **But** $x(t_0) \neq \hat{x}(t_0)$ since we had $\hat{u} \neq u$
- Hence we will again see a transient in the output.
- This transient is due to the mismatched states in the plant and the controller, i.e. $x(t_0) \neq \hat{x}(t_0)$, and is called windup.

Saturation, Slew Rate Limitations and Anti-windup Schemes

Anti-windup for IMC

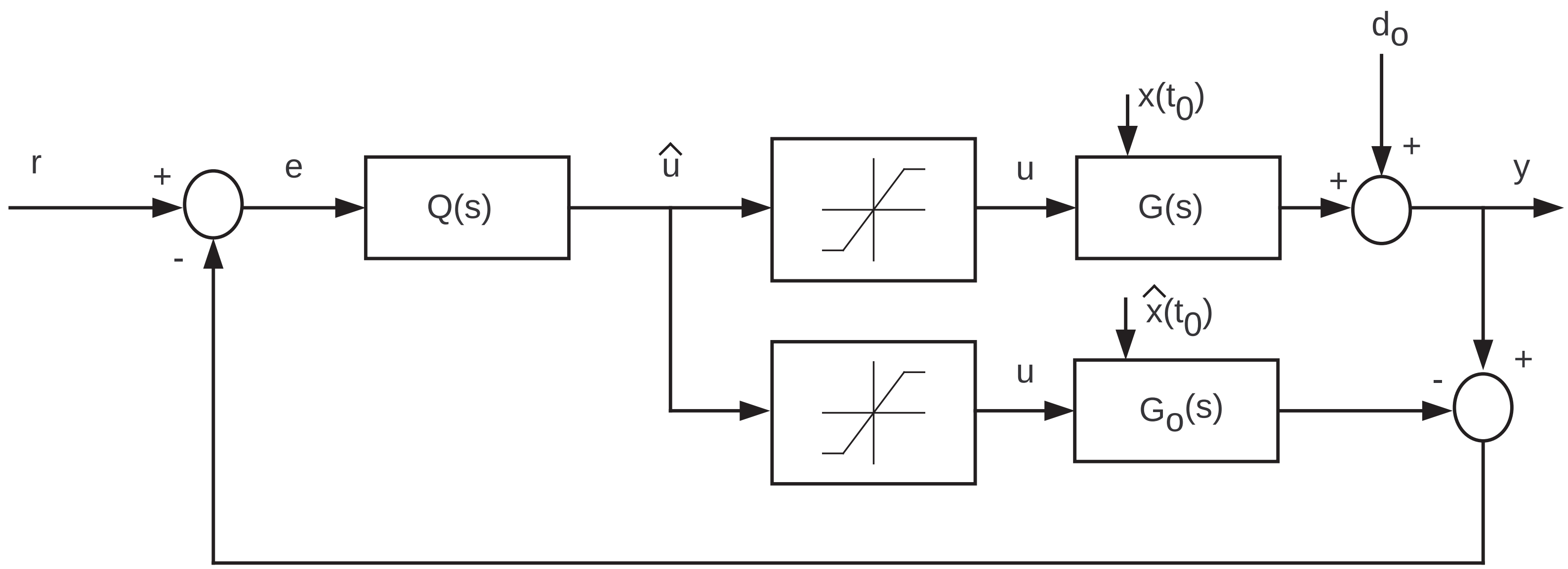
- Strategies aimed at reducing the effects of windup are, of course, called anti-windup.

Saturation, Slew Rate Limitations and Anti-windup Schemes

Anti-windup for IMC

- Strategies aimed at reducing the effects of windup are, of course, called anti-windup.

- **An initial idea for anti-windup** - Saturate the model input as well!

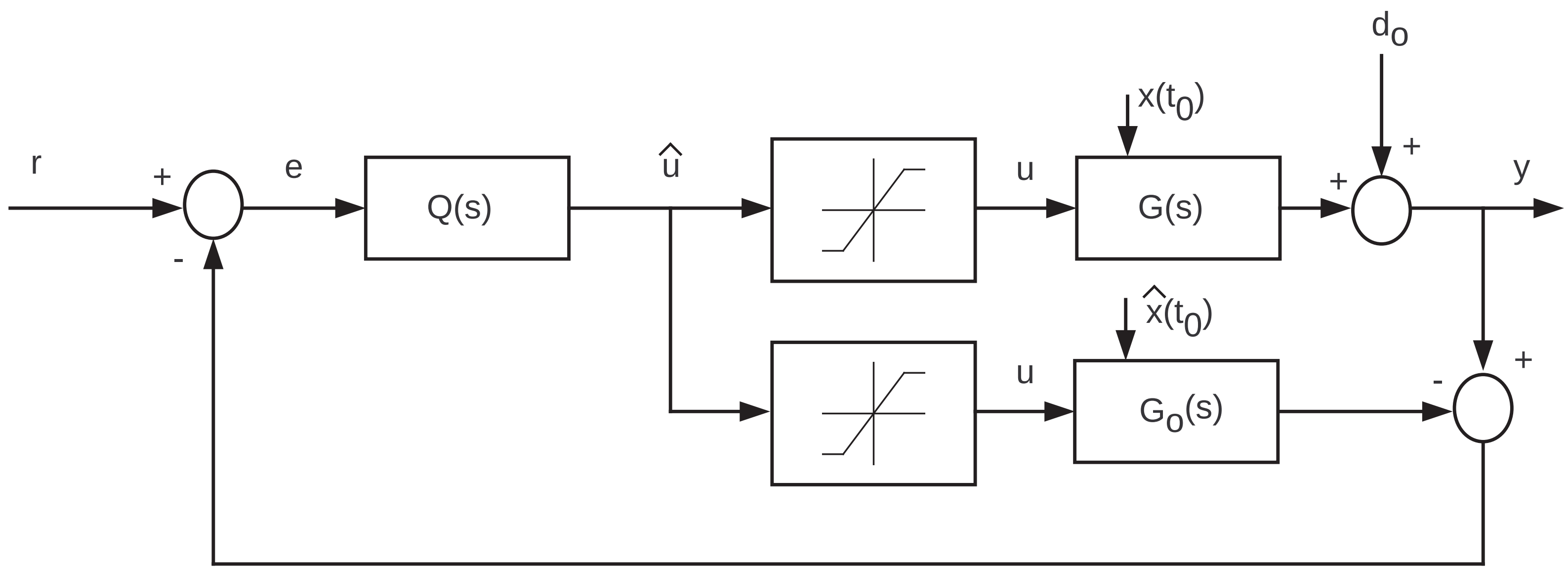


Saturation, Slew Rate Limitations and Anti-windup Schemes

Anti-windup for IMC

- Strategies aimed at reducing the effects of windup are, of course, called anti-windup.

- **An initial idea for anti-windup** - Saturate the model input as well!



- Now $x(t_0) = \hat{x}(t_0)$ during and after saturation.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The problems with this are:

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The problems with this are:
  1. The total system from $\hat{u}$ to y is nonlinear, so a linear compensator Q is no longer an inverse of the process.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The problems with this are:
  1. The total system from $\hat{u}$ to y is nonlinear, so a linear compensator Q is no longer an inverse of the process.
  2. e and $\hat{u}$ are completely independent of the saturation.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- The problems with this are:
  1. The total system from $\hat{u}$ to y is nonlinear, so a linear compensator Q is no longer an inverse of the process.
  2. e and $\hat{u}$ are completely independent of the saturation.

- For further insight into this problem, we digress slightly, and examine a feedback realisation of $Q(s)$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Assume, for simplicity, that $G_o(s)$ is minimum phase. Then

$$G_o^i = G_o^{-1}$$

and $Q(s) = F_q(s)G_o^i(s)$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Assume, for simplicity, that $G_o(s)$ is minimum phase. Then

$$G_o^i = G_o^{-1}$$

and $Q(s) = F_q(s)G_o^i(s)$.

- Lets also assume here that $F_q(s)$ is chosen to make $Q(s)$ bi-proper

$$Q(s) = \frac{n_n s^n + \cdot + n_1 s + n_0}{m_n s^n + \cdot + m_1 s + m_0}$$

where $n_n \neq 0$ and $m_n \neq 0$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Assume, for simplicity, that $G_o(s)$ is minimum phase. Then

$$G_o^i = G_o^{-1}$$

$$\text{and } Q(s) = F_q(s)G_o^i(s).$$

- Lets also assume here that $F_q(s)$ is chosen to make $Q(s)$ bi-proper

$$Q(s) = \frac{n_n s^n + \cdot + n_1 s + n_0}{m_n s^n + \cdot + m_1 s + m_0}$$

  where $n_n \neq 0$ and $m_n \neq 0$.

- The high frequency gain of $Q(s)$ is

$$q_\infty = \lim_{s \to \infty} Q(s) = \frac{n_n}{m_n} \neq 0.$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Assume, for simplicity, that $G_o(s)$ is minimum phase. Then

$$G_o^i = G_o^{-1}$$

and $Q(s) = F_q(s)G_o^i(s)$.

- Lets also assume here that $F_q(s)$ is chosen to make $Q(s)$ bi-proper

$$Q(s) = \frac{n_n s^n + \cdot + n_1 s + n_0}{m_n s^n + \cdot + m_1 s + m_0}$$

where $n_n \neq 0$ and $m_n \neq 0$.

- The high frequency gain of $Q(s)$ is

$$q_\infty = \lim_{s \to \infty} Q(s) = \frac{n_n}{m_n} \neq 0.$$

- We can then write

$$Q(s) = q_\infty + \overline{Q}(s)$$

where $\overline{Q}(s)$ is strictly proper.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now

$$
\begin{aligned}
\hat{u} &= Q(s)e \\
\text{then } e &= Q(s)^{-1}\hat{u} \\
&= \left(q_\infty^{-1} + \bar{Q}^{-1}(s)\right)\hat{u} \\
\therefore\ \hat{u} &= \frac{1}{q_\infty^{-1}}\left(e - \bar{Q}^{-1}(s)\hat{u}\right)
\end{aligned}
$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now

$$\hat{u} = Q(s)e$$

$$\text{then } e = Q(s)^{-1}\hat{u}$$

$$= \left(q_\infty^{-1} + \bar{Q}^{-1}(s)\right)\hat{u}$$

$$\therefore \hat{u} = \frac{1}{q_\infty^{-1}}\left(e - \bar{Q}^{-1}(s)\hat{u}\right)$$

- **Then,**

$$\bar{Q}^{-1}(s) = Q^{-1}(s) - q_\infty^{-1}$$

$$\text{then} \qquad \hat{u} = \frac{1}{q_\infty^{-1}}\left(e - \left(Q^{-1}(s) - q_\infty^{-1}\right)\hat{u}\right)$$

which is a feedback representation of $Q(s)$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now

$$\hat{u} = Q(s)e$$

$$\text{then } e = Q(s)^{-1}\hat{u}$$

$$= \left(q_\infty^{-1} + \bar{Q}^{-1}(s)\right)\hat{u}$$

$$\therefore \hat{u} = \frac{1}{q_\infty^{-1}}\left(e - \bar{Q}^{-1}(s)\hat{u}\right)$$

- **Then,**

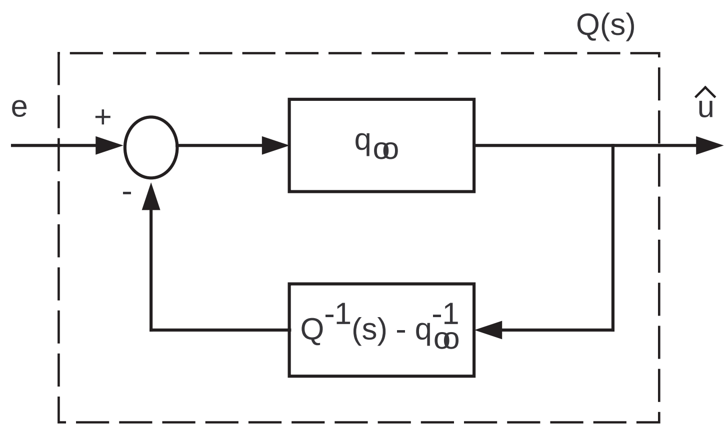
$$\bar{Q}^{-1}(s) = Q^{-1}(s) - q_\infty^{-1}$$

$$\text{then} \qquad \hat{u} = \frac{1}{q_\infty^{-1}}\left(e - \left(Q^{-1}(s) - q_\infty^{-1}\right)\hat{u}\right)$$

which is a feedback representation of $Q(s)$.

Saturation, Slew Rate Limitations and Anti-windup Schemes

Feedback Representation of $Q(s)$



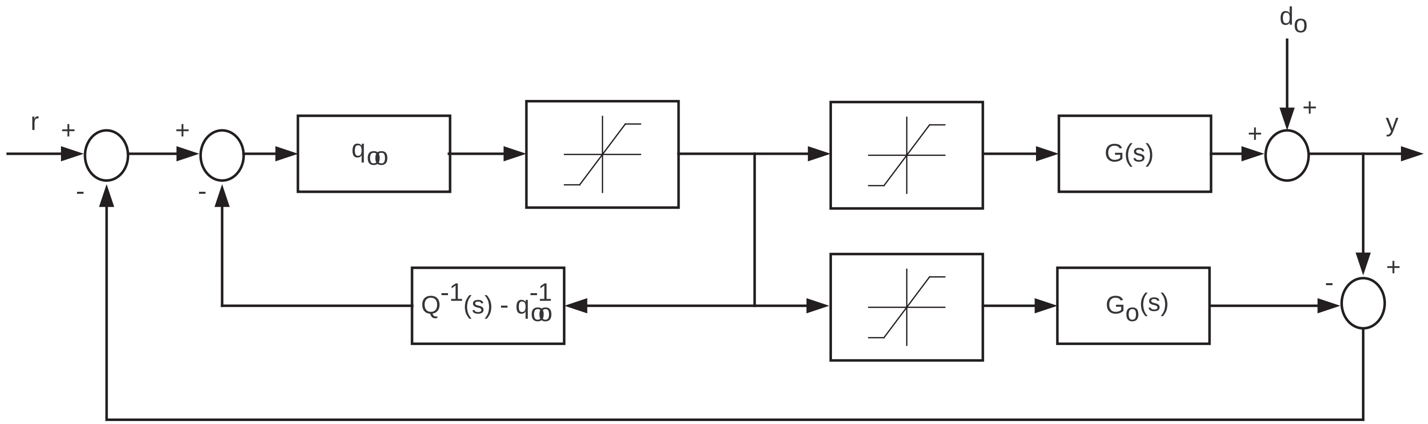
- Check:

$$
\begin{aligned}
\hat{u} &= q_\infty \left(e - \left(Q^{-1}(s) - q_\infty^{-1} \right) \hat{u} \right) \\
\frac{\hat{u}}{e} &= \frac{q_\infty}{1 + q_\infty Q^{-1}(s) - 1} \\
&= \frac{1}{Q^{-1}(s)} \\
&= Q(s)
\end{aligned}
$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now how can we use this to improve the anti-windup scheme for IMC.

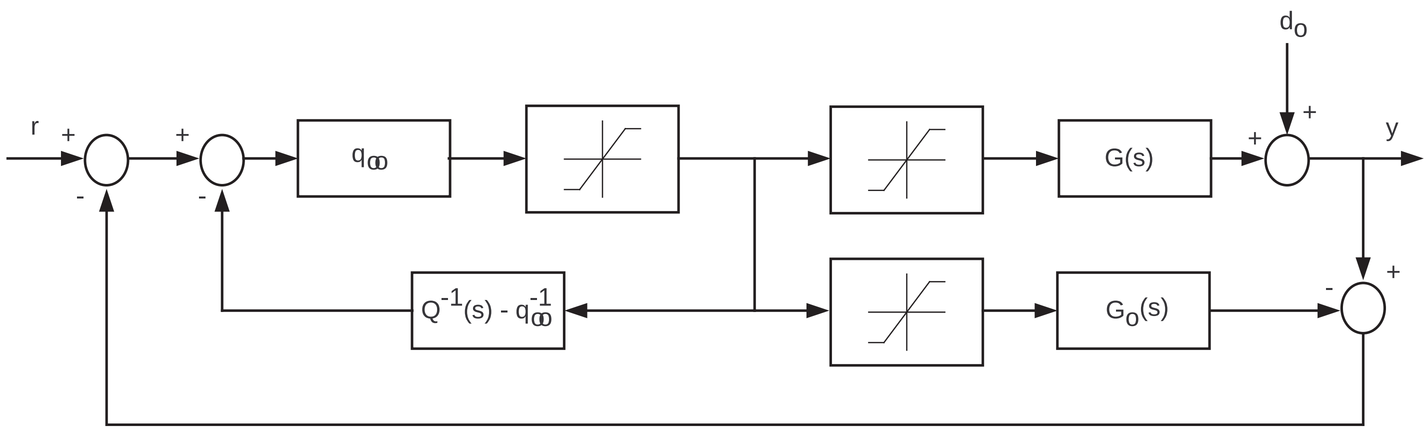
Anti-windup scheme utilising the feedback representation of $Q(s)$



Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now how can we use this to improve the anti-windup scheme for IMC.

Anti-windup scheme utilising the feedback representation of $Q(s)$

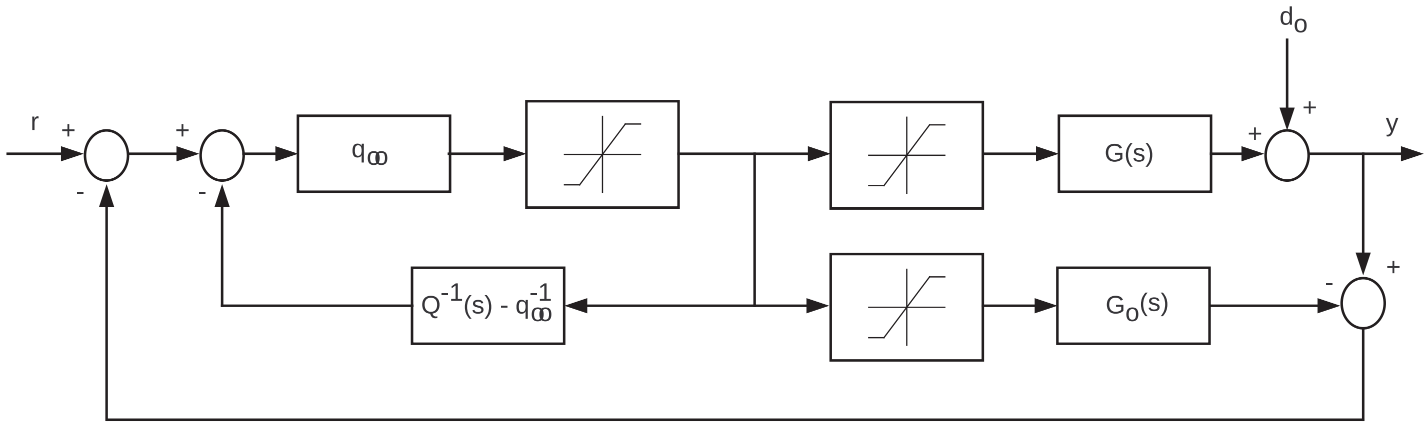


- In this scheme the controller state is updated based on the controller action which is effectively applied to the linear plant.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- Now how can we use this to improve the anti-windup scheme for IMC.

Anti-windup scheme utilising the feedback representation of $Q(s)$



- In this scheme the controller state is updated based on the controller action which is effectively applied to the linear plant.

- NOTE: For this to be feasible, we require $Q(s)$ to be stable and bi-proper.

Saturation, Slew Rate Limitations and Anti-windup Schemes

- It is easily seen that the saturation appearing in series with the nominal model, $G_o(s)$, is no longer required.

Saturation, Slew Rate Limitations and Anti-windup Schemes

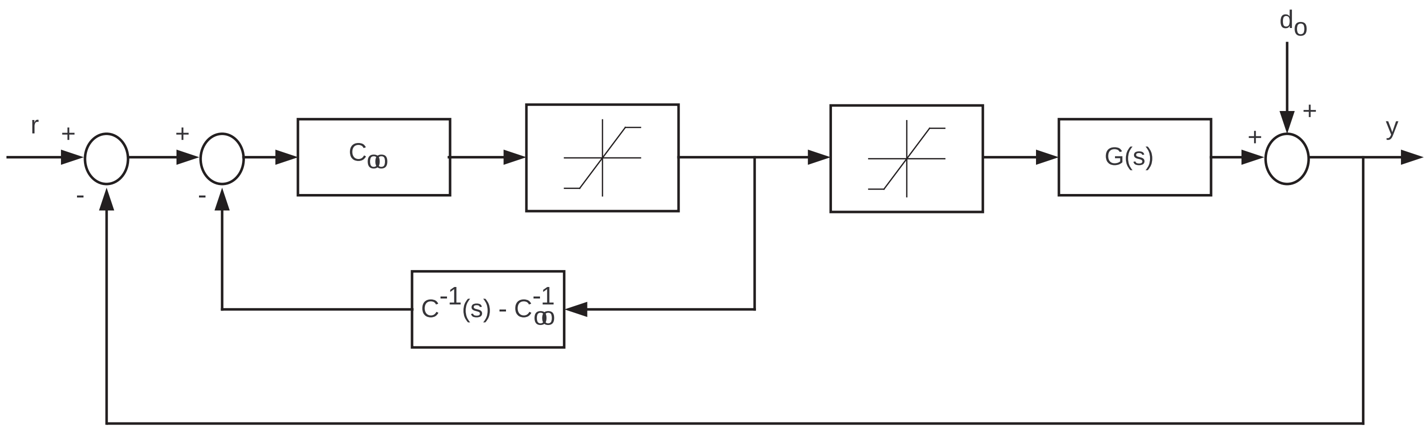
- It is easily seen that the saturation appearing in series with the nominal model, $G_o(s)$, is no longer required.
- The scheme can be generalised (including unstable plants).

Saturation, Slew Rate Limitations and Anti-windup Schemes

- It is easily seen that the saturation appearing in series with the nominal model, $G_o(s)$, is no longer required.
- The scheme can be generalised (including unstable plants).
- Let $C(s)$ be a bi-proper controller, then

$$C(s) = C_\infty + \bar{C}(s)$$

Anti-windup scheme for bi-proper $C(s)$



Saturation, Slew Rate Limitations and Anti-windup Schemes

Slew Rate Limitation

- Actuators may also be slew rate limited,

$$\dot{u}(t) = sat\{\dot{\hat{u}}(t)\} = \begin{cases} \sigma_{min} & \text{if } \dot{\hat{u}}(t) < \sigma_{min}, \\ \dot{\hat{u}}(t) & \text{if } \sigma_{min} \leq \dot{\hat{u}}(t) \leq \sigma_{max}, \\ \sigma_{max} & \text{if } \dot{\hat{u}}(t) > \sigma_{max}. \end{cases}$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

Slew Rate Limitation

- Actuators may also be slew rate limited,

$$\dot{u}(t) = sat\{\dot{\hat{u}}(t)\} = \begin{cases} \sigma_{min} & \text{if } \dot{\hat{u}}(t) < \sigma_{min}, \\ \dot{\hat{u}}(t) & \text{if } \sigma_{min} \leq \dot{\hat{u}}(t) \leq \sigma_{max}, \\ \sigma_{max} & \text{if } \dot{\hat{u}}(t) > \sigma_{max}. \end{cases}$$

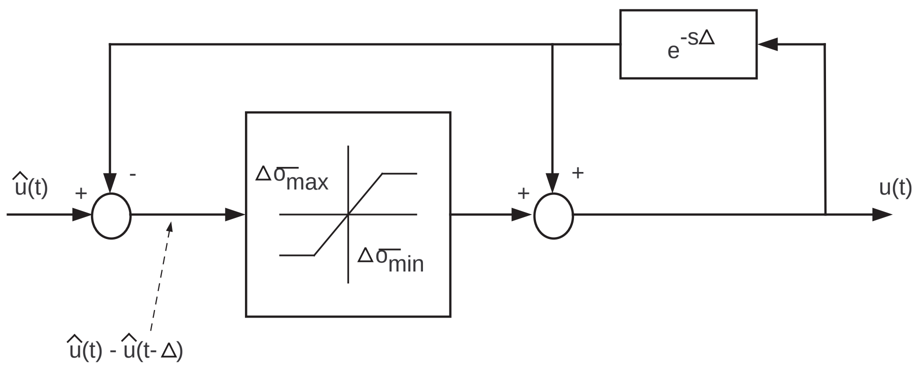
- We can approximate the derivative using Euler's Method

$$\dot{u}(t) \approx \frac{u(t) - u(t - \Delta)}{\Delta}$$

Saturation, Slew Rate Limitations and Anti-windup Schemes

- This leads to a Slew Rate Limiter (SRL)

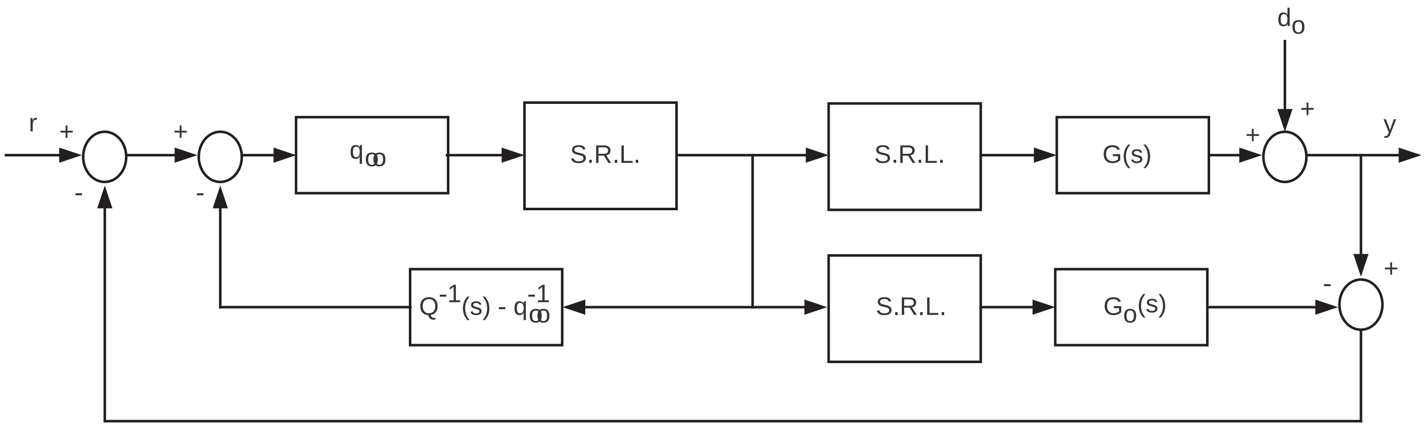
Model of Slew Rate Limiter



Saturation, Slew Rate Limitations and Anti-windup Schemes

- The previous idea for the anti-windup scheme can be applied to windup due to slew rate limitations.

Anti-windup Scheme for Slew Rate Limitation

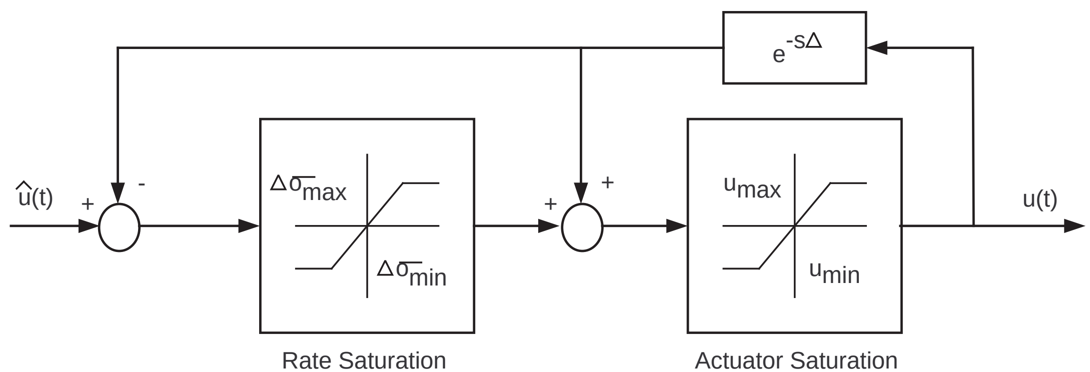


Saturation, Slew Rate Limitations and Anti-windup Schemes

Saturation and Slew Rate Limitation

- Saturation can be added to the slew rate limiter model.

Saturation and Slew Rate Limiter Model

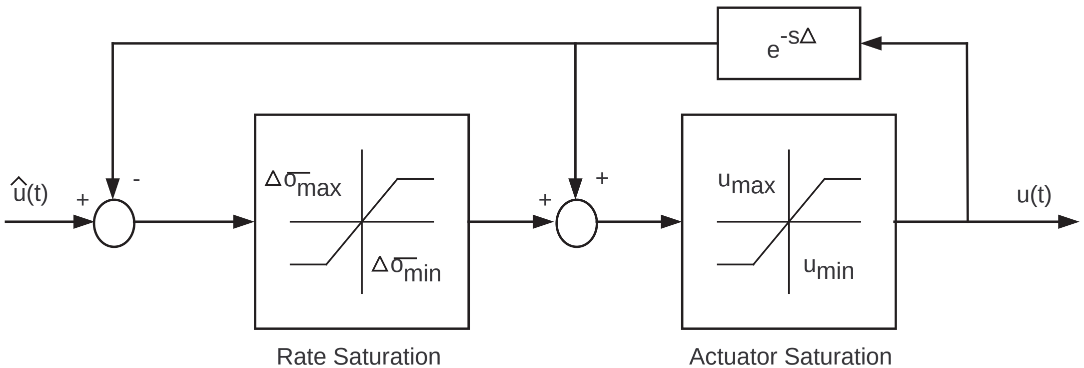


Saturation, Slew Rate Limitations and Anti-windup Schemes

Saturation and Slew Rate Limitation

- Saturation can be added to the slew rate limiter model.

Saturation and Slew Rate Limiter Model



- This model, which includes both saturation and slew rate limitation, can be incorporated in the same anti windup scheme as shown in the previous examples.