Introduction to Biomedical Engineering

Section 2: Control theory

Lecture 2.3: PID controller

POLYTECH
Peter the Great
St.Petersburg Polytechnic
University

OPEN
POLYTECH

Steady State Error

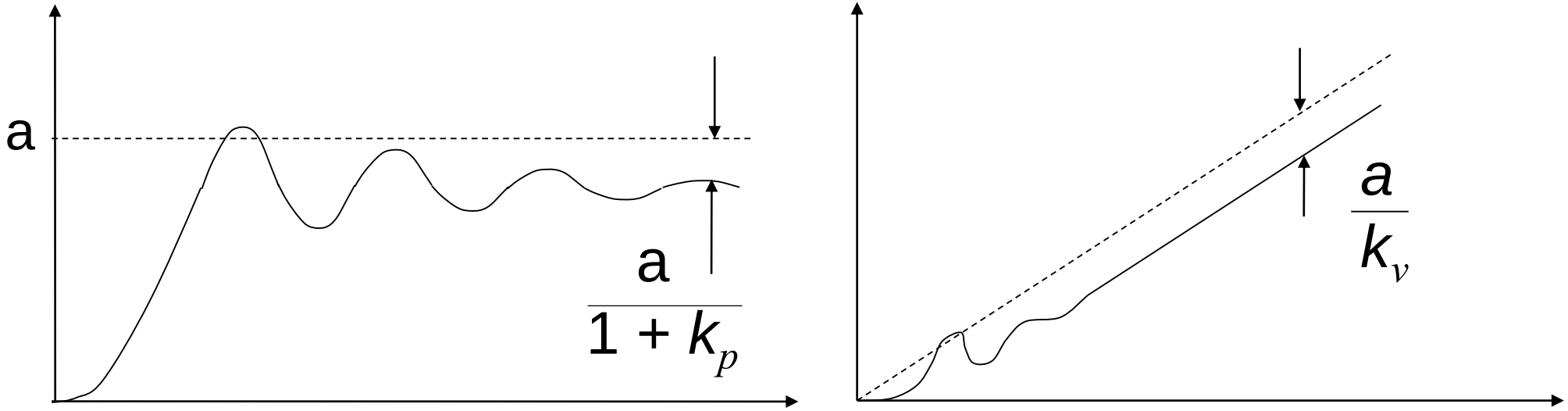
Previously we saw that the steady state performance of a control system can be found directly from its transfer function, using the Final Value Theorem applied to the error signal

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) \qquad E(s) = \frac{R(s)}{1+G(s)H(s)}$$

Where *E* is the error signal, *R* is the input signal, and *G* and *H* are the open loop and feedback loop gains.

Steady State Error

For a step and ramp inputs, the expression simplifies down to:



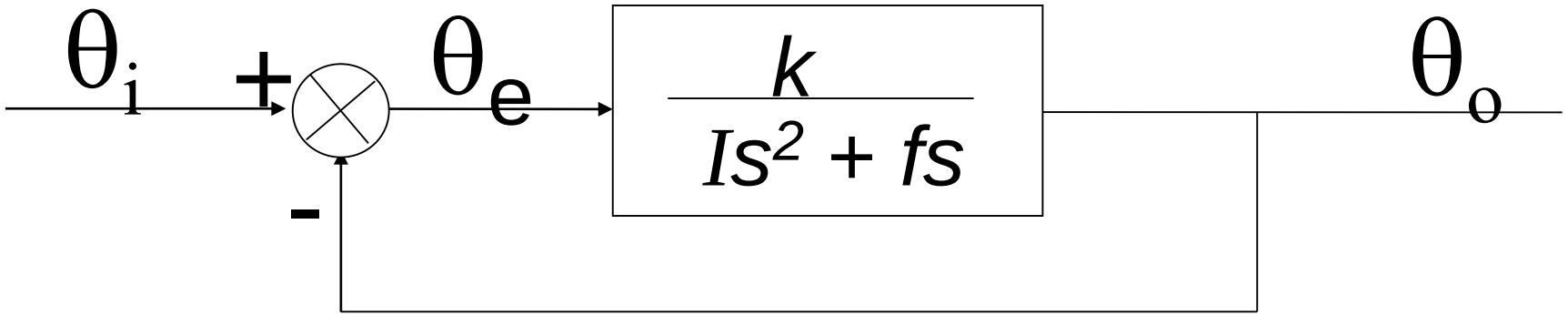
$$SSE = \frac{a}{1+k_p} \qquad SSVL = \frac{a}{k_v}$$

$$k_p = \lim_{s\to 0} G(s)H(s) \qquad k_v = \lim_{s\to 0} sG(s)H(s)$$

Both of which can be calculated from the plant and feedback gains, without having to calculate the closed loop transfer function

Servo Steady State Performance

For the DC Motor Servo we saw that for a step input the steady state error was zero, but was a finite value for a ramp input



$$k_p = \infty \rightarrow SSE = 0$$

$$k_p = \frac{k}{f} \rightarrow SSVL = \frac{af}{k}$$

System Type

Specifically, the positon error constant *kp* and velocity error constant *kv* depend upon **the number of poles at the origin** in the open loop system.

These correspond to roots in the denominator at s=0, and represent a pure integration.

$$\frac{1}{s}$$

System Type

Lets look more generally at the transfer function of our system and introduce the concept of system **type.** By factoring out any s terms from the denominator, we can write the transfer function in the following form:

$$G(s)H(s) = \frac{(s-z_1)(s-z_2)(s-z_3)\ldots}{s^p(s-\sigma_1)(s-\sigma_2)(s-\alpha_k+j\omega_k)(s-\alpha_k-j\omega_k)\ldots}$$

So, if there are *p poles at the origin*, the system is said to be a 'type *p*' system.

System Type IS NOT THE ORDER OF THE SYSTEM!!!

For example, the servo motor can be expressed as:

$$\frac{\Theta_o(s)}{\Theta_i(s)} = \frac{k}{s(Is+f)}$$

Type 1 System

System Type

Or the electro magnet system from lecture 2:

$$\frac{I(s)}{V(s)} = \frac{1}{(Ls + R)}$$

Type 0 System

Or the mass spring damper system

$$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + cs + k}$$

Type 0 System

System Type

Why introduce *more* jargon?

The system type quickly tells us the form of the SSE for our inputs without manipulation of block diagrams or converting to time domain. This gives an indication of the design of the controller

System Type – Calculating error

Essentially, for a zero error we want kp and kv to be infinite, or at least be a constant for a finite error, depending on our requirements

$$SSE = \frac{a}{1+k_p} \qquad SSVL = \frac{a}{k_v}$$

$$k_p = \lim_{s\to 0} G(s)H(s) \qquad k_v = \lim_{s\to 0} sG(s)H(s)$$

kv and kp are zero only when there is an s in every term of the denominator. Kv requires *two* as the formula requires the transfer function be multiplied by *s* before calculation.

System Type – Calculating error

The **position error constant** kp for a *type p* system is given by:

p > 0 $k_p = \lim_{s \to 0} G(s) = \infty$ no steady-state position error

p = 0 k_p is finite finite position error

The **velocity error constant** for a *type p* system is given by:

p > 1 $k_v = \lim_{s \to 0} sG(s) = \infty$ no velocity error

p = 1 k_v is finite steady state velocity lag

p = 0 $k_v = 0$ infinite lag (completely fails to track)

System Type – Calculating error

No. Integrators in denominator = system TYPE	Input type		
	Step $r(t) = a$ $R(s) = a/s$	Ramp $r(t) = at$ $R(s) = a/s^2$	Acceleration $r(t) = at^2/2$ $R(s) = a/s^3$
0	$e_{ss} = a/(1+k_p)$	$e_{ss} = \infty$	$e_{ss} = \infty$
1	$e_{ss} = 0$	$e_{ss} = a/k_v$	$e_{ss} = \infty$
2	$e_{ss} = 0$	$e_{ss} = 0$	$e_{ss} = a/k_a$

∞ Means system never settles

The more poles there are at the origin of the open-loop system, the better the steady-state tracking performance.

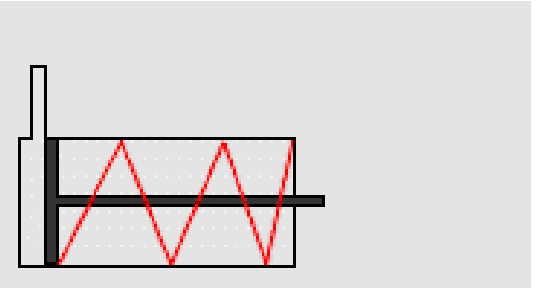
However, pure integrations have a highly destabilizing effect on the control system!

Type 0 Servo – Spring Return

Previously we have looked at servos with inertia and damping only, without any compliance/springs. This gives a Type-1 Transfer Function. However, when a spring is added, the system becomes Type 0.

Why would you add a spring?

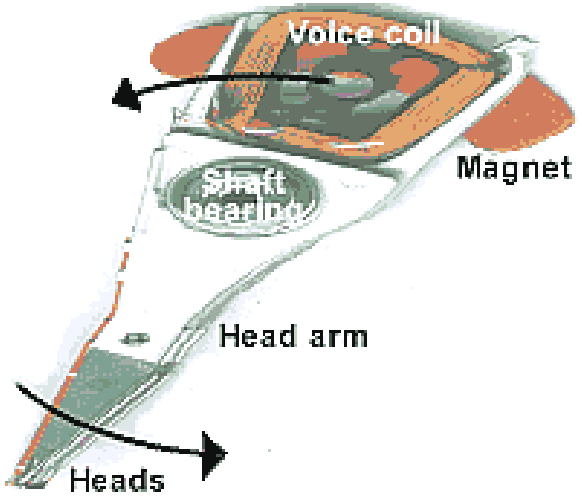
Some systems are not bidirectional e.g. a single acting piston in hydraulic system. Spring needed to return the piston to correct position in cycle



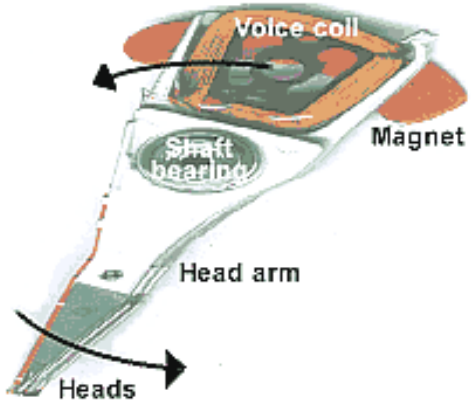
A spring is often added to servo actuators to return to a given position when system is off. HVAC systems use this for fire safety.

Type 0 Servo – Spring Return

The read and write heads in a hard drive are returned to the “home” position by a spring to prevent damage to the platter when power is off.



Type 0 Servo – Spring Return



The transfer function of this servo now becomes:

$$\sum T = I\frac{d^2\theta_o}{dt^2} \qquad I\frac{d^2\theta_o}{dt^2} = T_s + T_f + T_m$$

These oppose motion of motor

Time Domain $$I\frac{d^2\theta_o}{dt^2} = -k_s\theta_o - f\frac{d\theta_o}{dt} + k_s\theta_i$$

Laplace $$Is^2\Theta_o(s) = -k_s\Theta_o(s) - fs\Theta_0(s) + k_m\Theta_i(s)$$

Type 0 Servo – Spring Return

$$Is^2\Theta_o(s) + k_s\Theta_o(s) + fs\Theta_0(s) = k_m\Theta_i(s)$$

$$\Theta_o(s)\left(Is^2 + k_s + fs\right) = k_m\Theta_i(s)$$

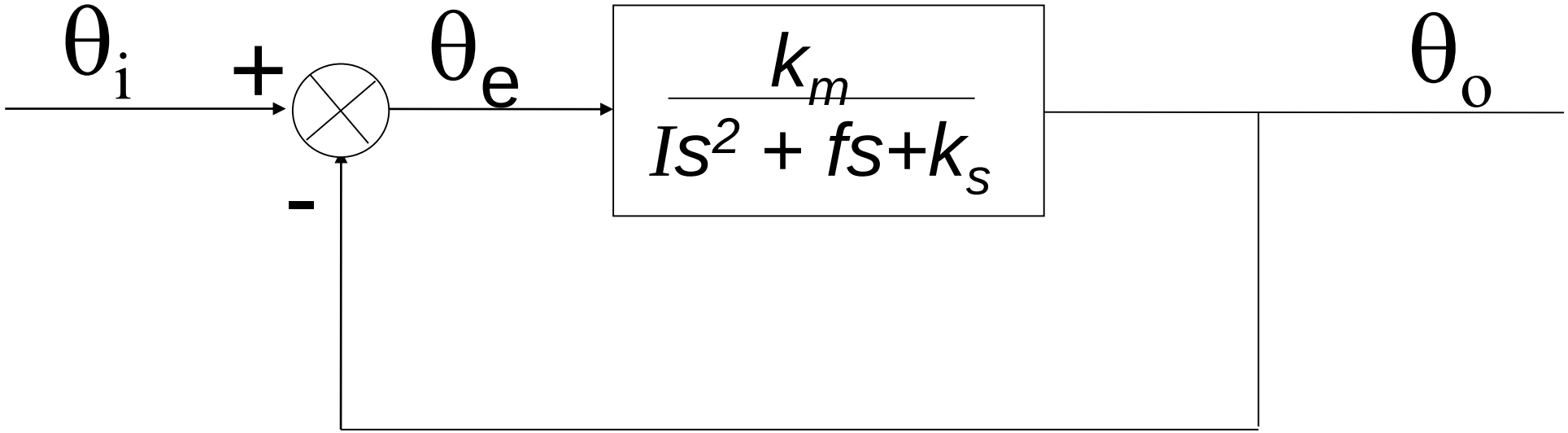
Transfer function

$$\frac{\Theta_o(s)}{\Theta_i(s)} = \frac{k_m}{Is^2 + fs + k_s} = G(s)$$

Cannot factorise – Type 0 System

Type 0 Servo – Spring Return

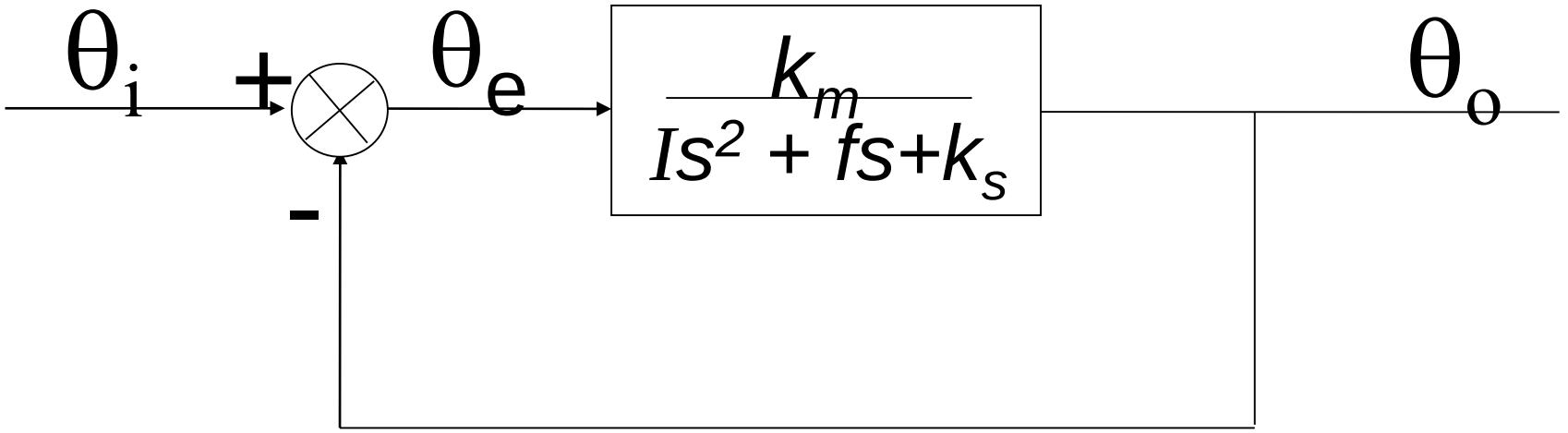
So the closed loop transfer function is now:



$$F(s) = \frac{G(s)}{1+G(s)} = \frac{k_m}{Is^2 + fs + k_s + k_m}$$

$$k_p = \lim_{s\to 0} G(s)H(s) = \frac{k_m}{Is^2 + fs + k_s} = \frac{k_m}{k_s}$$

Type 0 Servo – Spring Return



$$k_v = \lim_{s \to 0} sG(s)H(s) = \frac{k_m s}{Is^2 + fs + k_s} = \frac{0}{k_s}$$

$$SSE = \frac{a}{1 + k_p} = \frac{ak_s}{k_s + k_m} \qquad SSVL = \frac{a}{k_v} = \infty$$

Type 0 Servo – Spring Return

$$SSE = \frac{a}{1+k_p} = \frac{a\left(k_m + k_s\right)}{k_s + 2k_m} \quad SSVL = \frac{a}{k_v} = \infty$$

So now the servo will now only travel a fraction of the required distance for a step input, and will completely fail to track a ramp input, never settling to a steady state.

We need an improved controller for these systems!

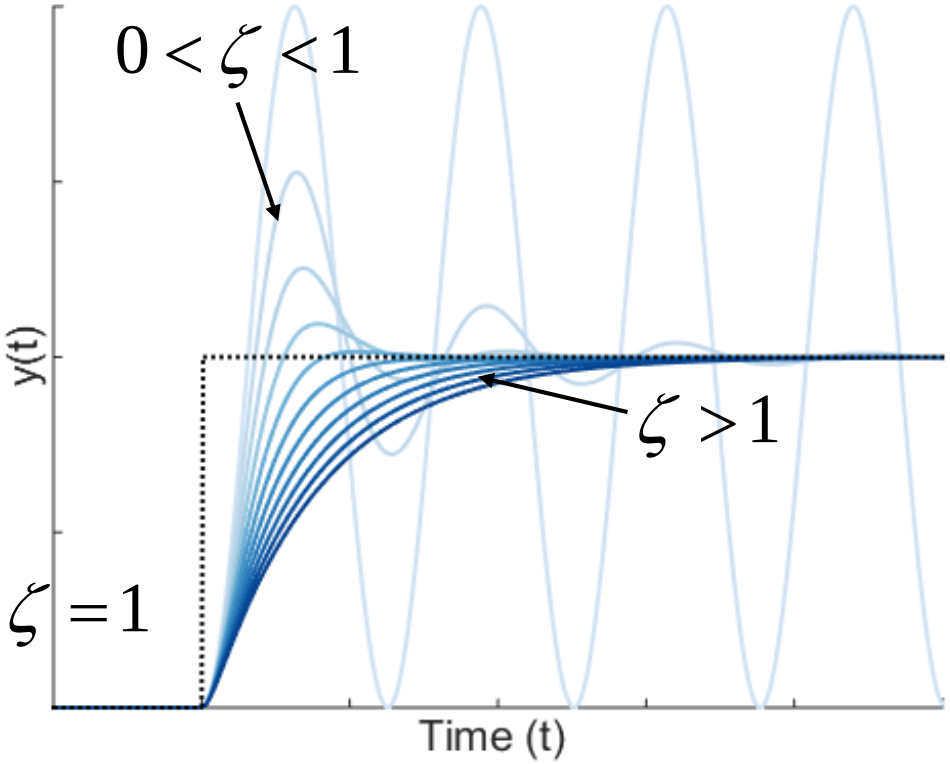
Proportional Control – Damping Ratio Trade off

We have seen the effect of damping ratio for our simple servo control system so far:

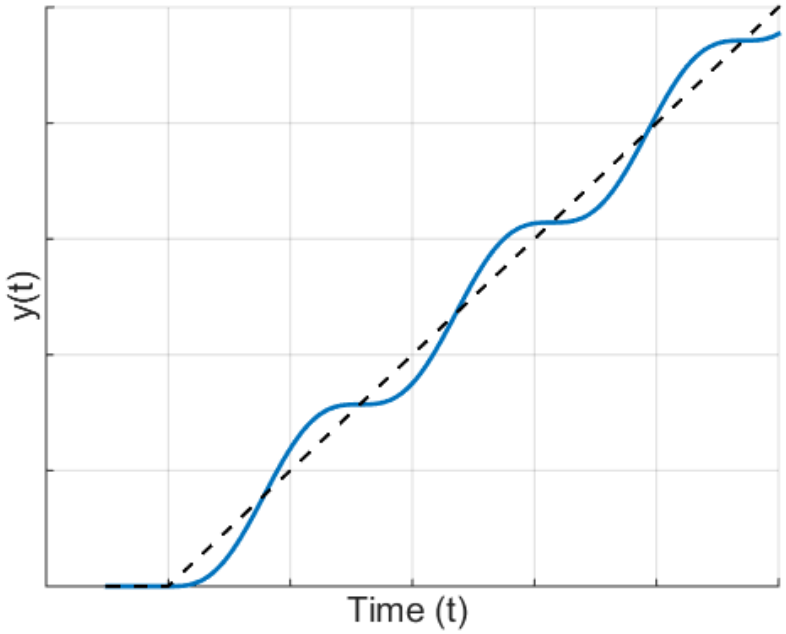
We have two conflicting requirements:

For transient response we want something around 0.6-0.8 to reduce overshoot and oscillations

$$\zeta = \frac{f}{2\sqrt{kI}}$$



Proportional Control – Damping Ratio Trade off



But for steady state response, we want the damping ratio as low as possible at the expense of oscillations

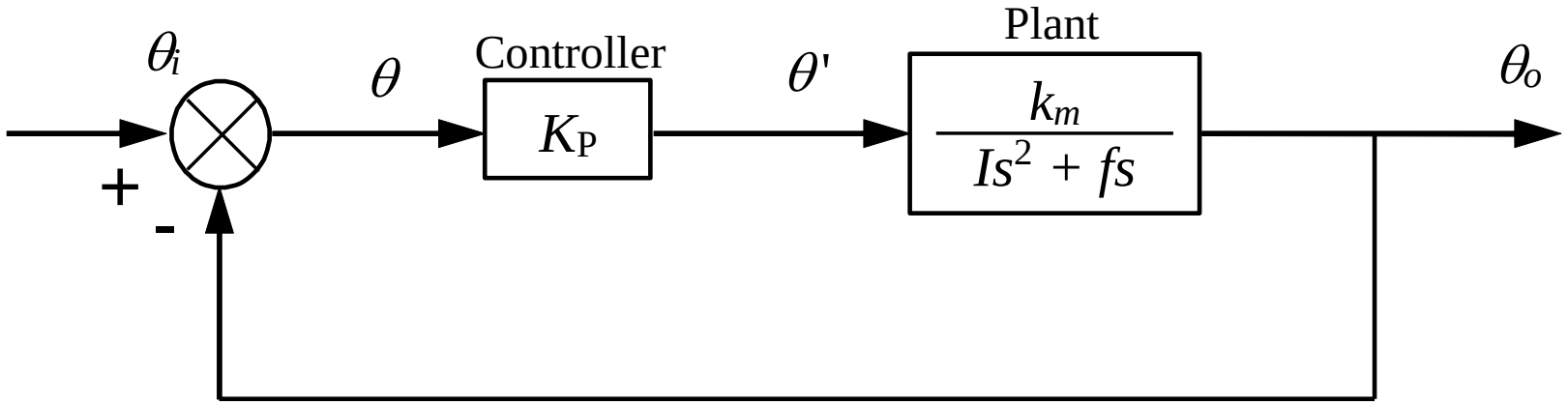
$$SSVL = \frac{af}{k} = -\frac{2\zeta a}{\omega_n}$$

For the our Type 1 servo this is trade off occurs if we want good ramp tracking, something we could possibly neglect in certain applications. However, the Type 0 servo SSVL renders it close to unusable in most control applications.

With current controller

Proportional Control – Damping Ratio Trade off

Commonly the challenge is to increase ζ, but with our current system, we do not have many options for adjustment



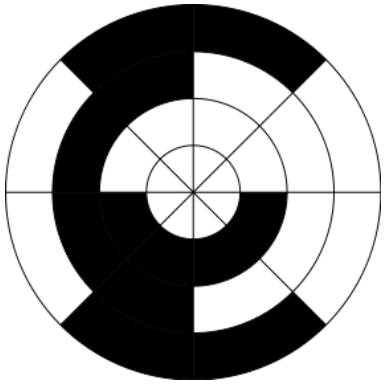
We can either increase our friction *f* but this increases wear on components reduces the life of the system. So that leaves the controller gain K_P to adjust, we can reduce it at the expense of steady state performance…

So we need a better controller!

Doing Better – Compensation methods

What the controller we have seen so far is known as a proportional controller, the control signal is directly proportional to the error signal, scaled by K_P. Compensation methods have been designed to improve **both** transient and steady state performance.

One of the earliest and simplest compensation methods is to consider not just the position but the velocity too.

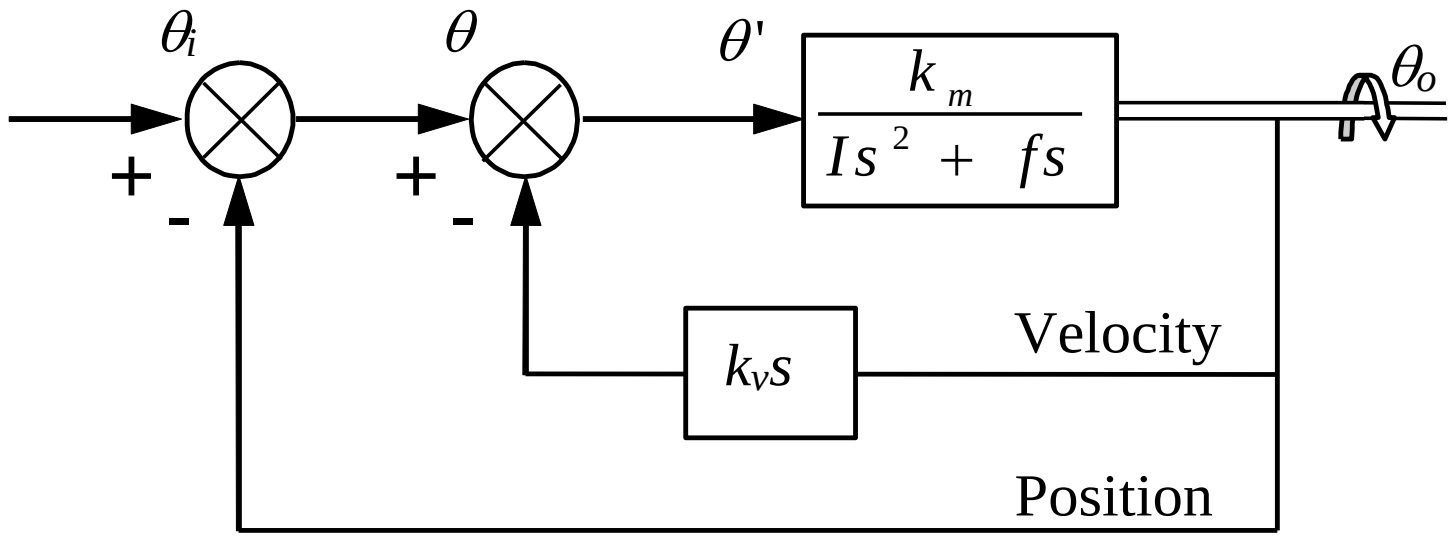


If we take the position from the shaft encoder, and also record the change over time, we can measure the speed.

Doing Better – Compensation methods

The block diagram of this is just *s* – as this is the differential operator, we also include another gain here k_v so we can choose the relative contributions of the two feedbacks.

So we have the following block diagram

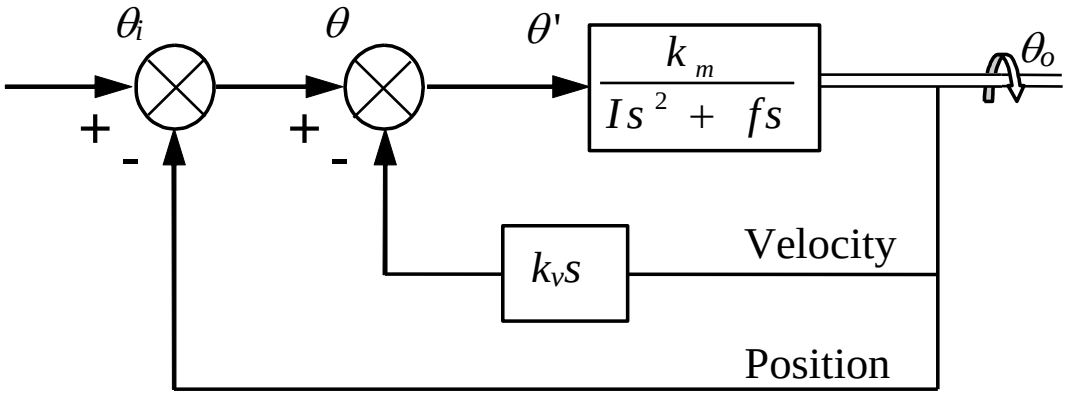


Inner comparator $\theta' = \theta - k_v s\theta_o$

Outer comparator $\theta = \theta_i - \theta_o$

$$\frac{\theta_o}{\theta_i} = \frac{k_m}{Is^2 + fs}$$

Doing Better – Compensation methods



If we calculate the new closed loop transfer function – work outwards!

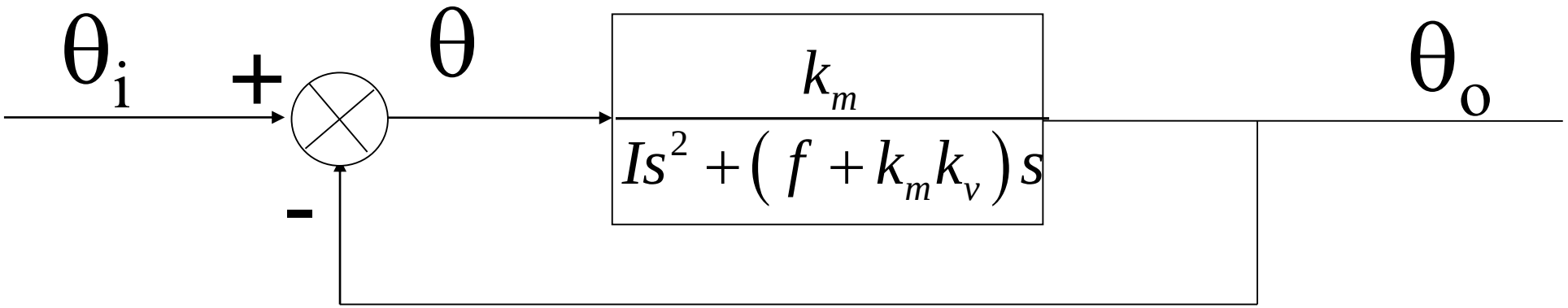
$$\frac{\theta_o}{\theta} = \frac{G}{1+GH} = F(s)$$

$$F(s) = \frac{k_m \big/ Is^2 + fs}{1 + \dfrac{k_m}{Is^2 + fs} k_v s}$$

$$= \frac{k_m}{Is^2 + fs + k_m k_v s}$$

$$= \frac{k_m}{Is^2 + (f + k_m k_v)s}$$

Doing Better – Compensation methods



Inner loop transfer function

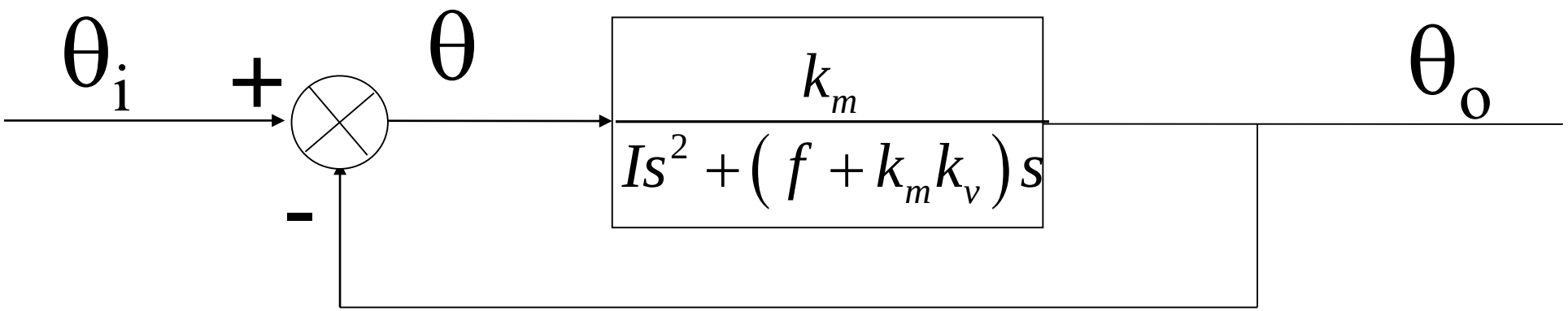
$$F(s) = \frac{k_m}{Is^2 + (f + k_m k_v)s}$$

Total Closed Loop transfer function

$$F_T(s) = \frac{F}{1+F}$$

$$F_T(s) = \frac{k_m}{Is^2 + (f + k_m k_v)s + k_m}$$

Doing Better – Compensation methods



$$F_T(s) = \frac{k_m / I}{s^2 + \left(\left(f + k_m k_v \right) / I \right) s + k_m / I}$$

$$\omega_n = \sqrt{\frac{k_m}{I}} \qquad \zeta = \frac{f + k_m k_v}{2\sqrt{kI}}$$

Doing Better – Compensation methods

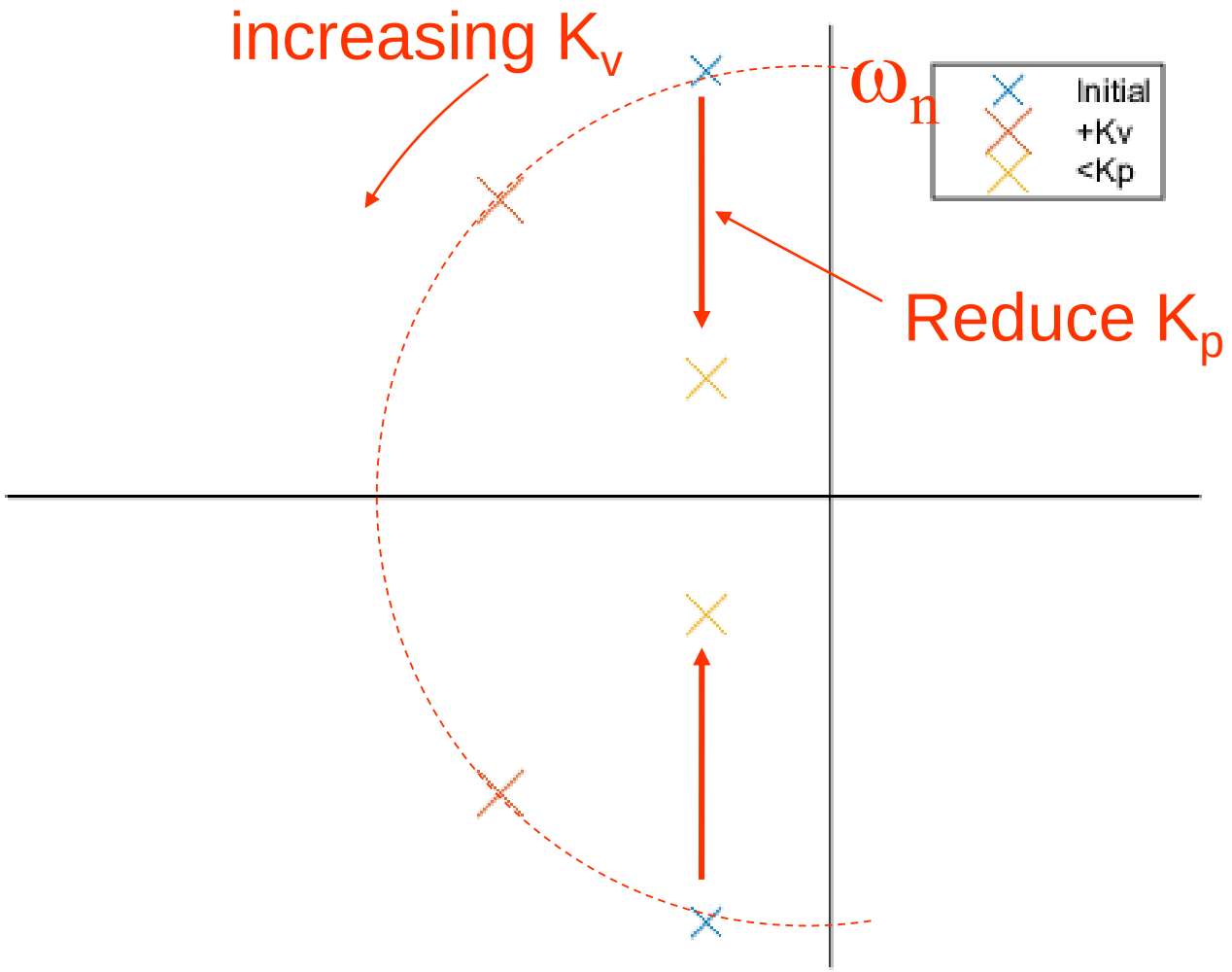
$$\omega_n = \sqrt{\frac{k_m}{I}} \qquad \zeta = \frac{f + k_m k_v}{2\sqrt{kI}}$$

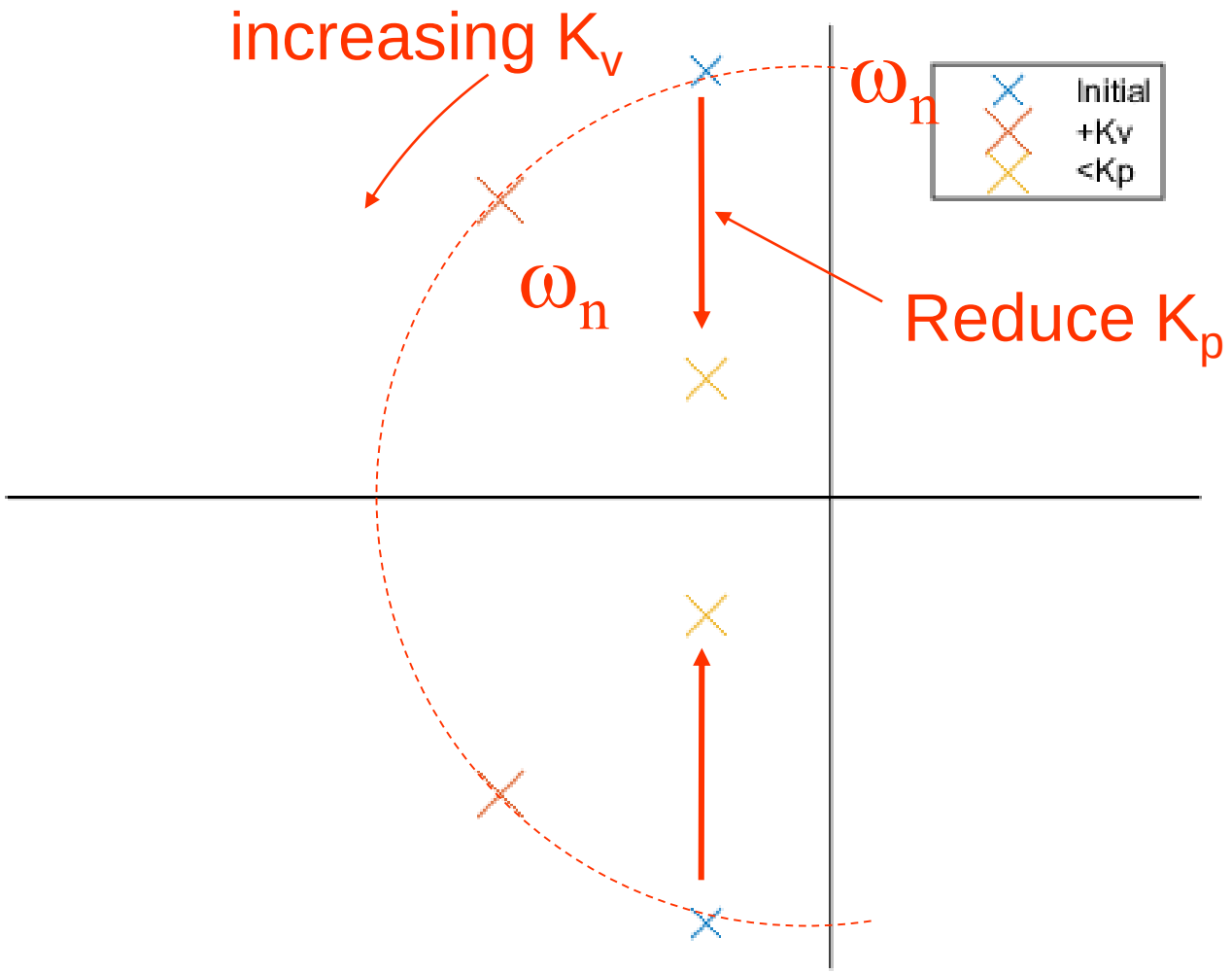
So by adding the extra loop, we are able to keep the natural frequency the same, *but increase the damping ratio*.

This means we can greatly improve the transient response compared to proportional control, where we can only scale the motor gain *km* through adjusting the parameter K_P

This is evident when considering the position of the poles of the system on the S-plane

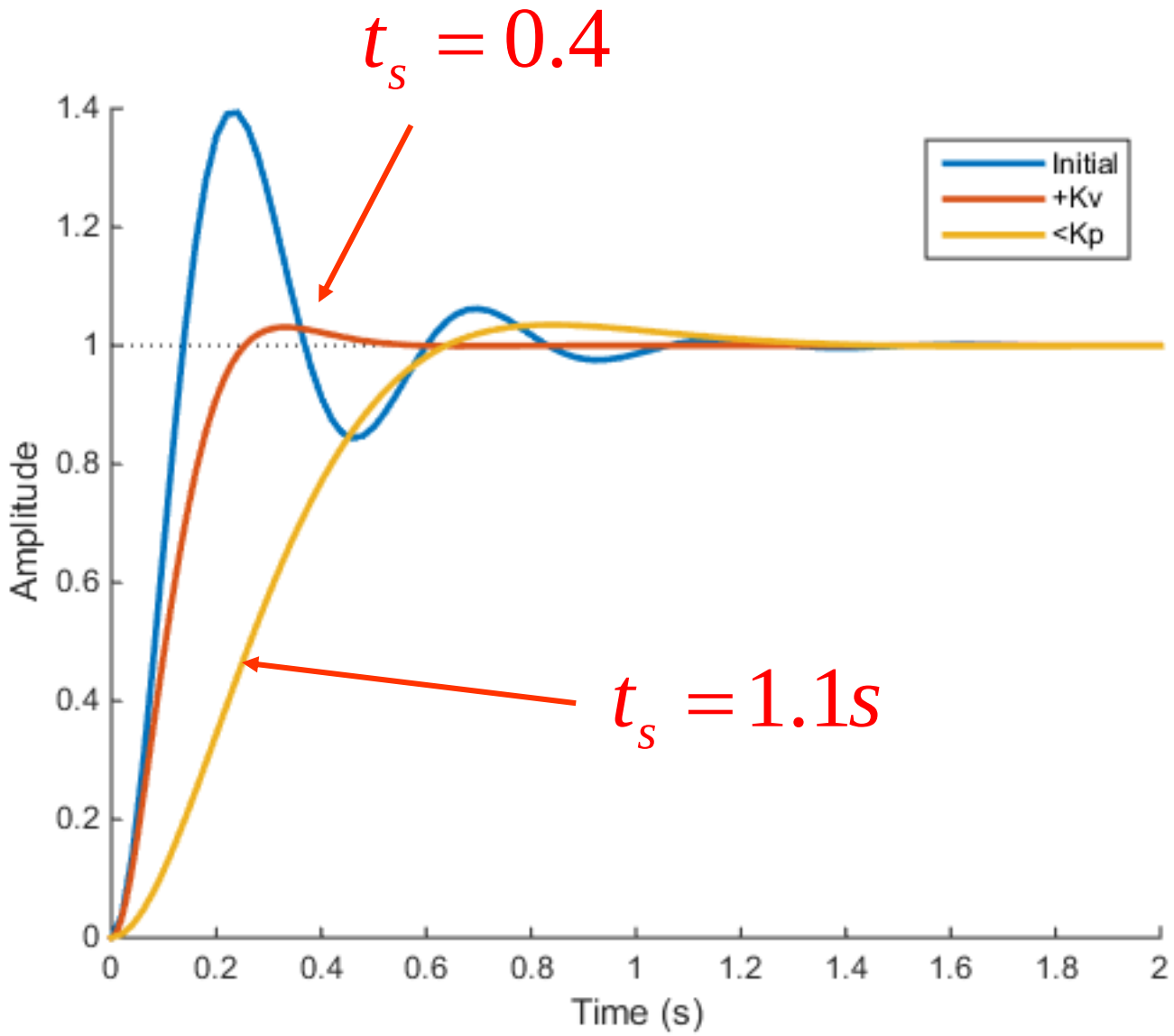
Adjusting the Motor gain using proportional control (K_P), can only move the poles closer to the real axis. But using velocity feedback (K_V), it is possible to move the poles away from the imaginary axis, thus reducing the decay time.





In this example, the initial zeta was ζ=0.3, in order to achieve the practical estimate of ζ=0.7 with proportional control, we have to increase both the rise time and settling time.

However, with velocity feedback we can achieve a vastly improved transient response.



Velocity feedback - disadvantages

This improved transient response comes at the expense of the steady state error, the expression for the SSVL becomes

$$k_v = \lim_{s\to 0} sF(s) = \frac{k_m}{Is^2 + (f + k_m k_v)s} = \frac{k_m}{f + k_m k_v}$$

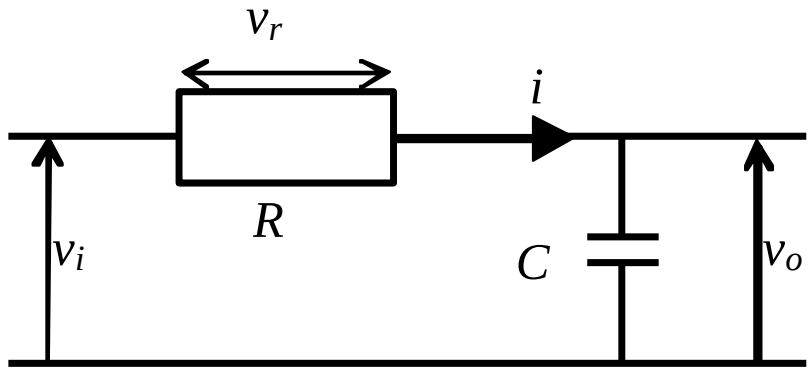
$$SSVL = a\frac{f + k_m k_v}{k}$$ Compared to $$\frac{af}{k}$$ originally

Using this feedback system allows for the designer to trade off the transient and steady state response dependent upon the requirements of the system.

Velocity feedback - disadvantages

However, using the velocity in this manner is very sensitive to errors or noise from the sensor, which give large instantaneous changes in the error signal.

To combat this, a low pass filter is normally employed in the feedback loop before the derivative term.



$$= \frac{1}{RCs + 1}$$

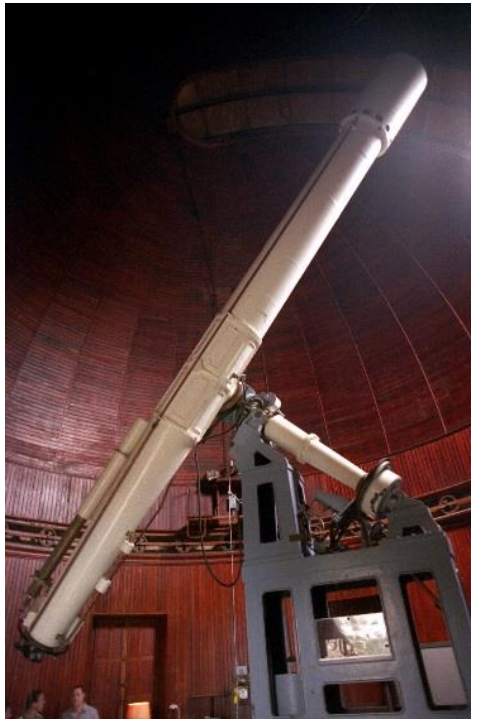
See lecture 2

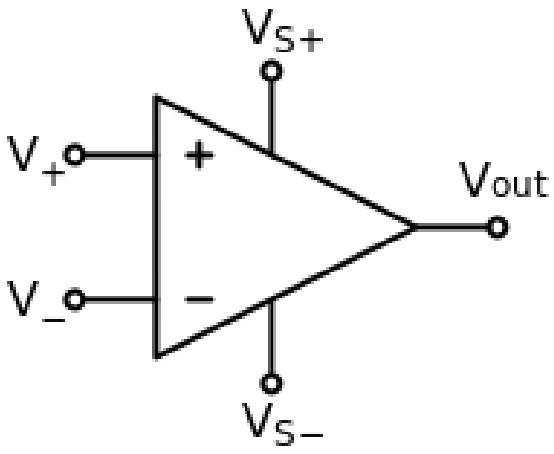
This makes the response more complex, and slightly reduces the improvements in transient response obtained

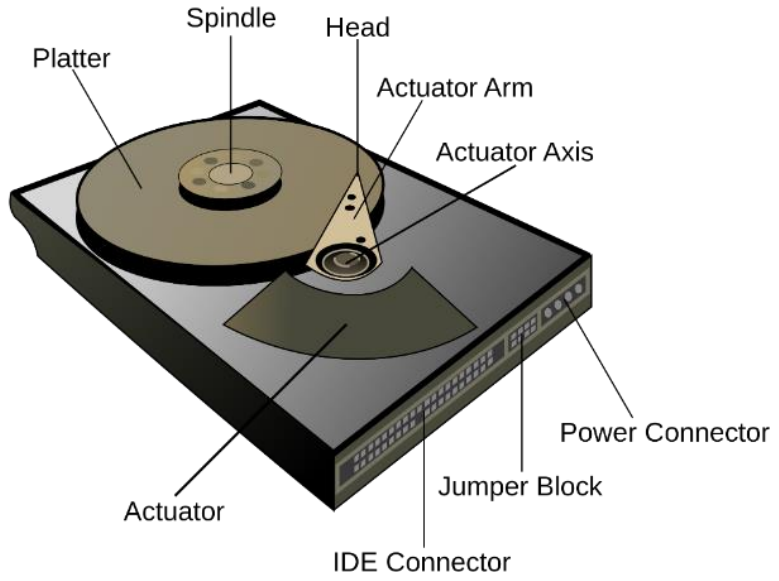
Minor Loop Feedback

Despite these drawbacks, velocity feedback, an example of a class of systems known more generally as *minor loop feedback,* is still effective in a number of applications.

Where transient response is critical



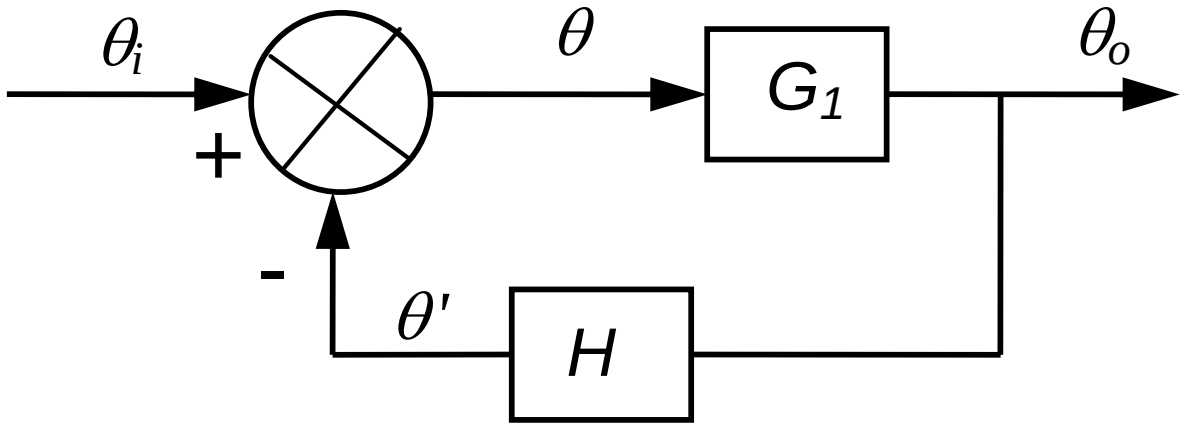




PID Control

This velocity feedback is rather specific to servomechanisms with a shaft encoder.

It is not always possible to have a sensor or a measurement from which both the variable and its derivative can be acquired simultaneously.



Instead, we can consider what happens to the *error* across time – something necessarily present in all closed loop systems!

PID Control

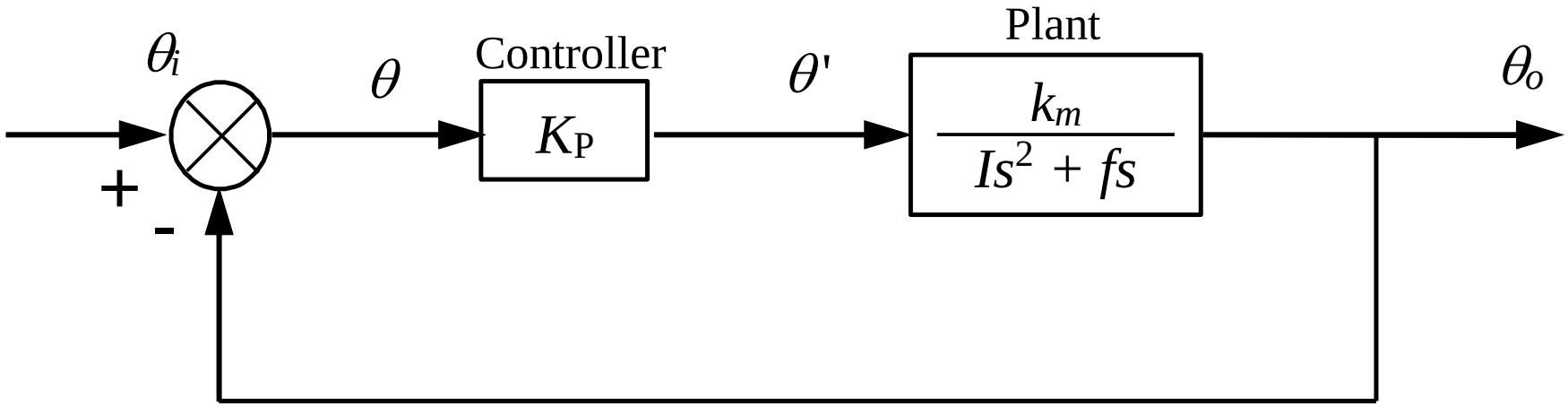
Doing so leads us to the **Proportional, Integral and Derivative (PID) Controllers**.

These controllers consist of three terms, all applied to the error signal. They are *incredibly* common in engineering, accounting for nearly 90%* of all use cases.

First, lets consider each of the three terms individually, and see how they change the closed loop transfer function

* Totally made up statistic

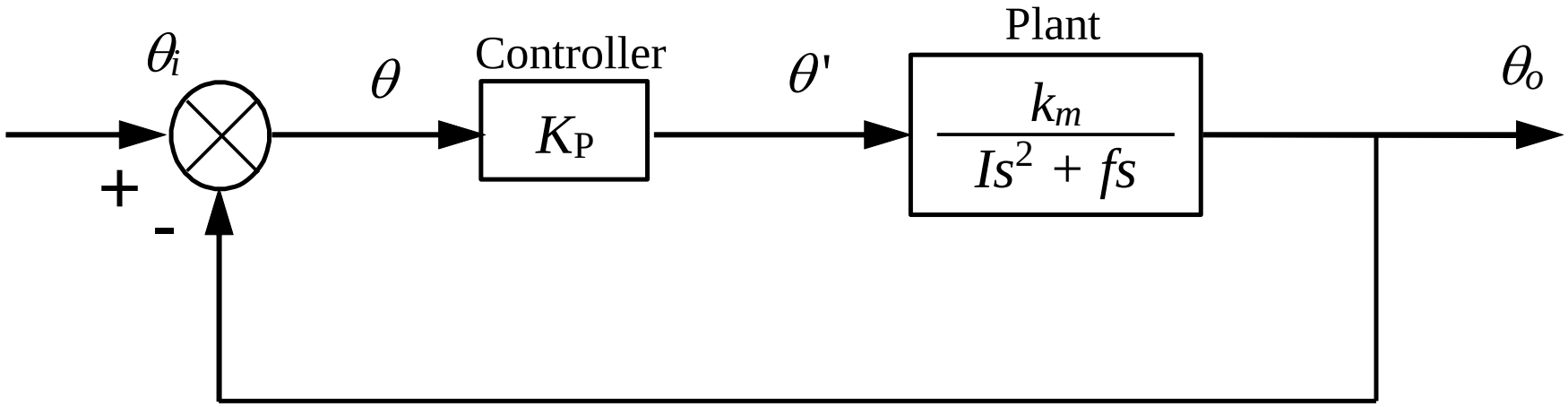
Proportional Error



$$F(s) = \frac{G(s)}{1 + G(s)} = \frac{K_P k_m}{Is^2 + fs + K_P k_m}$$

We have already seen this in our previous analysis of second order systems, and from changing k_m from last week. To summarise:

Proportional Error



$$F(s) = \frac{G(s)}{1 + G(s)} = \frac{K_P k_m}{Is^2 + fs + K_P k_m}$$

Low values of K_p give stable but slow responses, and high SSVL. High values reduce SSVL but response overshoots considerably

To understand why these overshoots occur, lets look at the error term over time for a purely proportional controller

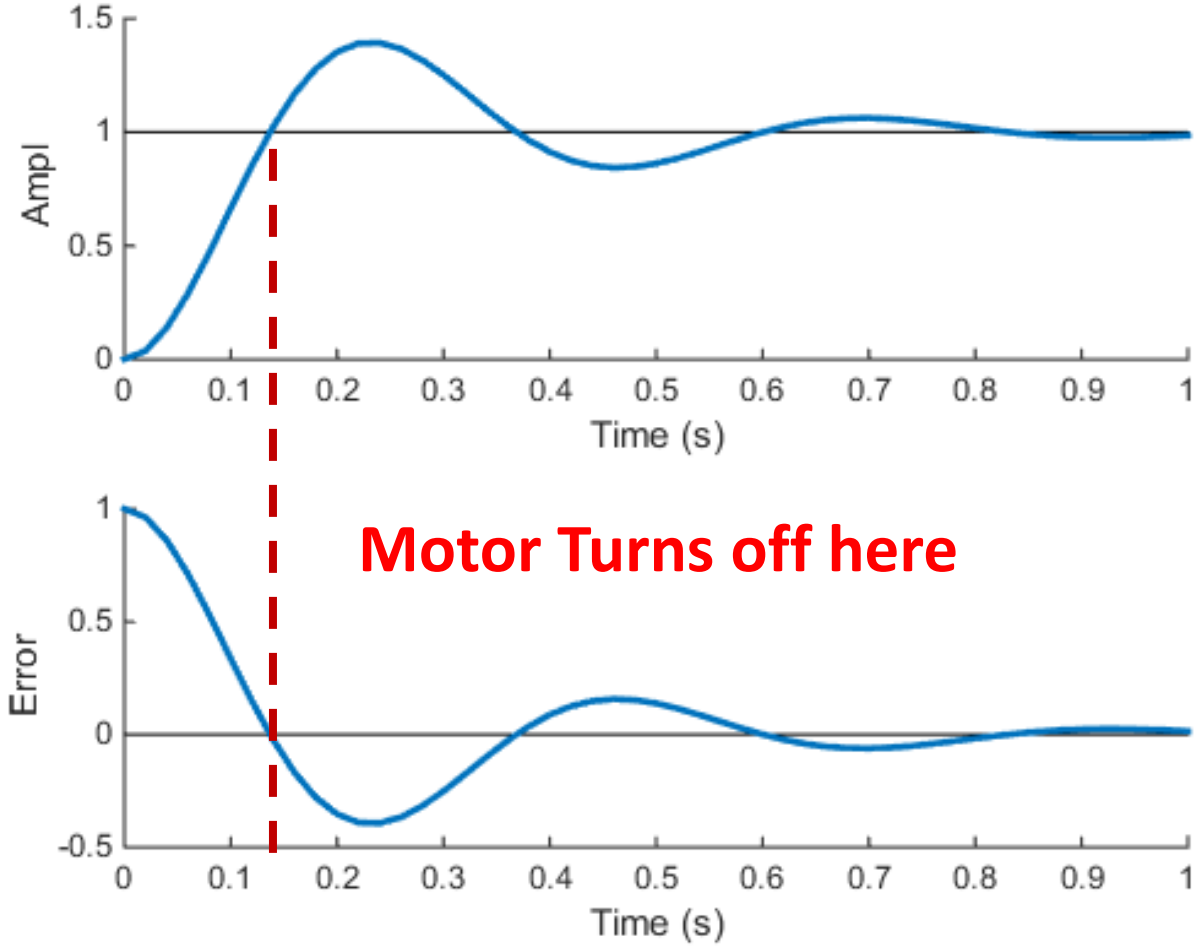
Proportional Error

The design goal for the controller is for the system to move to the input position *at rest*, *i.e.* the servo motor should move the attached load to required angle and maintain this position.

In other words *not this!*

Proportional Error

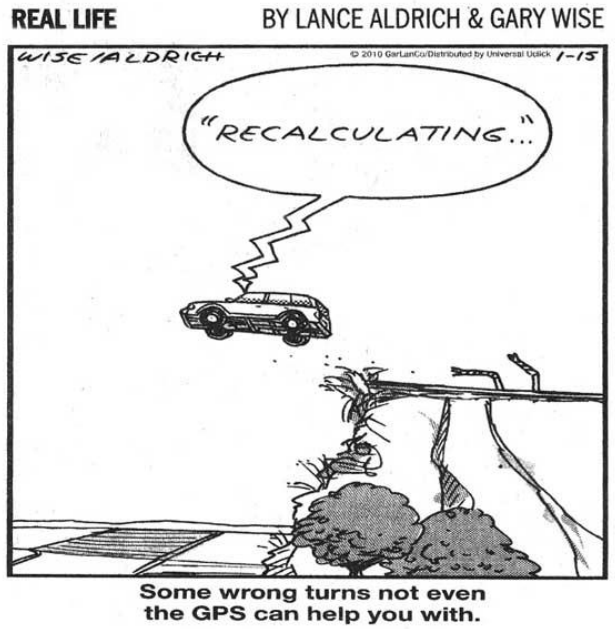
With proportional control the error *and thus the control signal* does not reach zero until the motor is at the desired position.



However, due to the inertia for the system, the motor continues to move even without a control signal, resulting in an overshoot

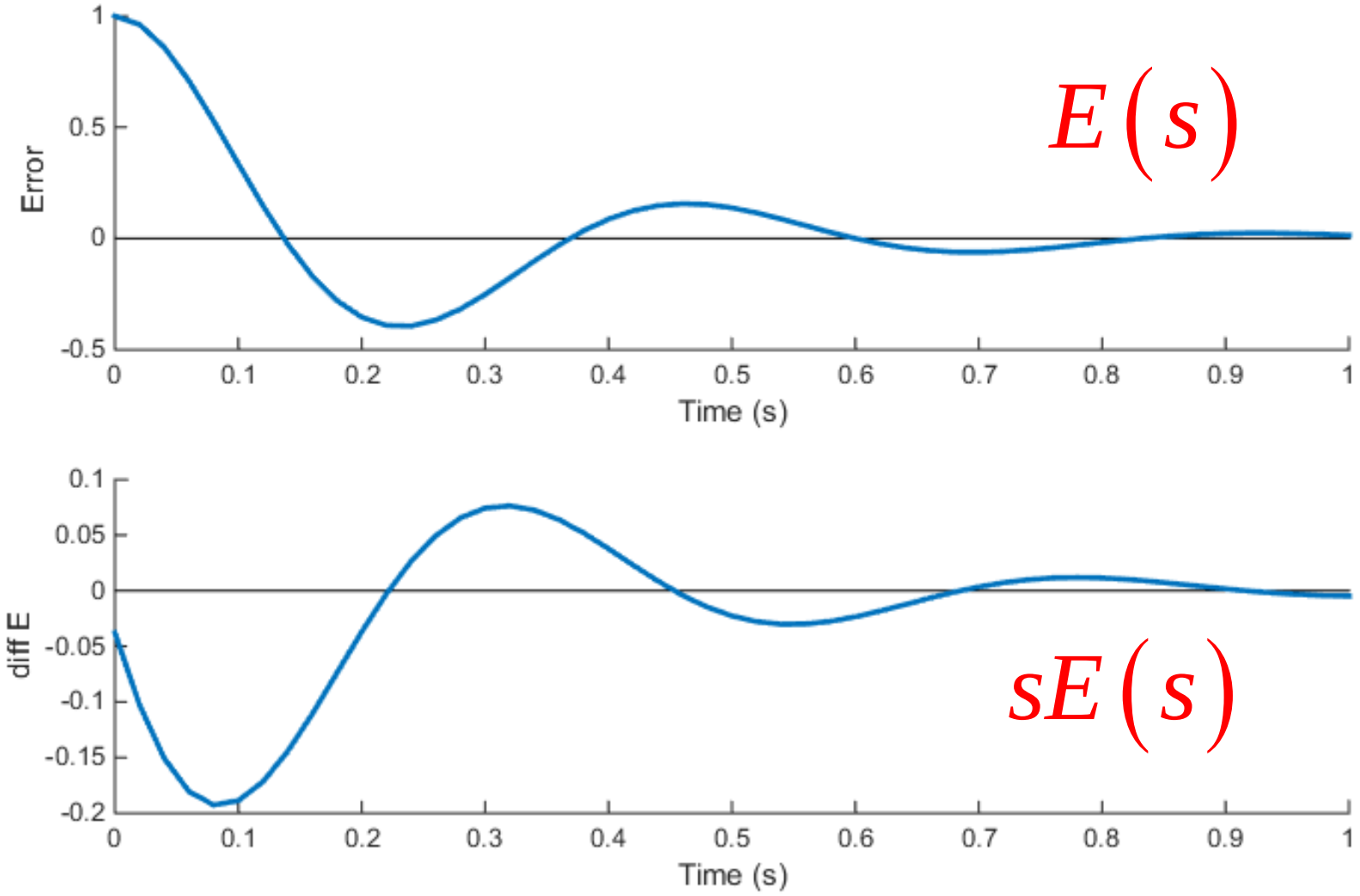
Proportional Error

Controlling the servo in the manner is like not applying the breaks until you reach your target position when driving!



In reality when driving you apply the breaks beforehand, and also dependent upon the speed you are travelling. You break earlier when you are driving faster (I hope!)

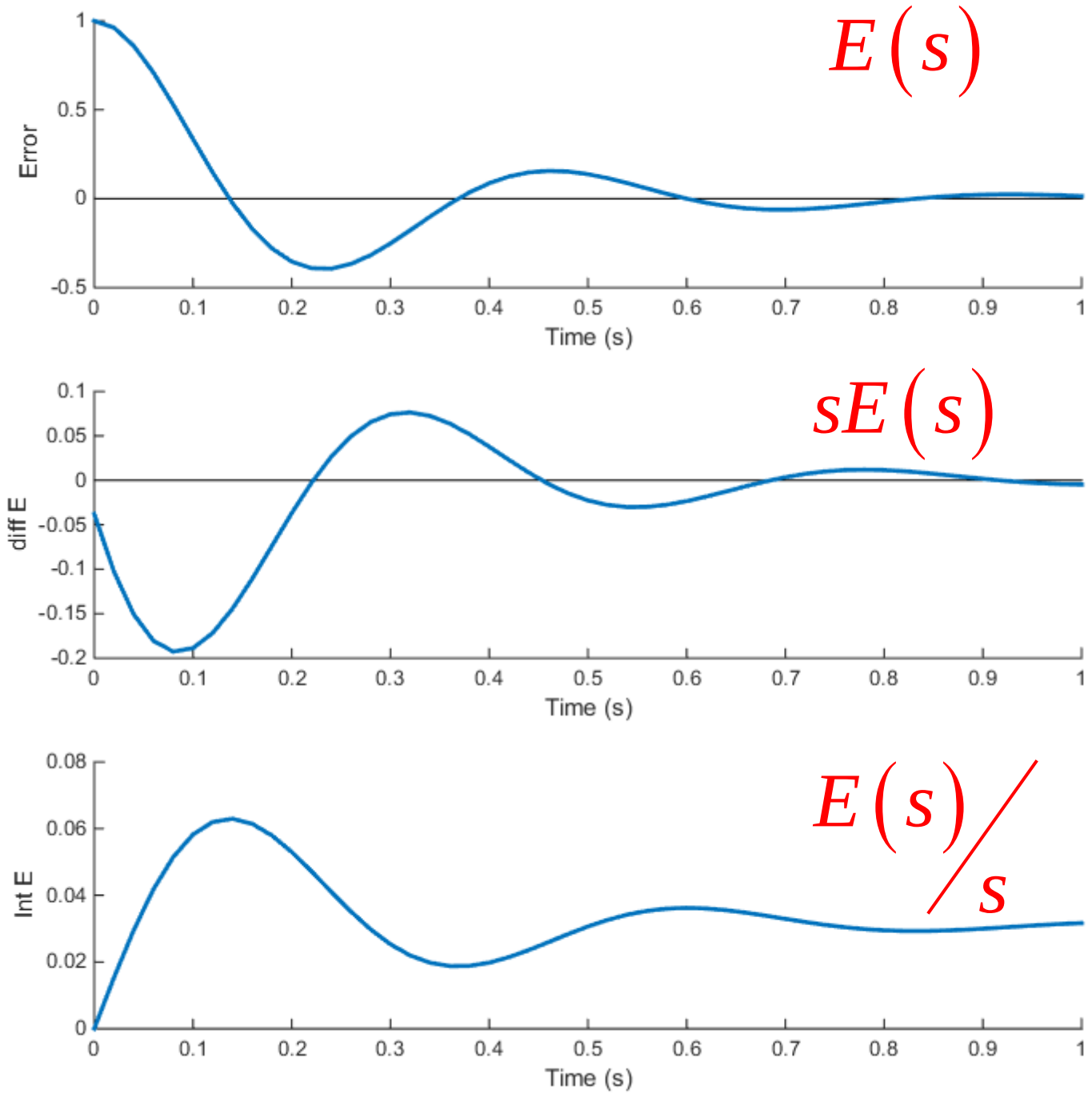
We can replicate this by considering the **derivative** of the error.



This is negative as the servo approaches the target, so offers a way of restraining the servos forward motion

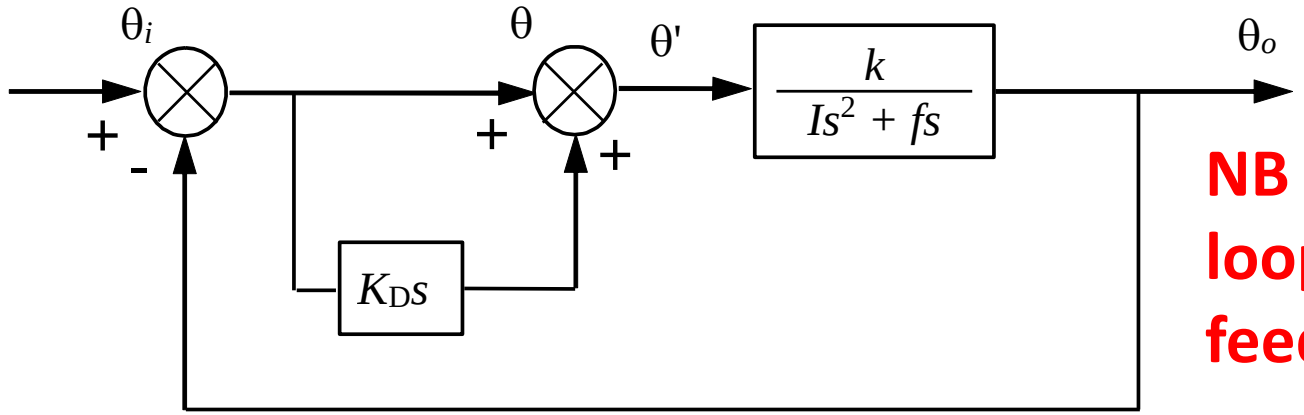
Further, if we consider the **integral** or total error over time:

This gradually increases over time, and can be used to magnify the control signal for small errors, and improve SSE



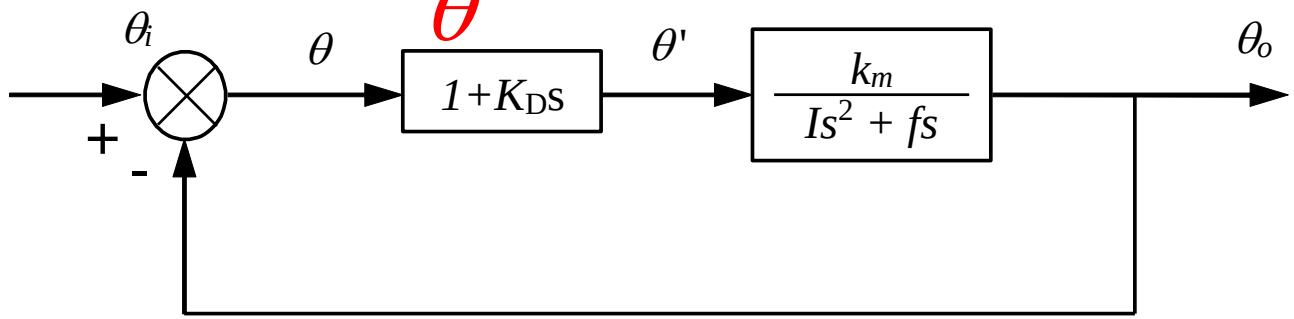
Derivative Error

First, lets consider a derivative controller with unity proportional gain



NB NOT minor loop feedback!

$$\theta' = \theta + K_D s\theta \qquad \frac{\theta'}{\theta} = 1 + K_D s \qquad \theta = \theta_i - \theta_o$$



The open loop transfer function now becomes:

$$G' = \frac{(1 + K_D s)k_m}{Is^2 + fs}$$

Type 1

Derivative Error

The open loop transfer function now becomes:

$$G' = \frac{(1 + K_D s) k_m}{Is^2 + fs}$$

Type 1

Giving a *closed loop transfer function* of

$$F(s) = \frac{G'(s)}{1 + G'(s)} = \frac{(1 + K_D s) k_m}{Is^2 + fs + (1 + K_D s) k_m}$$

$$F(s) = \frac{k_m K_D s + k_m}{Is^2 + (f + k_m K_D) s + k_m}$$

Derivative Error

$$F(s) = \frac{k_m K_D s + k_m}{Is^2 + (f + k_m K_D)s + k_m}$$

System is still second order, but now there is a zero in the numerator. The effect of this is subtle compared to the change in the poles.

More importantly, lets consider the SSVL of this system:

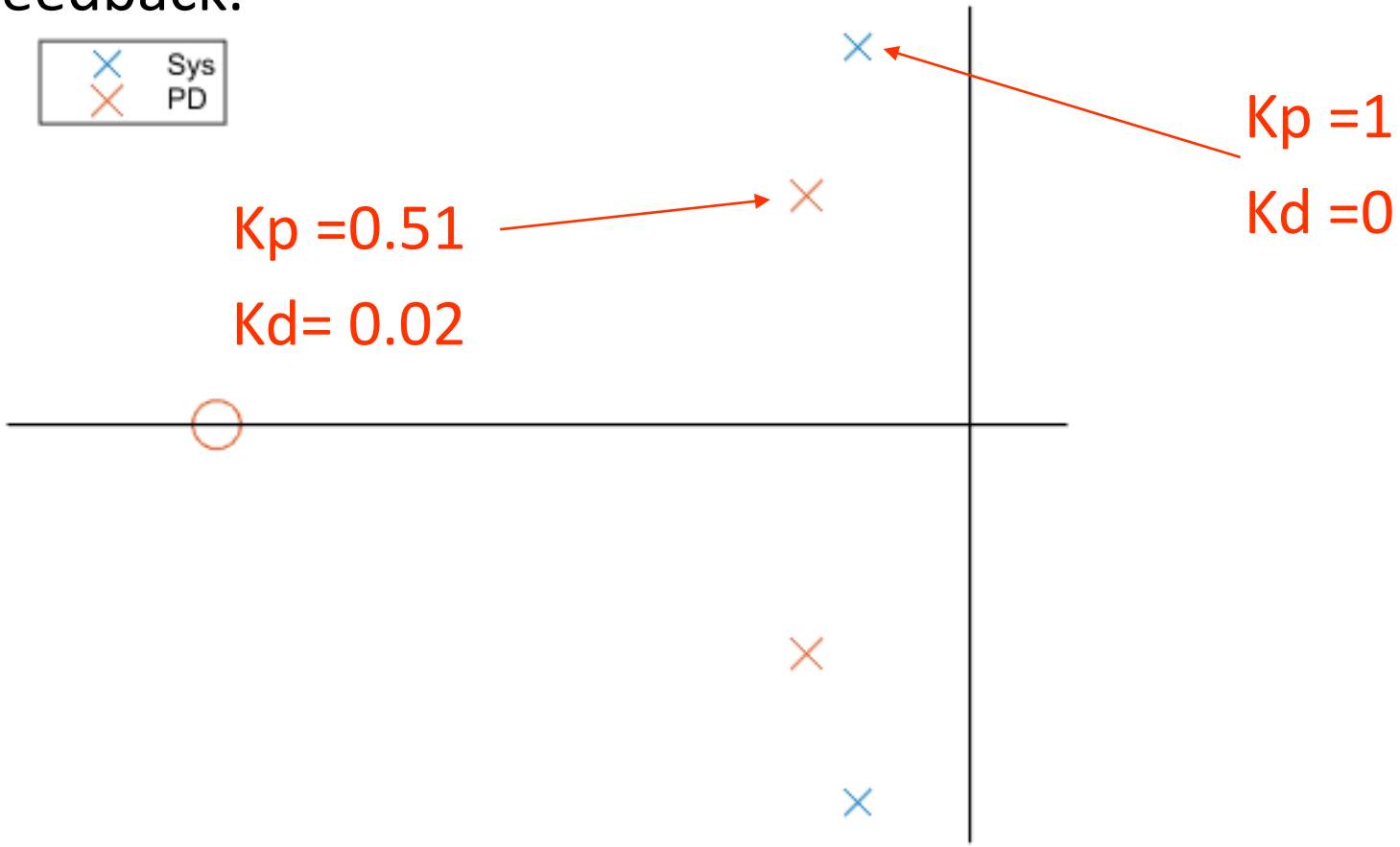
$$k_v = \lim_{s \to 0} sG'(s) = \frac{s(1 + K_D s)k_m}{s(Is + f)} = \frac{k_m}{f}$$

So unlike velocity feedback, the SSVL is unchanged by derivative error. So we can improve transient response without compromising steady state error

Hooray! Trade off avoided!

Derivative Error

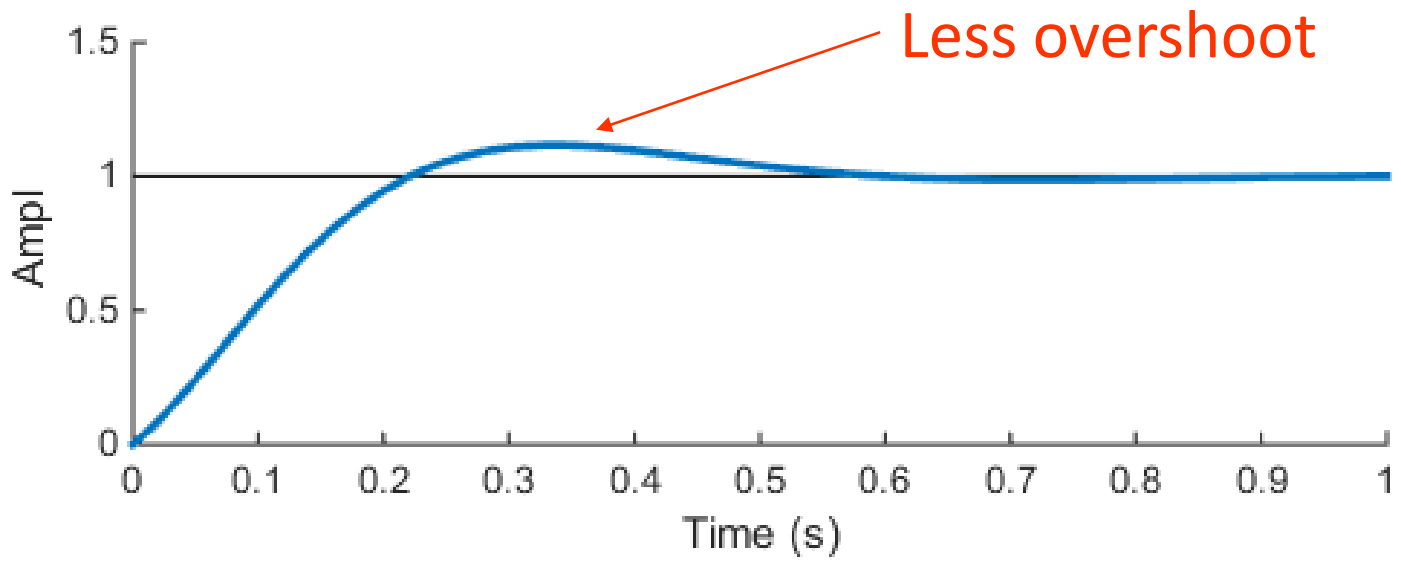
The effect on the step response is similar to that of the minor loop velocity feedback.

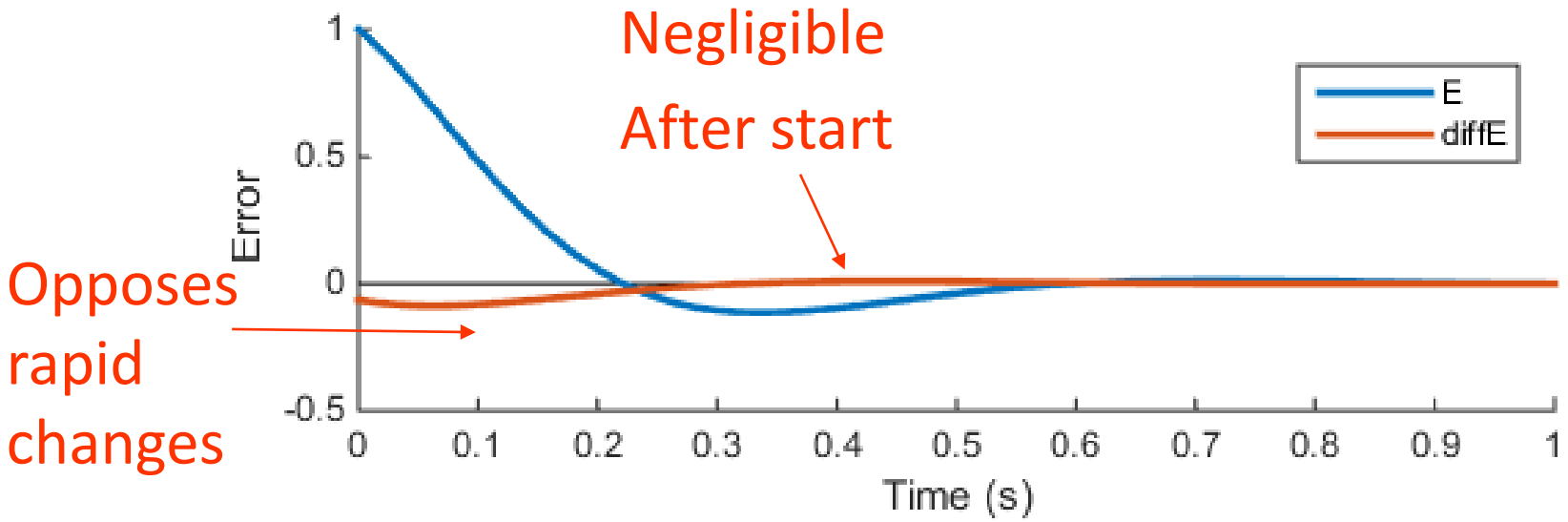


The poles of the system become less oscillatory and decay quicker. This combined with a reduced proportional gain, gives an improved transient response

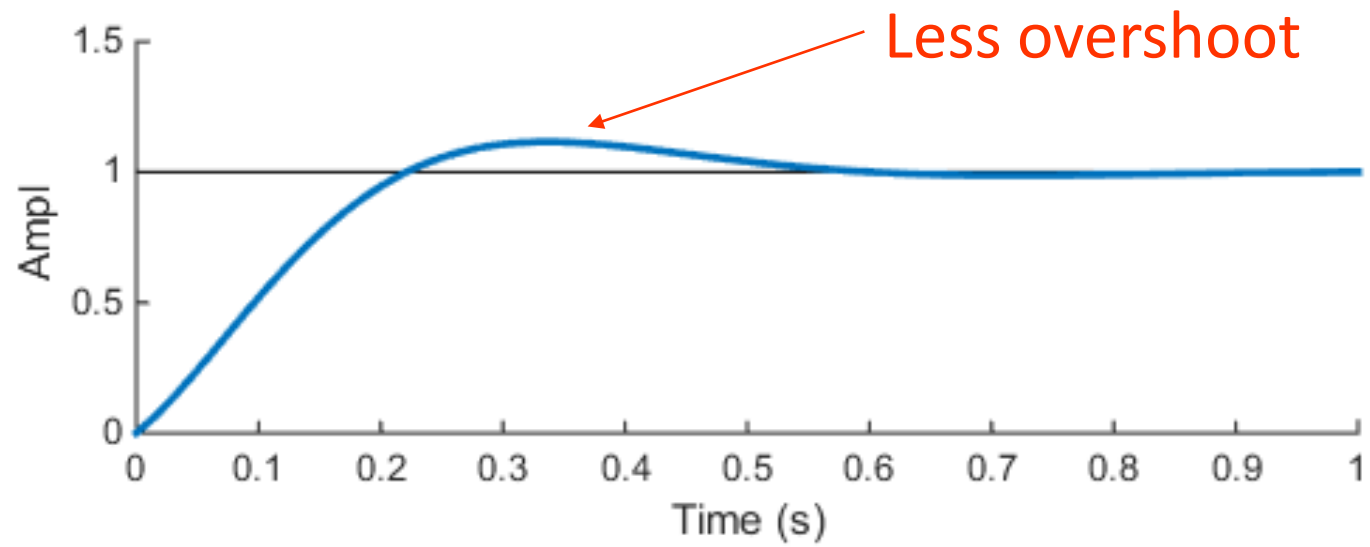
Derivative Error

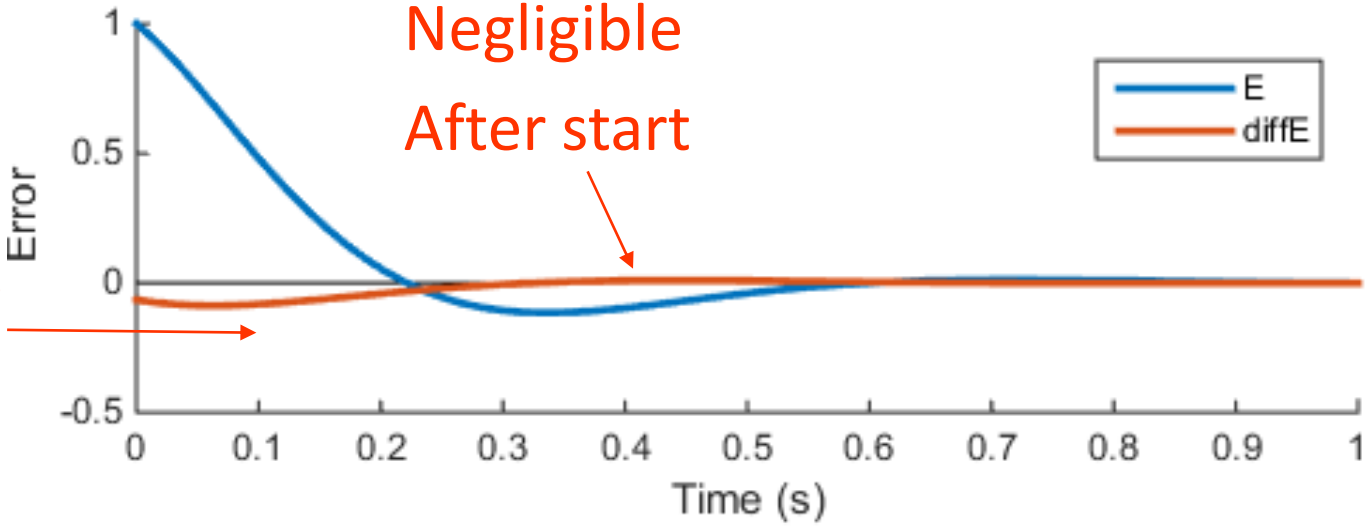
This is clear when looking at the step response, and the relative contributions of the two error terms.





Derivative Error

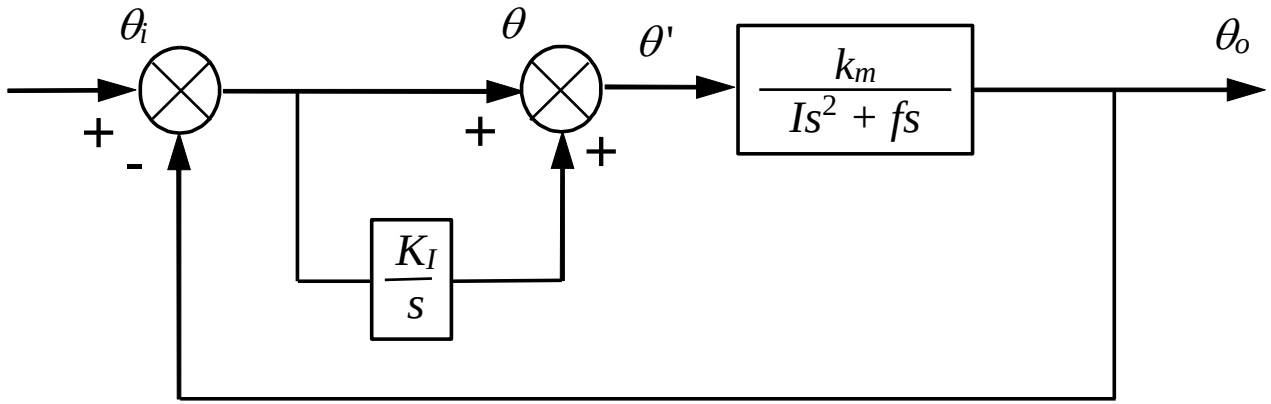




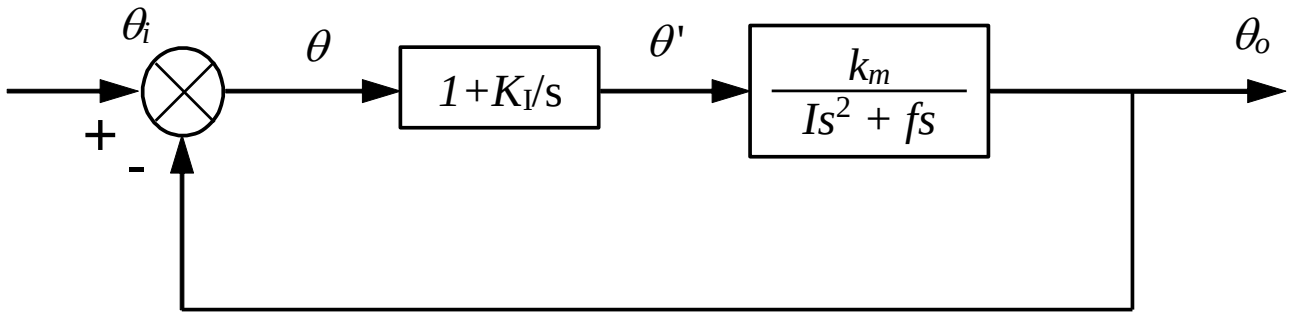
At the start, the derivative term is significant *and in the opposite direction to* the proportional error term, but becomes negligible as the system settles.

Integral Error

The integral of the error is used for correcting steady state errors, as it adds a pole at zero in the transfer function **Increases type**



$$\theta' = \theta + \frac{K_I \theta}{s} \qquad \frac{\theta'}{\theta} = 1 + \frac{K_I}{s}$$



Integral Error

$$G' = \frac{\left(1 + K_I / s\right) k_m}{Is^2 + fs} = \frac{\frac{1}{s}\left(s + K_I\right) k_m}{s\left(Is + f\right)} \quad G' = \frac{k_m s + k_m K_I}{s^2\left(Is + f\right)}$$

Type 2 system

With a close loop transfer function of

$$F(s) = \frac{G'(s)}{1 + G'(s)} = \frac{k_m s + k_m K_I}{s^2\left(Is + f\right) + k_m s + k_m K_I}$$

$$F(s) = \frac{k_m s + k_m K_I}{Is^3 + fs^2 + k_m s + k_m K_I}$$

3rd Order transfer function

Integral Error

$$F(s) = \frac{k_m s + k_m K_I}{Is^3 + fs^2 + k_m s + k_m K_I}$$

3rd Order transfer function

Increasing the system type improves the steady state response:

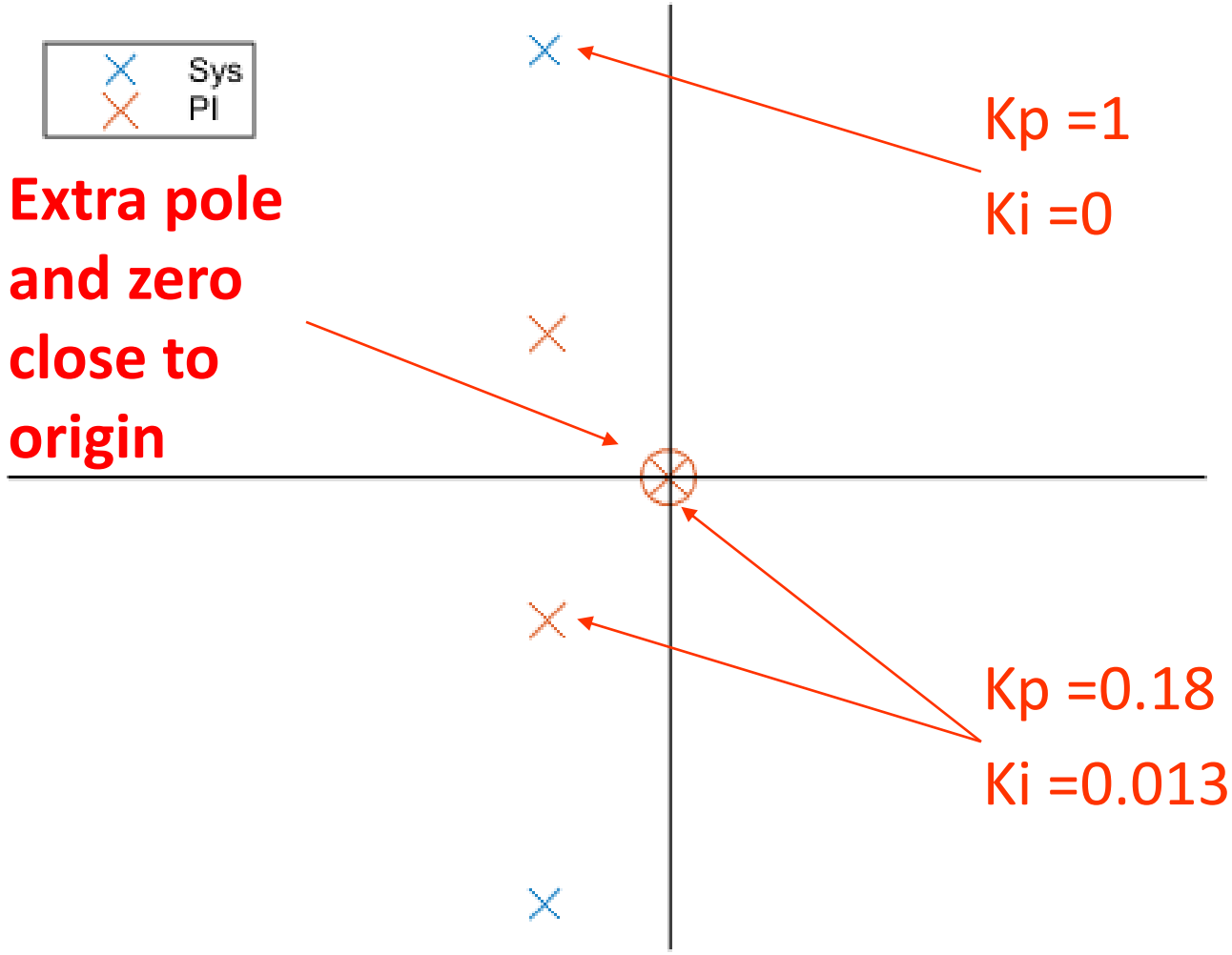
$$k_v = \lim_{s \to 0} sG'(s) \qquad = \frac{s\left(k_m s + k_m K_I\right)}{s^2\left(Is + f\right)}$$

$$k_v = \frac{k_m K_I}{0} = \infty \qquad SSVL = \frac{a}{k_v} = 0$$

Hooray! No velocity lag!

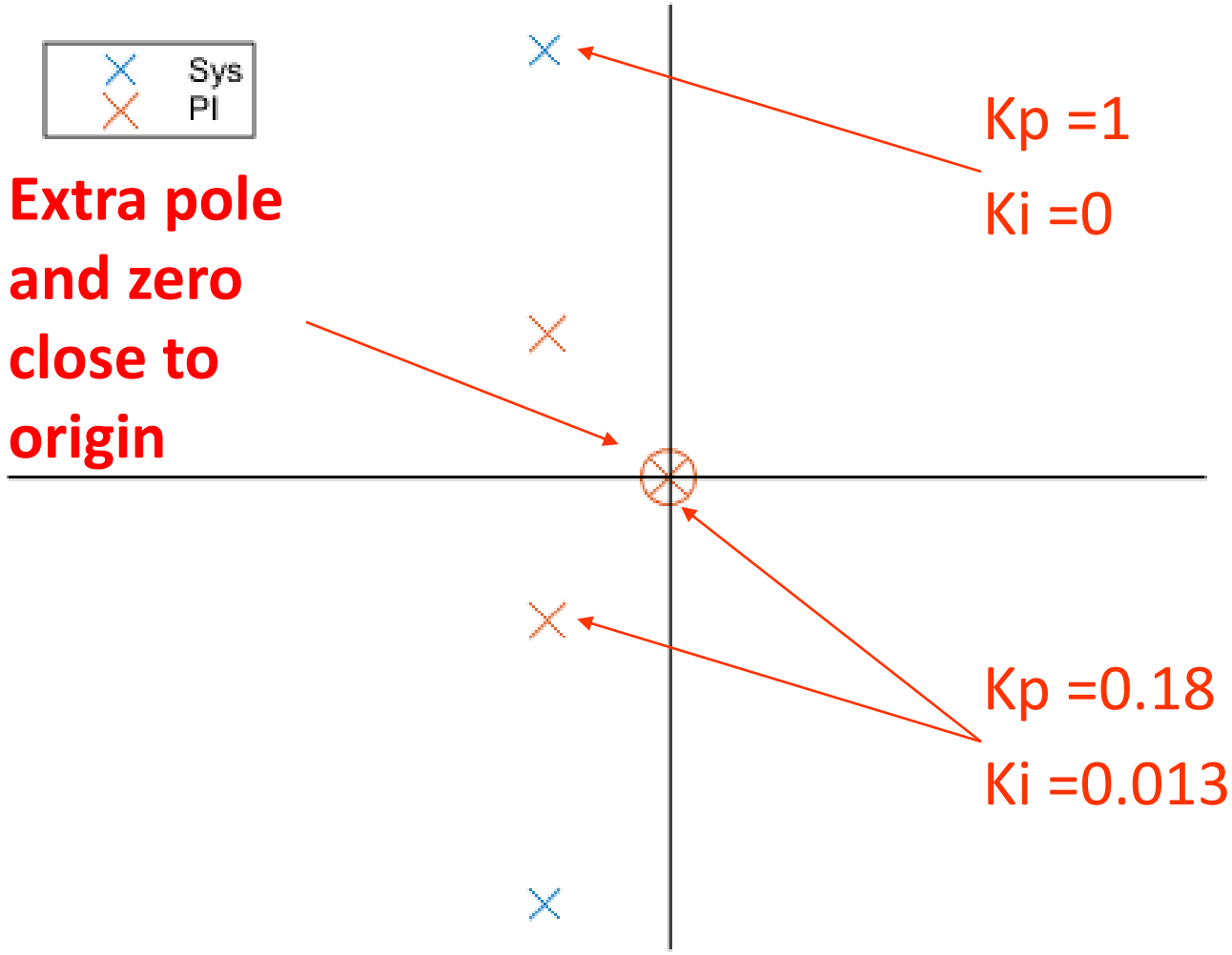
Integral Error

The effect of an extra pole close to origin – which would make our system *very slow* - is largely cancelled out by the nearby zero.

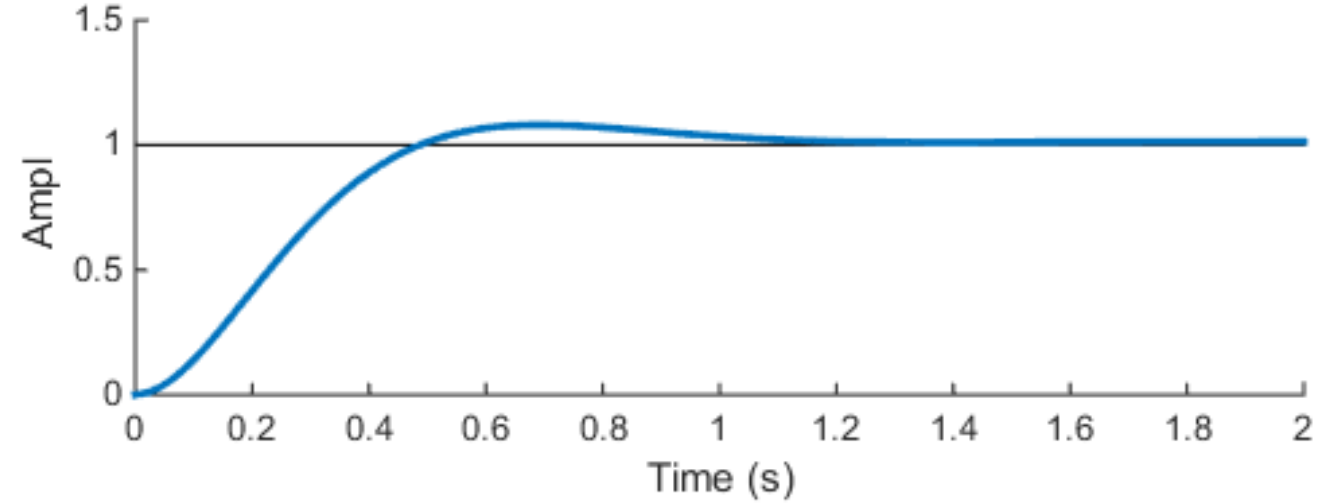


Integral Error

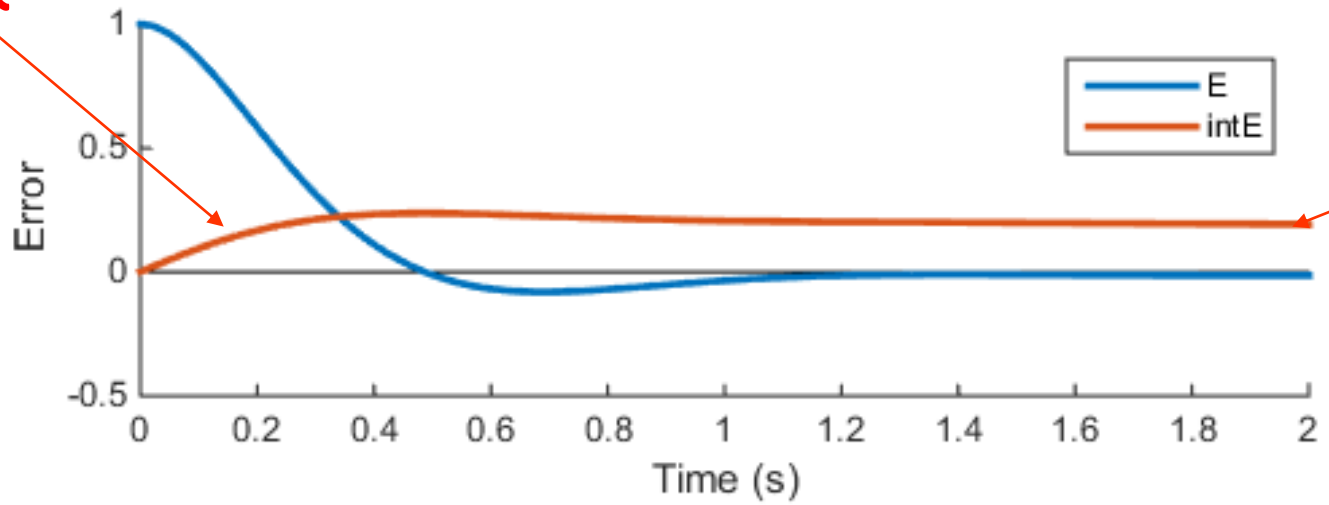
This means the system is broadly similar to a proportional controller, with the exception of improved steady state performance



Integral Error



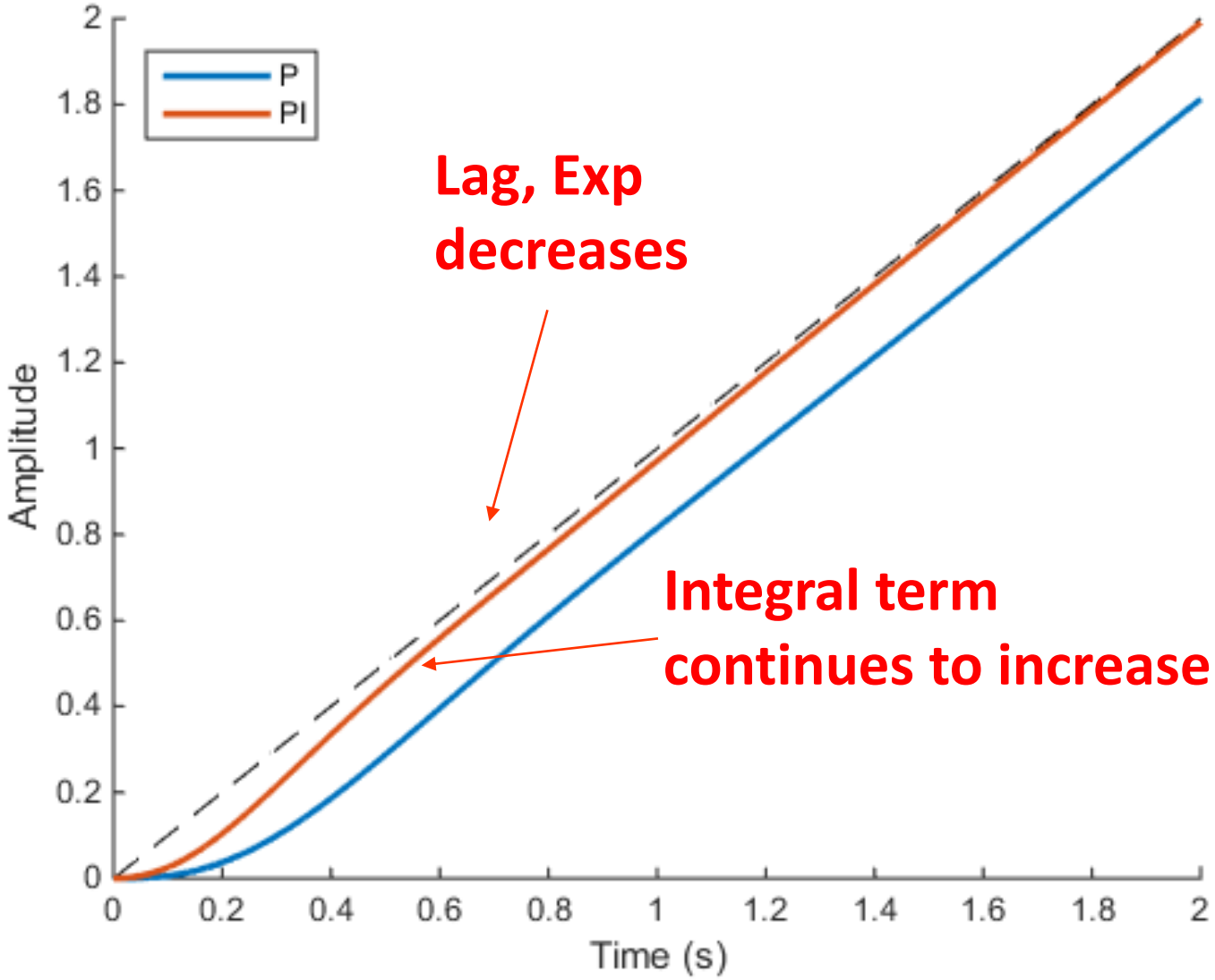
Gradually increases & Reinforces motion



Non Zero after settling

Integral Error

We can see that the integral term has enabled proper tracking of a ramp input



Integral Error

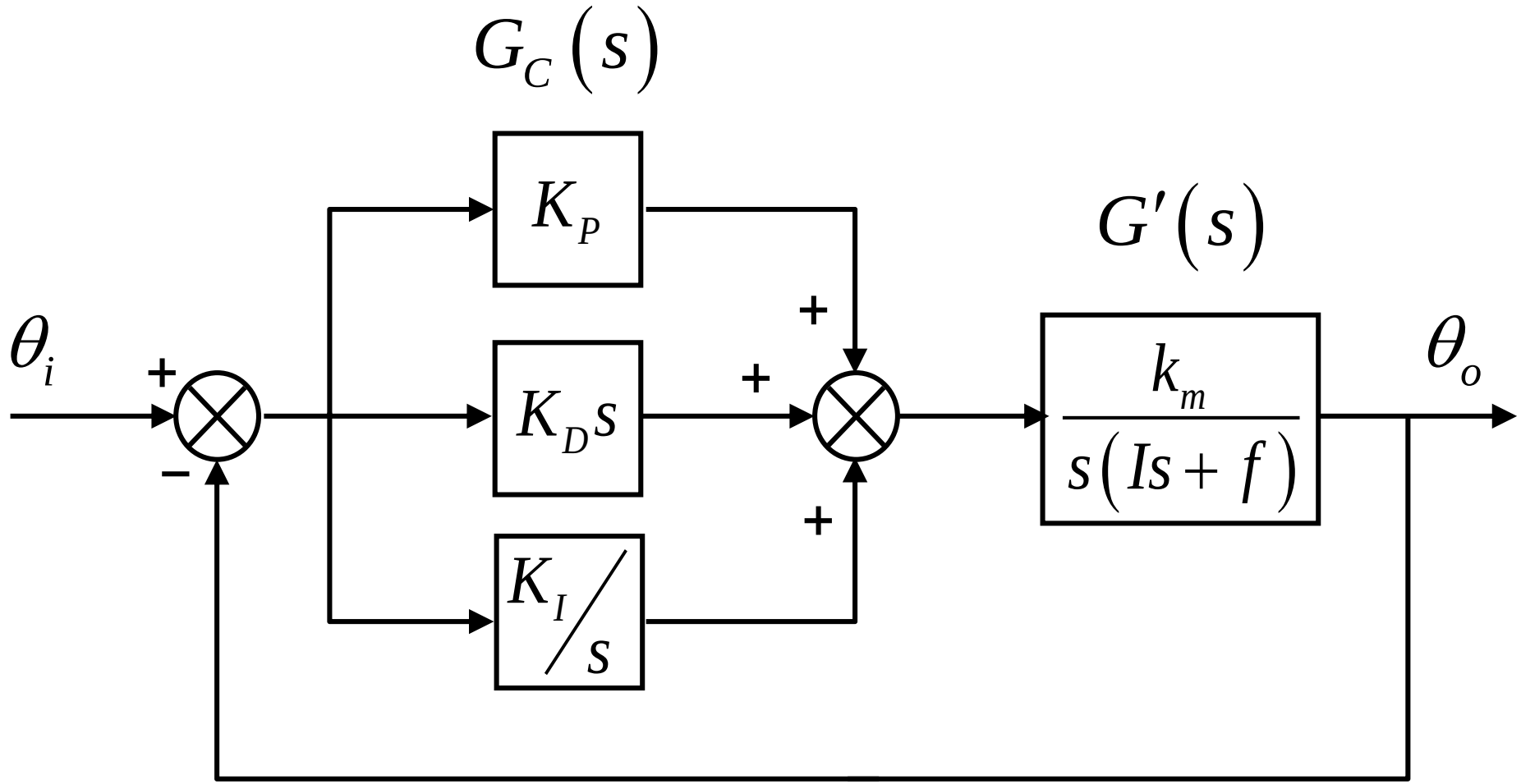
This is useful for overcoming steady state errors arising from nonlinearities not included in the model which prevent proper tracking in reality

Servo example with small error signals

Improve Type 0 Servo

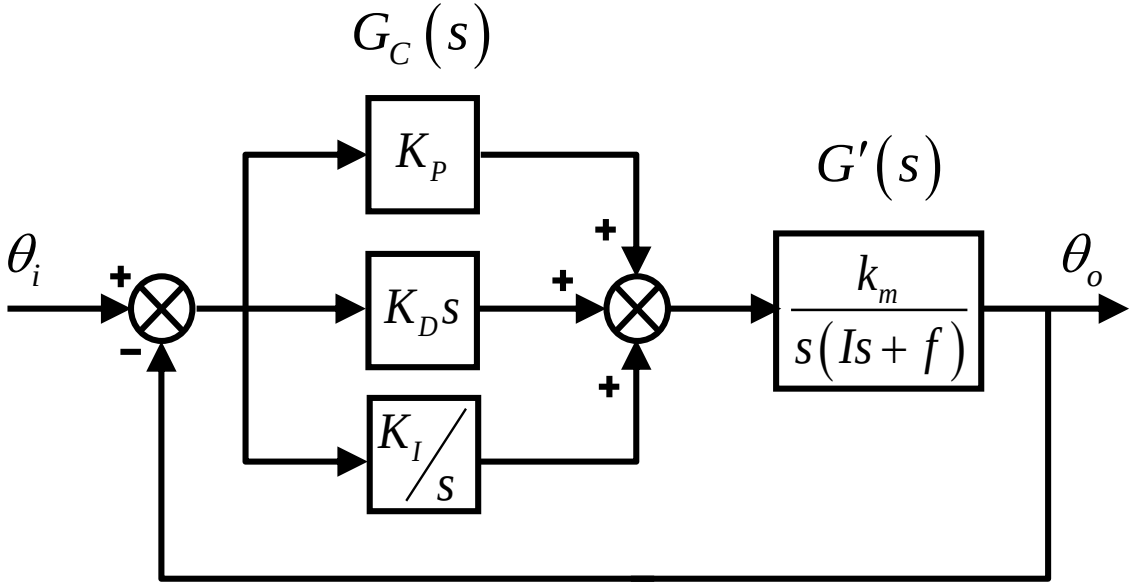
PID Control

Putting all three of these controllers together gives the complete PID



$$G_c(s) = K_P + K_D s + \frac{K_I}{s} = \frac{K_D s^2 + K_P s + K_I}{s}$$

PID Control

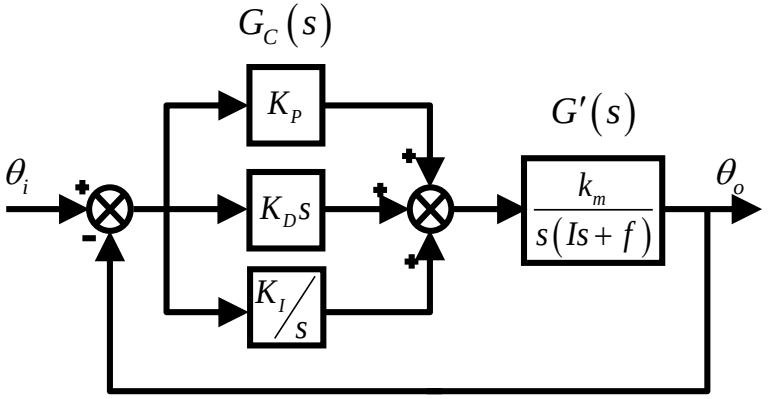


Open Loop gain $$G'(s) = \frac{k_m\left(K_D s^2 + K_P s + K_I\right)}{s^2\left(Is + f\right)}$$

The closed loop transfer function is then

$$F(s) = \frac{k_m\left(K_D s^2 + K_P s + K_I\right)}{s^2\left(Is + f\right) + k_m\left(K_D s^2 + K_P s + K_I\right)}$$

PID Control



$$F(s) = \frac{k_m K_D s^2 + k_m K_P s + k_m K_I}{Is^3 + fs^2 + k_m K_D s^2 + k_m K_P s + k_m K_I}$$

$$F(s) = \frac{k_m K_D s^2 + k_m K_P s + k_m K_I}{Is^3 + (f + k_m K_D) s^2 + k_m K_P s + k_m K_I}$$

Phew!

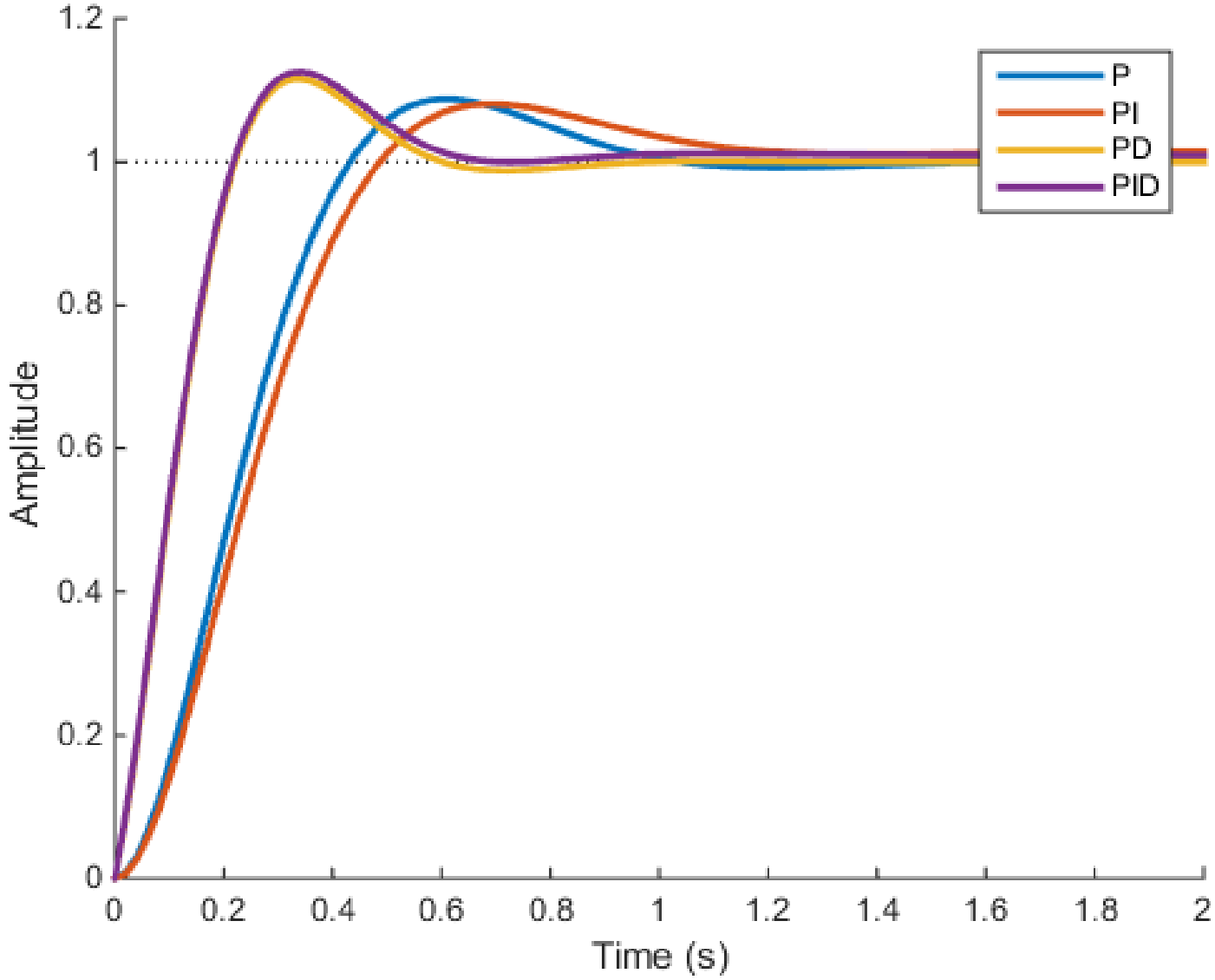
Thus by choosing the appropriate values of K_P K_D K_I

It is possible to design a controller with improved transient response **and** decreased/no steady state error

IN THEORY!

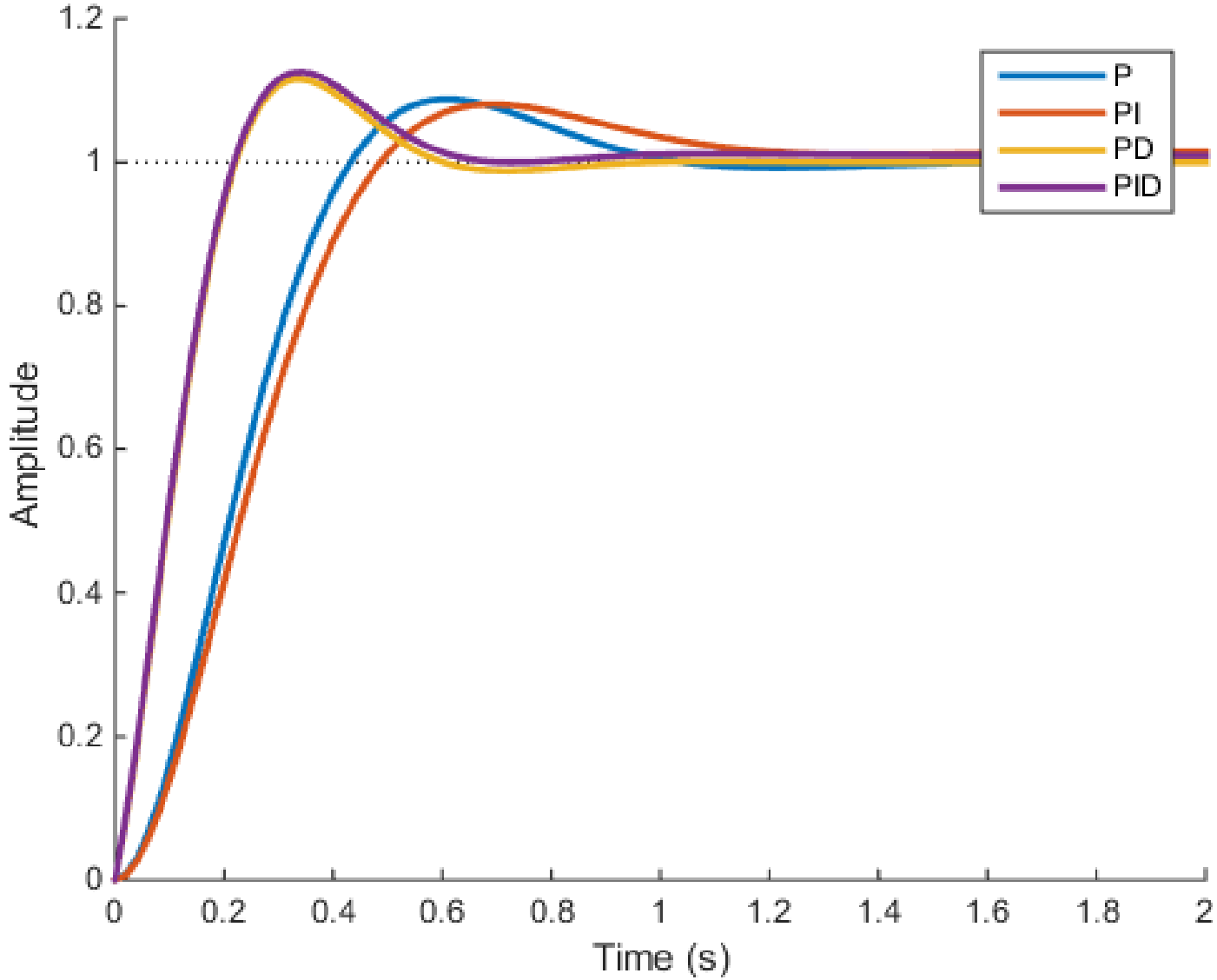
PID Control

There is no unique solution for the settings for the three gains, and so the values are dependent upon the specific system and the application.



PID Control

Selecting these parameters is known as **tuning**, and it was (and still is in machine learning circles) a very active area of research.



PID Control

Sometimes overshoot reduction is more important than settling time (heaters, ventilators).

In other cases reducing settling time may be crucial and some overshoot is an acceptable trade off – camera focus lens etc.

Many methods for “Tuning” these parameters exist, and often form the first estimate, before refinement based on testing real system.

PID Control

These controllers are used everywhere, and can even be applied with acceptable results with little or no knowledge of the system being controlled!

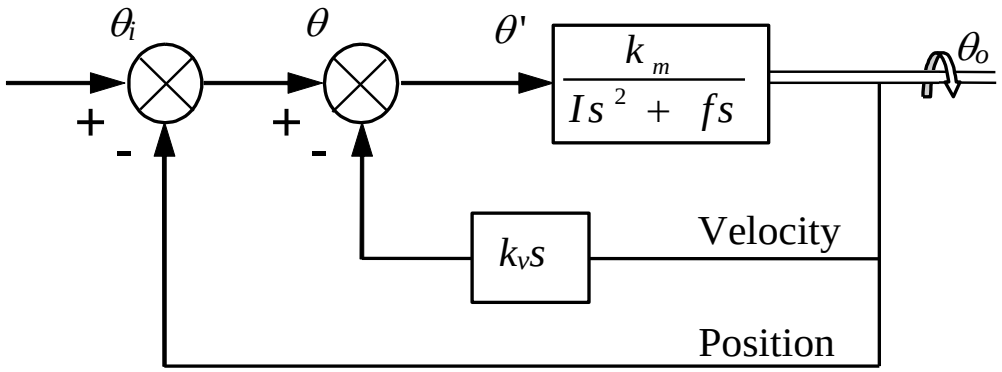
Summary

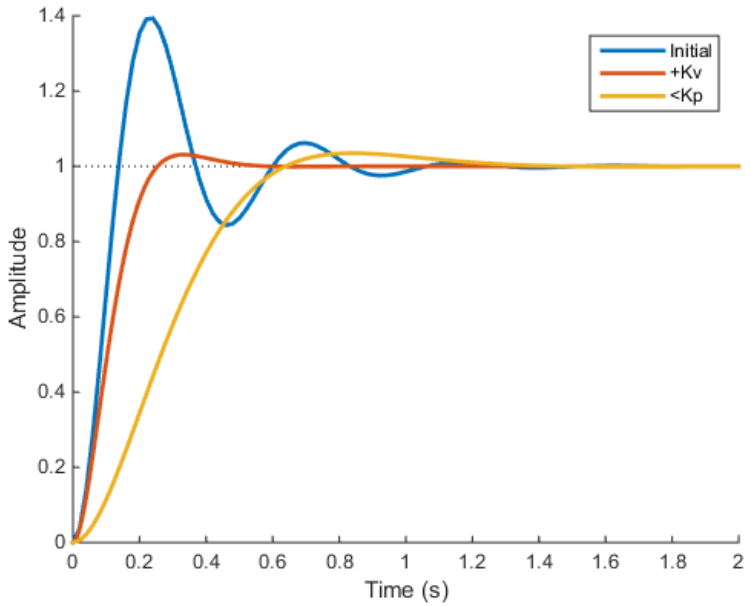
The system **type** is the number of poles in the denominator in *G(s)*, and it defines the steady state performance

	Input type		
No. Integrators in denominator = system TYPE	Step $r(t) = a$ $R(s) = a/s$	Ramp $r(t) = at$ $R(s) = a/s^2$	Acceleration $r(t) = at^2/2$ $R(s) = a/s^3$
0	$e_{ss} = a/(1+k_p)$	$e_{ss} = \infty$	$e_{ss} = \infty$
1	$e_{ss} = 0$	$e_{ss} = a/k_v$	$e_{ss} = \infty$
2	$e_{ss} = 0$	$e_{ss} = 0$	$e_{ss} = a/k_a$

Summary

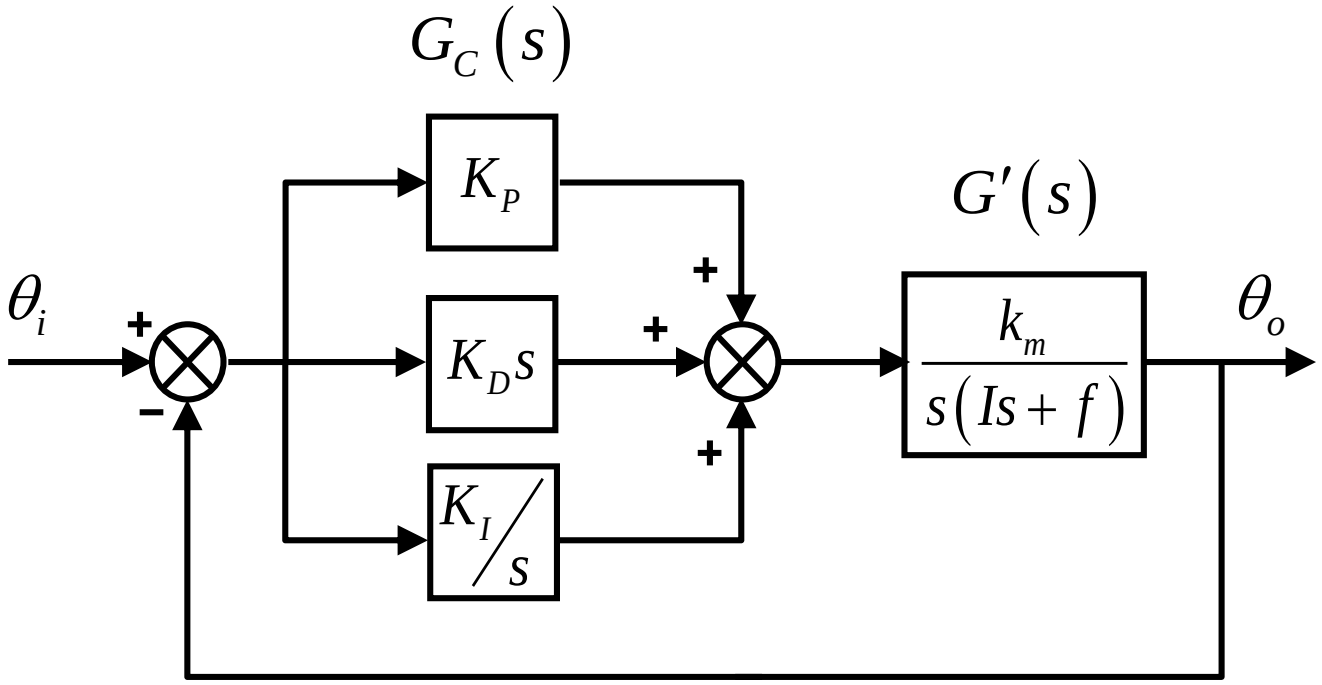
Minor loop feedback of a servo can improve the transient response at the expense of the steady state error





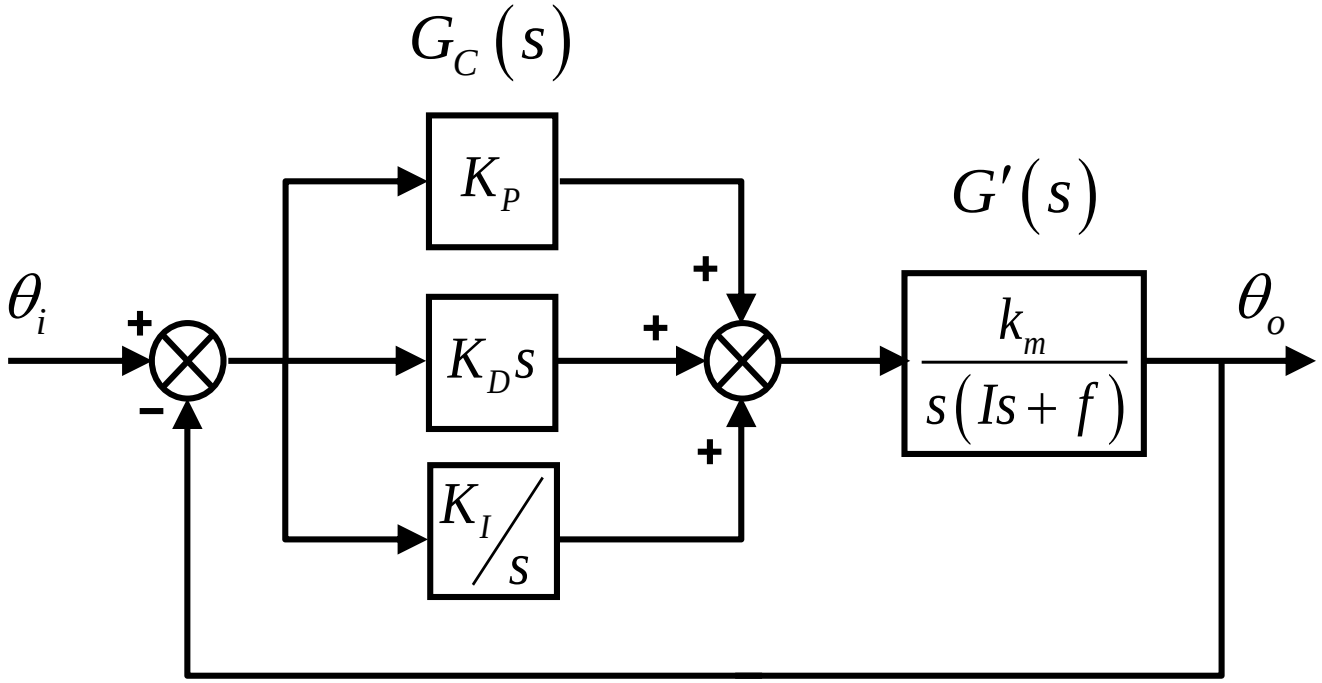
Summary

By far the most common class of controllers in industry is the **proportional – integral – derivative (PID)** controller



Through consideration of the change in error, and the total error over time, it is possible to improve the transient and steady state response of a controller through correct choice of gains

Summary



However, there does not exist one single answer for the correct choice of gain, so they must be selected by the engineer based on the specifications of the system

Thank you for your attention!

Some graphic material used in the course was taken from publicly available online resources that do not contain references to the authors and any restrictions on material reproduction.

POLYTECH
Peter the Great
St.Petersburg Polytechnic
University

OPEN
POLYTECH