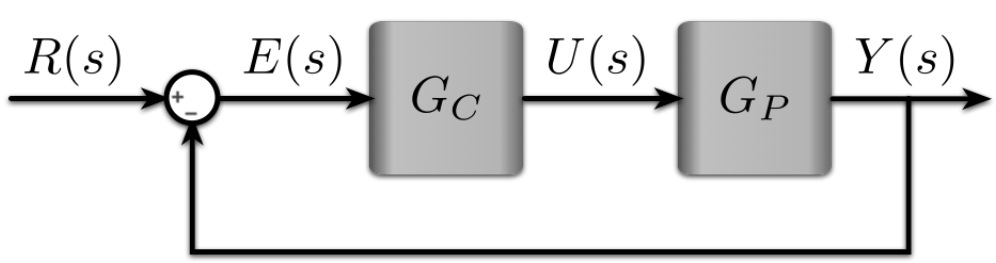
Implementing a PID Controller

Q: What do we need to implement a PID (or any feedback) controller?

- Measurement of the states (or outputs) we need to feed back ← or some way to reliably estimate
- Some way to do computation

Let's walk through what the algorithm should be:

Given a system in which measure or estimate all states that we need and a set of gains (we're not talking about choosing gains right now.)



1) Determine/define the reference command, $r(t)$
2) Measure/estimate states/output, $y(t)$
3) Calculate:
   a) error $= e = r - y$
   b) rate of change of error $= \dot{e}$
   c) integral of error $= \int_0^t e(t)\,dt$

   Q: How can we do this?

   Recognize that we'll be sampling at a (hopefully) regular rate
4) Calculate $u(t) = K_p e + K_I \int_0^t e(t)\,dt + K_d \dot{e}$
5) Goto step 2 and repeat

Implementing a PID Controller (cont.)

3) Calculate:

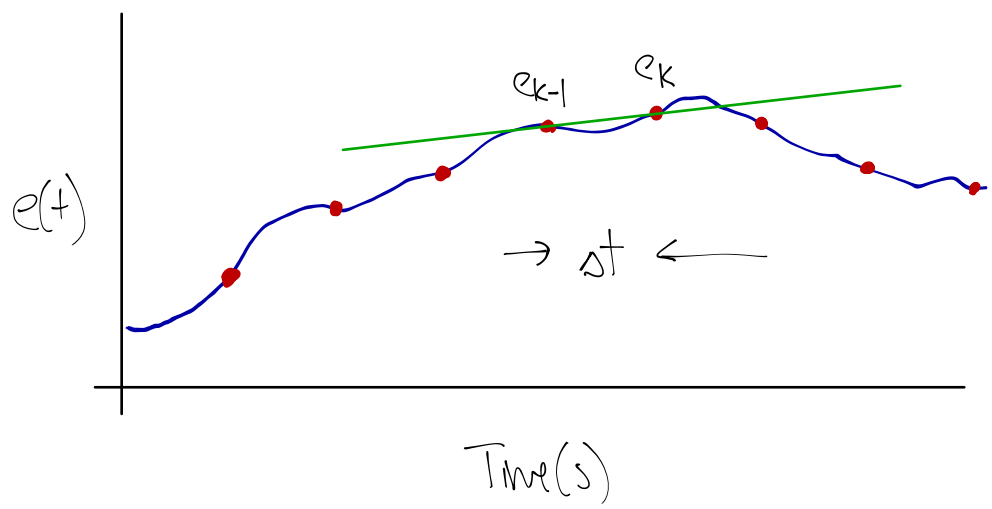
a) error = $e = r - y$

b) rate of change of error = $\dot{e}$

At current time

- $e(t)$ is easy
- How about $\dot{e}$

We can use a finite difference



← We're actually only able to process at discrete timesteps

Q: What is $\dot{e}$ between two of these points

- the slope / the line between them

So, we can get the rate of change of the error by:

$$\dot{e}_k = \frac{e_k - e_{k-1}}{\Delta t}$$

where
$\dot{e}_k \equiv$ rate of change at current time
$e_k \equiv$ error at current time
$e_{k-1} \equiv$ error the last time we measured
$\Delta t =$ time between measurments

And the derivative term becomes

$$D = K_d \left(\frac{e_k - e_{k-1}}{\Delta t} \right)$$

Q: What does this mean for our code?

- We have to know Δt
- We need to retain the value of the error from the previous calculation

Implementing a PID Controller (cont.)

2) Measure | estimate current state/output y_k

3) Calculate:

a) Δt = current time − last time we calculated

b) error = $e_k = r_k - y_k$

c) rate of change of error = $\dot{e}_k = \dfrac{e_k - e_{k-1}}{\Delta t}$

d) integral of error = $\int_0^t e(t)\,dt$

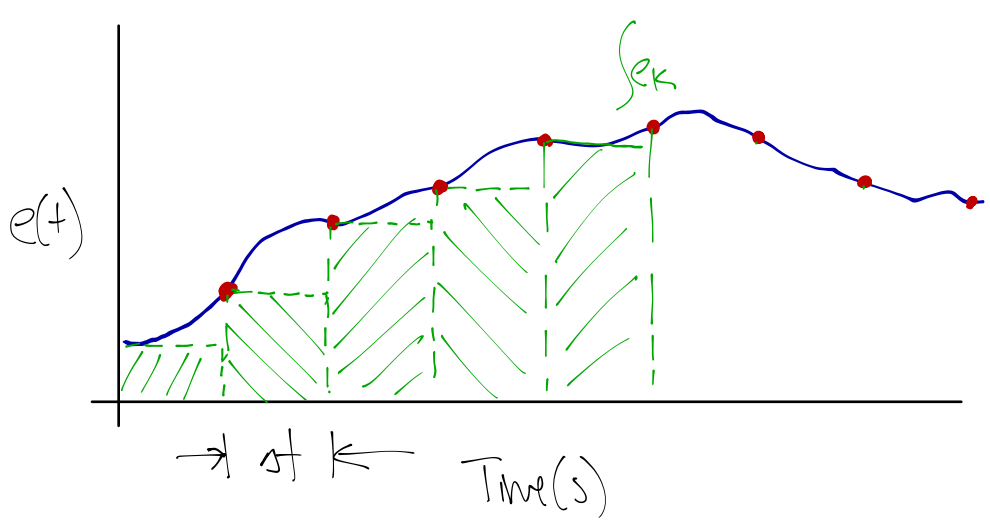
4) Calculate $u(t) = K_p e + K_I \int_0^t e(t)\,dt + K_d \dot{e}_k$

5) Save e_k for next time through the loop

6) Save current time for next time through the loop

7) Goto Step 2 and repeat

Q: How can we calculate the integral term?



← The integral is just the area under the curve.

We can use the running sum of that area to calculate the I term

(You probably learned to integrate this exact way, right?)

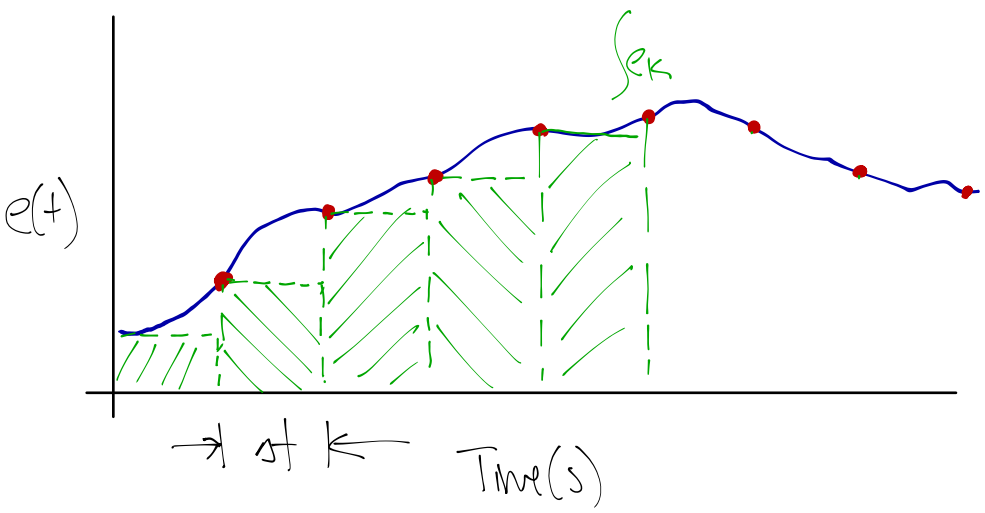
Between steps t_{k-1} and t_k, the area under the curve is just

Area = $e_k \Delta t$ ← just a single green rectangle in the figure above

The total area is just the running sum of these rectangles

$$\int e = \sum_{i=0}^{k} e_i \Delta t$$

Implementing a PID Controller (cont.)



Q: How can we implement this without having to remember the error signal for every time in the past?

The sum of the area at the current time is just the area up until that point, plus the new "rectangle" from the current time

Curr Int = Past Int + current area

$$\int e_k = \int e_{k-1} + e_k \Delta t = \underbrace{\sum_{i=1}^{k-1} e_i \Delta t}_{\text{just remember this sum, not each } e_i} + e_k \Delta t$$

So, we also need to save the past integral term for next time step

Implementing a PID Controller (cont.)

2) Measure | estimate current state/output y_k

3) Calculate:

a) Δt = current time − last time we calculated

b) error = $e_k = r_k - y_k$

c) rate of change of error = $\dot{e}_k = \dfrac{e_k - e_{k-1}}{\Delta t}$

d) integral of error = $\int e_k = \int e_{k-1} + e_k \Delta t$

4) Calculate $u(t) = K_p e + K_I \int e_k + K_d \dot{e}_k$

5) Save e_k for next time through the loop

6) Save current time for next time through the loop

7) Save current integral for next time step

8) Goto Step 2 and repeat

So, now our controller output at current time t_k is u_k:

$$u_k = K_p e_k + K_I \int e_k + K_d \left[\frac{e_k - e_{k-1}}{\Delta t} \right]$$

$$= K_p e_k + K_I \underbrace{\left[\int e_{k-1} + e_k \Delta t \right]}_{\text{every term here has a } \cdot \Delta t \text{ in it}} + \frac{K_d}{\Delta t} \left(e_k - e_{k-1} \right)$$

$$= K_p e_k + \underbrace{K_I \Delta t}_{} \left[\sum_i^{k-1} e_i + e_k \right] + \underbrace{\frac{K_d}{\Delta t}}_{} \left(e_k - e_{k-1} \right)$$

← Q: Is there any advantage to this form?

We only have to calculate these two terms once.

Implementing a PID Controller (cont.)

2) Measure | estimate current state/output y_k

3) Calculate:

a) Δt = current time − last time we calculated ← with some controllers Δt = constant is enforced for you

b) error = $e_k = r_k - y_k$

c) rate of change of error = $\dot{e}_k = \dfrac{e_k - e_{k-1}}{\Delta t}$

d) integral of error = $\int e_k = \int e_{k-1} + e_k \Delta t$

4) Calculate $u_k = k_p e_k + k_I \Delta t \left(\int e_k \right) + \dfrac{k_d}{\Delta t} \dot{e}_k$

5) Save e_k for next time through the loop

6) Save current time for next time through the loop

7) Save current integral for next time step

8) Goto Step 2 and repeat

Q: See any ways to further improve and/or problems with this formulation?

Let's look at the derivative term...

We can change the setpoint instantaneously. What happens in that case

$$e(t) = r(t) - y(t) \rightarrow \dot{e}(t) = \dot{r}(t) - \dot{y}(t)$$

Q: If $r(t)$ changes instantly, what is this?

Infinity.

Q: What does that mean for our controller output (as long as $k_d \neq 0$)?

Infinity!!! ← Not good!!!

This is called <u>derivative kick.</u>

Implementing a PID Controller (cont.)

"Fixing" Derivative Kick

$$e(t) = r(t) - y(t) \rightarrow \dot{e}(t) = \dot{r}(t) - \dot{y}(t)$$

We often have (mostly) constant reference commands.

Q: What does that mean for $\dot{r}(t)$?

It's (mostly) zero...

So, just ignore it... change derivative term to

$$D = k_d(\cancel{\dot{r}}^{\,0} - \dot{y}) = k_d(-\dot{y}) \longleftarrow \text{Derivative on Measurement}$$

or in our implementation:

$$D_k = -\frac{k_d}{\Delta t}(y_k)$$

Implementing a PID Controller (cont.)

2) Measure | estimate current state/output y_k

3) Calculate:

a) Δt = current time − last time we calculated ← with some controllers Δt = constant is enforced for you

b) error = $e_k = r_k - y_k$

c) rate of change of output/state − $\dot{y}_k = \frac{y_k - y_{k-1}}{\Delta t}$

d) integral of error = $\int e_k = \int e_{k-1} + e_k \Delta t$

4) Calculate $u_k = k_p e_k + k_I \Delta t \left(\int e_k\right) + \frac{k_d}{\Delta t}\left(-\dot{y}_k\right)$

5) Save y_k for next time through the loop

6) Save current time for next time through the loop

7) Save current integral for next time step

8) Goto Step 2 and repeat

Q: Other problems or improvements?

We aren't checking if u_k is a feasible input for our system

As is, we could exceed actuator limits, etc.

If we add that step:

if $u_k > u_{max}$

$u_k = u_{max}$

else if $u_k < u_{min}$

$u_k = u_{min}$

assuming symmetric limit − $u_{min} = -u_{max}$

if $u_k > u_{max}$

$u_k = u_{max}$

else if $u_k < -u_{max}$

$u_k = -u_{max}$

Q: What happens to the integral term?

Error can keep accumulating... integral term grows

Q: How long would it take the integral term to return to "normal"?

a nonzero finite amount of time

This is called Integral Windup

Implementing a PID Controller (cont.)

"Fixing" Integral Windup

We need to limit the max/min of the integral term itself (in addition to total output)

if $\int e_k > u_{max}$

$\int e_k = u_{max}$

else if $\int e_k < u_{min}$

$\int e_k = u_{min}$

Implementing a PID Controller (cont.)

1) Determine/define the reference command, $r(t)$

2) Measure/estimate current state/output y_k

3) Calculate:

   a) Δt = current time − last time we calculated ← with some controllers Δt = constant is enforced for you

   b) error = $e_k = r_k - y_k$

   c) rate of change of output/state − $\dot{y}_k = \dfrac{y_k - y_{k-1}}{\Delta t}$

   d) integral of error = $\int e_k = \int e_{k-1} + e_k \Delta t$

4) Check limits on integral term

   if $\int e_k > u_{max}$
   $\int e_k = u_{max}$
   else if $\int e_k < u_{min}$
   $\int e_k = u_{min}$

5) Calculate $u_k = k_p e_k + k_I \Delta t \left(\int e_k\right) + \dfrac{k_d}{\Delta t}\left(-\dot{y}_k\right)$

6) Check limits on u_k

   if $u_k > u_{max}$
   $u_k = u_{max}$
   else if $u_k < u_{min}$
   $u_k = u_{min}$

7) Save y_k for next time through the loop

8) Save current time for next time through the loop

9) Save current integral for next time step

10) Goto step 2 and repeat

Q: Other potential problems? Improvements?

- still relies on good measurements/estimates of states/output
- We've given no thought to what $r(t)$ should be ← $r(t)$ can dramatically change a system resp.