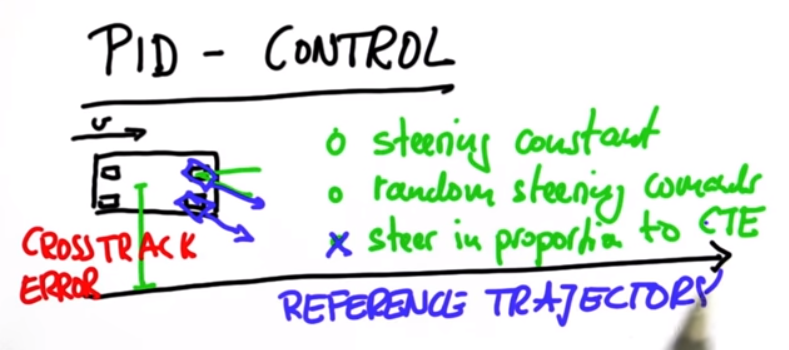
PID Control

P- Proportional, I- Integral, D-Derivative.

Example:

If we have a car with a fixed forward velocity, and an ability to control the steering angle of the car.

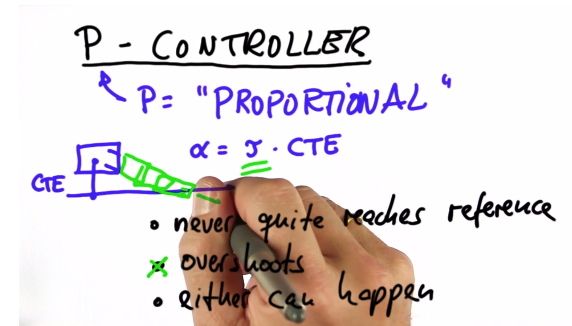
A solution is to make our steering angle proportional to the cross track error. Which is the distance between the vehicle and the reference trajectory.



This is an example of a proportional controller.

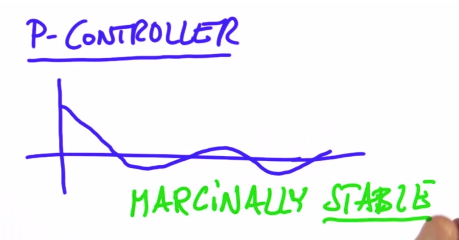
A proportional controller has 1 issue though.

It overshoots. In this example, as the car approaches the reference trajectory, in order to minimize the cross track error, it will pass the answer because the wheels are still pointing in the same direction, making it go past the line.



A P controller just oscillates around the reference, without actually converging.

This is called Marginally Stable/ Stable.

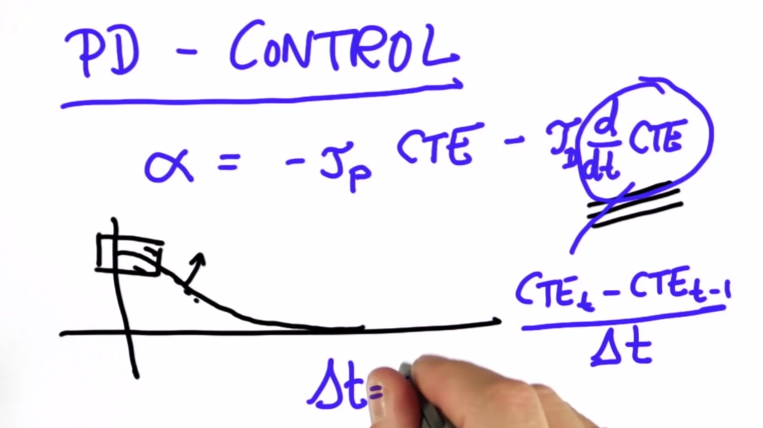


How do we avoid overshoot?

We add a derivative portion to the controller.

PD Control

The Derivative portion will steer the vehicle in the opposite direvtion proportional to the rate of change of the cross track error. This will lower the overshoot of the P controller. Now the controller will mostly converge to 0 cross track error.



**Systematic Bias**

Common problem in robotics.

Example: we got a car and believe the wheels are 100% aligned, but it turns out the mechanic aligned the wheel a little bit on an angle. In this situation, the controller is not general enough to still ensure the car has zero cross track error.

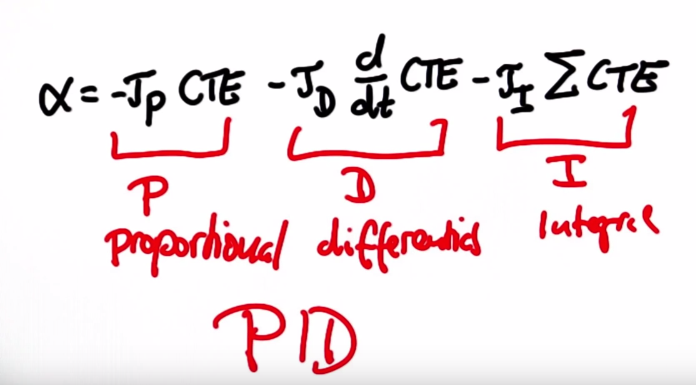
It would result in a larger error. That is equivalent to an offset in the cross track error.



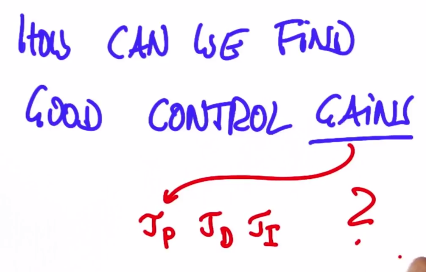
PID Controller.

Now to handle this issue, we need to keep track of the bias over time, to compensate for it. It helps minimize error that over a long period of time, we notice we can’t get rid of.

Adjusting to the bias (a sustained situation of large error), we track the Integral (Or In this case for a digital controller, the sum for all time) of the cross track error over time.



Now how do we find good control gains?



A great solution for this other than trial and error is an algorithm called **TWIDDLE**.

Or it is also called **Coordinate Ascent**.

In Twiddle, the algorithm optimize based on a goodness value of how well the control parameters work.

