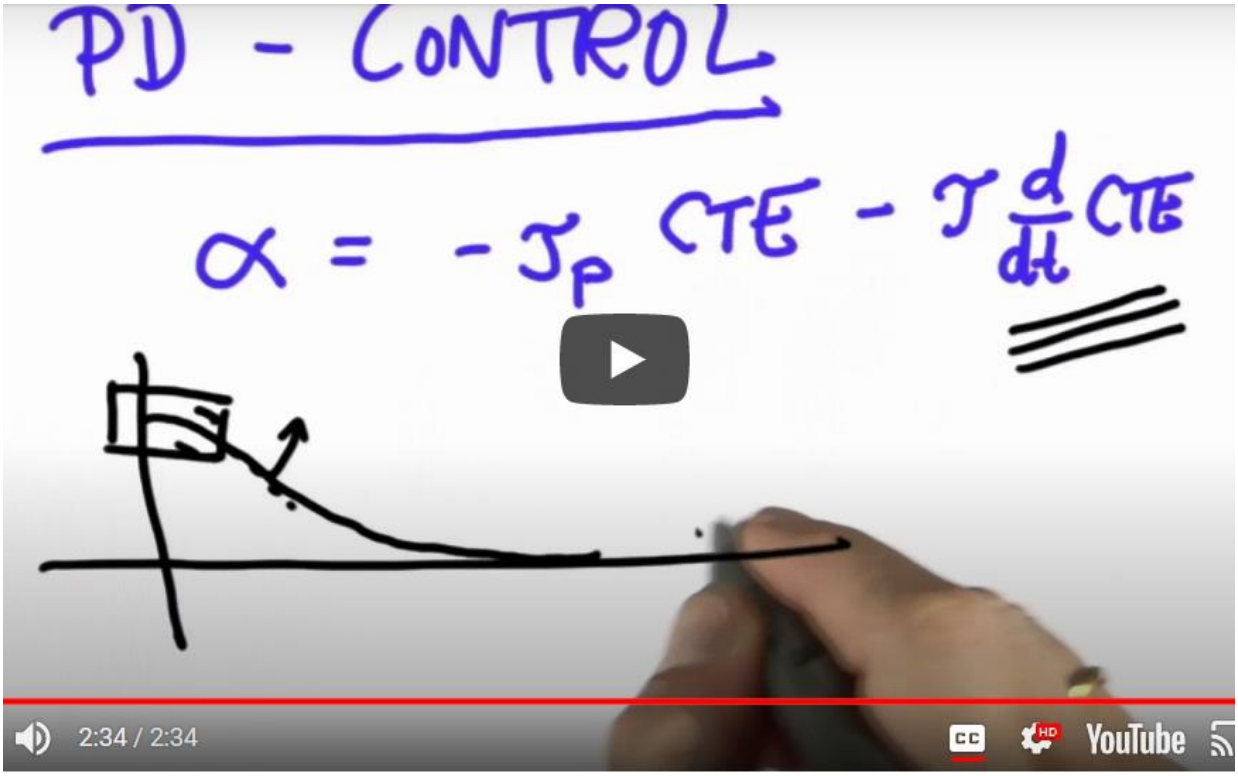
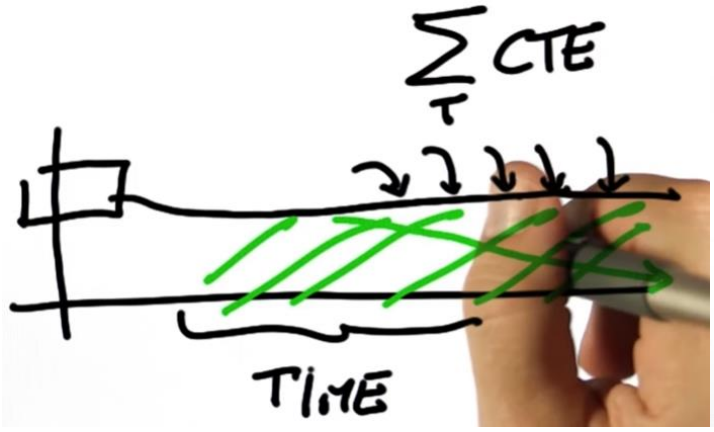
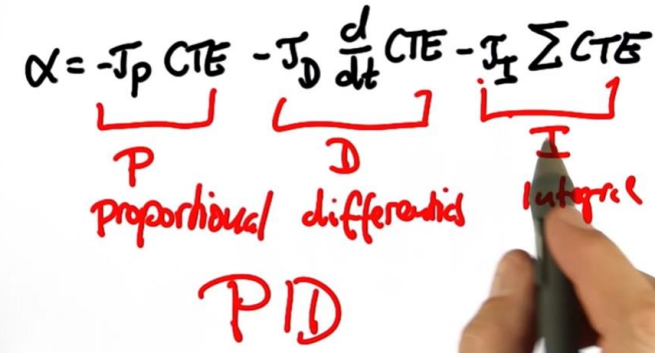


llowing quiz you'll implement the PD controller.

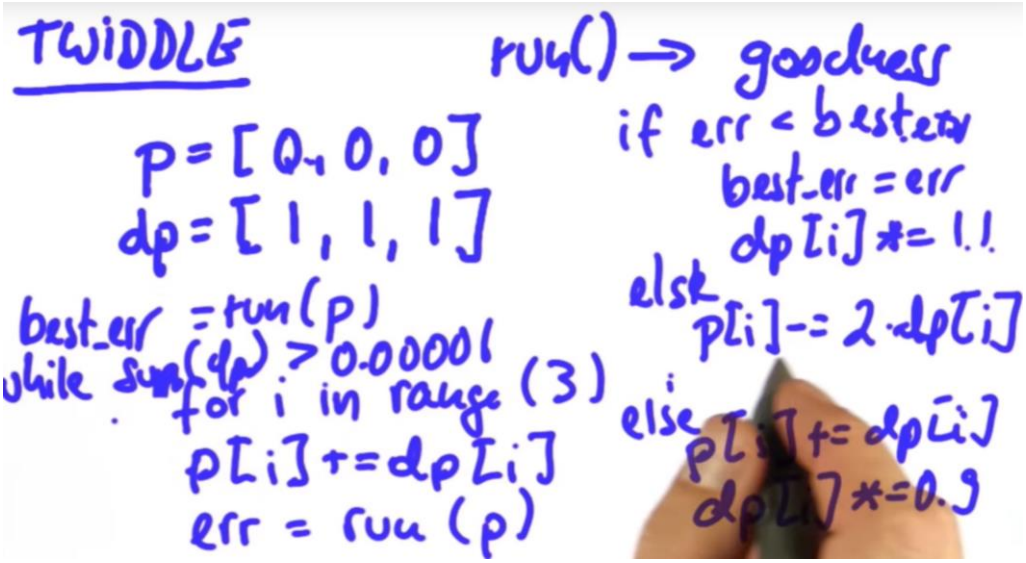
```
        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)
    return x_trajectory, y_trajectory

robot = Robot()
robot.set(0, 1, 0)

def run(robot, tau_p, tau_d, n=100, speed=1.0):
    x_trajectory = []
    y_trajectory = []
    # TODO: your code here
    prev_error = robot.y
    for i in range(n):
        error = robot.y
        diff_error = error - prev_error
        steer = -tau_p * error + (-tau_d * diff_error)
        prev_error = error
        robot.move(steer, speed)

        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)

    return x_trajectory, y_trajectory

x_trajectory, y_trajectory = run(robot, 0.2, 3.0)
```

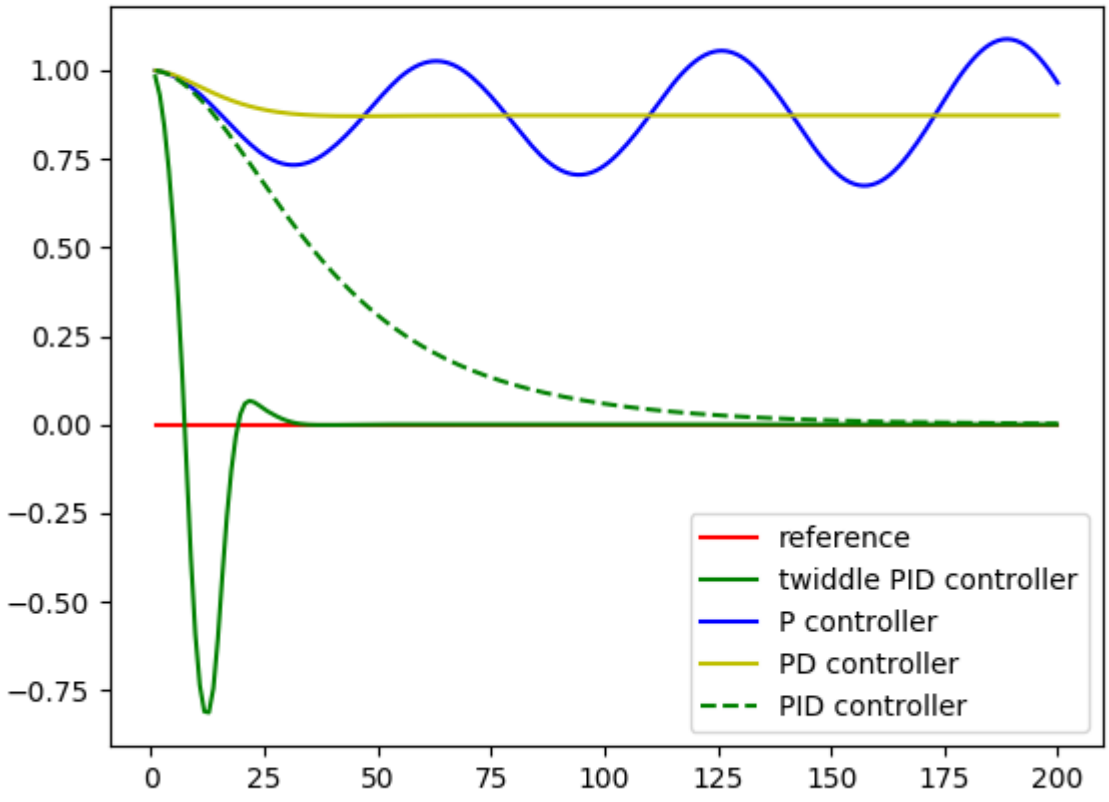
The integral factor is needed when there is a systemic bias

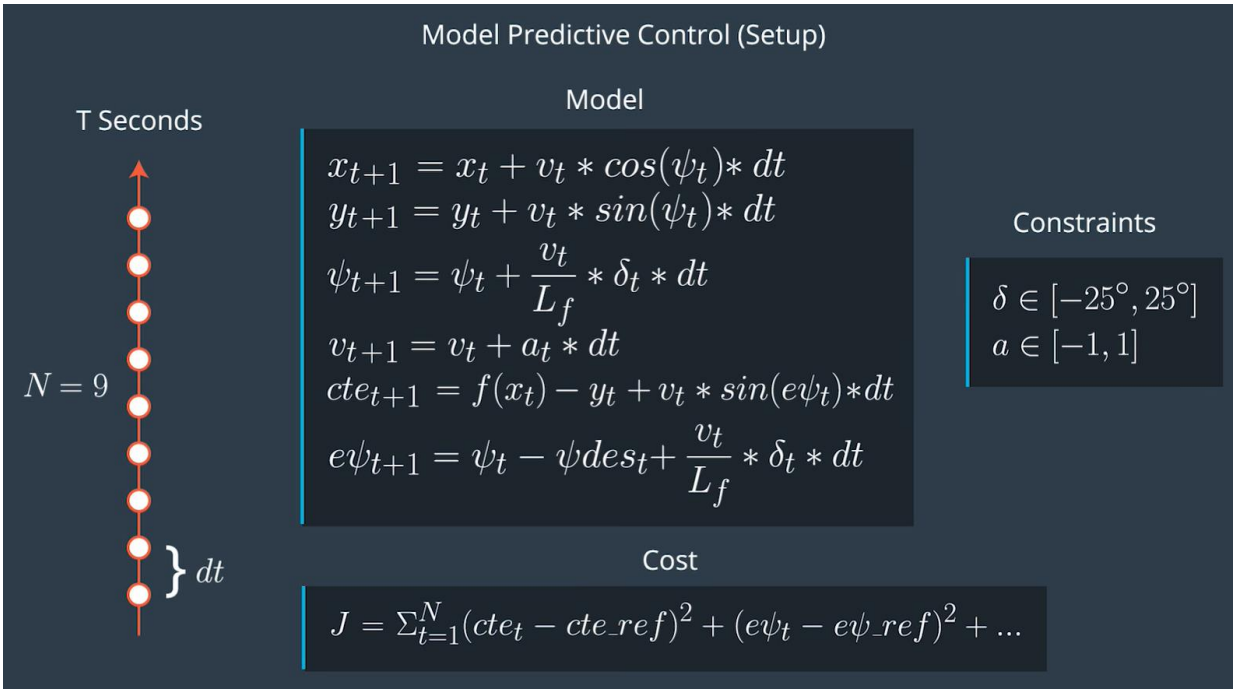


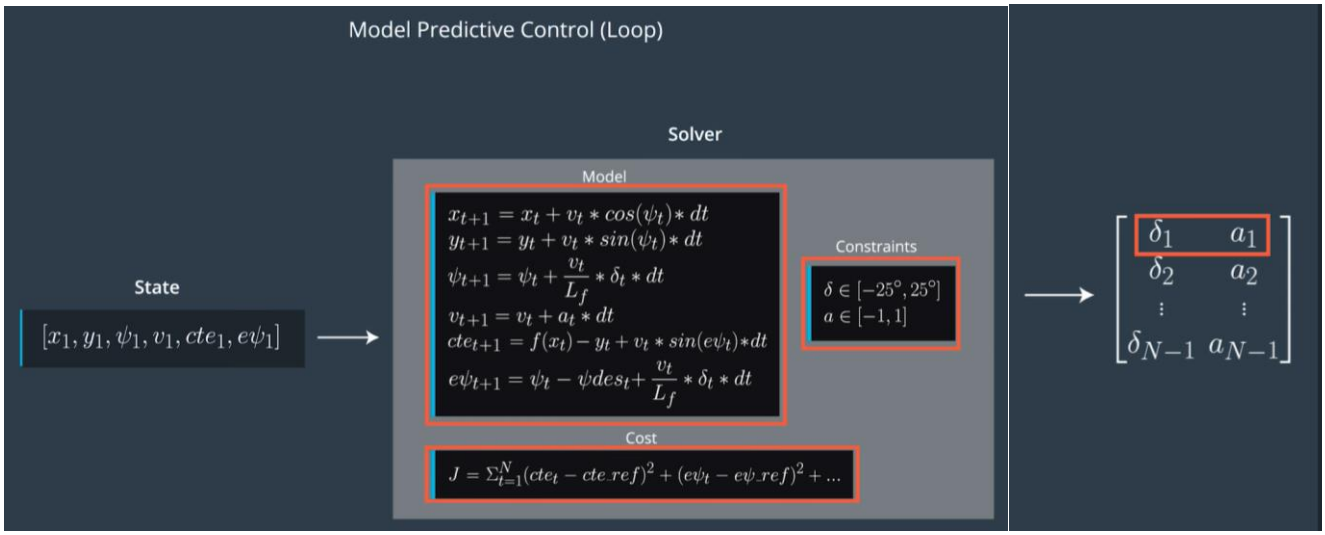


To find the most optimal parameters, let's implement the following algorithm:









Latency

In a real car, an actuation command won't execute instantly - there will be a delay as the command propagates through the system. A realistic delay might be on the order of 100 milliseconds.

This is a problem called "latency", and it's a difficult challenge for some controllers - like a PID controller - to overcome. But a Model Predictive Controller can adapt quite well because we can model this latency in the system.

PID Controller

PID controllers will calculate the error with respect to the present state, but the actuation will be performed when the vehicle is in a future (and likely different) state. This can sometimes lead to instability.

The PID controller could try to compute a control input based on a future error, but without a vehicle model it's unlikely this will be accurate.

Model Predictive Control

A contributing factor to latency is actuator dynamics. For example the time elapsed between when you command a steering angle to when that angle is actually achieved. This could easily be modeled by a simple dynamic system and incorporated into the vehicle model. One approach would be running a simulation using the vehicle model starting from the current state for the duration of the latency. The resulting state from the simulation is the new initial state for MPC.

Thus, MPC can deal with latency much more effectively, by explicitly taking it into account, than a PID controller.