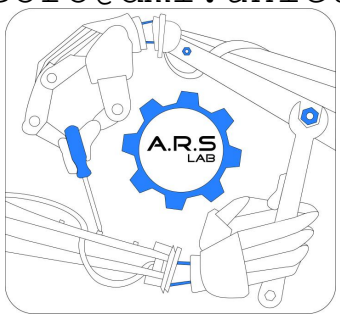
Controlling an Arm

Corrado Santoro

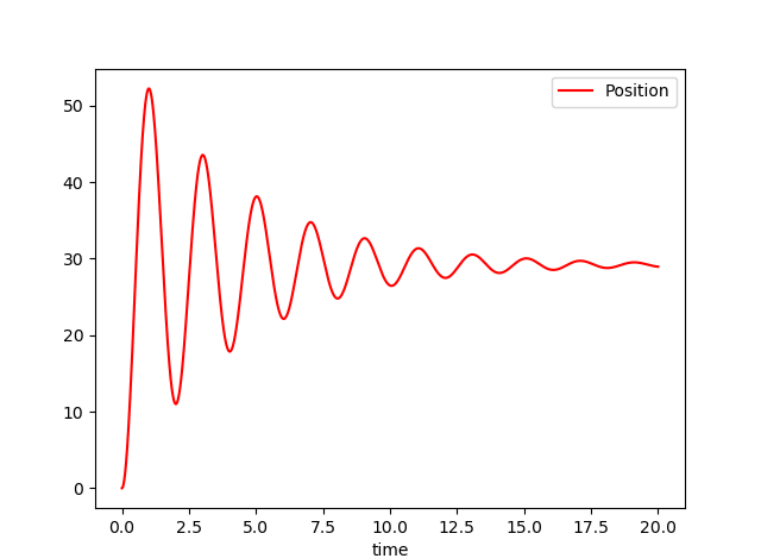
ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



Robotic Systems

Let's analise the trend of simulation

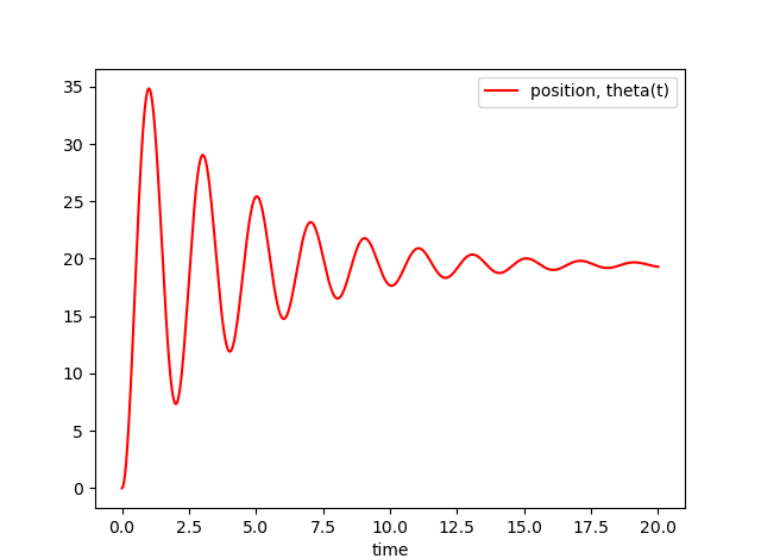
$$M = 1\ Kg, b = 0.8, r = 0.6\ m, M_o = 3\ Nm$$



With 3 Nm of torque we reach a position of (approx) 30 degrees

Let's analise the trend of simulation

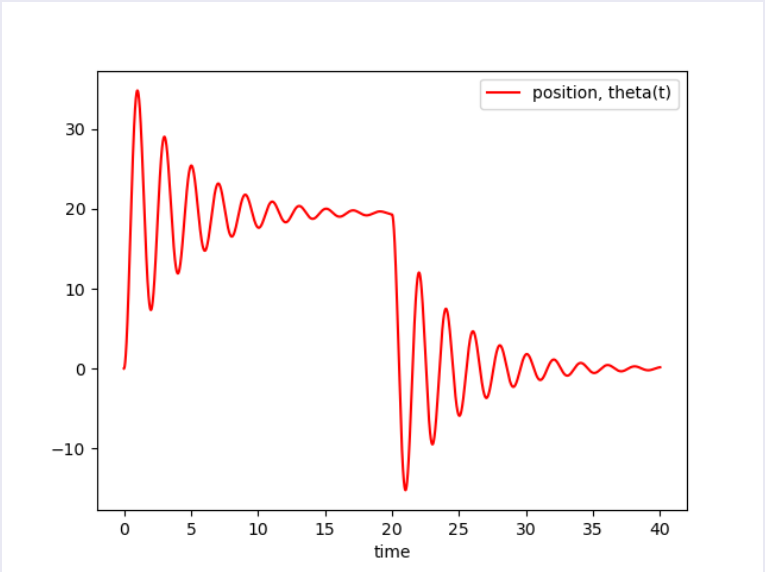
$$M = 1\ Kg, b = 0.8, r = 0.6\ m, M_o = 2\ Nm$$



With 2 Nm of torque we reach a position of (approx) 20 degrees

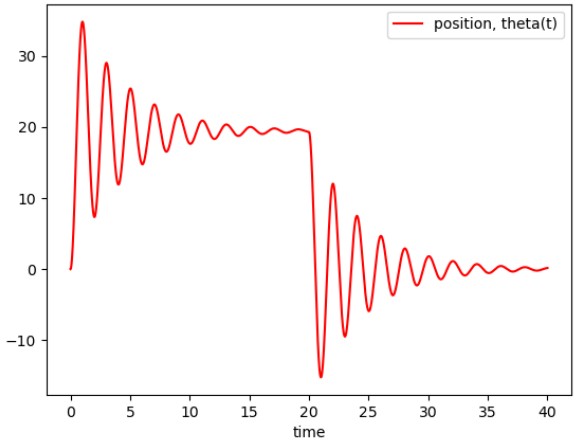
Let's analise the trend of simulation

Let's turn off the motor $M_o = 0$ after 20 seconds...



The mass returns to the equilibrium position $\theta = 0$

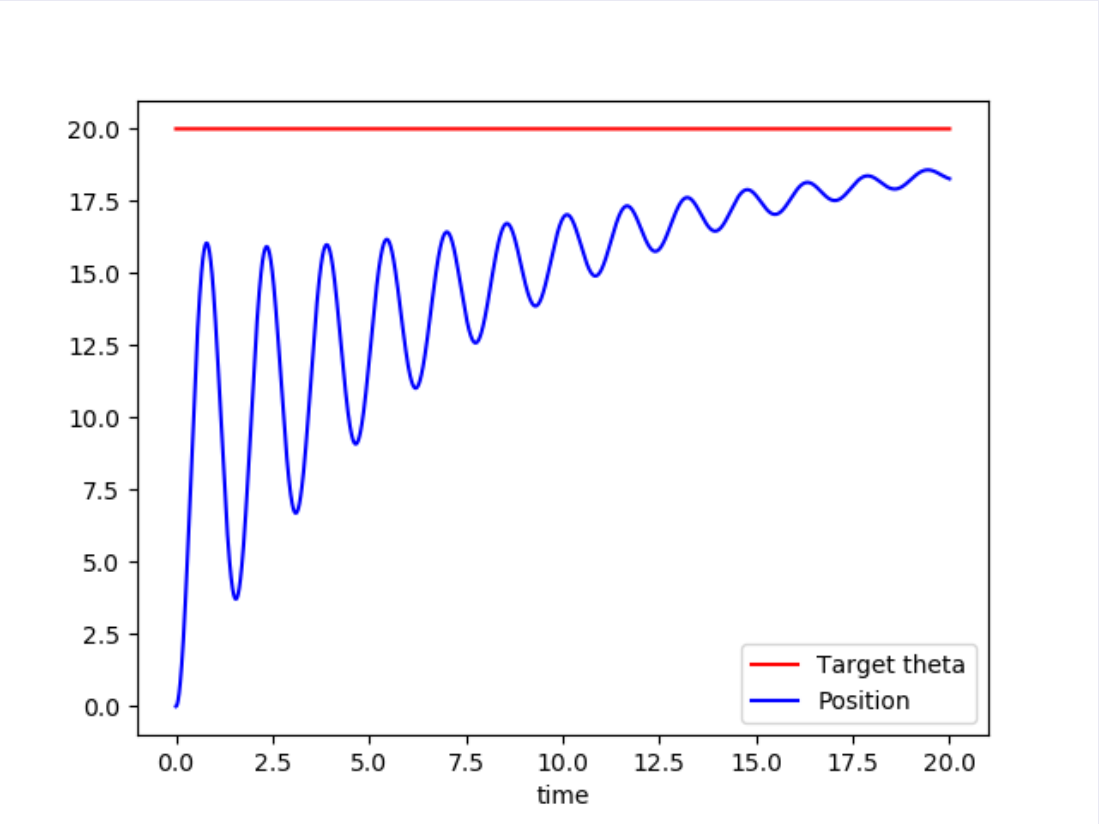
Controlling Arm Position



- As the simulation suggests, to reach and maintain a position we must continuosly apply a certain torque
- This is due to the presence of gravity
- A proportional controller does not work because, when the error is 0, “removing” the torque implies to abandon the target position
- We need (also in this case) an **accumulator**, so at least a **proportional-intergral (PI)** controller is needed
- The bad news is the presence of oscillations

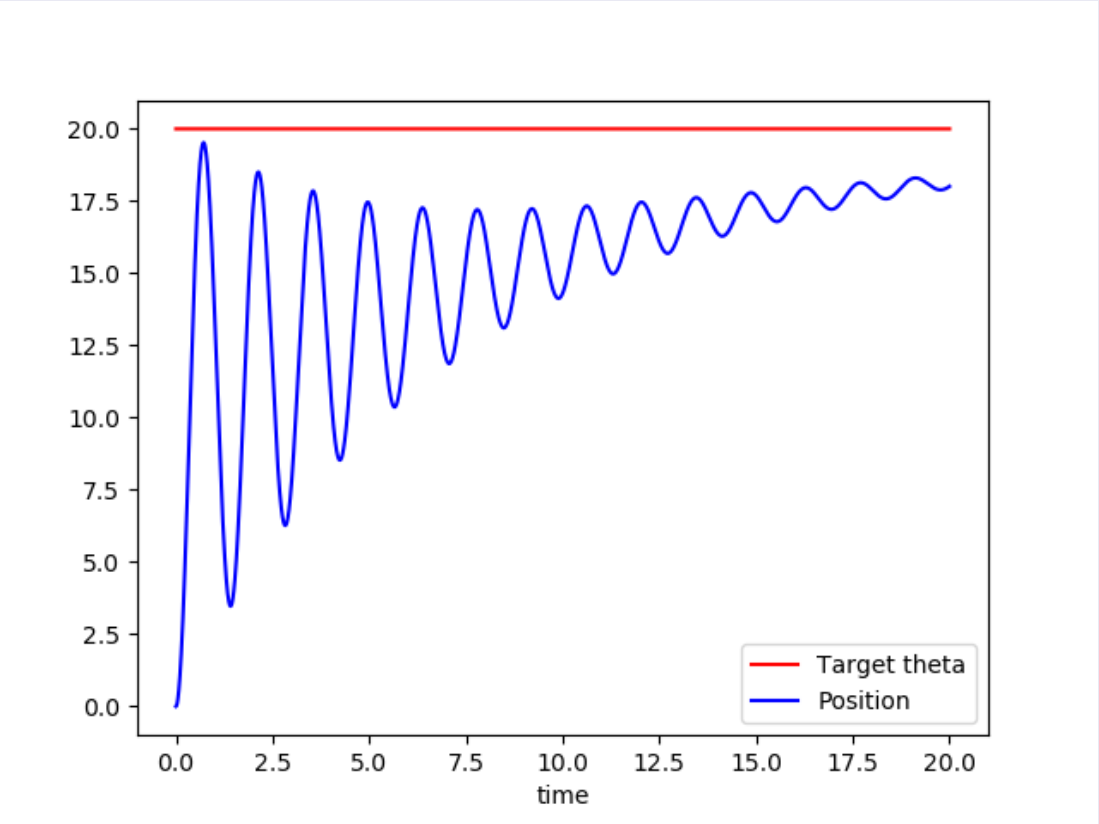
Using a PI Controller

$$Target = 20\ degrees, K_P = 4, K_I = 1$$



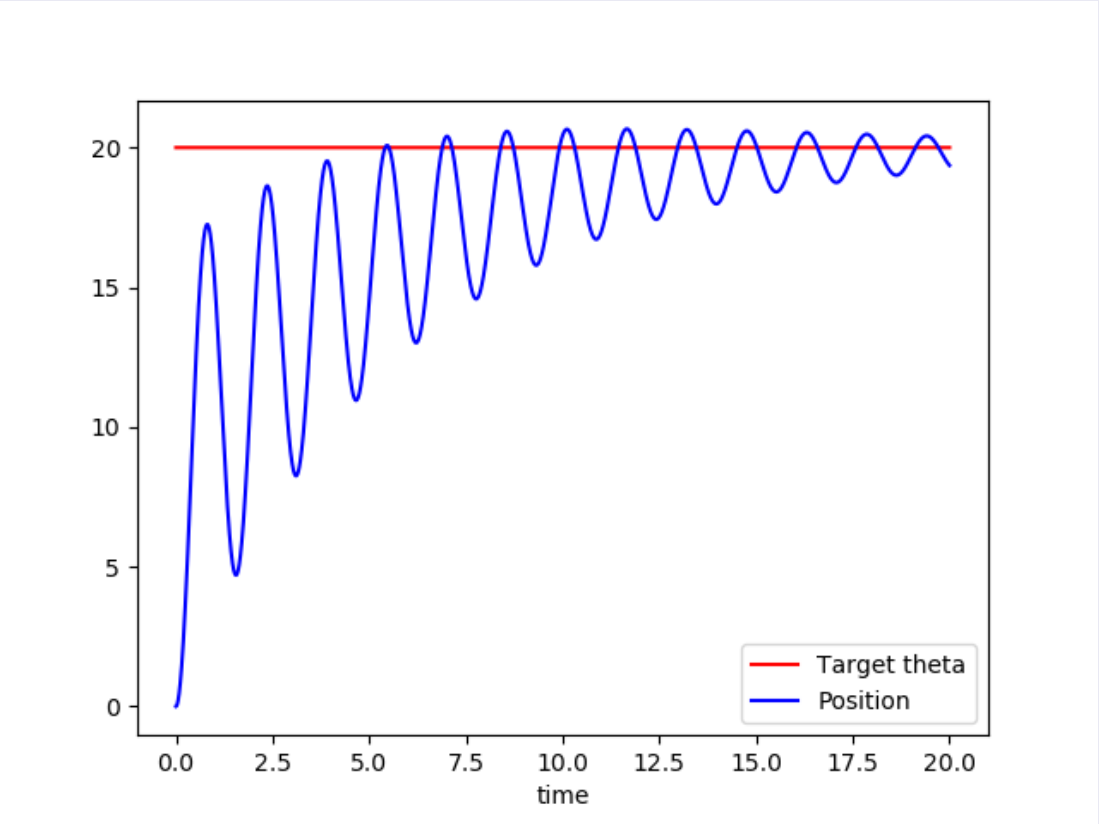
Using a PI Controller

$$Target = 20\ degrees, K_P = 6, K_I = 1$$



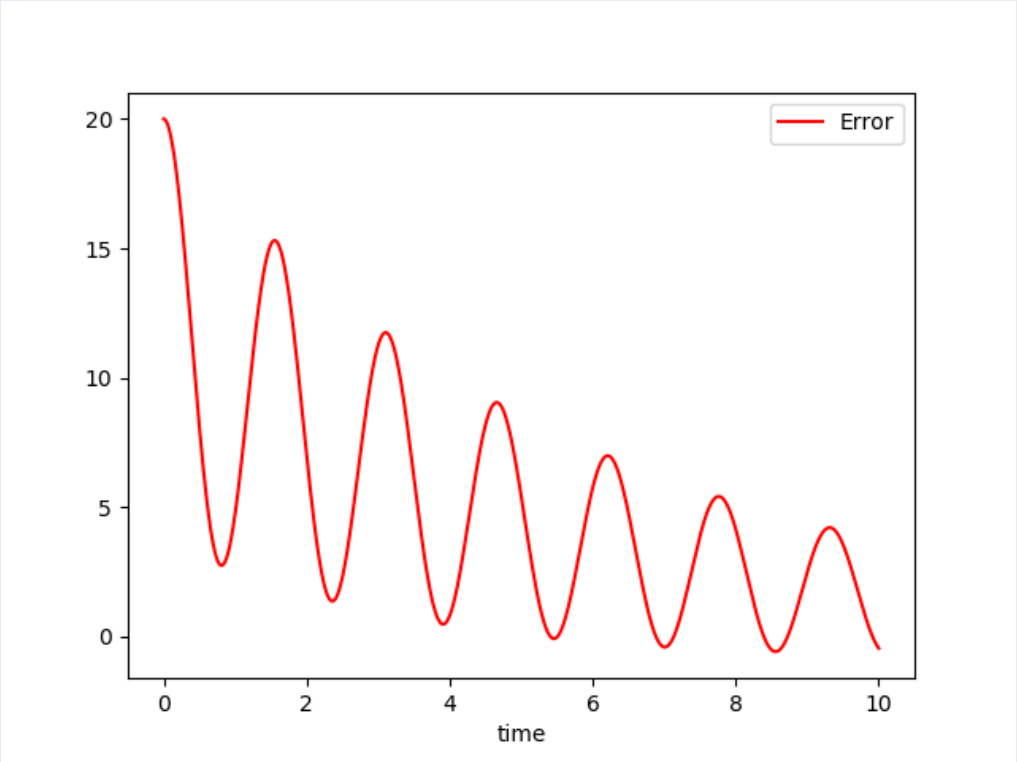
Using a PI Controller

$$Target = 20\ degrees, K_P = 4, K_I = 2$$



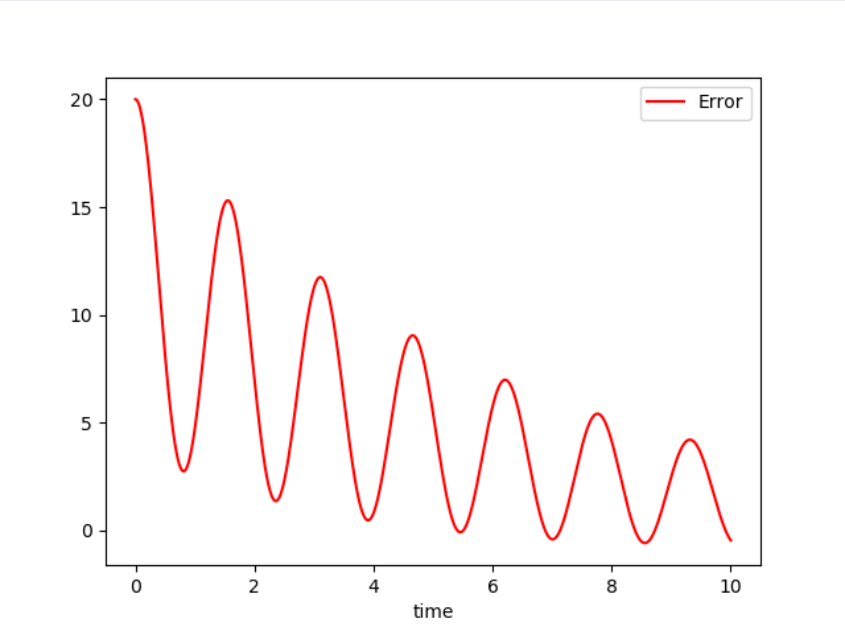
Arm Position Control

Let's Analyse the trend of the **error**



- When the error **decreases**, the control action is OK
- When the error **increases**, we need more control action

Let's **derivate** the **error**



$$\dot{e} = \frac{de(t)}{dt}$$

- When the error **decreases**, $\dot{e} < 0$, the control action is OK
- When the error **increases**, $\dot{e} > 0$, we can increase control action

The PID Controller

Let's add a factor **proportional to the derivative of the error**:

$$ControlAction = PI(error) + K_D\dot{e}$$

- When the error **decreases**, $\dot{e} < 0$, so the derivative action acts a as "brake" reducing the control action (this is OK, because the trend of the error is, in any case, decreasing)
- When the error **increases**, $\dot{e} > 0$, so the derivative factor **improves** the control action

The PID Controller

Discretizing the derivative, we have:

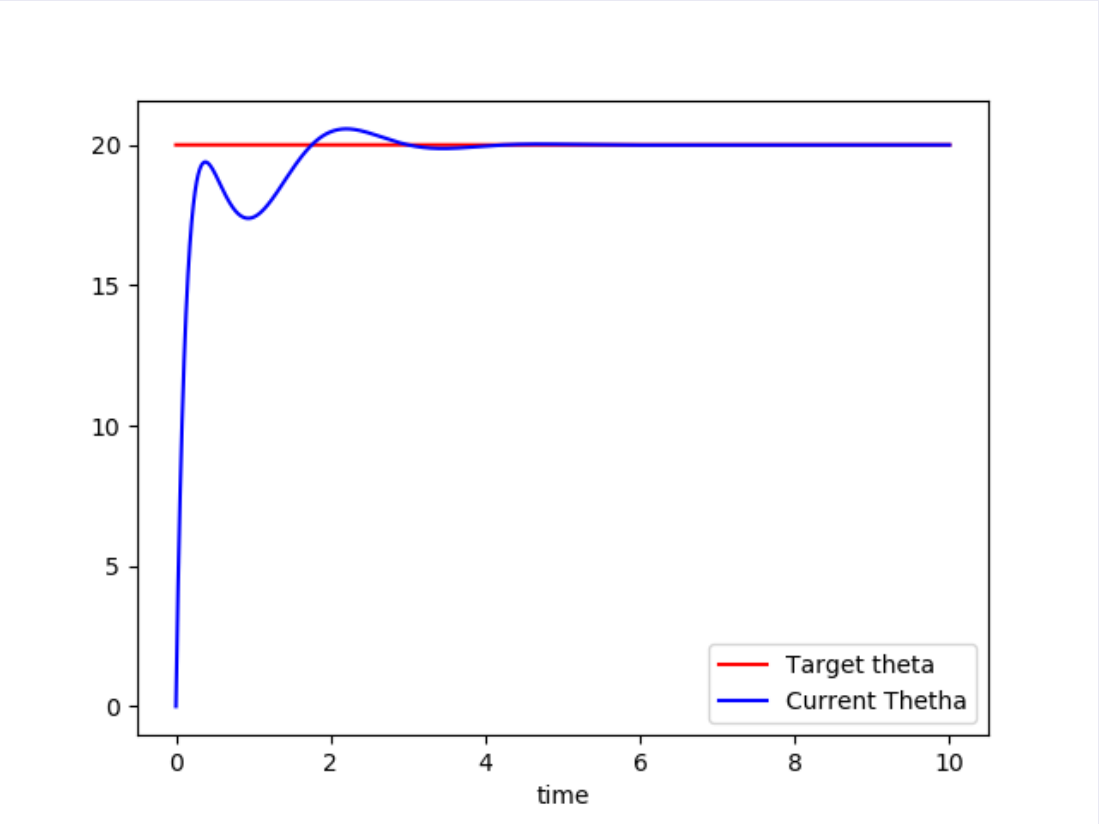
$$ControlAction = PI(error) + K_D \frac{e(t) - e(t - \Delta T)}{\Delta T}$$

PID = Proportional-Integral-Derivative

```
class PID:

    def __init__(self, kp, ki, kd):
        self.p = Proportional(kp)
        self.i = Integral(ki)
        self.kd = kd
        self.prev_error = 0


    def evaluate(self, delta_t, target, current):
        error = target - current

        derivative  = (error - self.prev_error) / delta_t

        self.prev_error = error

        return self.p.evaluate(target, current) + \
           self.i.evaluate(delta_t, target, current) + \
           derivative * self.kd
```

Using a PID Controller

$$Target = 20\ degrees, K_P = 8, K_I = 30, K_D = 5$$

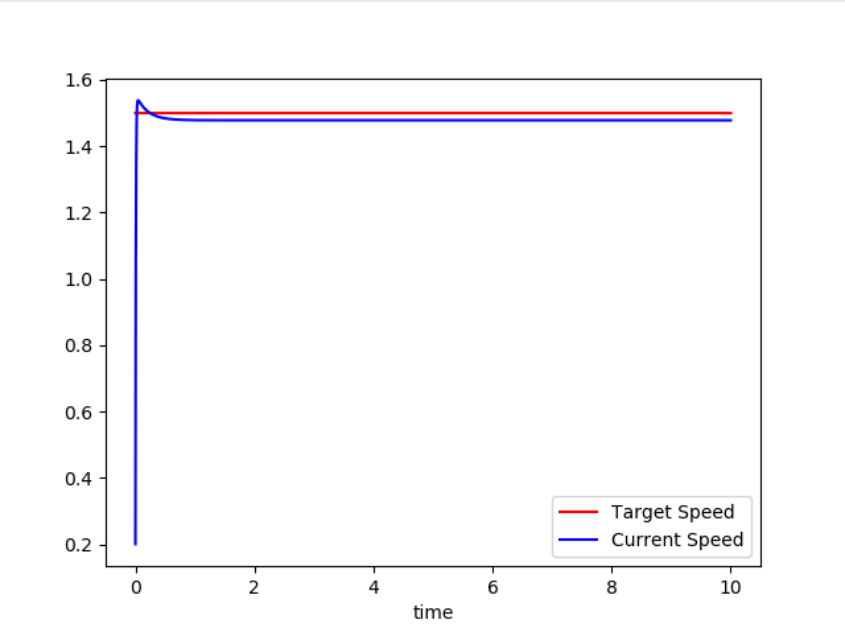


Arm Speed Control

The PID Controller

- Also in controlling the speed of the arm a PID controller is needed (PI surely!)
- The aim is to let the integral part compensate the variation of the weight of the mass on the basis of the angle and also push the arm towards rotation
- For these reasons costants may be high

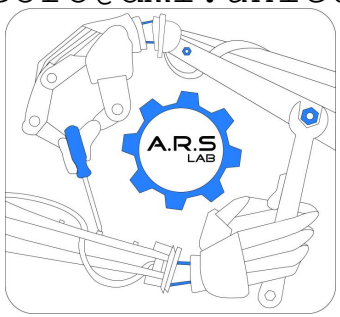
$$Target = 1.5\ rad/s, K_P = 80, K_I = 400, K_D = 0$$



Controlling an Arm

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



Robotic Systems