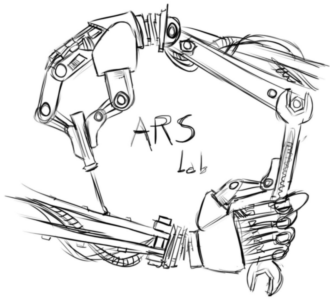
Il Controllo Derivativo
I Controllori Proporzionale-Integrale-Derivativo (PID)

Corrado Santoro

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



Programmazione Sistemi Robotici

Controllori Proporzionale-Integrale: Riassunto

Controllo P

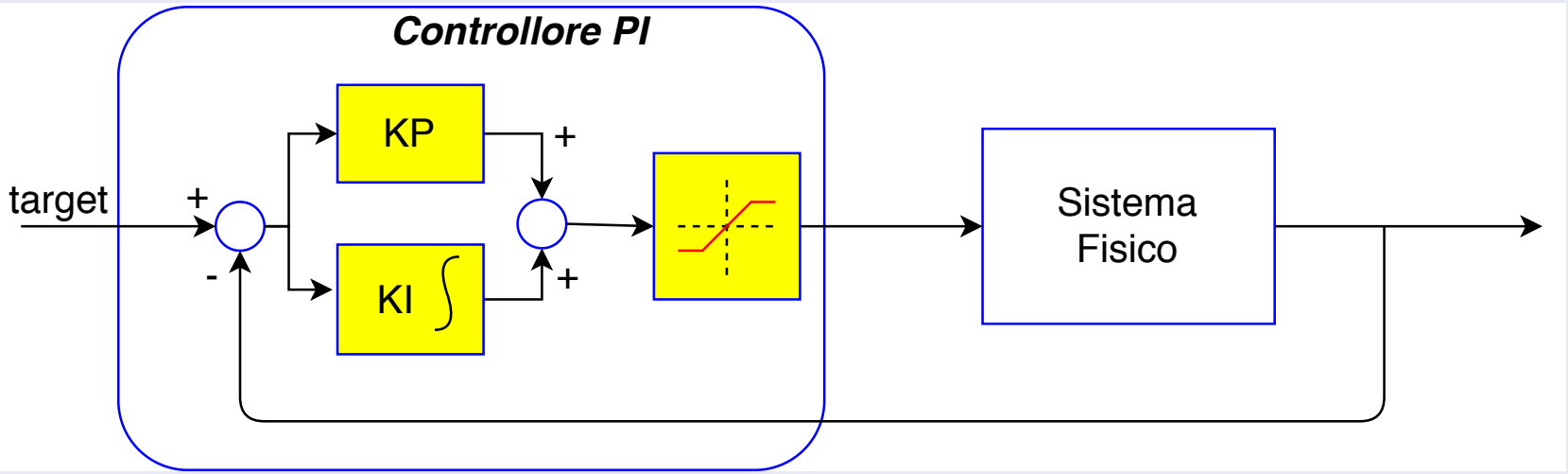
- Genera un'azione di controllo **proporzionale** all'errore tra valore target e valore della variabile da controllare
- Genera un'azione di controllo **istantanea** che tuttavia **non permane nel tempo**

Controllo I

- **Accumula** un'azione di controllo **proporzionale** all'errore tra valore target e valore della variabile da controllare
- Genera un'azione di controllo **ritardata** che **permane nel tempo**

Controllori Proporzionale-Integrale: Riassunto

Controllo PI



Controllore Derivatore

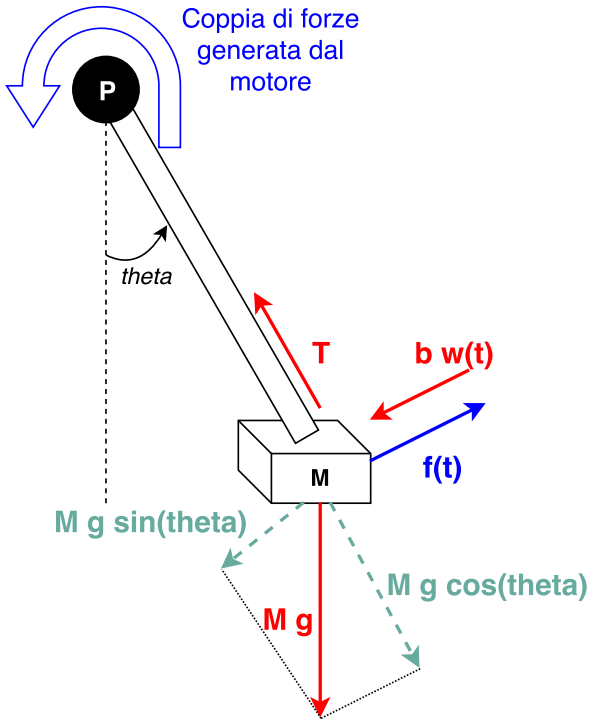
Controllo D

- Il controllore PI può non essere sufficiente nel caso di sistemi particolarmente “lenti” (sistemi meccanici con **elevata inerzia**)
- Il controllore PI **non può** ridurre la dinamica più di tanto senza provocare sovraelongazione e oscillazioni smorzate
- L’aggiunta di un’azione **derivativa** sull’errore $K_d \frac{de(t)}{dt}$ permette di agire sull’**andamento dell’errore**
- Poichè i sistemi dinamici agiscono sempre con un certo ritardo agli ingressi, l’informazione sull’andamento dell’errore permette al controllore di capire come sta evolvendo il sistema
- Un’azione basata sulla conoscenza dell’evoluzione del sistema consente di capire “in che direzione siamo andando” e pertanto **anticipare** l’azione di controllo
- Per tale motivo, i derivatori sono detti **sistemi anticipatori**

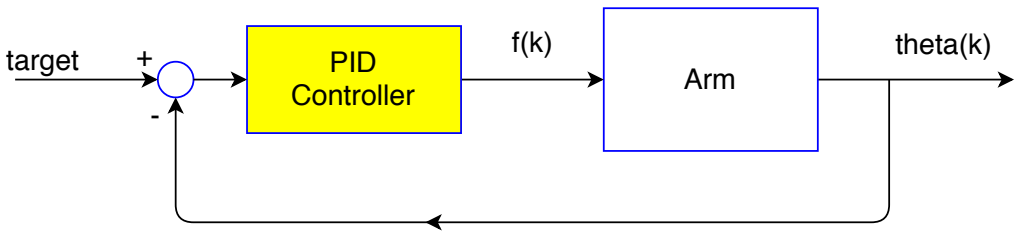
Esempio di Controllore PID

Controllo in posizione di un braccio robotico

- Consideriamo il sistema del secondo ordine che modella un giunto robotico
- Controlliamo **posizione angolare** con un PI (l'azione integrale è necessaria—è presente la gravità)

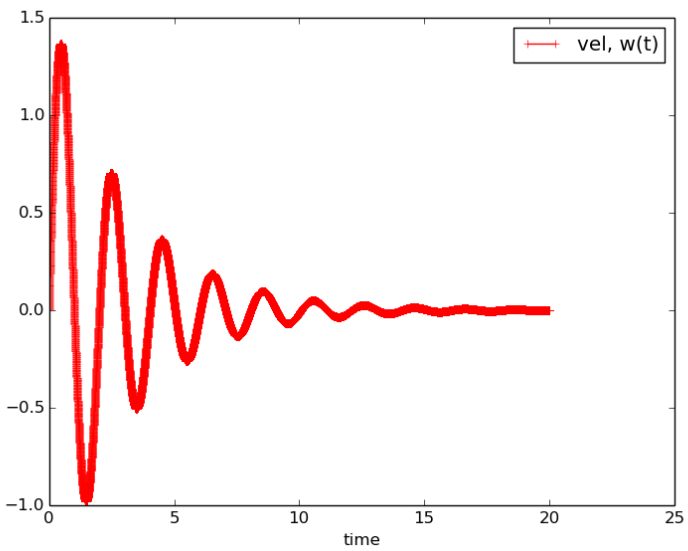


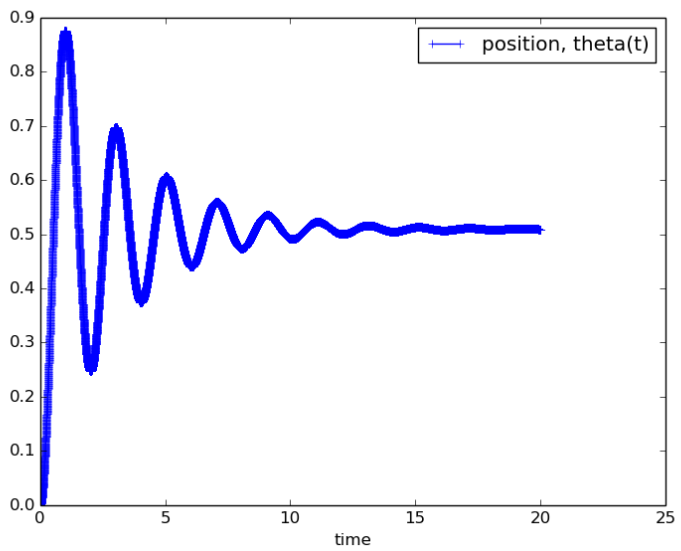
$$\begin{cases} \dot{\theta} &= \omega \\ \dot{\omega} &= -g\theta - \frac{b}{M}\omega + \frac{1}{M}f \end{cases}$$



Sistema a Ciclo Aperto

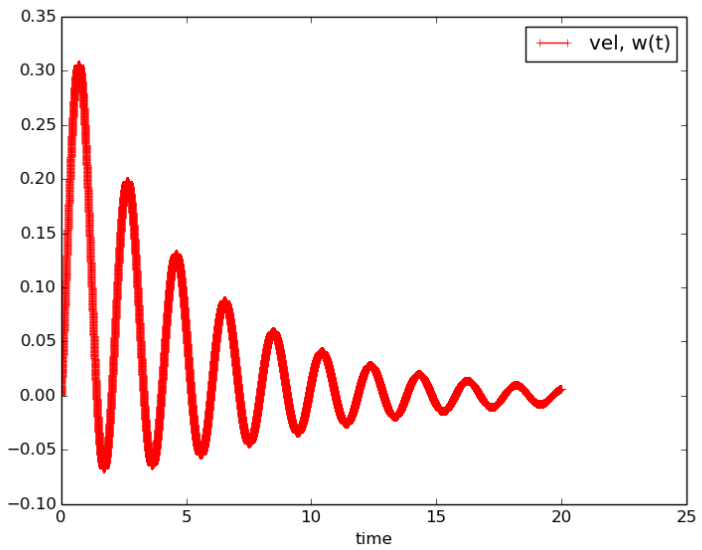
Risposta al gradino, $f(t) = 30, M = 6kg, b = 4$

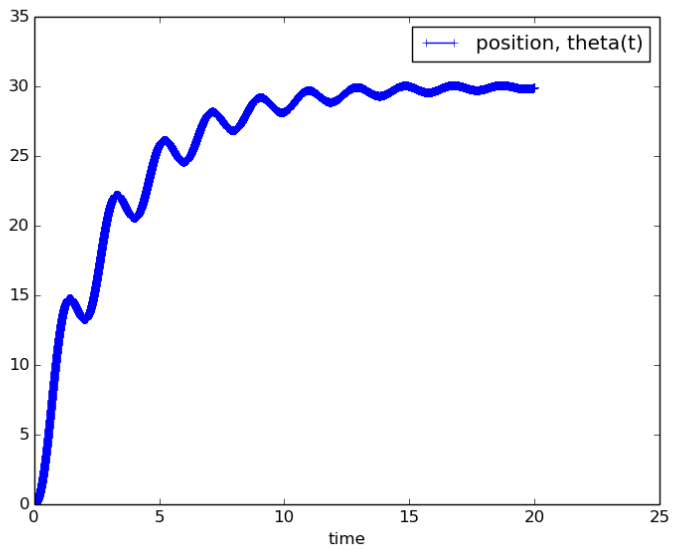




Sistema a Ciclo Chiuso

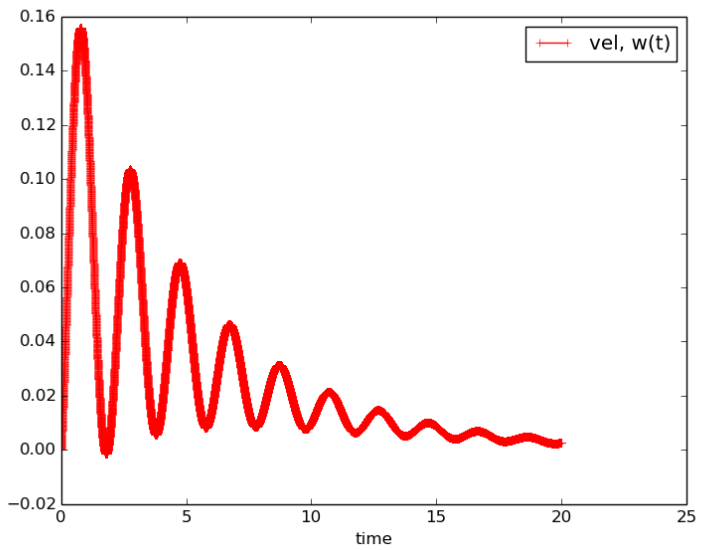
$target = 30\ gradi, K_P = 5, K_I = 20, Sat = 120N$

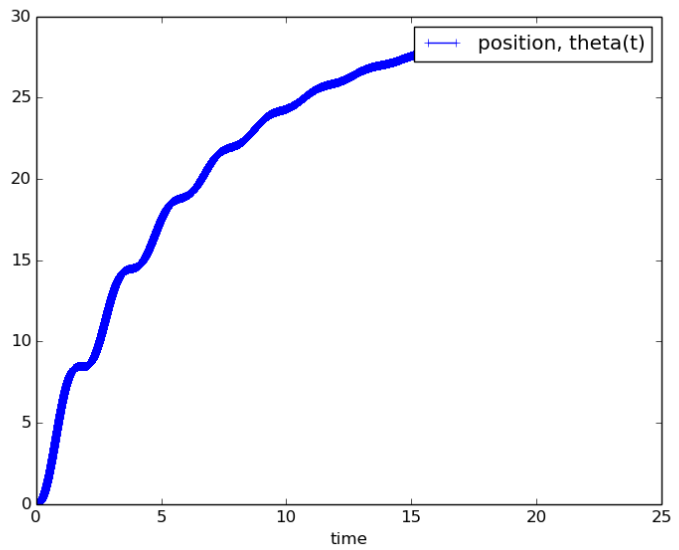




Sistema a Ciclo Chiuso

$target = 30\ gradi, K_P = 2, K_I = 10, Sat = 120N$





Controllore PI su sistema oscillante

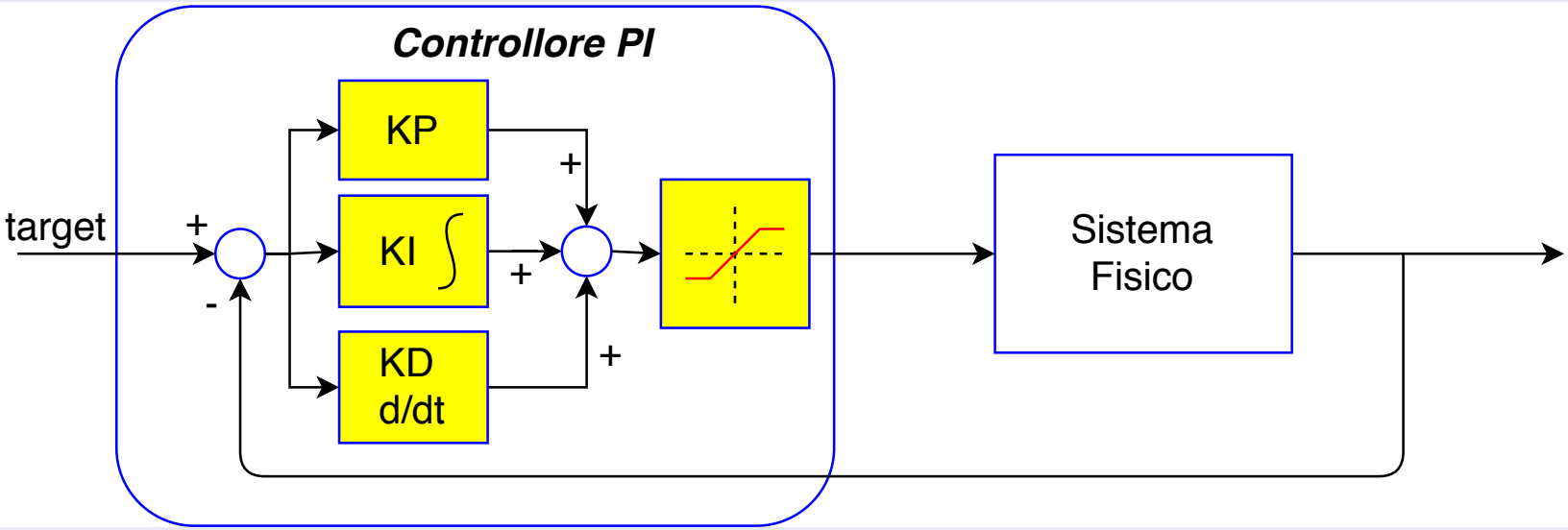
Effetto del controllo PI su un sistema oscillante

- Valori alti di K_P e K_I migliorano la dinamica, ma provocano la comparsa delle oscillazioni anche nel transitorio
- Valori bassi di K_P e K_I riducono le oscillazioni ma peggiorano la dinamica

Controllo "D"

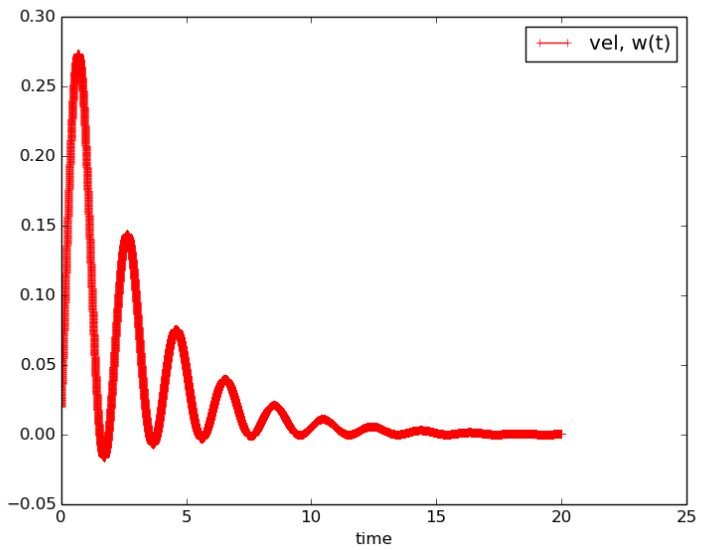
- L'aggiunta di un **derivatore** permette di "prevedere" l'andamento dell'errore e anticipare la risposta del sistema
- Come risultato si ha l'eliminazione delle oscillazioni senza alterare la dinamica

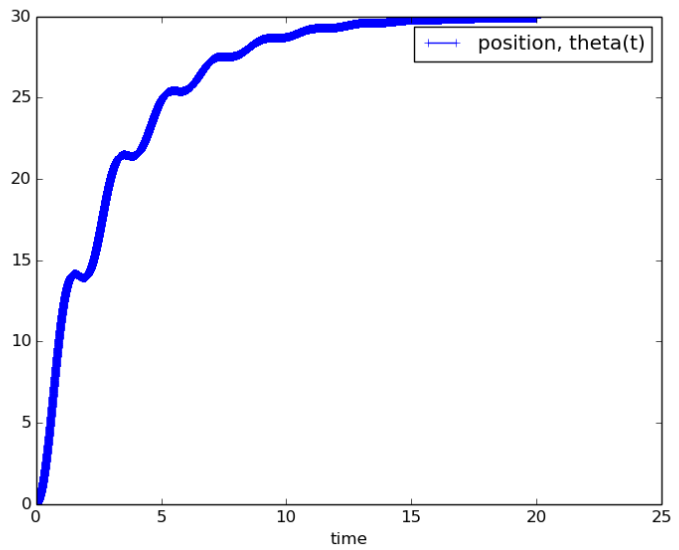
Controllo PID



Sistema a Ciclo Chiuso

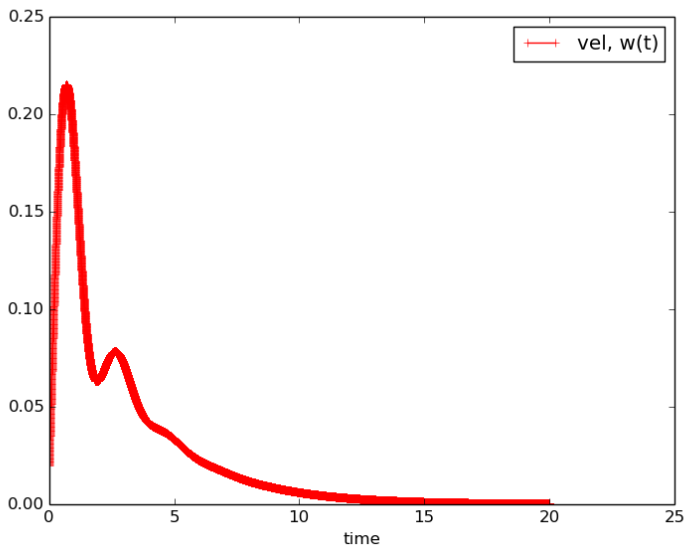
$target = 30\ gradi, K_P = 5, K_I = 20, K_D = 2, Sat = 120N$

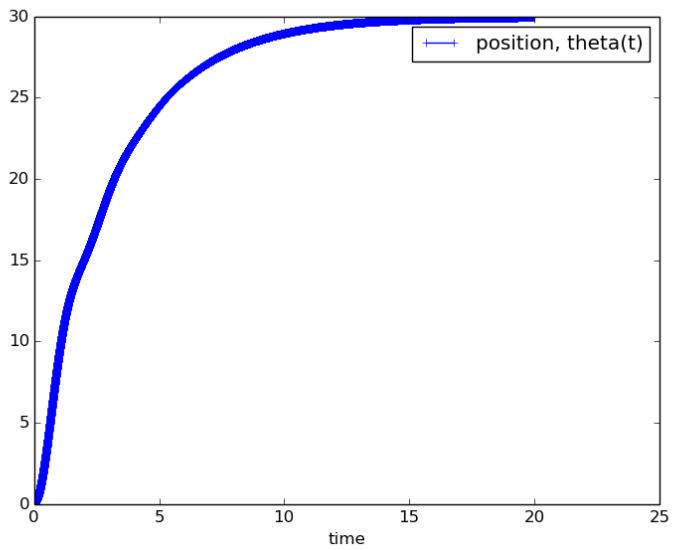




Sistema a Ciclo Chiuso

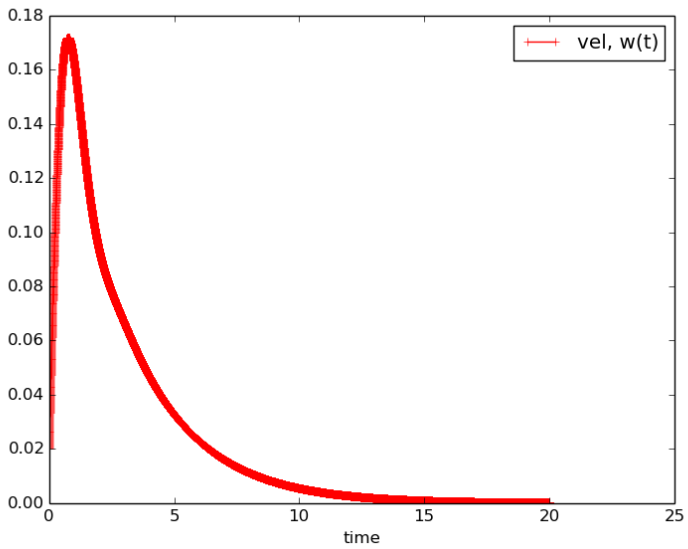
$target = 30\ gradi, K_P = 5, K_I = 20, K_D = 10, Sat = 120N$

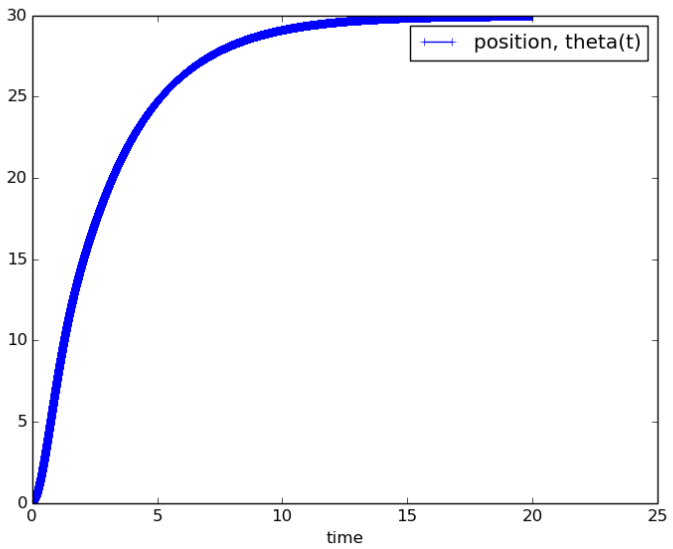




Sistema a Ciclo Chiuso

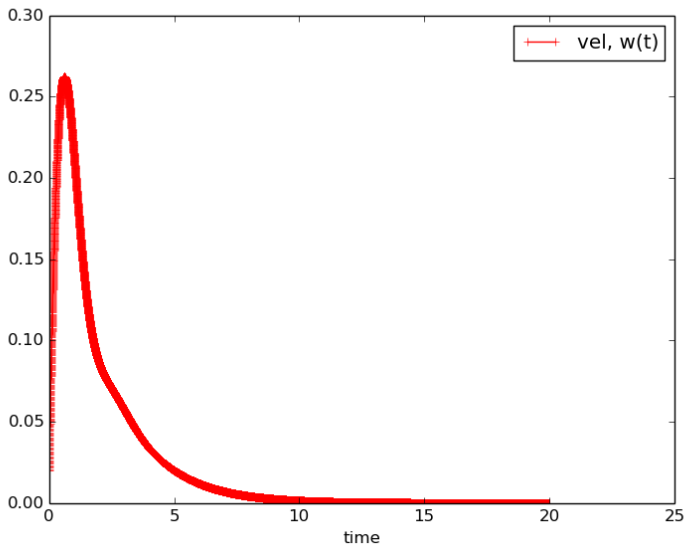
$target = 30\ gradi, K_P = 5, K_I = 20, K_D = 20, Sat = 120N$

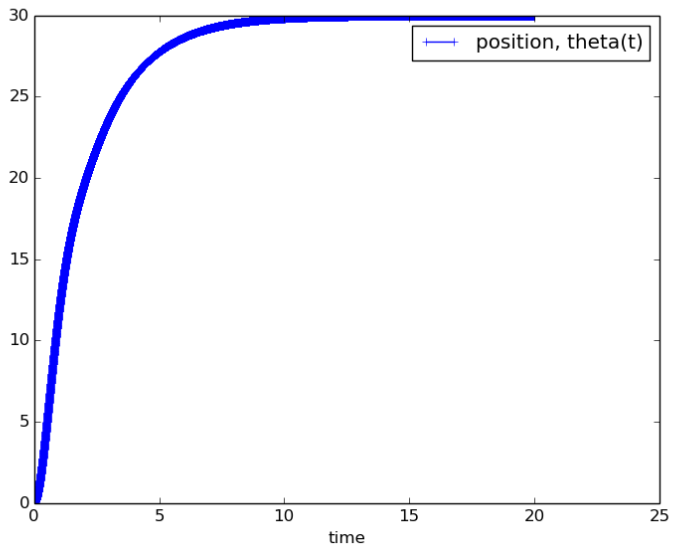




Sistema a Ciclo Chiuso

$target = 30\ gradi, K_P = 10, K_I = 30, K_D = 20, Sat = 120N$





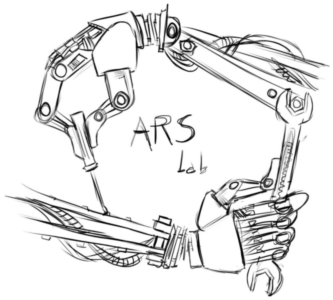
Controllo PID con Saturazione e Anti-Windup

```
class PIDSat:

    def __init__(self, kp, ki, kd, sat):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.saturation = sat
        self.integral = 0
        self.prev_error = 0
        self.saturation_flag = False

    def evaluate(self, target, current, delta_t):
        error = target - current
        if not(self.saturation_flag):
            self.integral = self.integral + error * delta_t
        deriv = (error - self.prev_error) / delta_t
        self.prev_error = error
        output = self.kp * error + self.ki * self.integral + self.kd * deriv
        if output > self.saturation:
            output = self.saturation
            self.saturation_flag = True
        elif output < -self.saturation:
            output = -self.saturation
            self.saturation_flag = True
        else:
            self.saturation_flag = False
        return output
```

Il Controllo Derivativo
I Controllori Proporzionale-Integrale-Derivativo (PID)

Corrado Santoro

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



Programmazione Sistemi Robotici