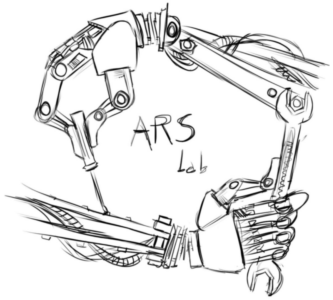
Controllo PI con saturazione

Corrado Santoro

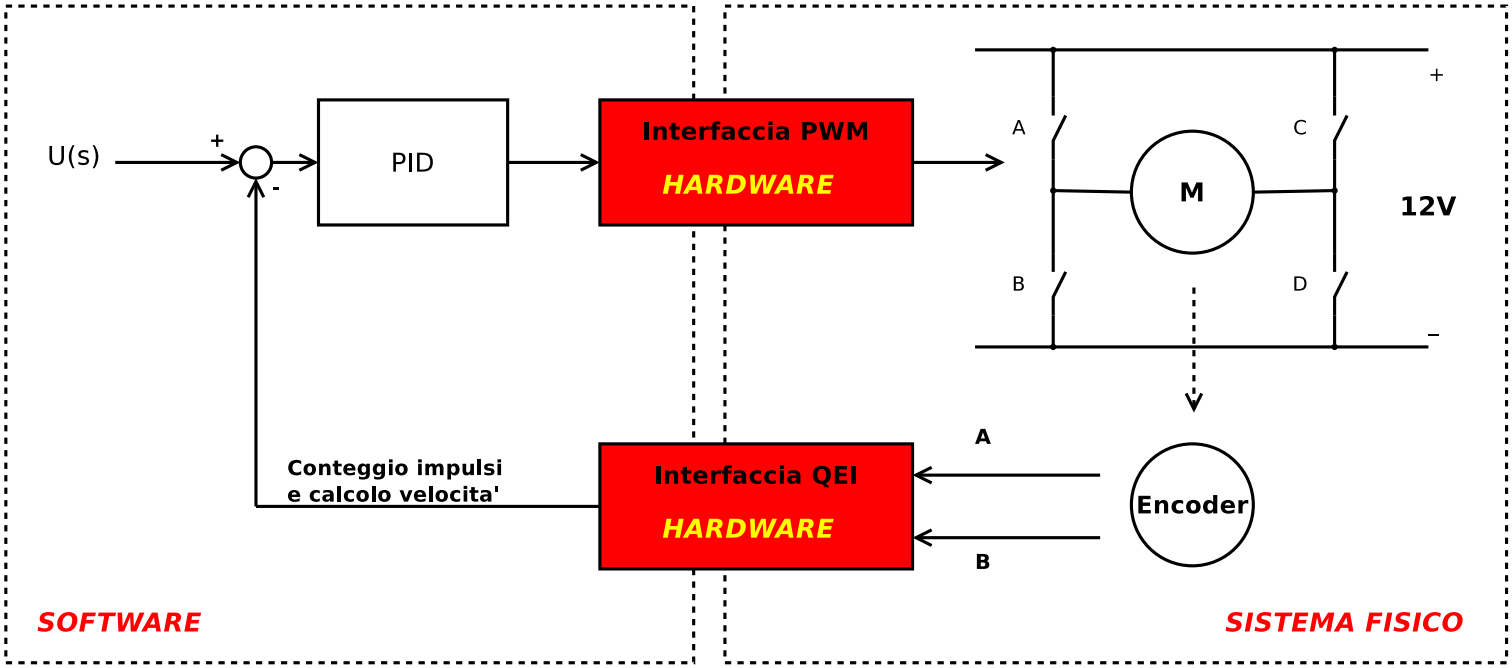
ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



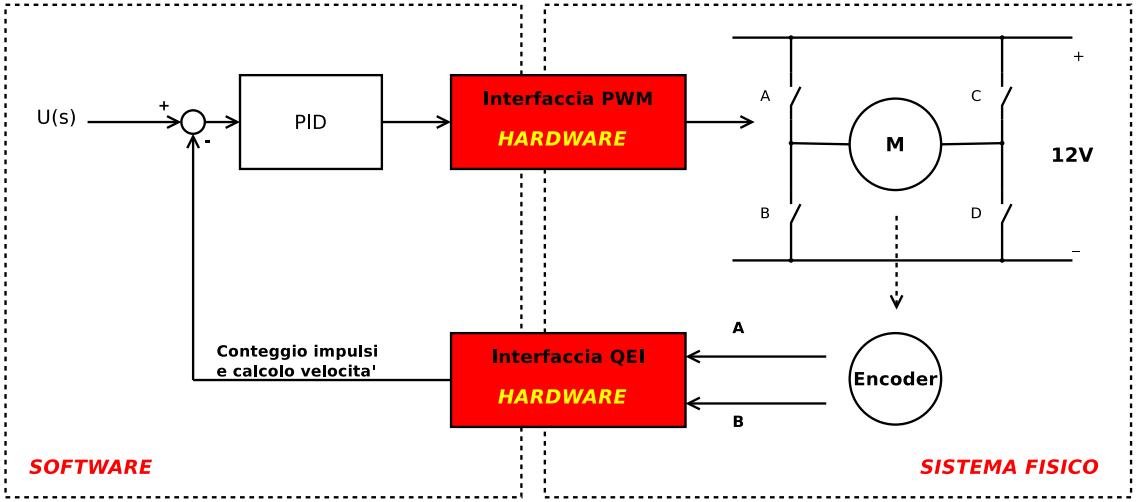
Programmazione Sistemi Robotici

Schema controllo PI di un motore in CC

- Il sistema complessivo per il controllo di un motore in CC è il seguente, ed include
  - L'algoritmo del PI (software)
  - L'intefaccia PWM più il ponte-H (hardware)
  - Il sistema motore + encoder (sistema fisico)
  - L'intefaccia QEI (hardware)

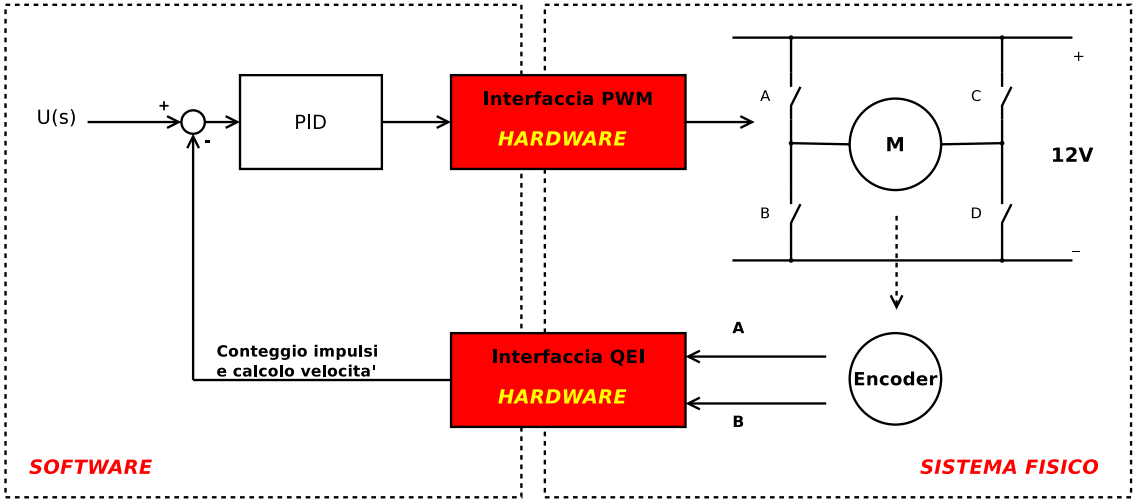


Limite del PWM



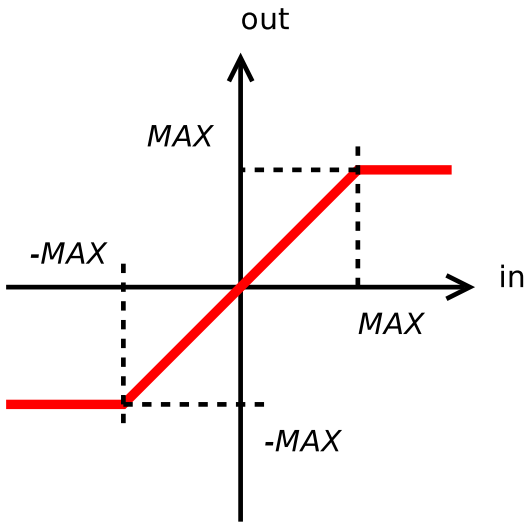
- Ogni sistema fisico ha un **limite** oltre il quale non si può andare
- Nel caso del motore in CC tale limite è rappresentato dalla sua coppia e velocità, che si ottengono quando il motore è alimentato alla sua tensione massima
- La tensione massima corrisponde al valore massimo che possiamo fornire al circuito PWM

Limite del PWM



- Il circuito generatore di PWM “accetta” valori nell’intervallo $[-MAX, MAX]$
- Tuttavia, l’output del PI può essere *qualunque numero* (il PI implementa una formula matematica)
- Pertanto è necessario **limitare** cioè **saturare** l’uscita del PI in modo che non vada oltre i limiti accettati dal circuito PWM

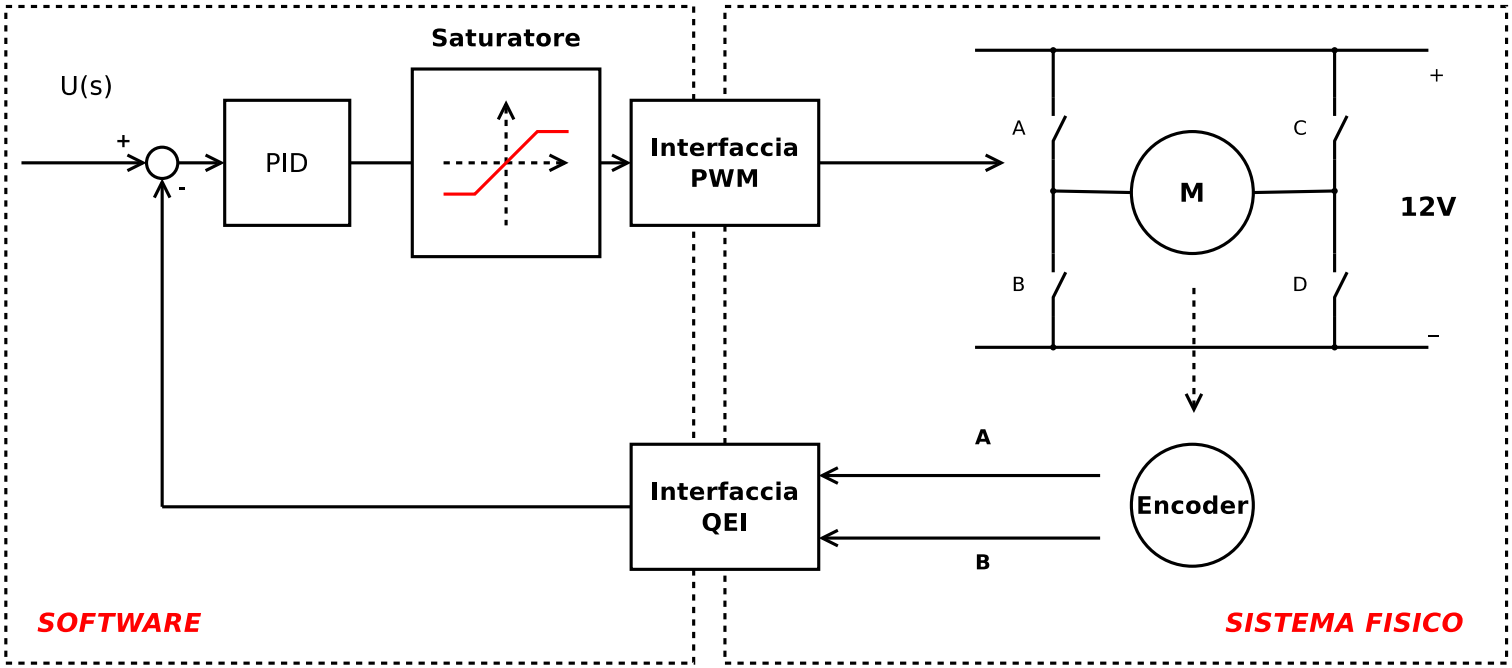
Saturazione



- Un blocco **saturatore** è rappresentato con il grafico *in/out* mostrato in figura
- Quando $-MAX \leq in \leq MAX$, l'uscita "ricopia" l'ingresso
- Quando $in < -MAX$ (risp. $in > MAX$), l'uscita vale $-MAX$ (risp. MAX)
- Questo si traduce nel seguente codice:

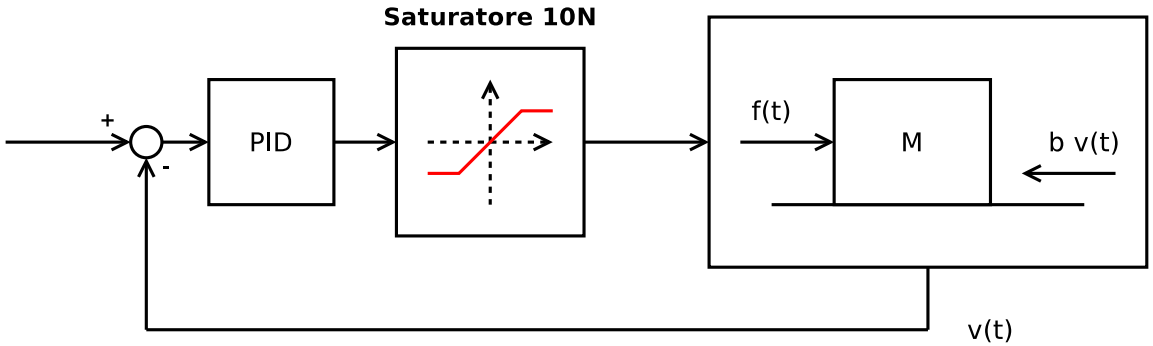
```
...
if (in < -MAX) out = -MAX;
else if (in > MAX) out = MAX;
else out = in;
...
```

PI con saturazione



- Il sistema completo è mostrato in figura
- Il codice del saturatore di solito è implementato all'interno della funzione del PI

Esempio di PI con saturazione



- Supponiamo il classico esempio di controllo (simulato) di massa su piano con attrito
- Consideriamo che l'output del PI (forza di spinta della massa) sia saturato ad un valore che non può superare i 10N
- $M = 1Kg, \quad b = 0.5Ns/m, \quad SAT = 10N$

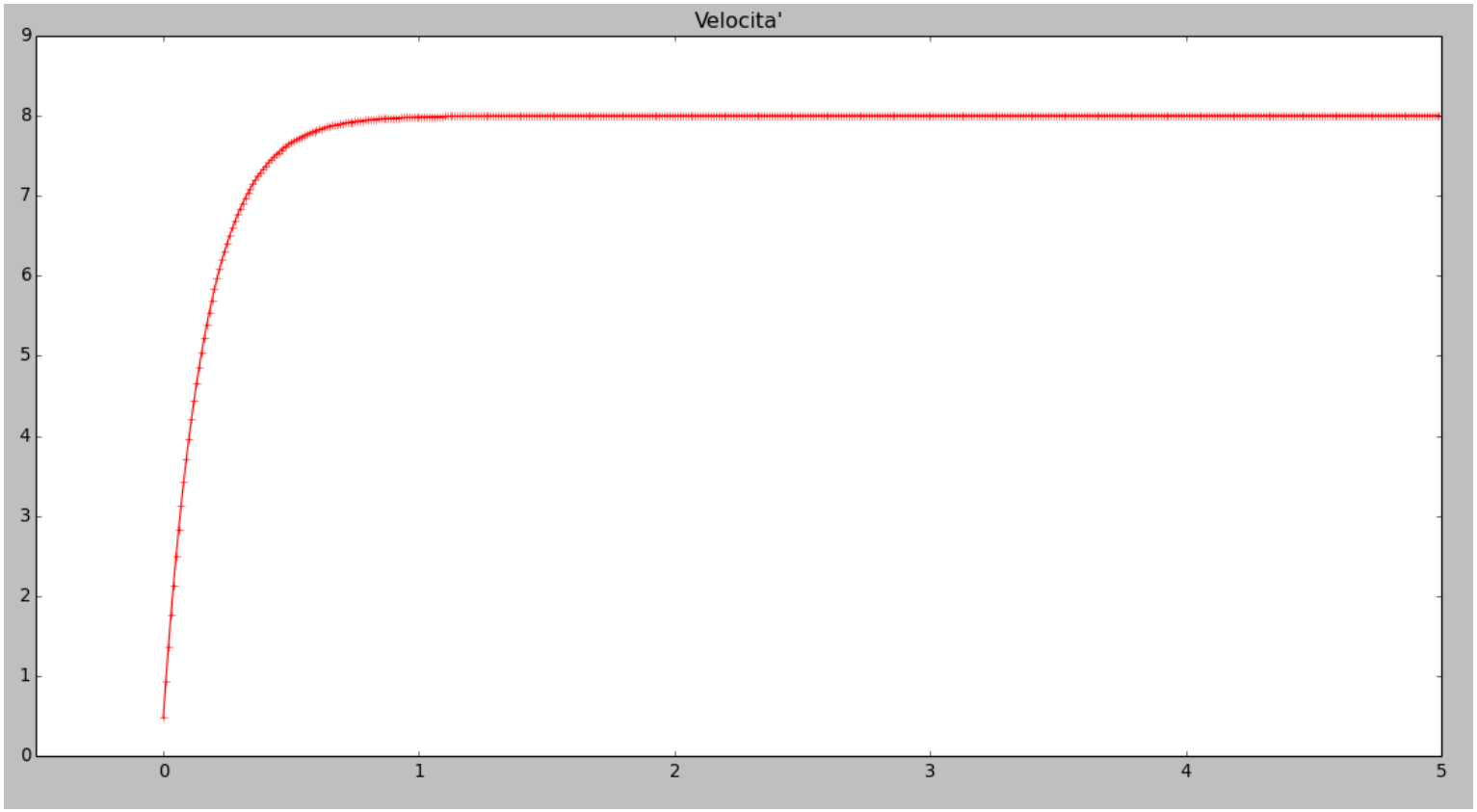
Codice del PI con saturazione (source file)

- Il codice del PI con saturazione diventa il seguente

```
class PI_Sat_Controller:

    def __init__(self, kp, ki, sat):
        self.__kp = kp
        self.__ki = ki
        self.__intergral_term = 0
        self.__saturation = sat

    def evaluate(self, target, measure, delta_t):
        self.__error = target - measure
        self.__intergral_term = self.__intergral_term + self.__error * delta_t
        output = self.__error * self.__kp + self.__intergral_term *
        self.__ki

        if output > self.__saturation:
            output = self.__saturation
        if output < -self.__saturation:
            output = -self.__saturation

        return output
```

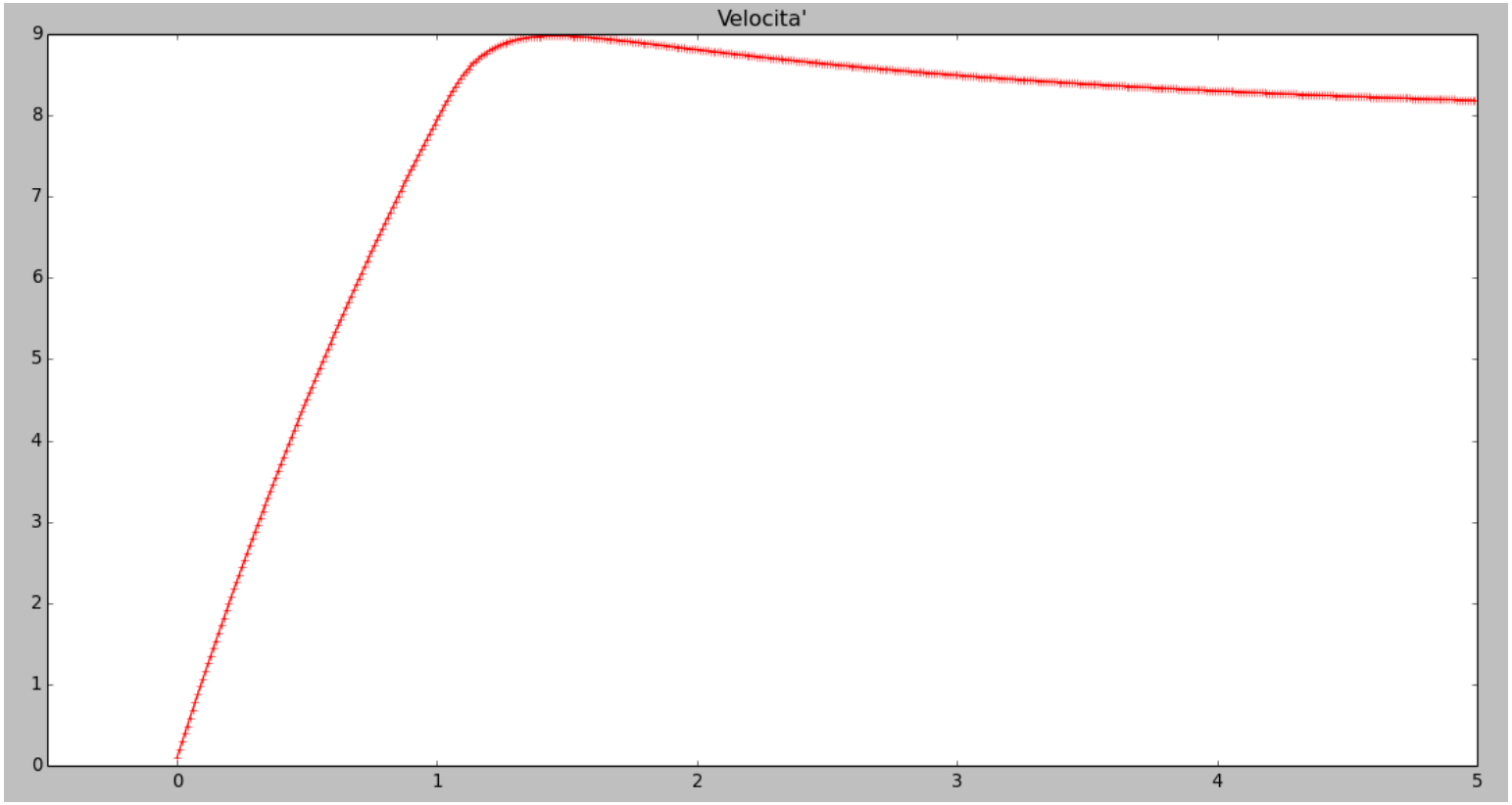
Risposta del sistema SENZA saturazione

$$v_{target} = 8m/s$$
$$K_p = 6,\ K_i = 3$$



Risposta del sistema CON saturazione

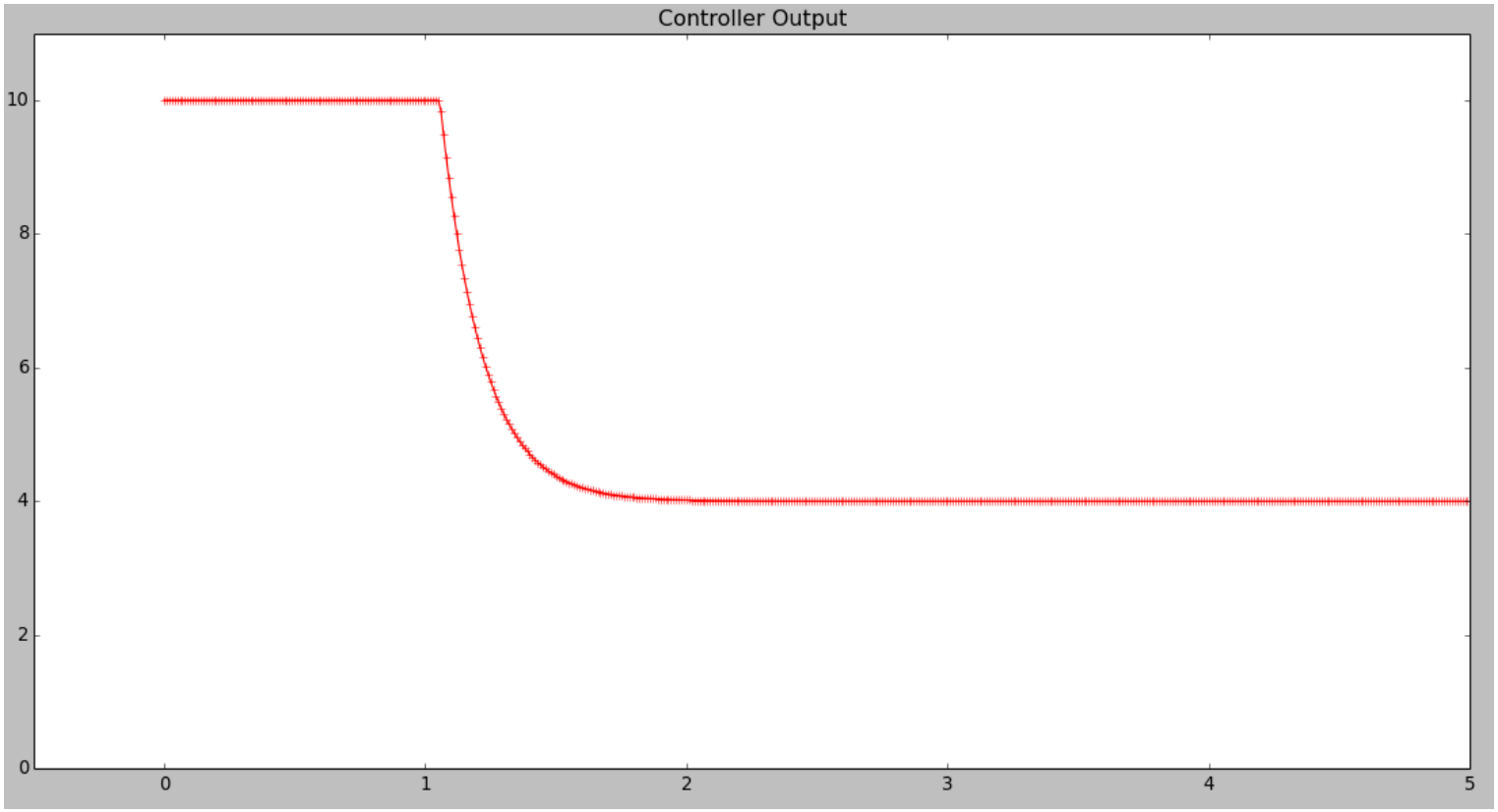
$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6, K_i = 3$$



- E' comparsa una **brutta sovraelongazione** (detta **windup**)
- Il sistema è diventato più lento

Uscita del PID CON saturazione

$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6, K_i = 3$$



- Durante il primo secondo di tempo, il sistema è **saturato**

Saturazione e ottimizzazione anti-windup

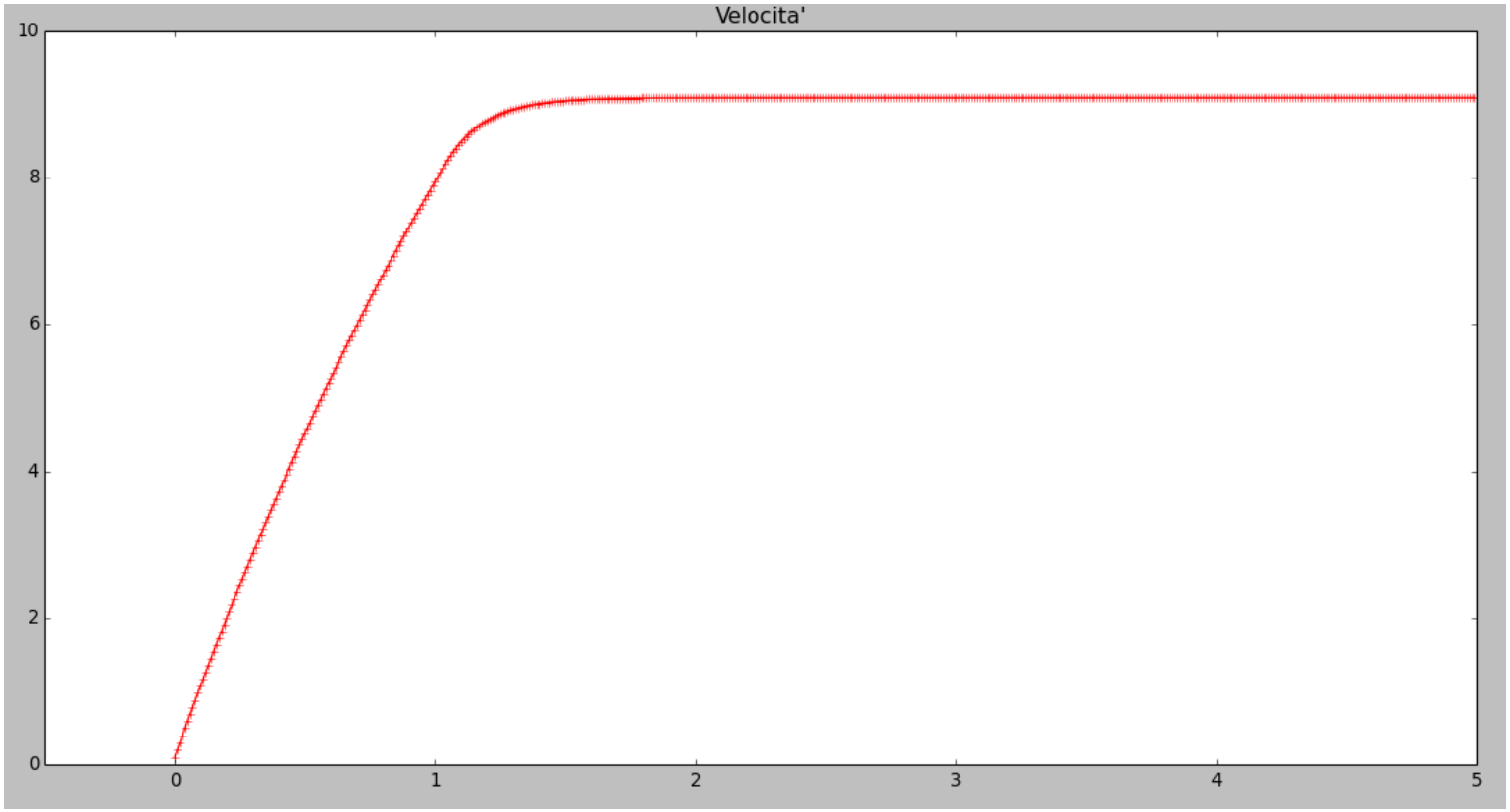
- Quando il sistema è saturato, l'errore non potrà mai ridursi secondo quanto ci si aspetta
- Cioè $e_{sat}(t) > e_{nonsat}(t)$, l'errore in saturazione è sempre più grande dell'errore quando non c'è la saturazione
- Poichè l'integratore **accumula** l'errore, quando siamo in saturazione è **inutile** accumulare errore che **non potrà ridursi**
- Per tale motivo, in *codizioni di saturazione*, si preferisce **non calcolare il termine integrale**
- Tale ottimizzazione è detta **anti-windup**

Codice del PI con saturazione e anti-windup

```
class PI_Sat_AntiW_Controller:

    def __init__(self, kp, ki, sat):
        self.__kp = kp
        self.__ki = ki
        self.__intergral_term = 0
        self.__saturation = sat
        self.__saturation_flag = False

    def evaluate(self, target, measure, delta_t):
        self.__error = target - measure

        if not(self.__saturation_flag):
            self.__intergral_term = self.__intergral_term + self.__error * delta_t

        output = self.__error * self.__kp + self.__intergral_term * self.__ki

        if output > self.__saturation:
            self.__saturation_flag = True
            output = self.__saturation
        elif output < -self.__saturation:
            self.__saturation_flag = True
            output = -self.__saturation
        else:
            self.__saturation_flag = False

        return output
```

Risposta del sistema CON saturazione e ANTI-WINDUP

$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6,\ K_i = 3$$

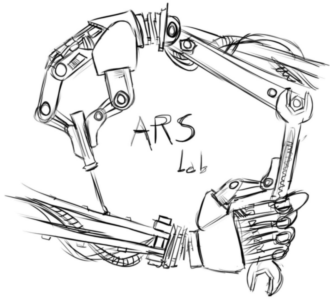


- Con l'anti-windup è scomparsa la sovraelongazione

Controllo PI con saturazione

Corrado Santoro

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



Programmazione Sistemi Robotici