Controllo PID con saturazione

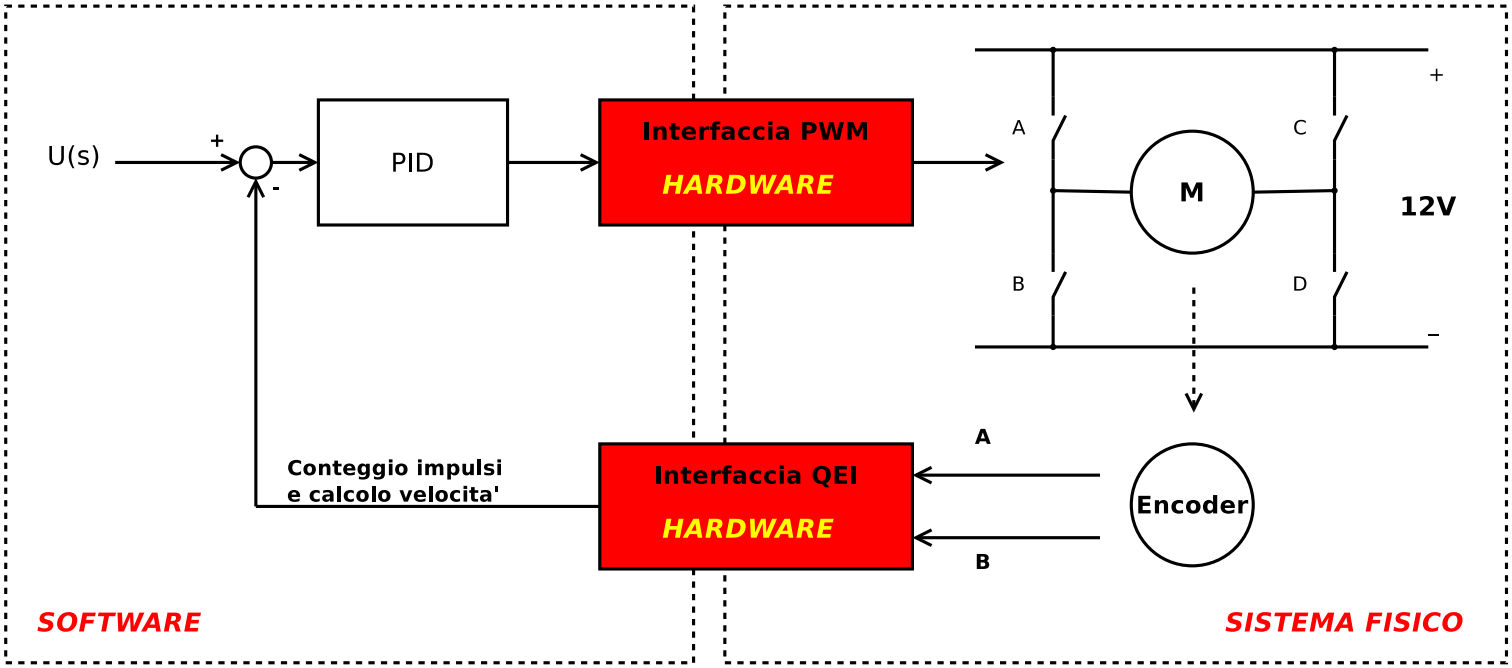
Corrado Santoro

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

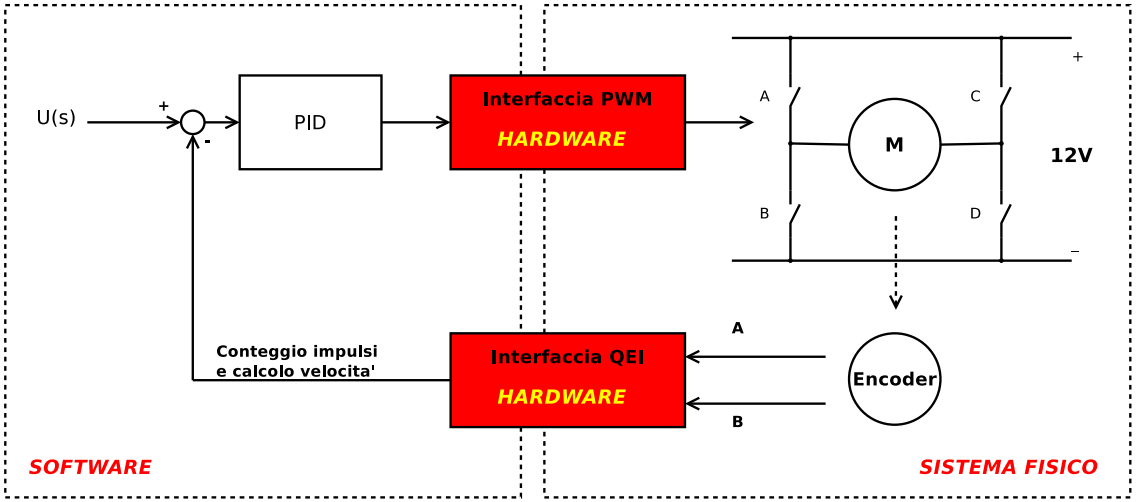
Programmazione Sistemi Robotici

Schema controllo PID di un motore in CC

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

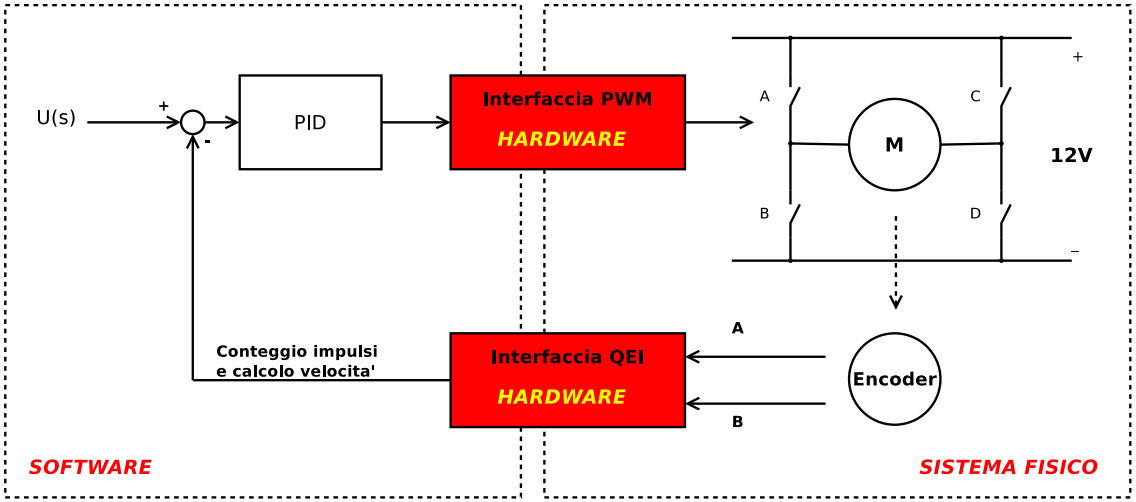


Limite del PWM



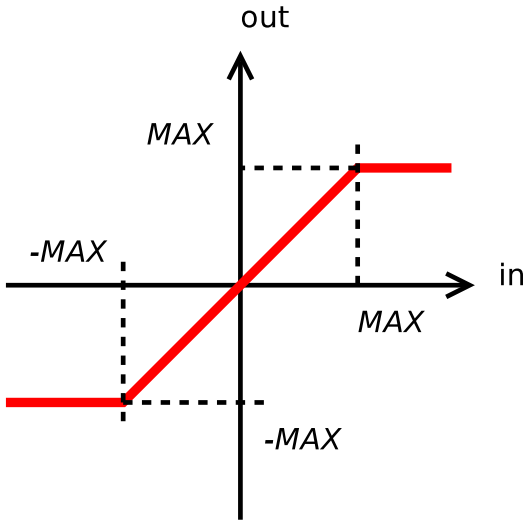
- Ogni sistema fisico ha un **limite** oltre il quale non si può andare
- Nel caso del motore in CC tale limite è rappresentato dalla sua coppia e velocità, che si ottengono quando il motore è alimentato alla sua tensione massima
- La tensione massima corrisponde al valore massimo che possiamo fornire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

Limite del PWM



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

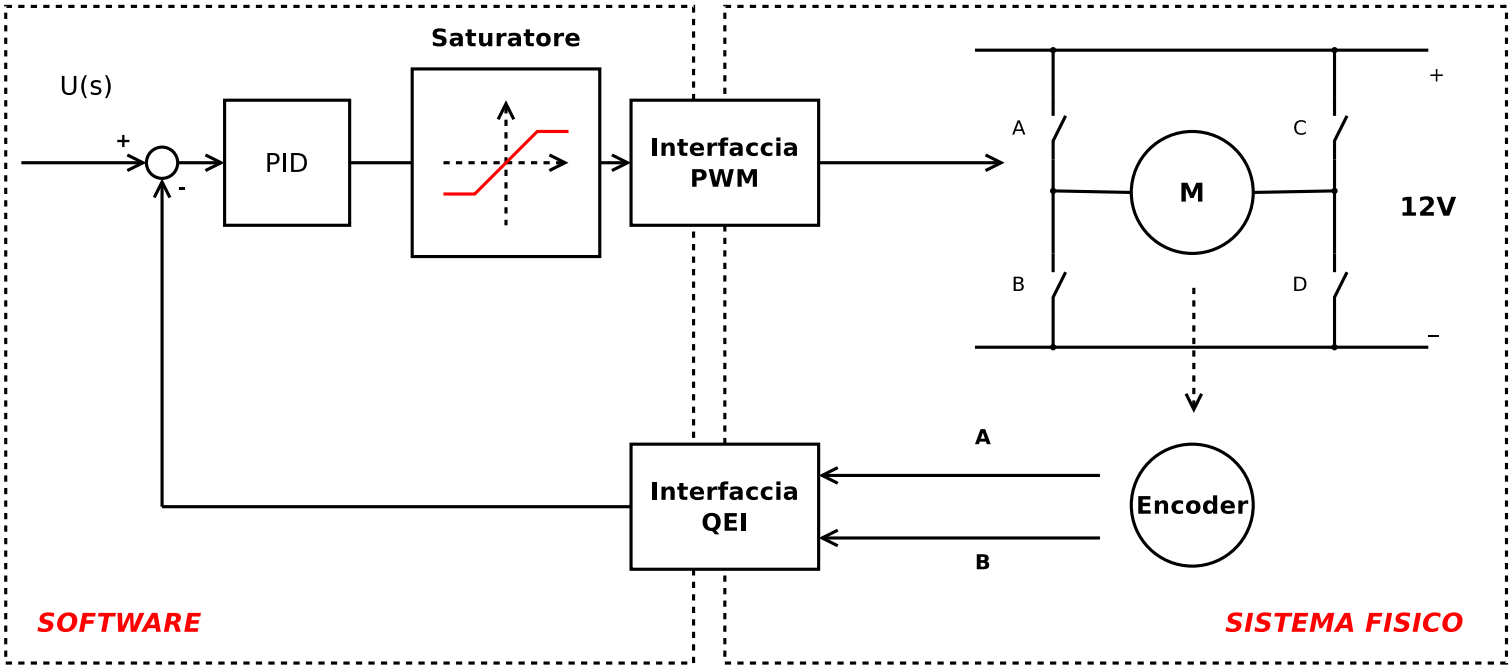
Saturazione



- 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:

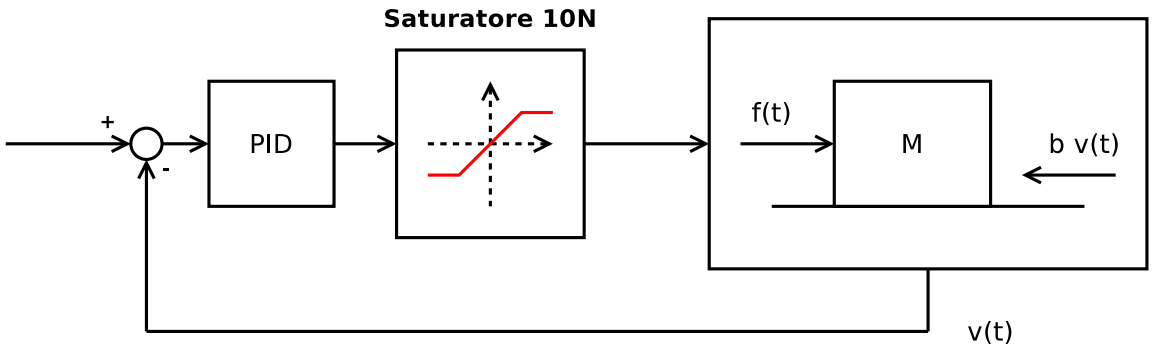
```
...
if (in < -MAX) out = -MAX;
else if (in > MAX) out = MAX;
else out = in;
...
```

PID con saturazione



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

Esempio di PID con saturazione



- Supponiamo il classico esempio di controllo (simulato) di massa su piano con attrito
- Consideriamo che l'output del PID (forza di spinta della massa) sia saturato ad un valore che non può superare i 10N

Codice del PID con saturazione (header file)

- Il codice del PID con saturazione diventa il seguente

```
/*
 * pid_saturation.h
 */
#include "dynamic_system.h"
#include <stdbool.h>

class PID_Saturation : public DynamicSystem {

public:
    PID_Saturation(float kp, float ki, float kd, float saturation, float delta_t);
    float evaluate(float input);

private:
    float m_kp, m_ki, m_kd, m_saturation, m_out_i, m_prev_input;
    bool m_saturation_flag;
};
```

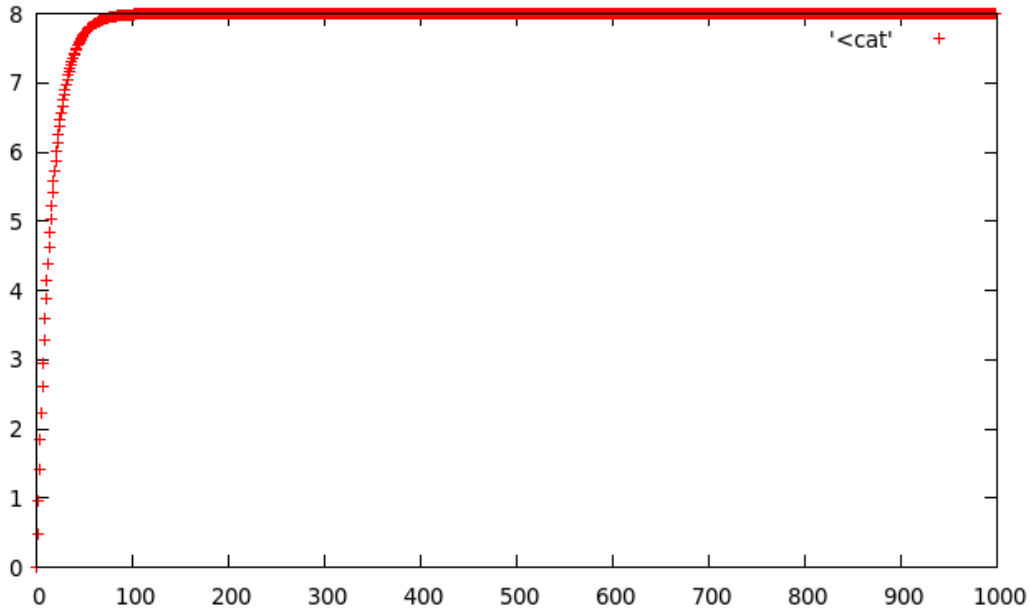
Codice del PID con saturazione (source file)

- Il codice del PID con saturazione diventa il seguente

```
/*
 * pid_saturation.cpp
 */
#include "pid_saturation.h"

PID_Saturation::PID_Saturation(float kp, float ki, float kd,
                               float saturation, float delta_t)
    : DynamicSystem(delta_t), m_kp(kp), m_ki(ki), m_kd(kd),
      m_saturation(saturation), m_out_i(0), m_prev_input(0), m_saturation_flag(false)
{ }

float PID_Saturation::evaluate(float input)
{
    float deriv = (input - m_prev_input) / m_delta_t;
    m_prev_input = input;

    m_out_i = m_out_i + m_ki * input * m_delta_t;

    float output = input * m_kp + m_out_i + deriv * m_kd;

    if (output > m_saturation) {
        output = m_saturation;
    }
    else if (output < - m_saturation) {
        output = - m_saturation;
    }

    return output;
}
```

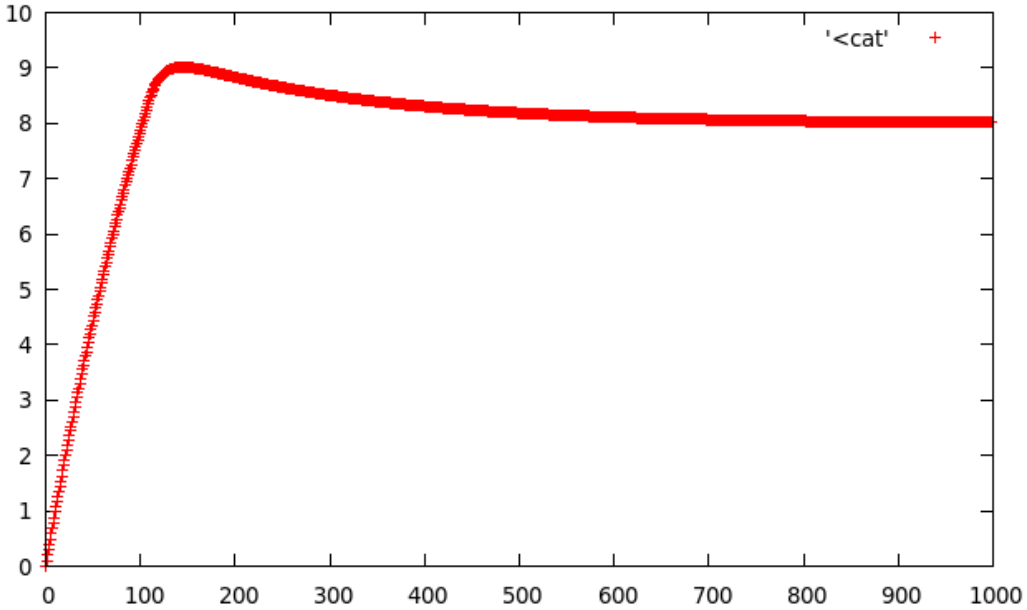
Risposta del sistema SENZA saturazione

$$v_{target} = 8m/s$$
$$K_p = 6,\ K_i = 3,\ K_d = 0$$



Risposta del sistema CON saturazione

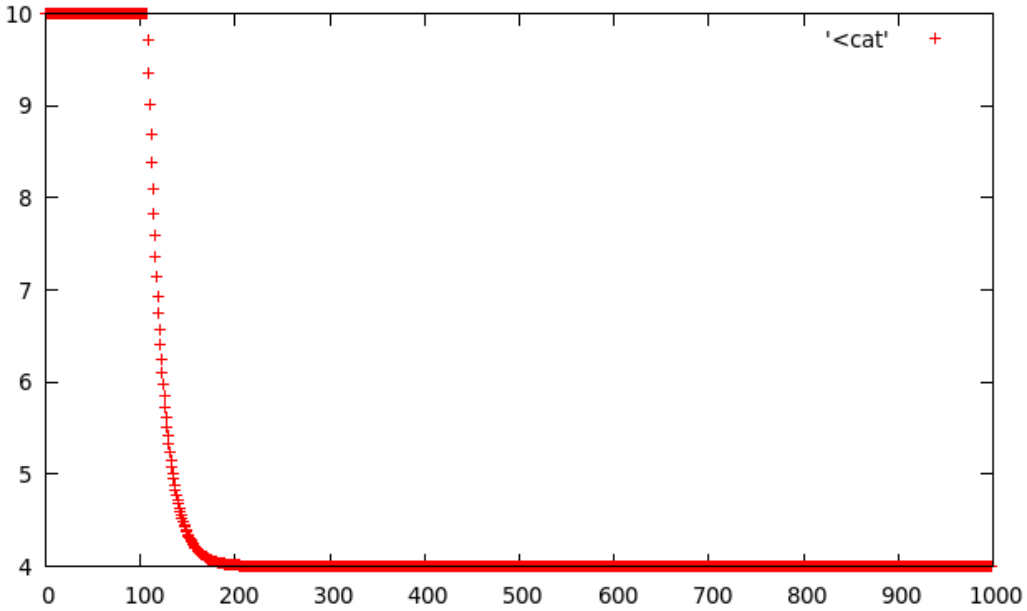
$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6, K_i = 3, K_d = 0$$



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

Uscita del PID CON saturazione

$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6, K_i = 3, K_d = 0$$



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

Saturazione e ottimizzazione anti-windup

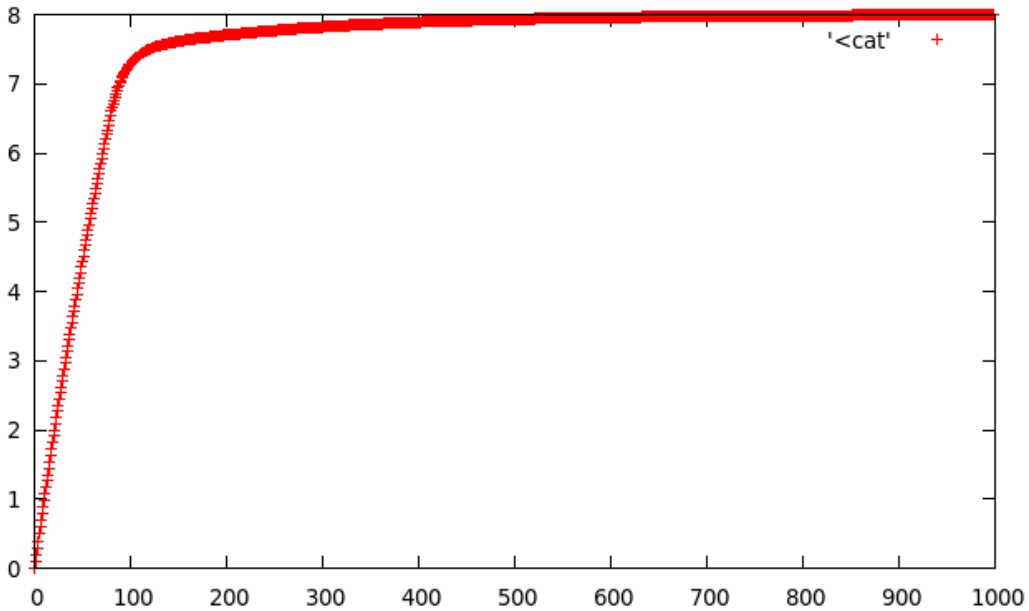
- Quando il sistema è saturato, l'errore non potrà mai ridursi secondo quanto ci si aspetta
- Cioè $e_{sat}(t) > e_{nonsat}(t)$, l'errore in saturazione è sempre più grande dell'errore quando non c'è la saturazione
- Poichè l'integratore **accumula** l'errore, quando siamo in saturazione è **inutile** 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 PID con saturazione e anti-windup

```
#define USE_ANTI_WIND_UP /* define or comment this to enable/disable anti-wind-up */

float PID_Saturation::evaluate(float input)
{
    float deriv = (input - m_prev_input) / m_delta_t;
    m_prev_input = input;

#ifdef USE_ANTI_WIND_UP
    /* if ANTIWINDUP is active, do not integrate when the system is in
        saturation */
    if (!m_saturation_flag)
        m_out_i = m_out_i + m_ki * input * m_delta_t;
#else
    m_out_i = m_out_i + m_ki * input * m_delta_t;
#endif

    float output = input * m_kp + m_out_i + deriv * m_kd;

    if (output > m_saturation) {
        output = m_saturation;
        m_saturation_flag = true;
    }
    else if (output < - m_saturation) {
        output = - m_saturation;
        m_saturation_flag = true;
    }
    else
        m_saturation_flag = false;

    return output;
}
```

Risposta del sistema CON saturazione e ANTI-WINDUP

$$v_{target} = 8m/s, sat = 10N$$
$$K_p = 6,\ K_i = 3,\ K_d = 0$$



- Con l'anti-windup è scomparsa la sovraelongazione

Controllo PID 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