Simulazione di un Sistema “Massa con attrito” e suo controllo PID

Corrado Santoro

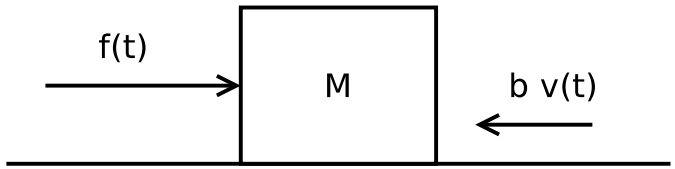
ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici

Sistema Massa con attrito

Sia data una massa su un piano alla quale e' applicata una forza $f(t)$; sia b il coefficiente di attrito, determinare la legge del moto.

$a(t) =$ accelerazione, $\quad v(t) =$ velocità



$$f(t) = Ma(t) + bv(t)$$

$$f(t) = M\frac{dv(t)}{dt} + bv(t)$$

Discretizzazione Sistema

Discretizziamo il sistema supponendo un tempo di campionamento T:

$$f(t) = M\frac{dv(t)}{dt} + bv(t)$$

$$f(k) = M\frac{v(k+1) - v(k)}{T} + bv(k)$$

$$Tf(k) = Mv(k+1) - Mv(k) + bTv(k)$$

$$Tf(k) + Mv(k) - bTv(k) = Mv(k+1)$$

$$v(k+1) = \frac{T}{M}f(k) + (\frac{M - bT}{M})v(k)$$

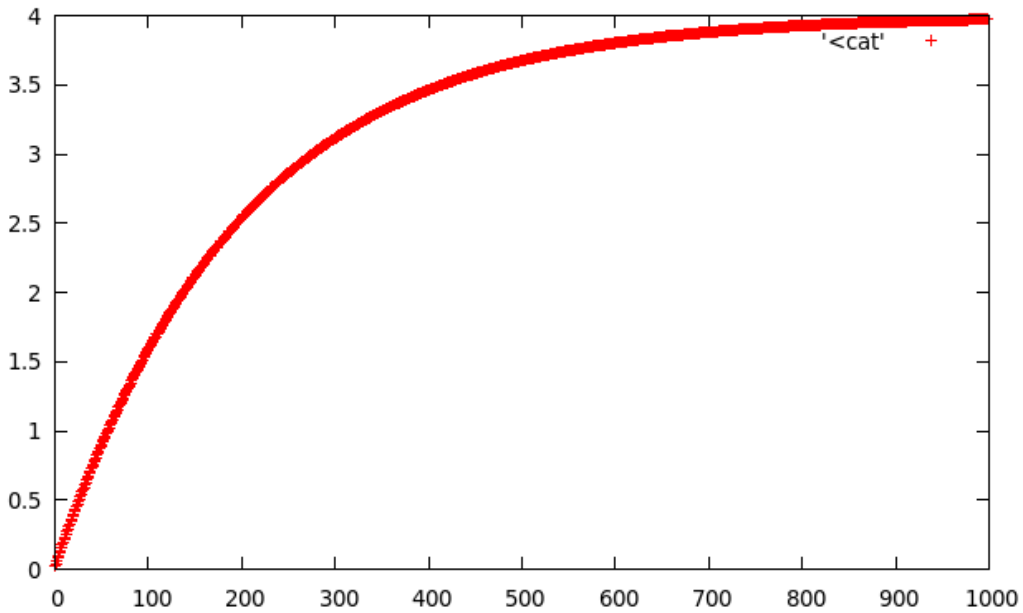
Implementazione Sistema

$$v(k+1) = \frac{T}{M} f(k) + (\frac{M - bT}{M}) v(k)$$

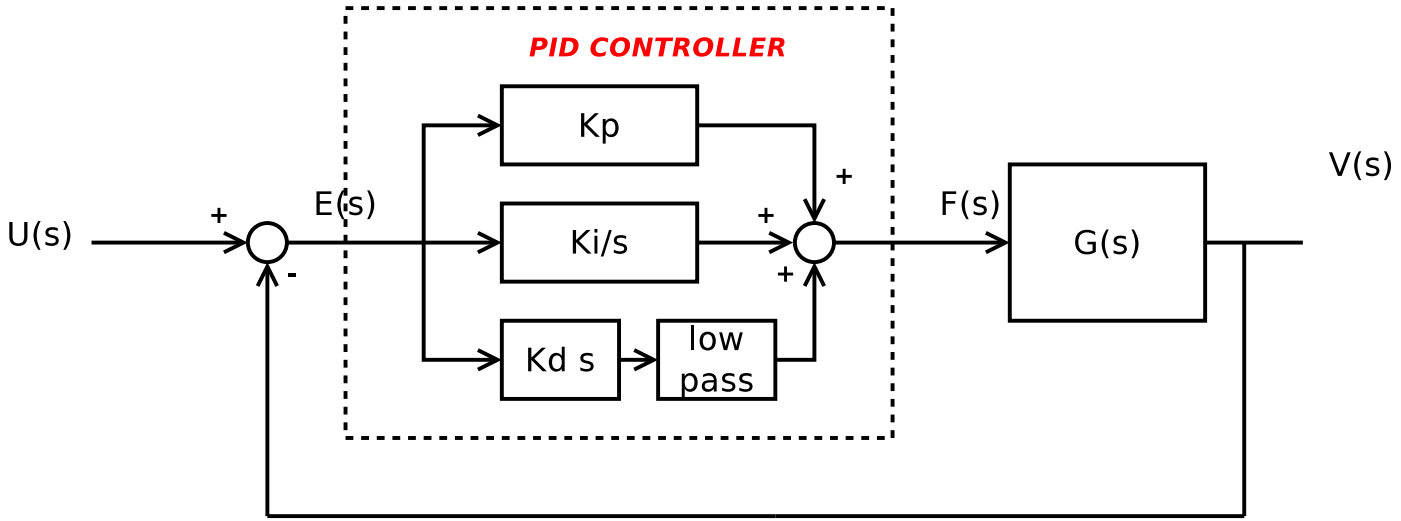
```
float v_k = 0;

float T = 0.01; // tempo di campionamento = 10 ms
float M = 1;    // massa = 1 Kg
float b = 0.5; // attrito = 0.5 Kg/s

float fdt(float f_k)
{
    float v_k1 = (T/M)*f_k + (M - b*T) / M * v_k;
    v_k = v_k1;
    return v_k1;
}
```

Implementazione Sistema

```
int main(int argc, char *argv[])
{
    int i;
    for (i = 0; i < 10*100;i++) { // 10 secondi di simulazione
        printf("%f\n", fdt(2)); // gradino di 2N
    }
}
```



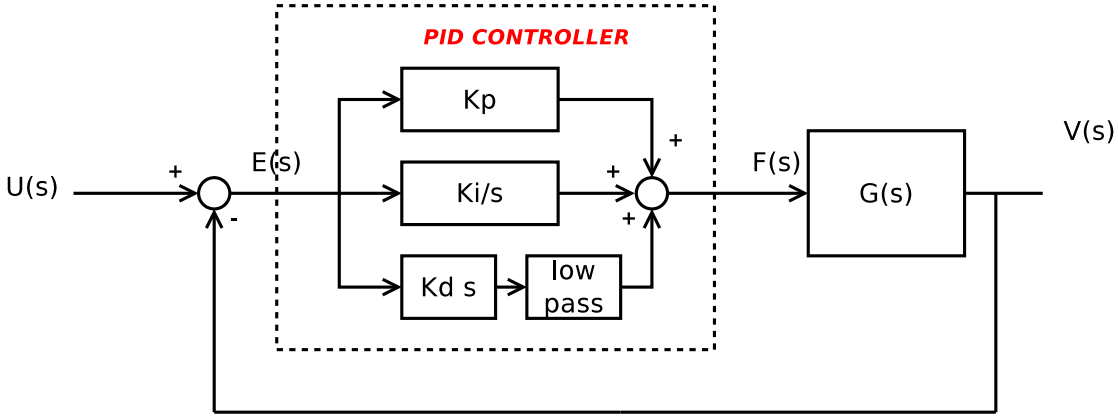
Implementazione PID



Discretizziamo le tre azioni (ignoriamo il filtro passa-basso) e calcoliamo i contributi sull'uscita $f(t)$ delle tre azioni $f_p(t)$, $f_i(t)$, $f_d(t)$:

Proporzionale $f_p(k) = K_p e(k)$

Implementazione PID



Integrale

$$F_i(s) = \frac{K_i}{s} E(s)$$

$$sF_i(s) = K_i E(s)$$

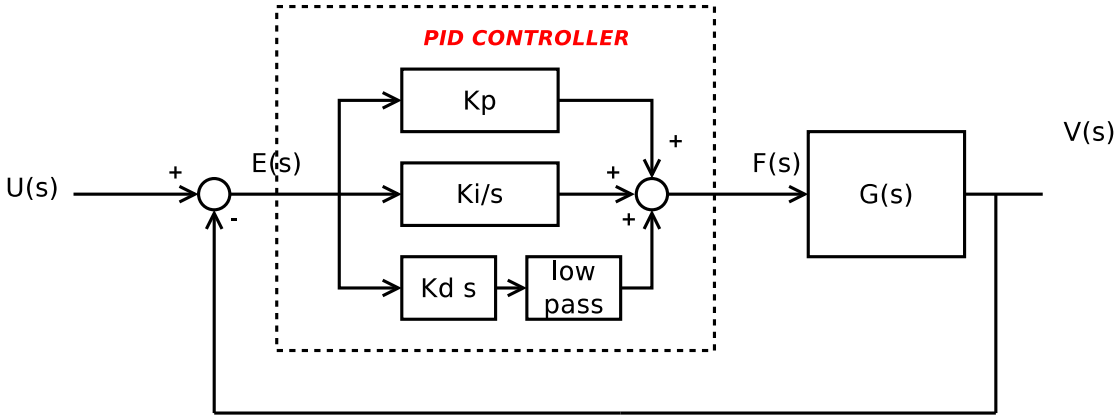
$$\frac{df_i(t)}{dt} = K_i e(t)$$

$$\frac{f_i(k) - f_i(k-1)}{T} = K_i e(k)$$

$$f_i(k) - f_i(k-1) = TK_i e(k)$$

$$f_i(k) = f_i(k-1) + TK_i e(k)$$

Implementazione PID



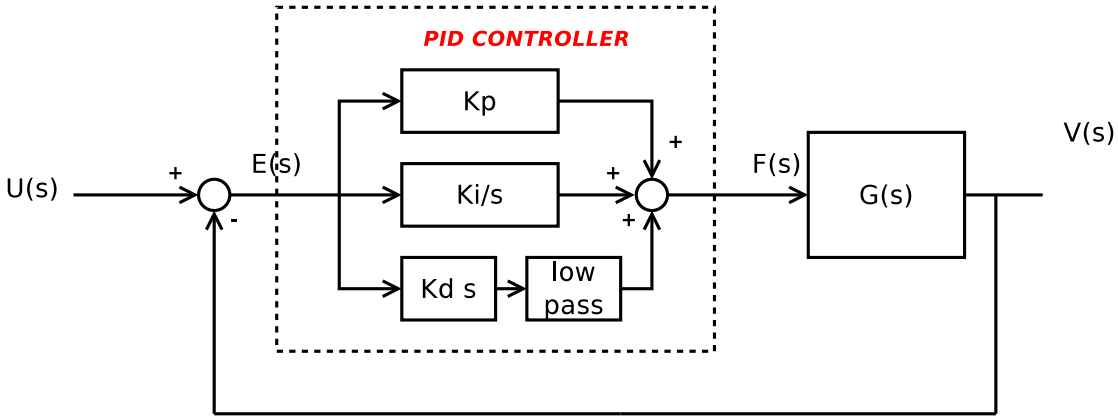
Derivativa

$$F_d(s) = K_d s E(s)$$

$$f_d(t) = K_d \frac{de(t)}{dt}$$

$$f_d(k) = K_d \frac{e(k) - e(k-1)}{T}$$

Implementazione PID



Proporzionale $f_p(k) = K_p e(k)$

Integrale $f_i(k) = f_i(k-1) + TK_i e(k)$

Derivativa $f_d(k) = K_d \frac{e(k) - e(k-1)}{T}$

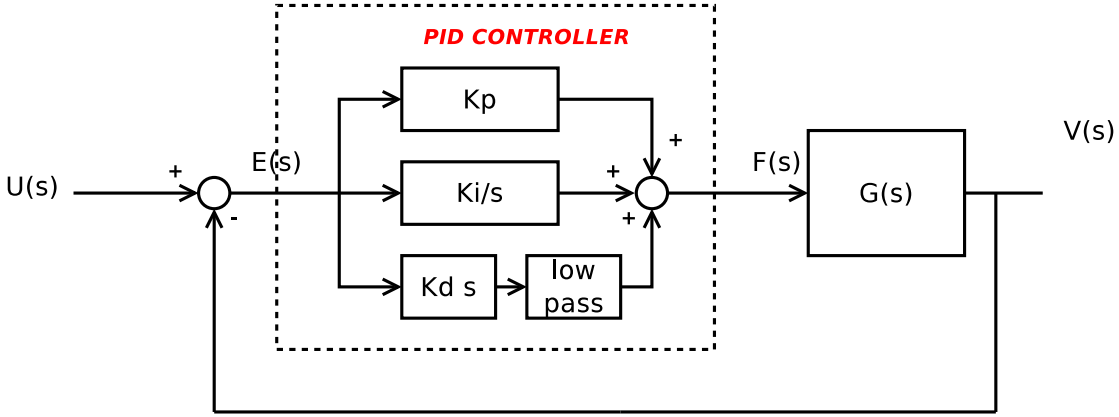
Uscita $f(k) = f_p(k) + f_i(k) + f_d(k)$

Implementazione PID

$$\begin{aligned}
\textit{Proporzionale} \quad & f_p(k) = K_p e(k) \\
\textit{Integrale} \quad & f_i(k) = f_i(k-1) + TK_i e(k) \\
\textit{Derivativa} \quad & f_d(k) = K_d \frac{e(k) - e(k-1)}{T} \\
\textit{Uscita} \quad & f(k) = f_p(k) + f_i(k) + f_d(k)
\end{aligned}$$

```
float Kp;
float Ki;
float Kd;
float out_I = 0;
float error_1 = 0;

float PID(float target, float current)
{
    float error = target - current;
    float out_P = Kp * error;
    float out_I = out_I + Ki * error * T;
    float out_D = Kd * (error - error_1) / T ;
    float output = out_P + out_I + out_D;

    return output;
}
```

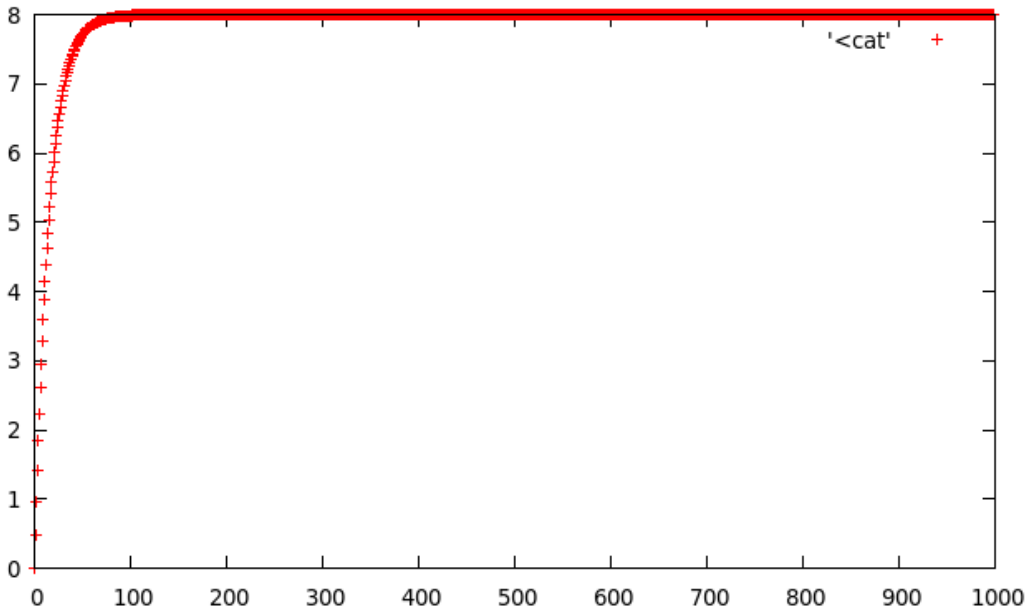
Implementazione Sistema di Controllo Completo




```
int main(int argc, char *argv[])
{
    int i;
    float current_speed;
    float target_speed = 8; // 8 m/s

    Kp = atof(argv[1]);
    Ki = atof(argv[2]);
    Kd = atof(argv[3]);

    current_speed = 0;
    for (i = 0; i < 10*100;i++) { // 10 secondi di simulazione
        float pi_output = PID(target_speed, current_speed);
        current_speed = fdt(pi_output);
        printf("%f\n", current_speed);
    }
}
```

Risposta del sistema di controllo

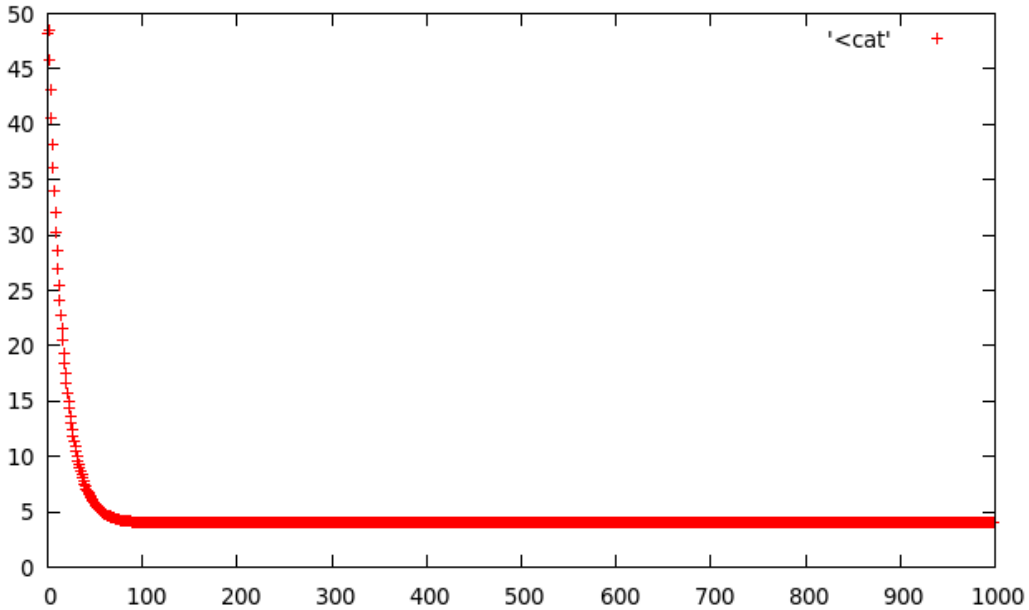
$$v_{target} = 8m/s$$
$$K_p = 6,\ K_i = 3,\ K_d = 0$$



Raggiungiamo il target dopo circa 100 campioni (1 secondo).

Output del PID

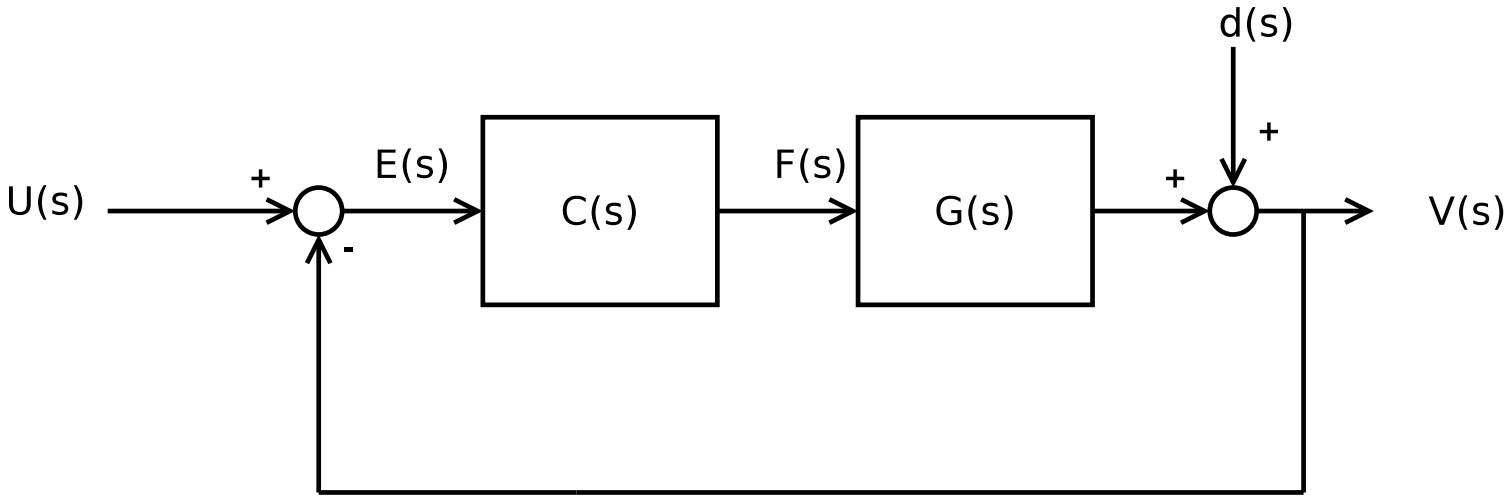
$$v_{target} = 8m/s$$
$$K_p = 6,\ K_i = 3,\ K_d = 0$$



Il controllore genera una “botta” iniziale di quasi 50N, per poi decrescere gradualmente.

Presenza di un disturbo

Supponiamo che, nel sistema, intervenga un **disturbo**, cioè degli eventi generici che **alterano la grandezza che vogliamo controllare**:



Il controllore **reagirà** di conseguenza, intervenendo e **riportando** il valore della grandezza al target desiderato (qualora non si vada oltre i limiti fisici).

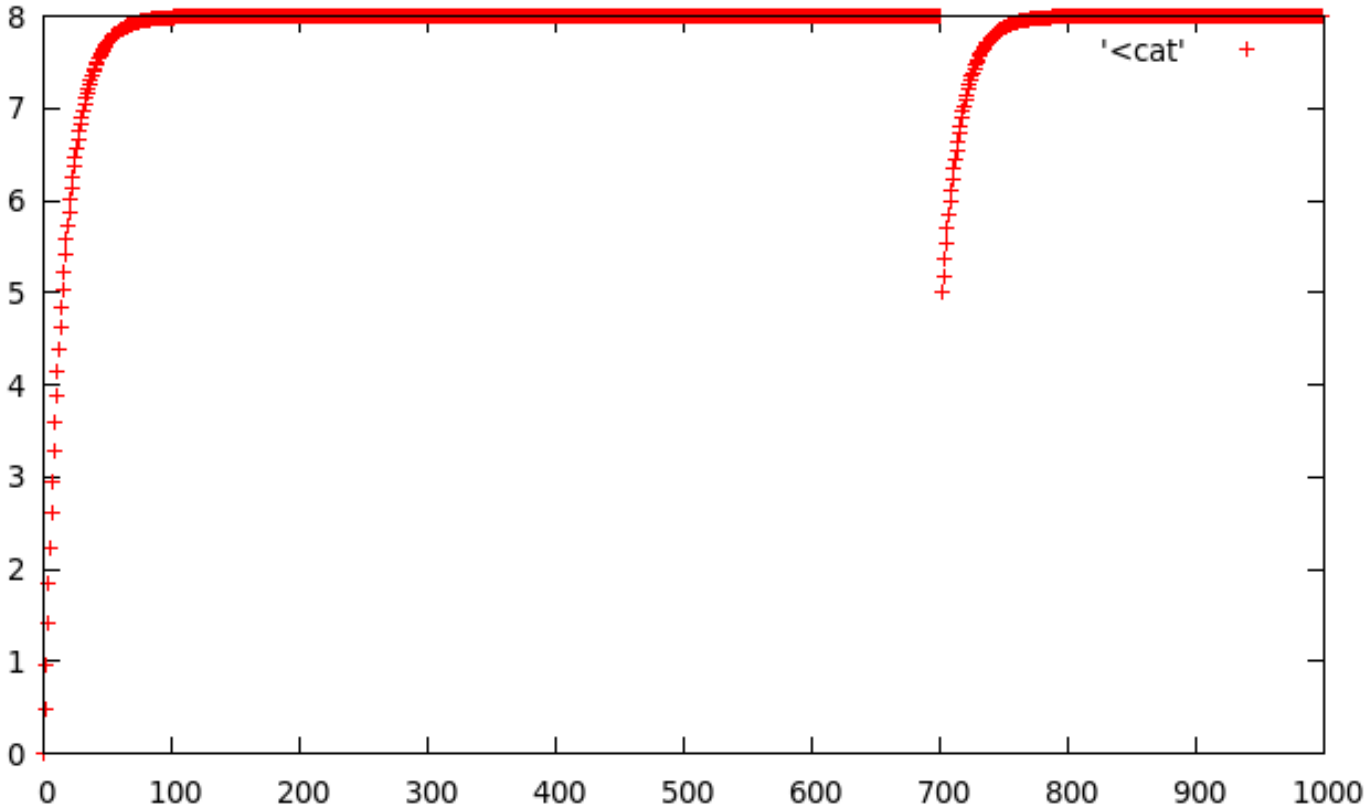
Simulazione del disturbo

Supponiamo che dopo 7 secondi compaia un disturbo che riduce la velocità di un fattore 3:

```
float climb(int i)
{
    if (i > 7*100)
        return 3;
    else
        return 0;
}

int main(int argc, char *argv[])
{
    int i;
    float current_speed;
    float target_speed = 8; // 8 m/s

    Kp = atof(argv[1]);
    Ki = atof(argv[2]);
    Kd = atof(argv[3]);

    current_speed = 0;
    for (i = 0; i < 10*100;i++) { // 10 secondi di simulazione
        float pi_output = PID(target_speed, current_speed);
        current_speed = fdt(pi_output) - climb(i);
        printf("%f\n", current_speed);
    }
}
```

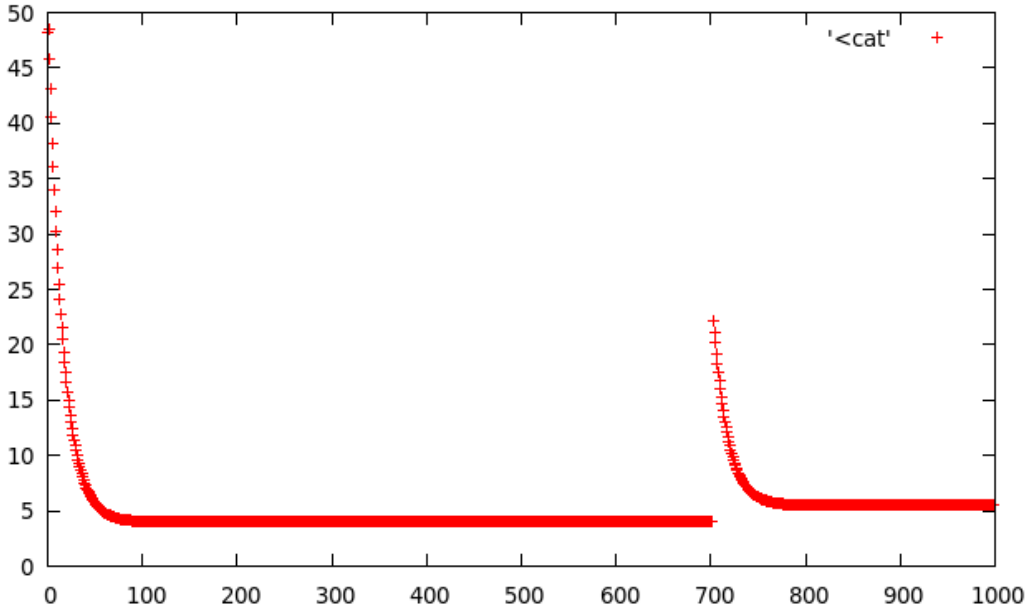
Risposta del sistema di controllo con disturbo

$v_{target} = 8m/s$
$K_p = 6, K_i = 3, K_d = 0$
Disturbo dopo 7 secondi



Output del PID con disturbo

$$v_{target} = 8m/s$$
$$K_p = 6,\ K_i = 3,\ K_d = 0$$
Disturbo dopo 7 secondi



Simulazione di un Sistema “Massa con attrito” e suo controllo 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