V.A.B.

Modellizzazione e controllo di un veicolo auto bilanciato

A.A. 19/20 - Settembre 2020

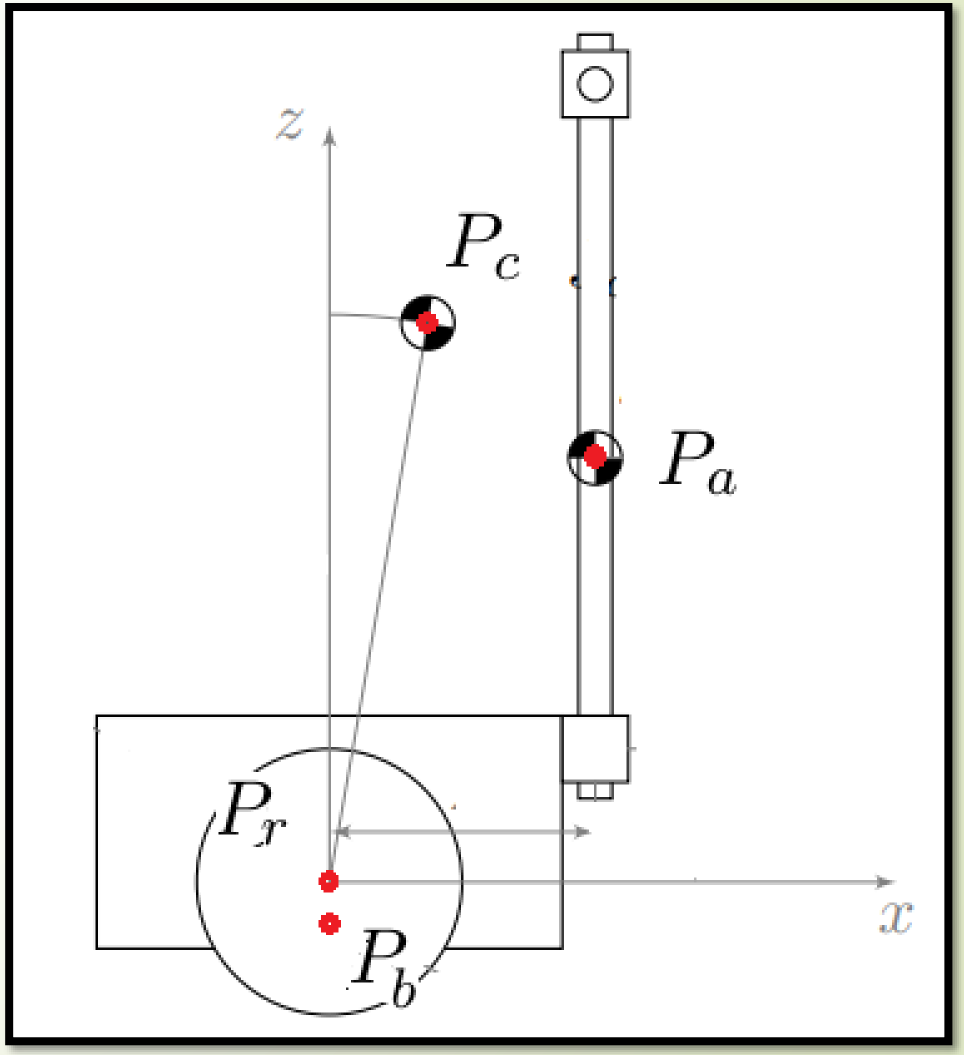
Calegari Andrea – 1041183
Piffari Michele - 1040658

Lista dei contenuti

- Modellizzazione
- Equazioni del moto
- Linearizzazione
- Controllo
- Simulazione
- OPC-UA
- Xenomai

Modellizzazione – Componenti del V.A.B

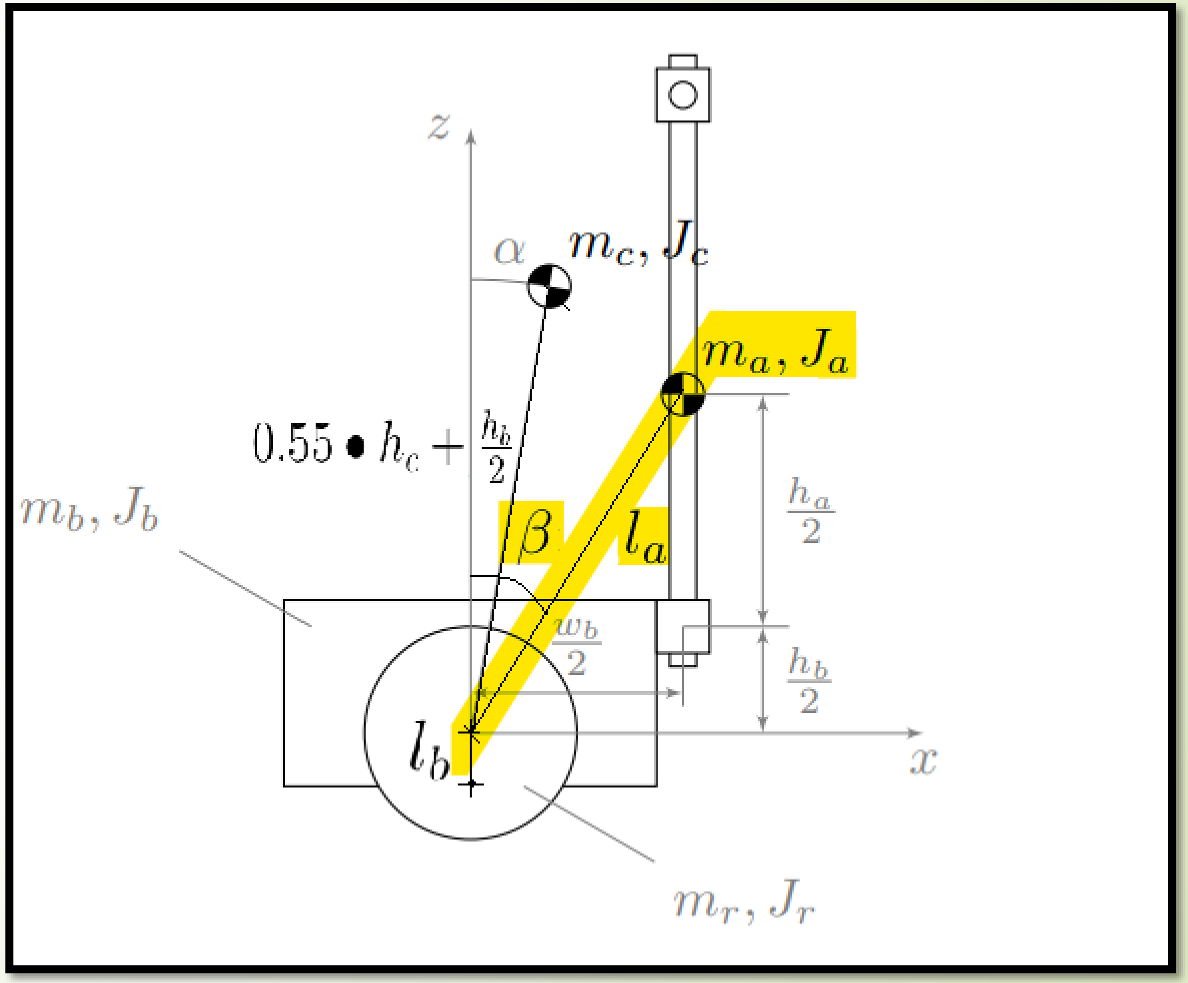
- Asta → P_a
- Utente a bordo → P_c
- Chassis → P_b
- Ruota → P_r → origine degli assi
- DOF:
  - Asta-Utente-Chassis →1 d.o.f.
    - θ
  - Ruota → 1 d.o.f.
    - ϕ → puro rotolamento



Modellizzazione – Asta

$$P_a = \begin{pmatrix} r\,\phi(t) + l_a \sin(\beta + \theta(t)) \\ l_a \cos(\beta + \theta(t)) \\ \theta(t) \end{pmatrix}$$

$$M_a = \begin{pmatrix} m_a & 0 & 0 \\ 0 & m_a & 0 \\ 0 & 0 & m_a\, l_a{}^2 + J_a \end{pmatrix}$$



Modellizzazione – Asta, Lagrangiana

$$T_a = \frac{1}{2}\frac{dP_a}{dt} M_a \left(\frac{dP_a}{dt}\right)^T$$

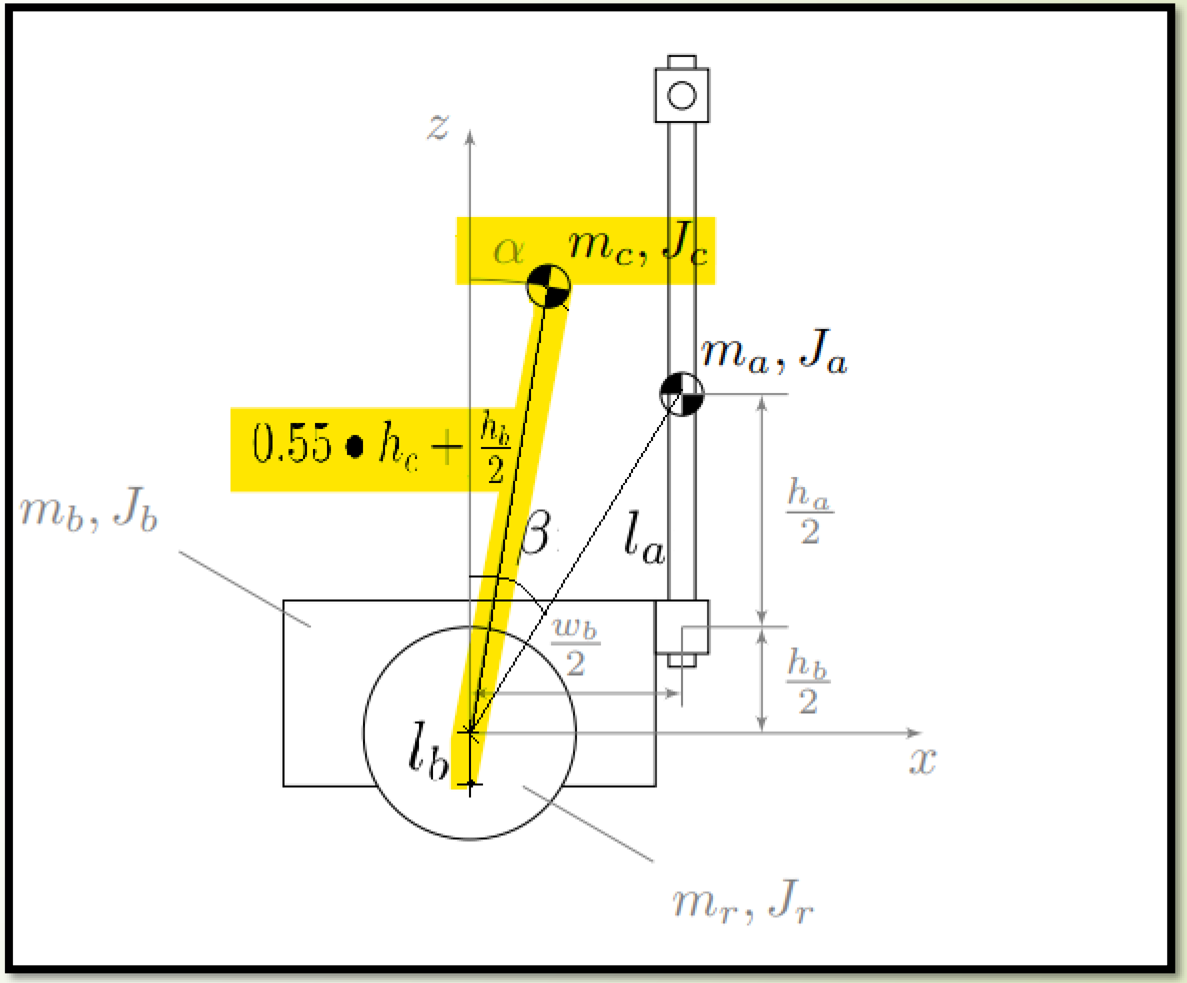
$$U_a = gM_a l_a \cos(\beta + \theta(t))$$

$$L_a = T_a - U_a$$

Modellizzazione – Uomo

$$P_c = \begin{pmatrix} r\,\phi(t) + l_c \sin(\alpha + \theta(t)) \\ l_c \cos(\alpha + \theta(t)) \\ \alpha + \theta(t) \end{pmatrix}$$

$$M_c = \begin{pmatrix} m_c & 0 & 0 \\ 0 & m_c & 0 \\ 0 & 0 & m_c {l_c}^2 + J_c \end{pmatrix}$$



Modellizzazione – Uomo, J_c

- $M_c = 70\ kg = 2.20462\ * 70\ lb$
- $h_c = 177\ cm = altezza\ [cm]$
- $J_c = (1732353 \cdot hc\ +\ 619298 \cdot M_c -\ 277625773)10^{-7}$

Modellizzazione – Uomo, Lagrangiana

$$T_c = \frac{1}{2}\frac{dP_c}{dt}M_c\left(\frac{dP_c}{dt}\right)^T$$

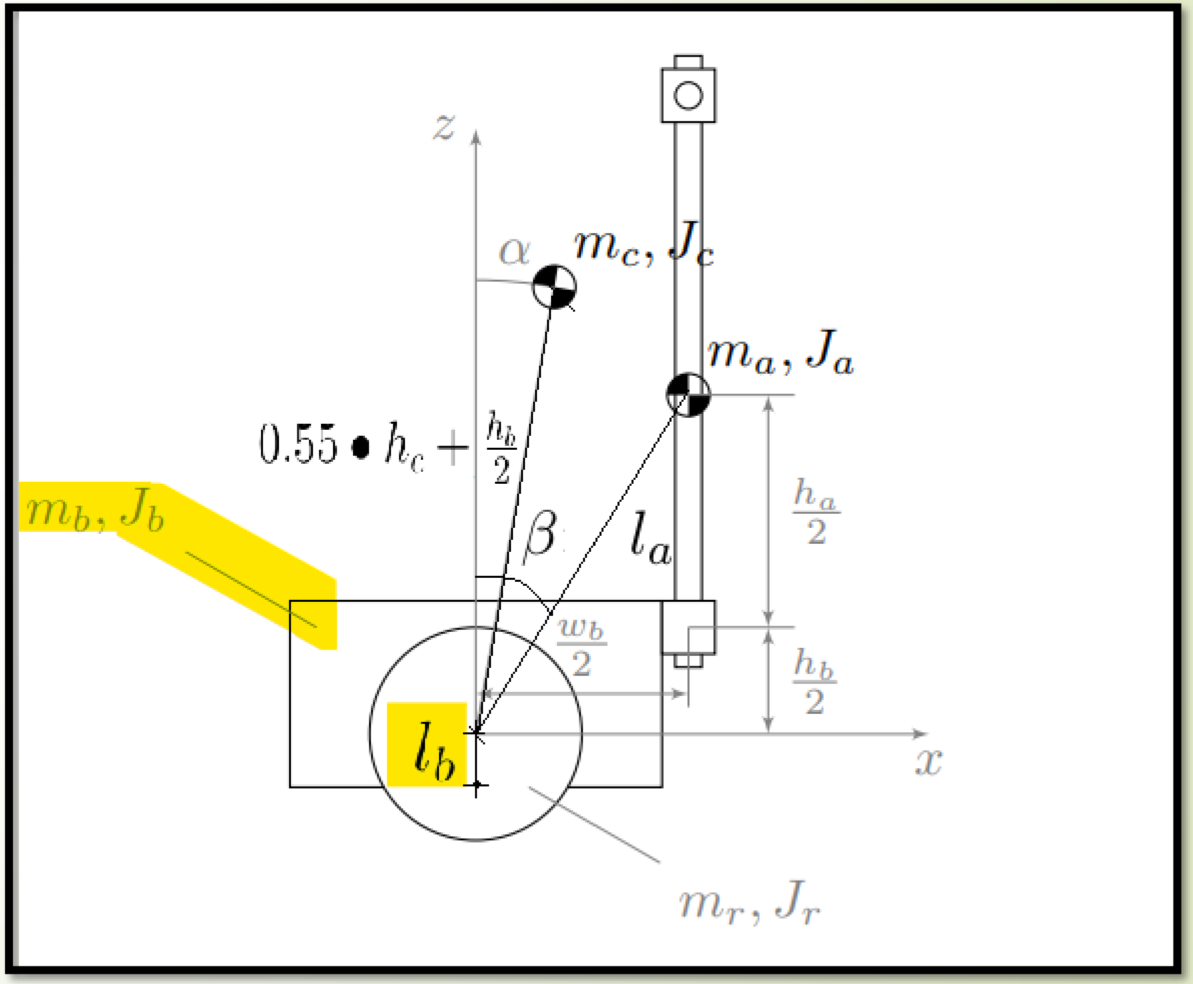
$$U_c = gM_c l_c \cos(\alpha + \theta(t))$$

$$L_c = T_c - U_c$$

Modellizzazione – Chassis

$$P_b = \begin{pmatrix} r\,\phi(t) - l_b \sin(\theta(t)) \\ -l_b \cos(\theta(t)) \\ \theta(t) \end{pmatrix}$$

$$M_b = \begin{pmatrix} m_b & 0 & 0 \\ 0 & m_b & 0 \\ 0 & 0 & m_b\,{l_b}^2 + J_b \end{pmatrix}$$



Modellizzazione – Chassis, Lagrangiana

$$T_b = \frac{1}{2}\frac{dP_b}{dt} M_b \left(\frac{dP_b}{dt}\right)^T$$

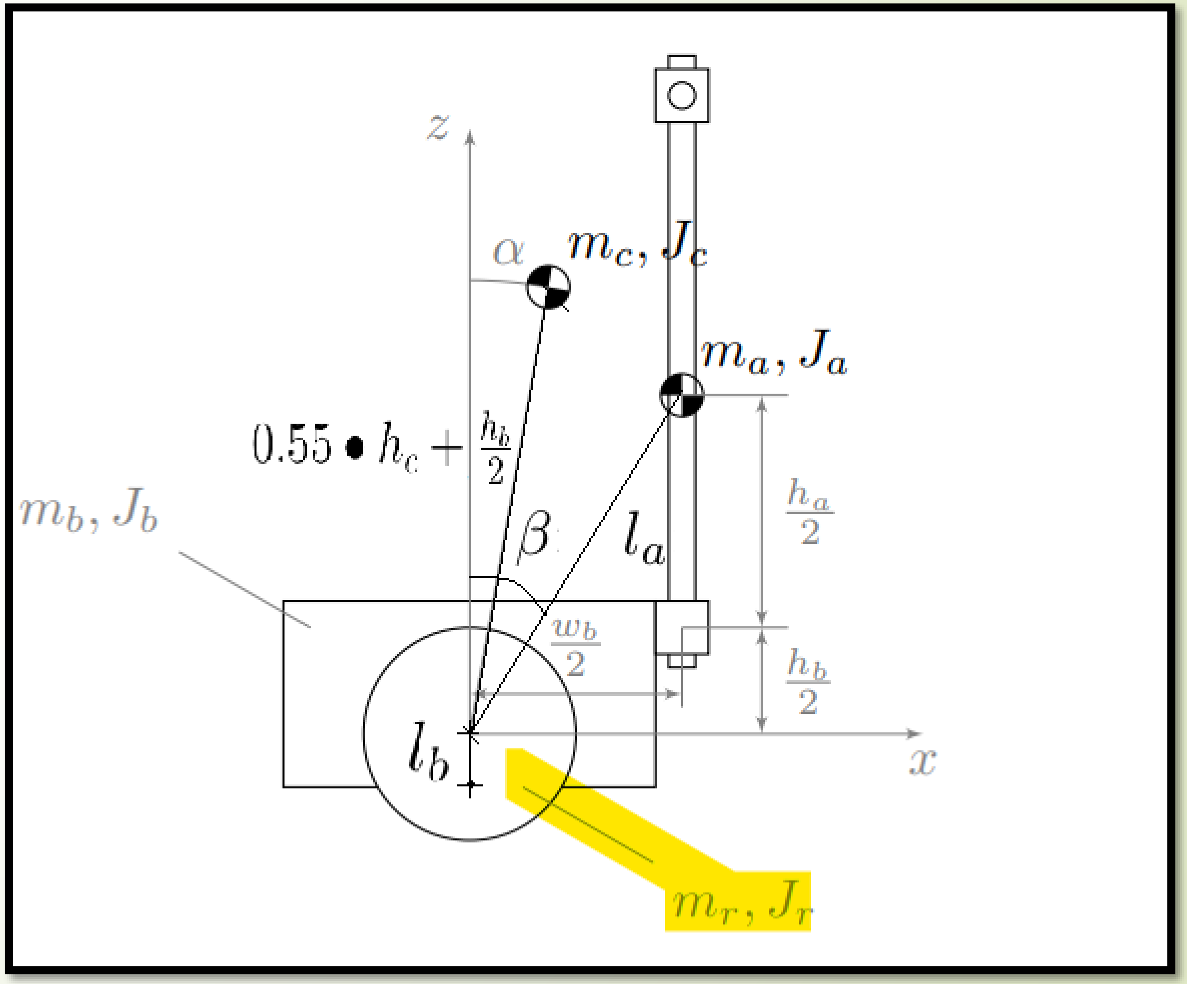
$$U_b = -\, g M_b l_b \cos(\theta(t))$$

$$L_b = T_b - U_b$$

Modellizzazione – Ruota

$$P_r = \begin{pmatrix} r\,\phi(t) \\ 0 \\ \phi(t) \end{pmatrix}$$

$$M_r = \begin{pmatrix} m_r & 0 & 0 \\ 0 & m_r & 0 \\ 0 & 0 & J_r \end{pmatrix}$$



Modellizzazione – Ruota, Lagrangiana

$$T_r = \frac{1}{2}\frac{dP_r}{dt}M_r\left(\frac{dP_r}{dt}\right)^T$$

$$U_r = 0$$

$$L_r = T_r$$

Equazioni del moto, sistema

$$\begin{cases} L = L_a + L_b + L_c + 2 * L_r \\ \dfrac{d}{dt}\left(\dfrac{\partial}{\partial \dot{\phi}} L\right) - \dfrac{\partial}{\partial \phi} L = \dfrac{C_m}{\tau} \\ \dfrac{d}{dt}\left(\dfrac{\partial}{\partial \dot{\theta}} L\right) - \dfrac{\partial}{\partial \theta} L = -2 * C_m \end{cases}$$

Dove il rapporto di trasmissione τ è dato da:

$$\tau = \tau_1 * \tau_2 = 0.1 * \frac{Z_{in}}{Z_{out}} = 0.1 * \frac{22}{26} = 0.085$$

Equazioni del moto, sistema

$$\begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = - \begin{bmatrix} -l_a m_a r sin\left(\beta + \theta(t)\right)\dot{\theta}^2 + l_b m_b r sin\left(\theta(t)\right)\dot{\theta}^2 - l_c m_c r sin\left(\alpha + \theta(t)\right)\dot{\theta}^2 \\ -g\left(l_c m_c \cos\left(\alpha + \theta(t)\right) + l_a m_a \sin\left(\beta + \theta(t)\right) - l_b m_b \sin\left(\theta(t)\right)\right)\dot{\theta} - l_a m_a r sin\left(\beta + \theta(t)\right)\dot{\theta}\dot{\phi} + l_b m_b r sin\left(\theta(t)\right)\dot{\theta}\dot{\phi} - l_c m_c r sin\left(\alpha + \theta(t)\right)\dot{\theta}\dot{\phi} \end{bmatrix} + \begin{bmatrix} \frac{C_m}{\tau} \\ -2C_m \end{bmatrix}$$

$$M_1 = l_a m_a r cos\left(\beta + \theta(t)\right) - l_b m_b r cos\left(\theta(t)\right) + l_c m_c r cos\left(\alpha + \theta(t)\right)$$

$$M_2 = 2m_r r^2 + 2J_r + m_a r^2 + m_b r^2 + m_c r^2$$

$$M_3 = J_a + J_b + J_c + 2l_a^2 m_a + 2l_b^2 m_b + 2l_c^2 m_b$$

$$M_4 = l_a m_a r cos\left(\beta + \theta(t)\right) - l_b m_b r cos\left(\theta(t)\right) + l_c m_c r cos(\alpha + \theta(t)$$

Equazioni del moto, soluzioni?

$$\begin{cases} \ddot{\phi} = f_{\phi}(M_c, C_m, \theta, \dot{\theta}) \\ \ddot{\theta} = f_{\theta}(M_c, C_m, \theta, \dot{\theta}) \end{cases}$$

matlabFunction($\ddot{\phi}$,'File','phi_secondo');
matlabFunction($\ddot{\theta}$,'File','theta_secondo');

Le equazioni così ottenute saranno la base di partenze in due step successivi:

- Linearizzazione
- Applicazione del controllore al sistema reale

Linearizzazione, definizione dello stato

$$x = \begin{bmatrix} x_1 \rightarrow \phi \\ x_2 \rightarrow \dot{\phi} \\ x_3 \rightarrow \theta \\ x_4 \rightarrow \dot{\theta} \end{bmatrix}$$

$$\begin{cases} \mathbf{f_1} \rightarrow \dot{x_1}(t) = \dot{\phi} = x_2(t) = 0 \text{ (poichè all'equilibrio)} \\ \mathbf{f_2} \rightarrow \dot{x_2}(t) = \ddot{\phi} \\ \mathbf{f_3} \rightarrow \dot{x_3}(t) = \dot{\theta} = x_4(t) = 0 \text{ (poichè all'equilibrio)} \\ \mathbf{f_4} \rightarrow \dot{x_4}(t) = \ddot{\theta} \\ \mathbf{g_1} \rightarrow y_1(t) = \phi = x_1 \\ \mathbf{g_2} \rightarrow y_2(t) = \dot{\phi} = x_2 \\ \mathbf{g_3} \rightarrow y_3(t) = \theta = x_3 \\ \mathbf{g_4} \rightarrow y_4(t) = \dot{\theta} = x_4 \end{cases}$$

Linearizzazione, matrici di stato

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases}$$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} \quad B = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix} \quad C = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \\ \frac{\partial g_4}{\partial x_1} & \frac{\partial g_4}{\partial x_2} & \frac{\partial g_4}{\partial x_3} & \frac{\partial g_4}{\partial x_4} \end{bmatrix} \quad D = \begin{bmatrix} \frac{\partial g_1}{\partial u} \\ \frac{\partial g_2}{\partial u} \\ \frac{\partial g_3}{\partial u} \\ \frac{\partial g_4}{\partial u} \end{bmatrix}$$

Linearizzazione, equilibrio

$$\ddot{\theta} = f_{\theta}(m_c, C_m, \theta, \dot{\theta}) = 0$$

$$\theta_1 = -0{,}0117 \rightarrow \text{equilibrio instabile}$$

$\theta_2 = 3{,}1298 \rightarrow$ equilibrio stabile (ma non utile ai fini del controllo)

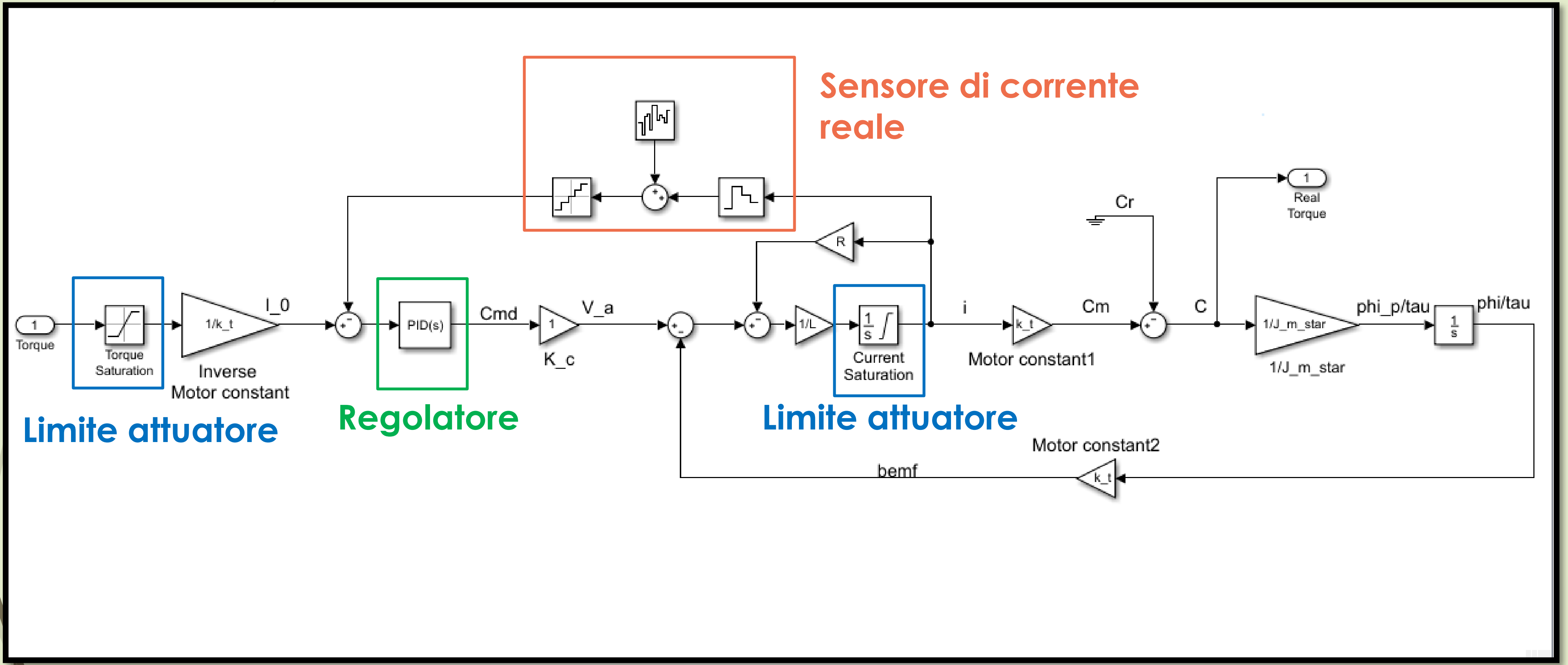
<u>Si andrà a linearizzare intorno all' equilibrio instabile</u>

Linearizzazione, matrici di stato

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -15.2286 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 5.4392 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 2.7498 \\ 0 \\ -0.2612 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Motore, modello



Motore, regolatore

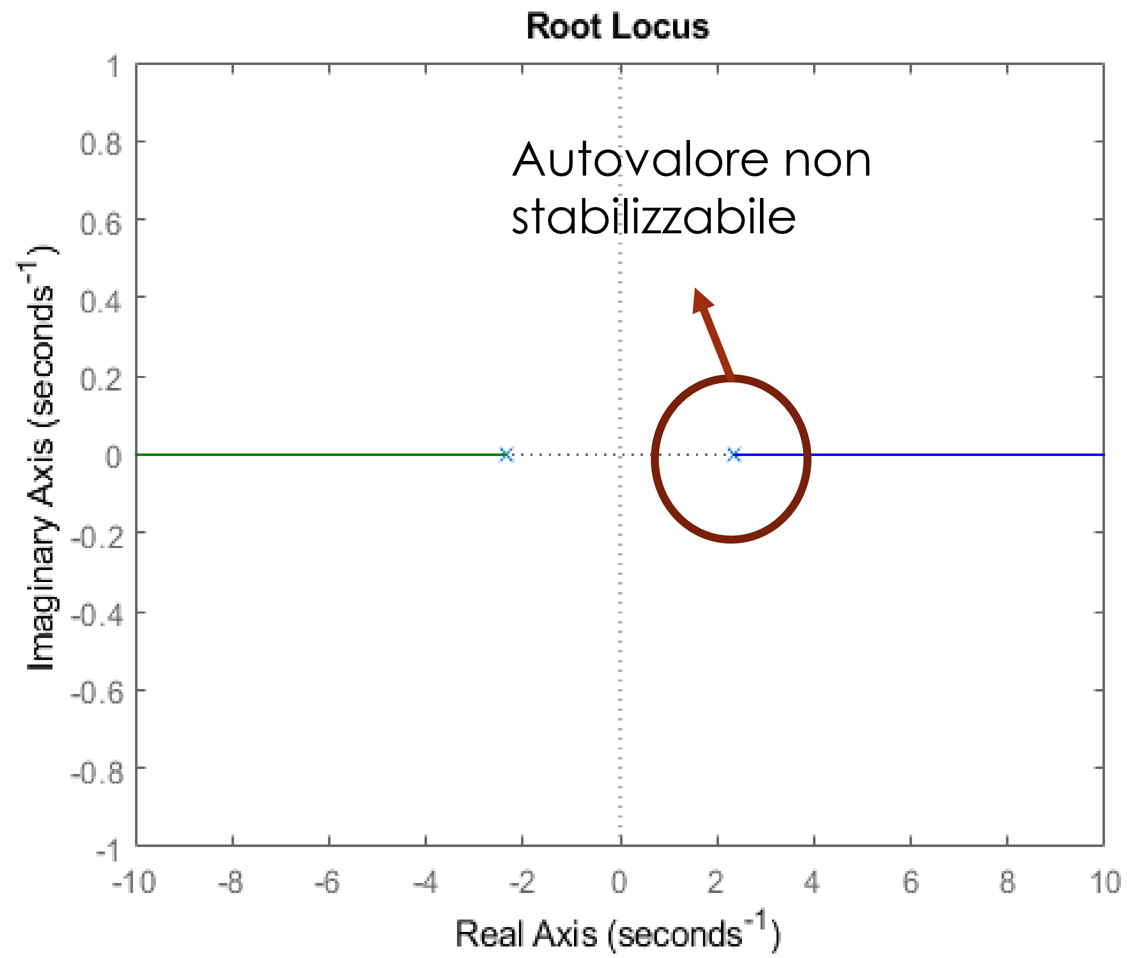
- $\frac{\tau_{em}}{\tau_e} = \frac{\frac{J_m + J_r \tau^2}{K^2}}{\frac{L_a}{R_a}} = \frac{0{,}0183\ s}{1{,}22 \cdot 10^{-3} s} \gg 4 \rightarrow$per garantire che $\tau_{em} \gg 4\tau_e$
  così da poter apportare semplificazioni nei poli

- Utilizzo un PI come regolatore

- $R(s) = k_{gi}\left(1 + \frac{1}{s\tau_e}\right) = 2\pi f_c \tau_e R\left(1 + \frac{1}{s\tau_e}\right)$
  - dove $f_c = 1000\ Hz$

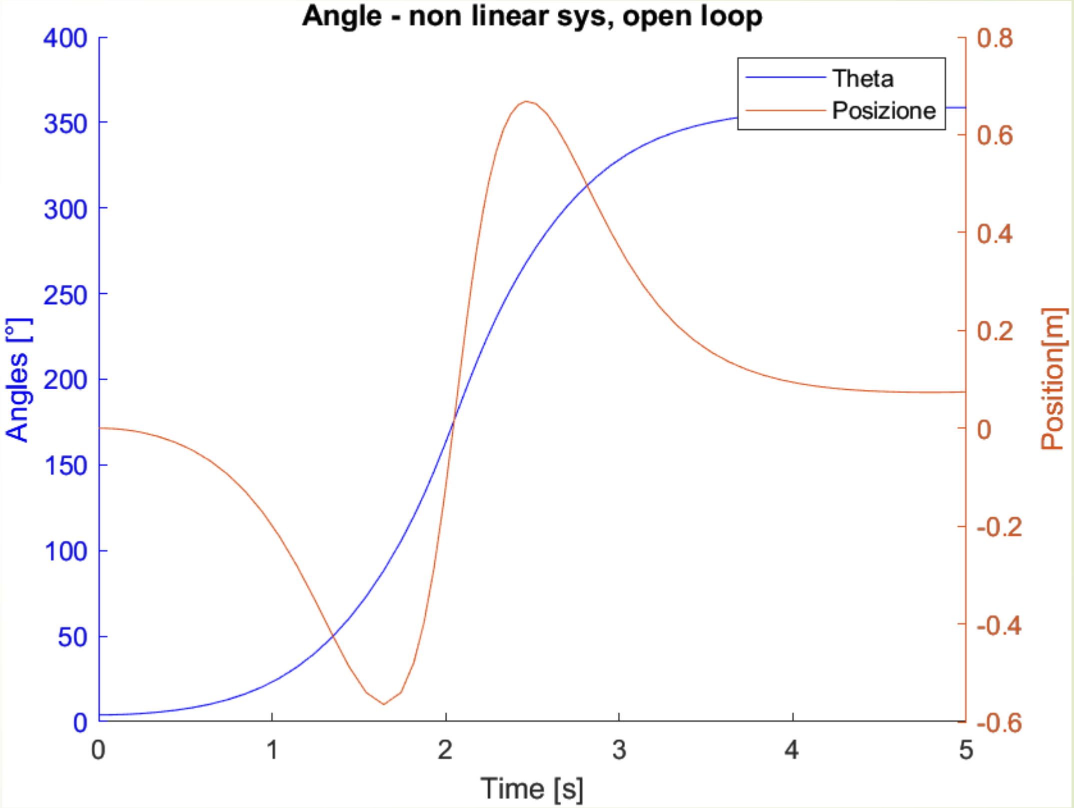
Controllo, analisi preliminari

- Per verificare la bontà della modellizzazione fatta fino a questo punto, abbiamo verificato l'instabilità del sistema senza controllo
- Dalla teoria sappiamo che se la matrice A del sistema presenta almeno un autovalore con parte reale strettamente positiva, il sistema è instabile
- Comando Matlab: $eig(A)$



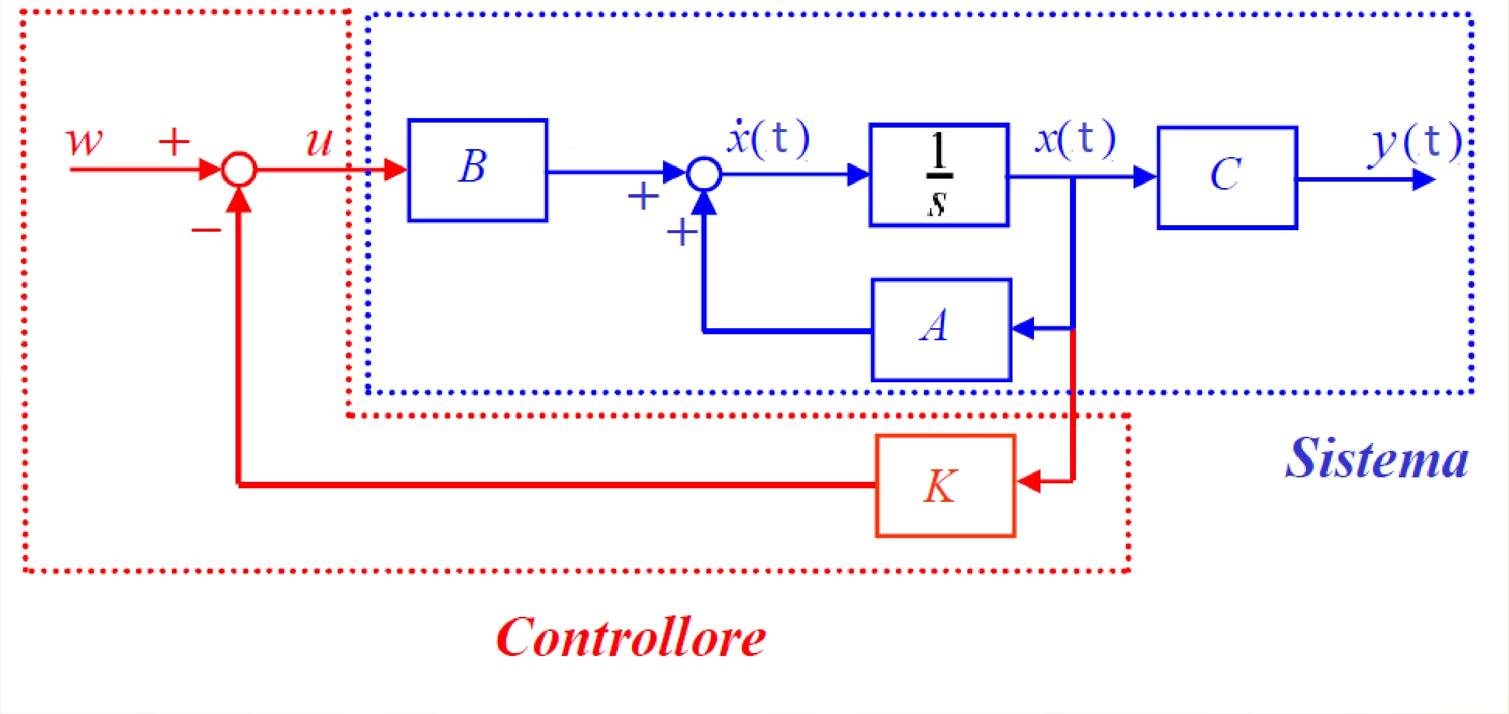
Controllo, analisi preliminari

- Comportamento del sistema non linearizzato
- Il V.A.B. compie un giro completo su sé stesso: questo sarebbe fisicamente impossibile per
  - Presenza di una superficie su cui appoggia;
  - Presenza di attriti che limitano il movimento, non permettendogli di completare l'oscillazione;
- Quindi, come è logico che sia, si rende necessario l'inserimento di un controllore.



Definizione del controllore, pole placement

- Operazione eseguita sul sistema linearizzato
- $A_{cl} = A - BK = matrice\ di\ stato\ in\ anello\ chiuso$



Controllo, retroazione dello stato

- La retroazione dello stato ci permette di posizionare i poli del sistema e quindi di imporre al sistema la dinamica a nostro piacimento;
- Coppie di poli distanziate in frequenza di una decade;
- Il polo più veloce (0.2 Hz) è stato assegnato alla parte di controllo dell'angolo θ, essendo che si preferisce raggiungere il più velocemente possibile l'equilibrio dello chassis, piuttosto che la posizione spaziale desiderata per il V.A.B. stesso;
- La frequenza dei poli è stata posizionata anche tenendo conto dei limiti di coppia imposti dai motori (che presentano appunto una corrente limitata);

Controllo, pole placement

- Lo smorzamento scelto è stato di: $\xi = 0.7$
- Le frequenze scelte sono le seguenti:

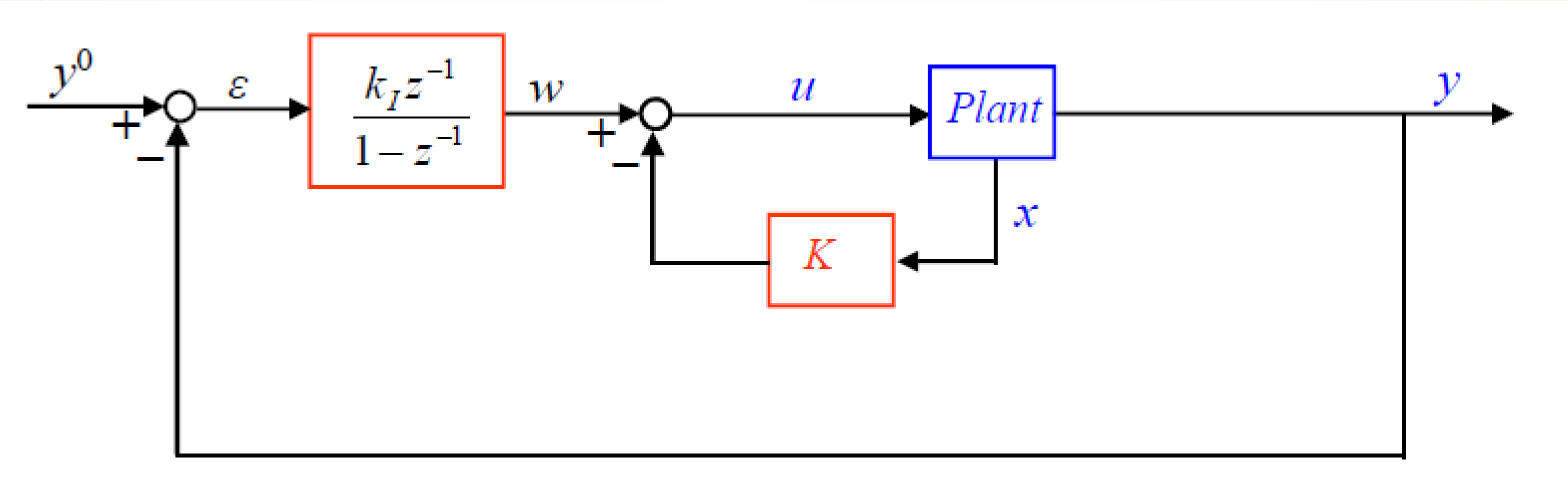
$$f_\theta = 0.2\ Hz \quad \Rightarrow \quad \mathrm{Pole}_\theta = \begin{pmatrix} -0.8796 + 0.8974\mathrm{i} \\ -0.8796 - 0.8974\mathrm{i} \end{pmatrix}$$

$$f_\phi = 0.02\ Hz \quad \Rightarrow \quad \mathrm{Pole}_\phi = \begin{pmatrix} -0.0880 + 0.0897\mathrm{i} \\ -0.0880 - 0.0897\mathrm{i} \end{pmatrix}$$

- Per individuare il vettore dei pesi K, è stato sfruttato il comando *place* di Matlab, che ha dato come risultato i seguenti valori:

$$K = [-0{,}0023 \quad -0{,}0278 \quad -28{,}1397 \quad -7{,}7022]$$

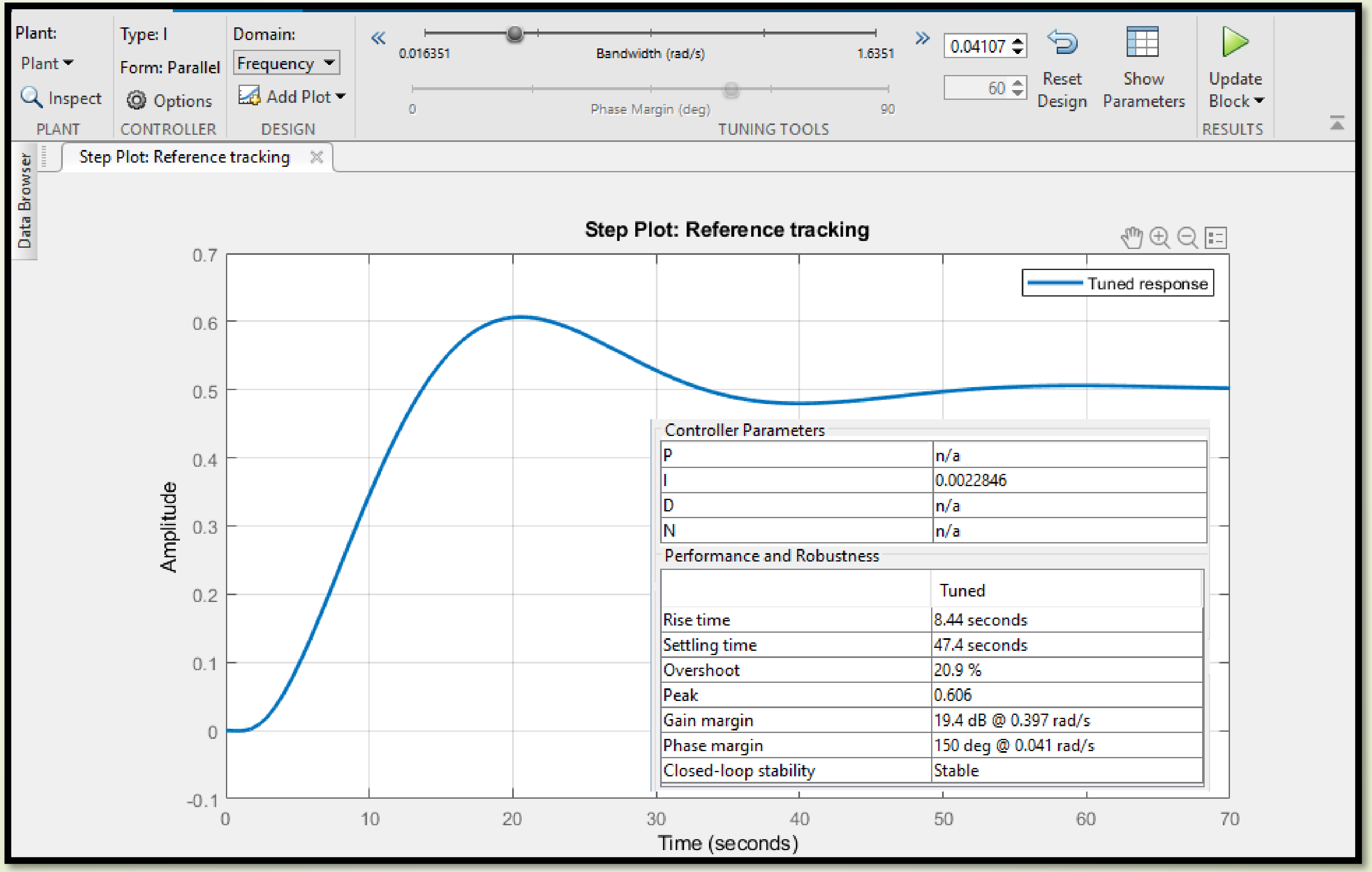
Controllo, integratore



$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ x_I(k+1) = x_I(k) + \epsilon(k) \\ u(k) = -Kx(k) + K_I x_I(k) \end{cases}$$

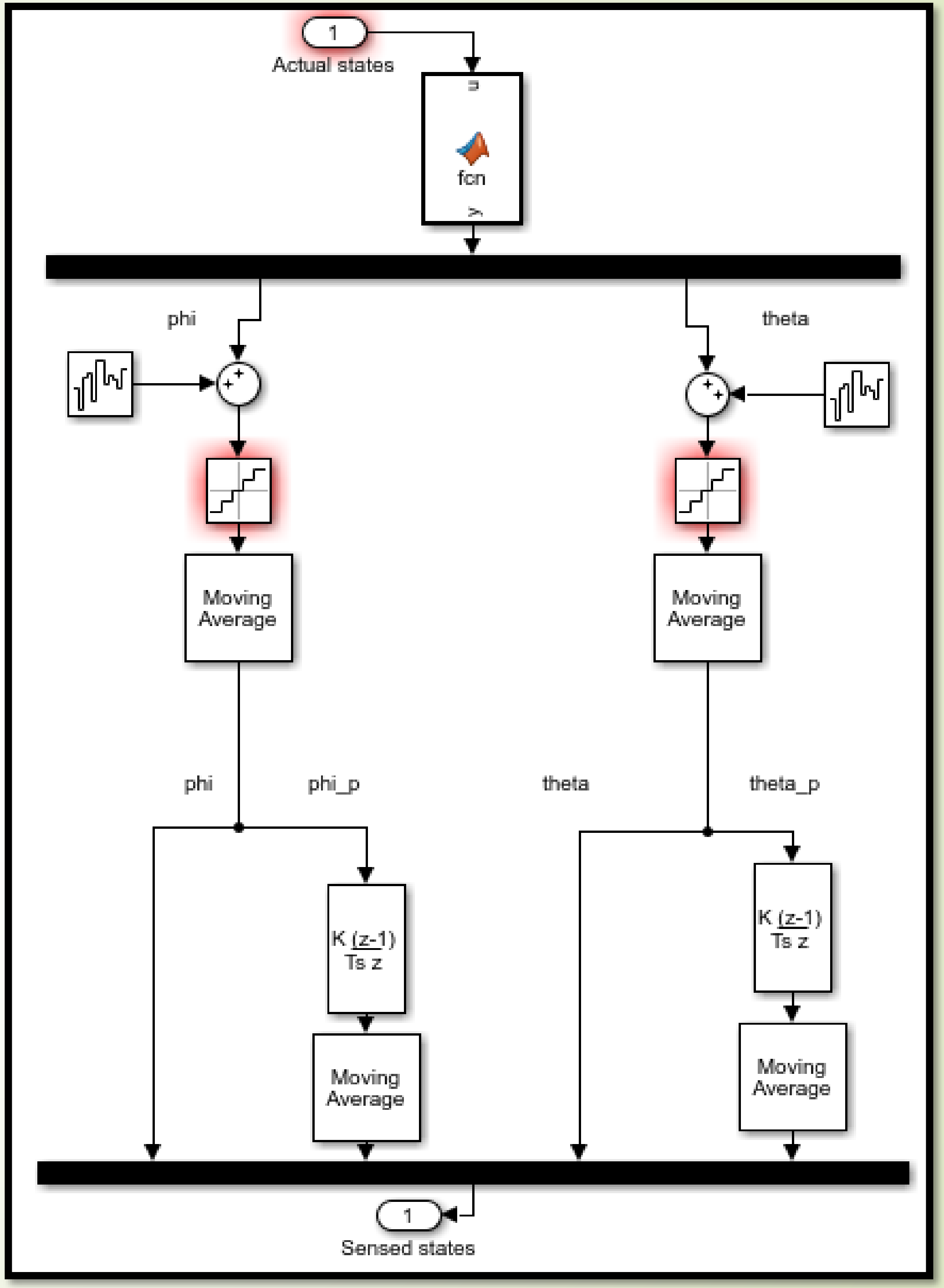
Controllo, posizionamento dell’integratore

- Posizionamento polo dell’integratore tramite il tool di Matlab per il tuning;

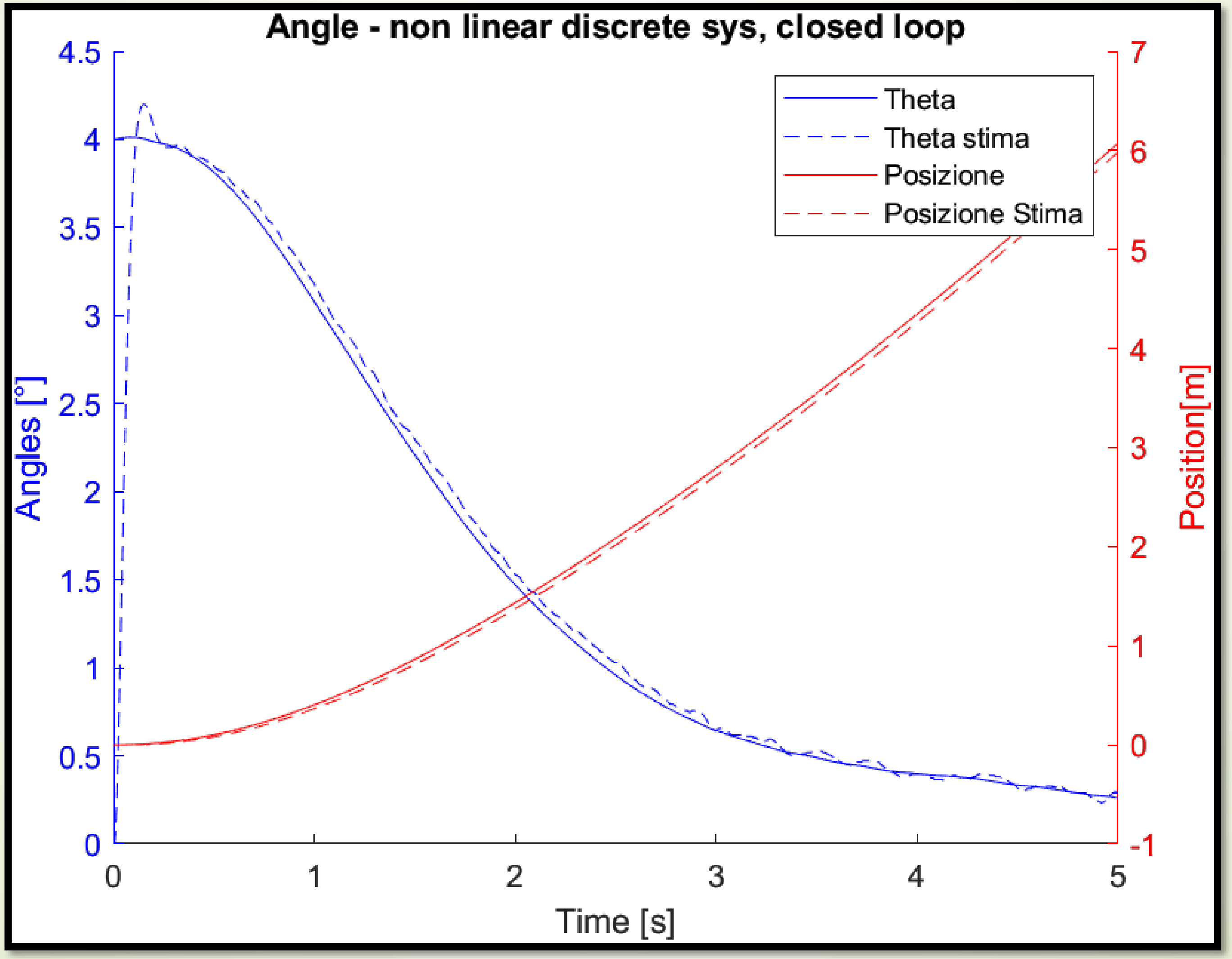


Sensori, moving average

- ⬆ Facilità di implementazione lato codice
- ⬆ Semplicità di progettazione del filtro
- ⬇ Ritardo nell'uscita
- ⬇ Lentezza nell'andare a regime

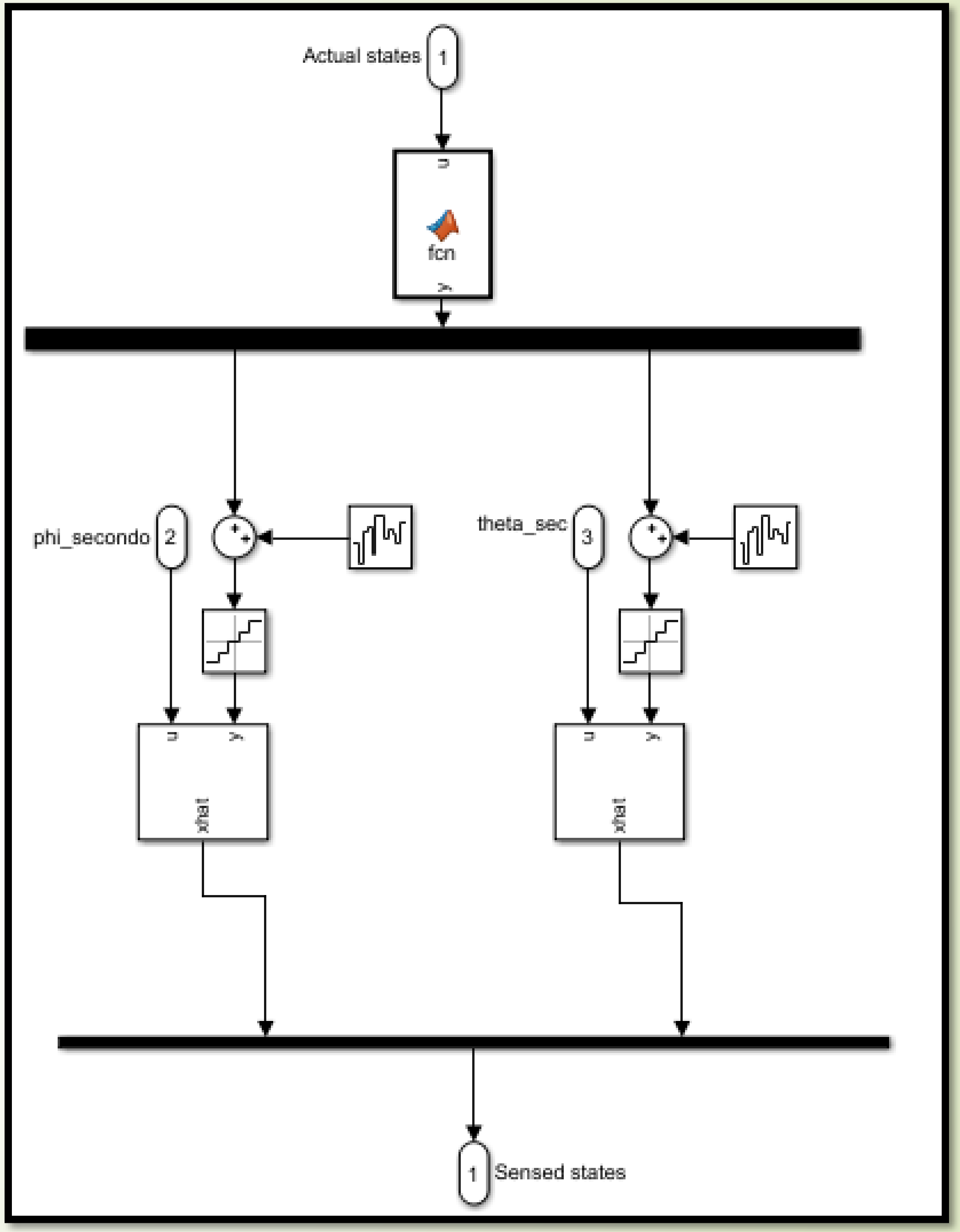


Sensori, moving average result



Sensori,
filtro di Kalman

- ↑ Strumento potente
- ↑ Molto più robusto
- ↑ Va a regime velocemente
- ↓ Tuning del rumore
- ↓ Identificazione delle matrici A, B, C



Sensori, sistema dinamico del filtro

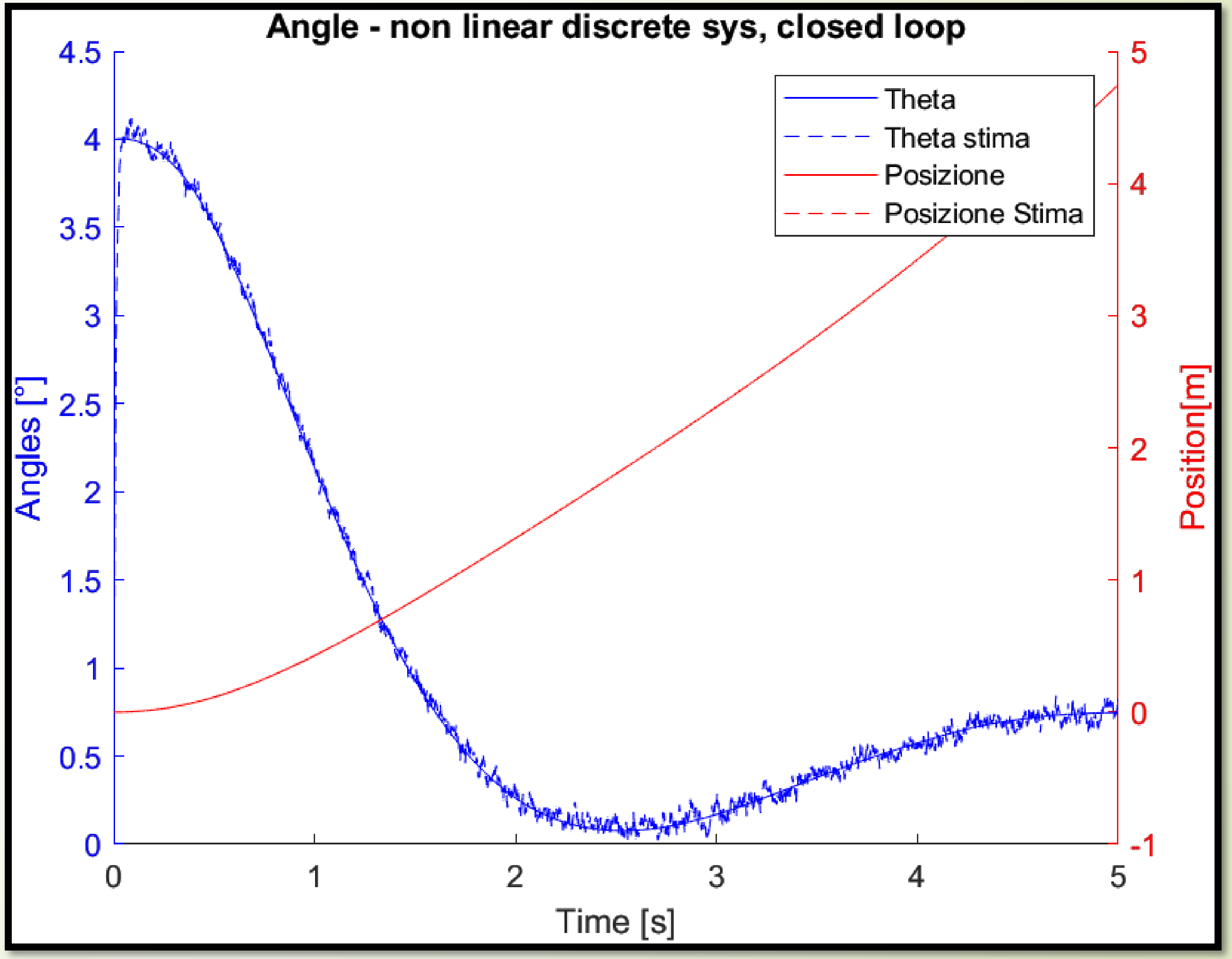
$$\begin{cases} x(t+1) = x(t) + v(t)\Delta t + \frac{1}{2}\Delta \mathrm{t}^2 \mathrm{a(t)} \\ v(t+1) = v(t) + a(t)\Delta t \\ a(t) = u(t) \end{cases}$$

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2}\Delta \mathrm{t}^2 \\ \Delta t \end{bmatrix}, \mathrm{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

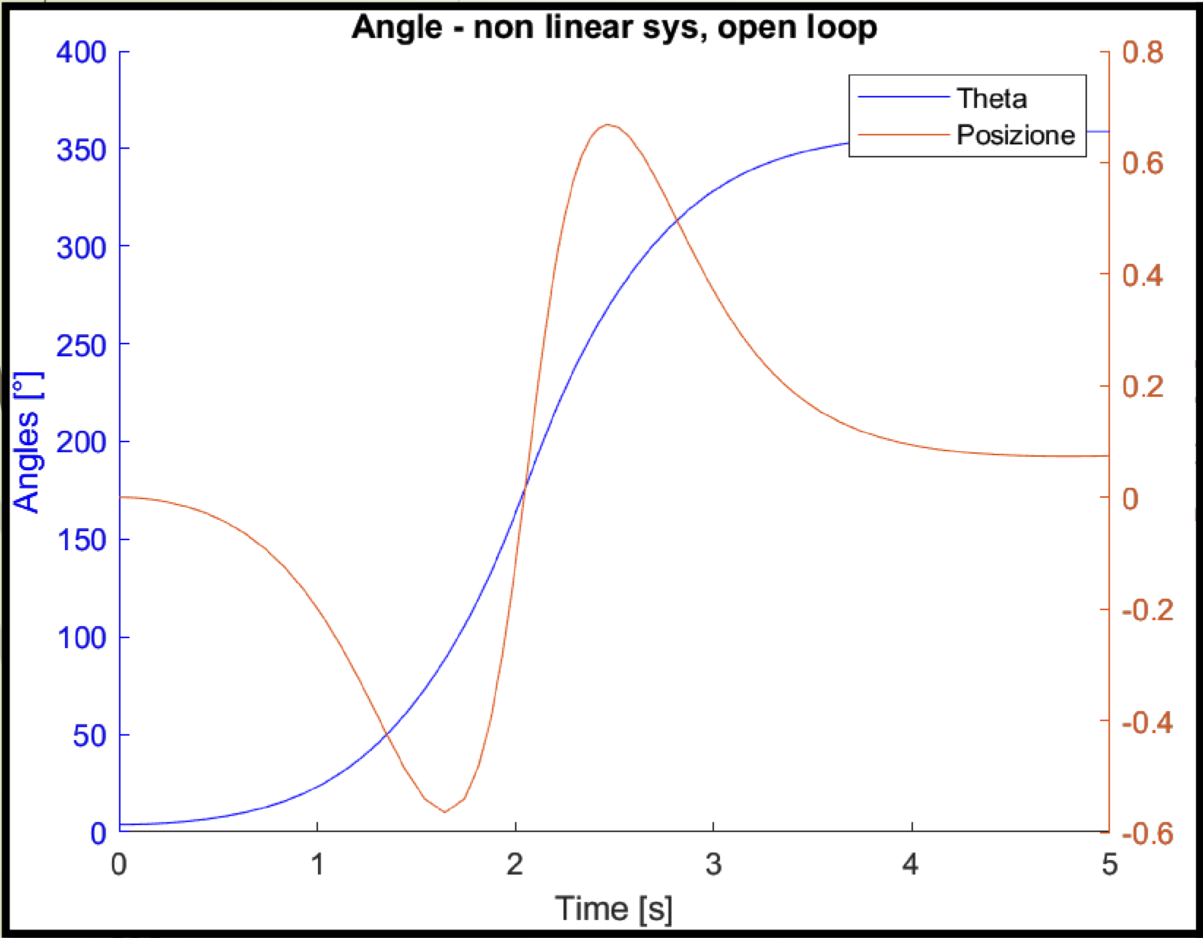
$$stato = \tilde{x} = \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} posizione \\ velocità \end{bmatrix}$$

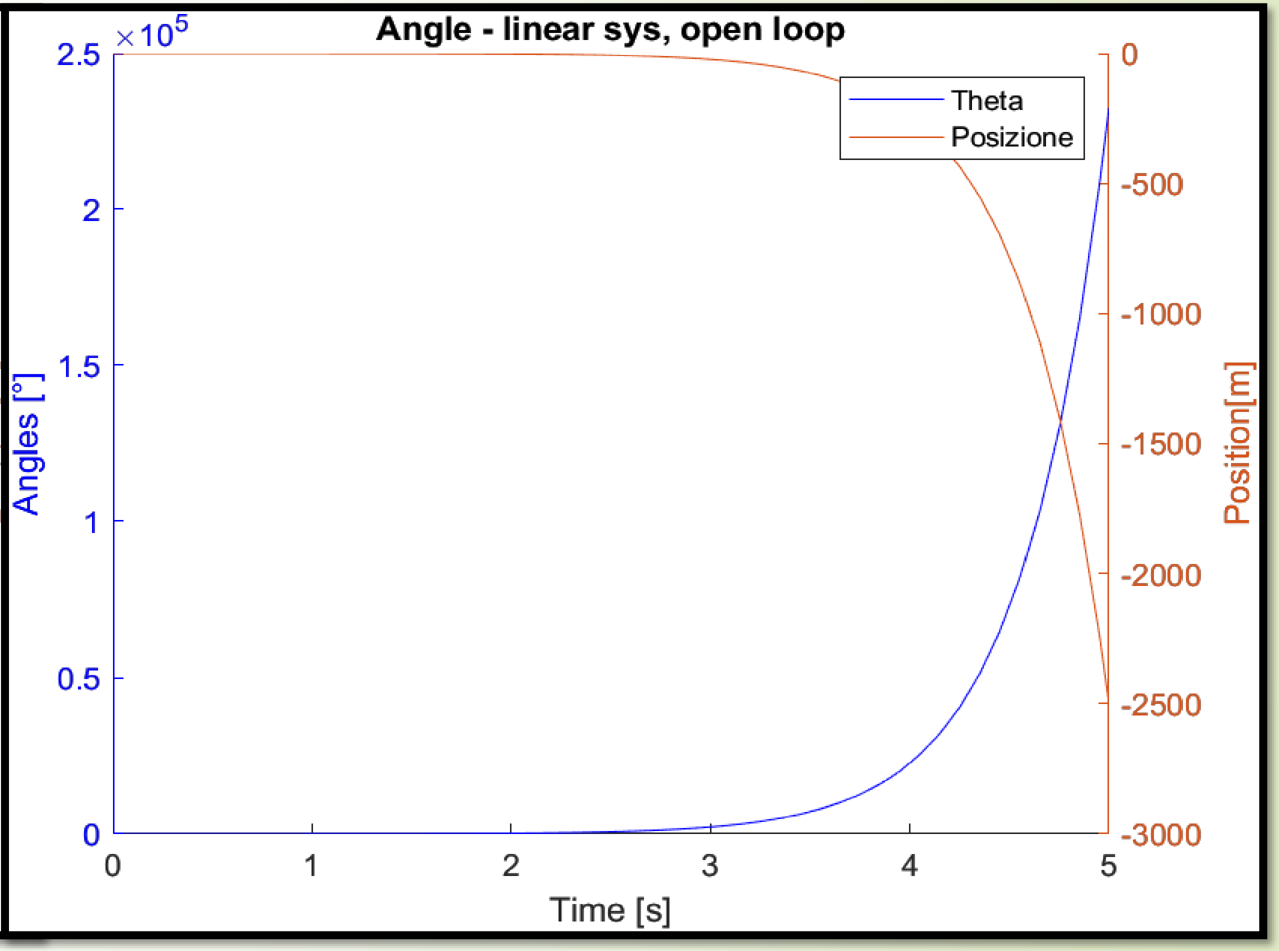
Sensori, Kalman output

- Stima dei parametri del rumore tramite trial and error

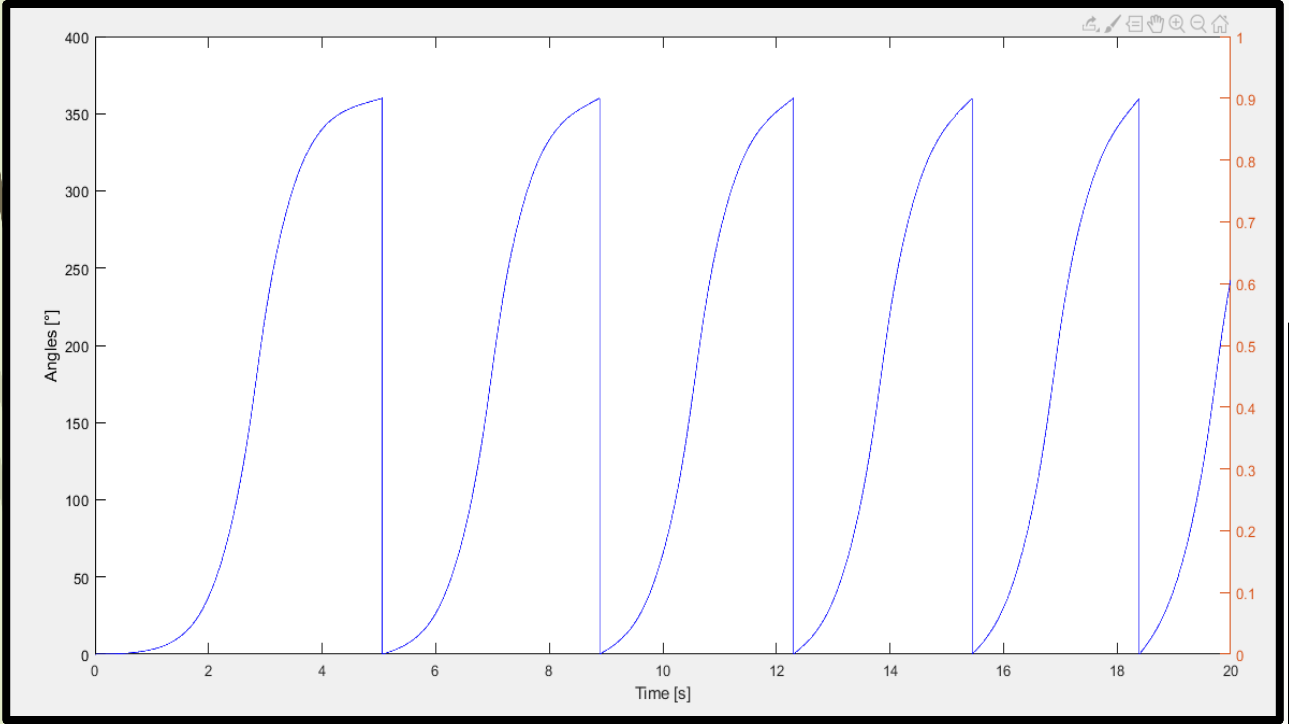


Simulazione, open loop



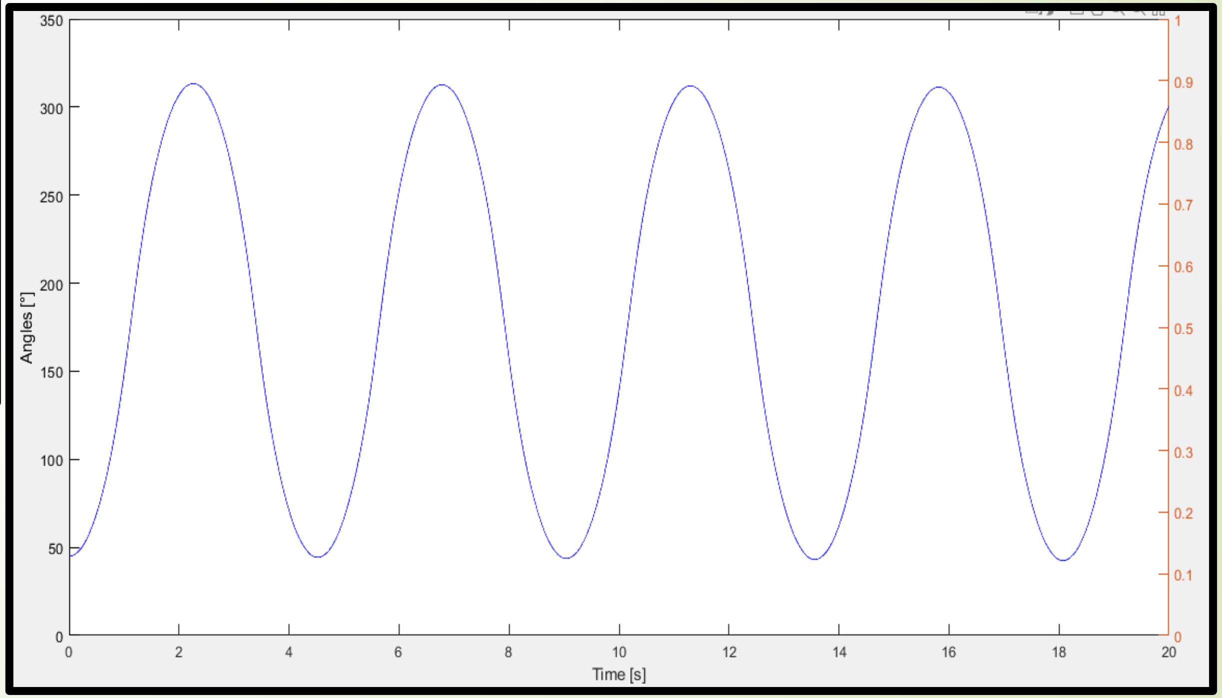


Simulazione, open loop con posizioni di partenza differenti

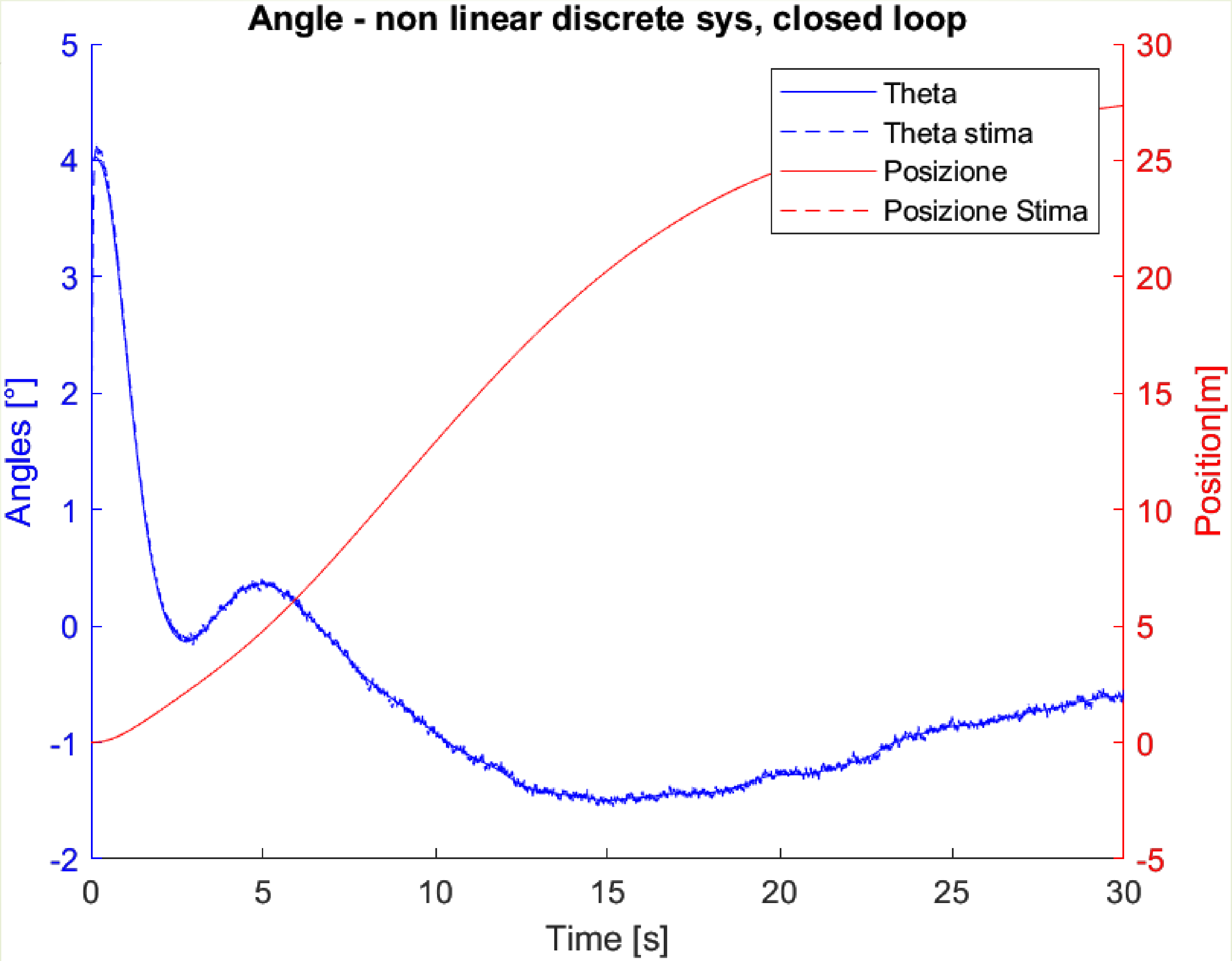


$\theta(0) = 0\ deg$

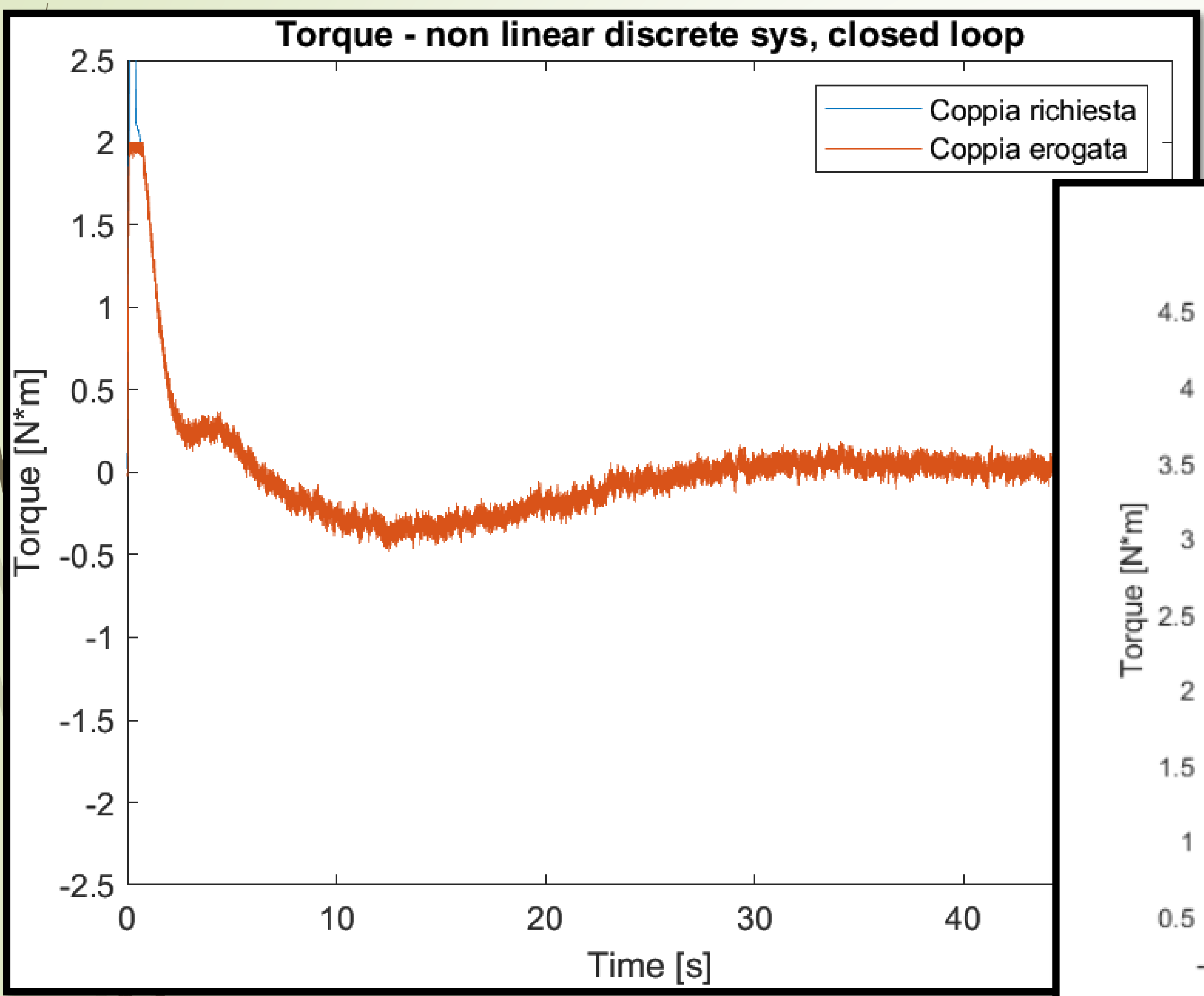
$\theta(0) = 45\ deg$

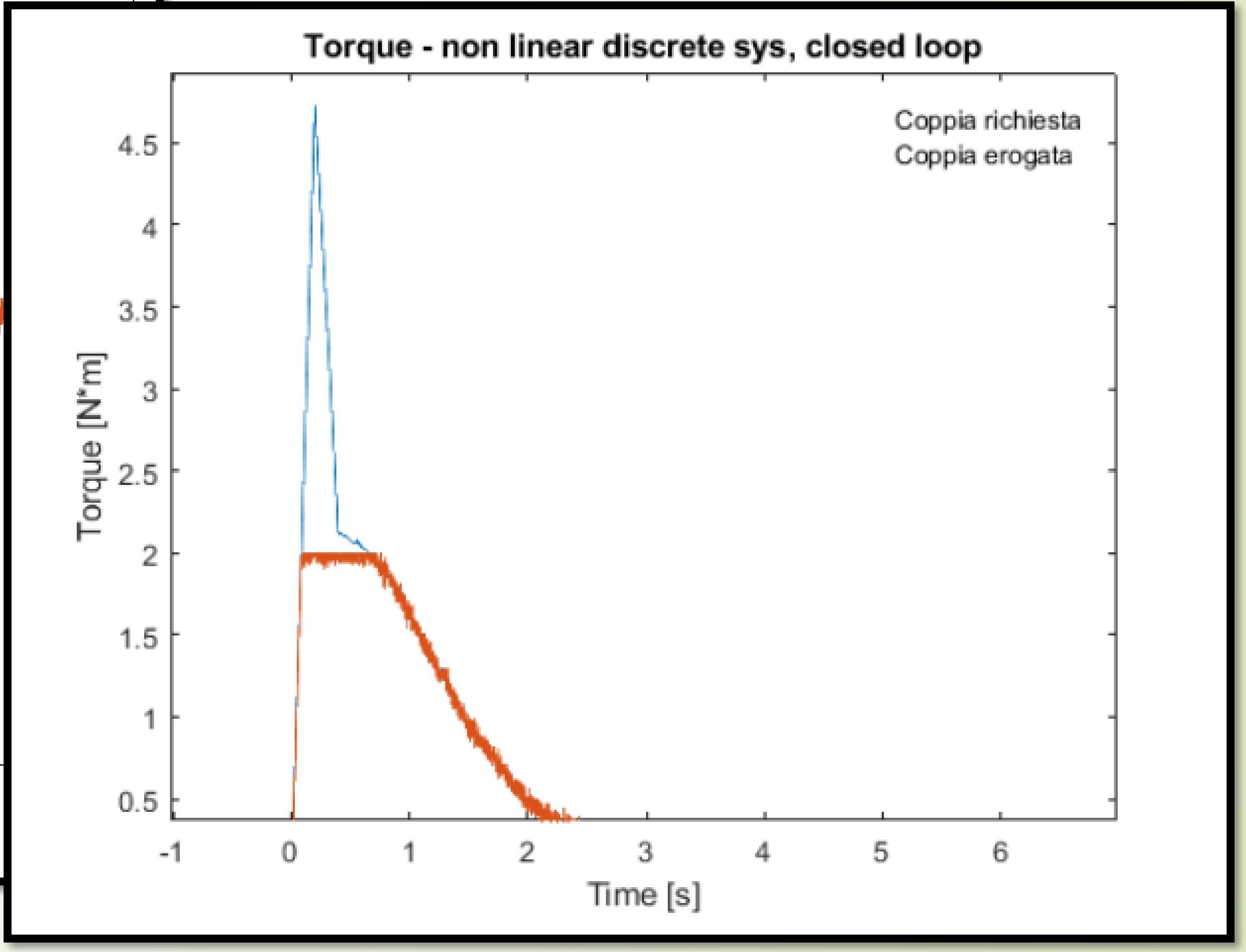


Simulazione, closed loop

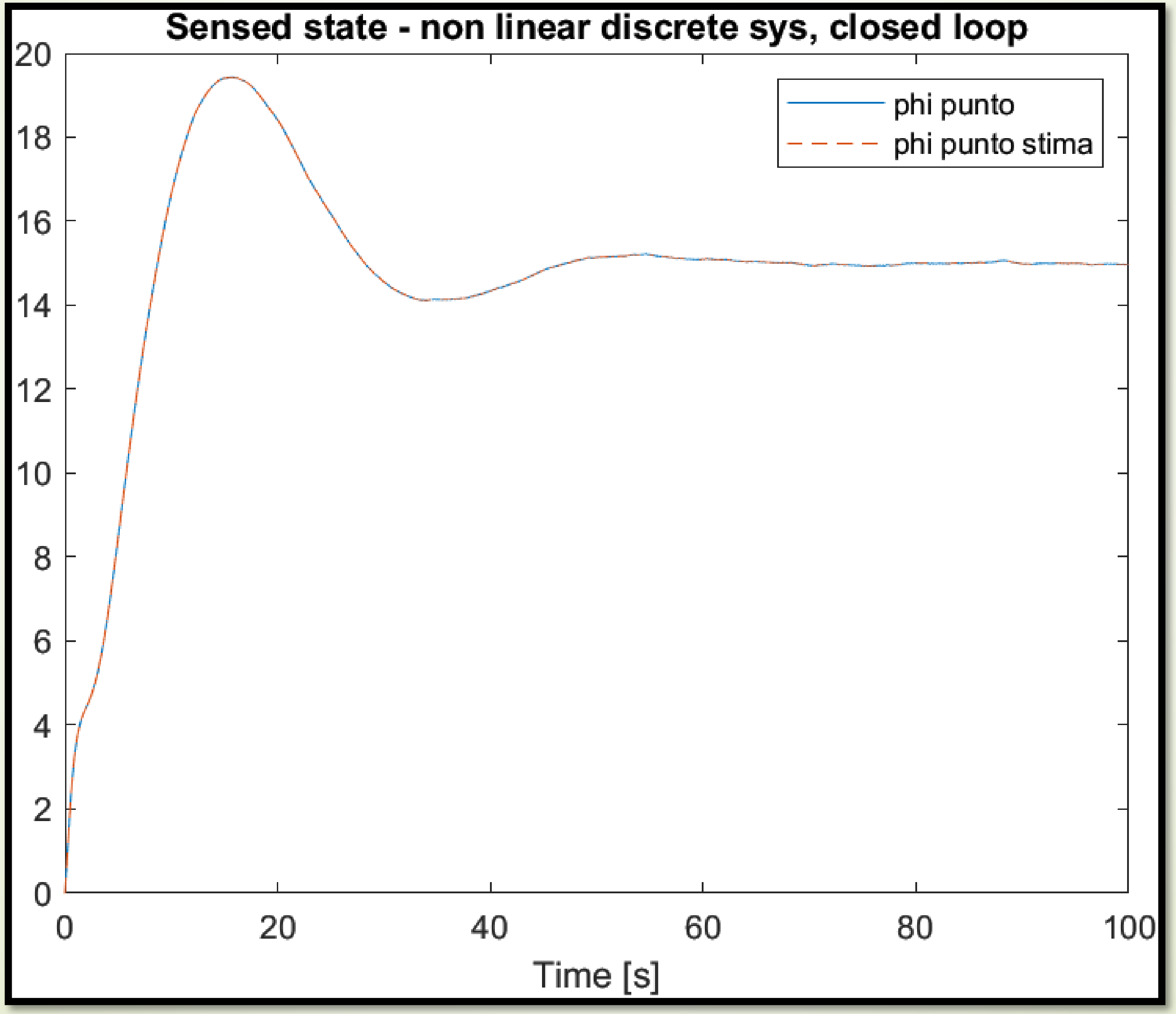


Simulazione, coppia





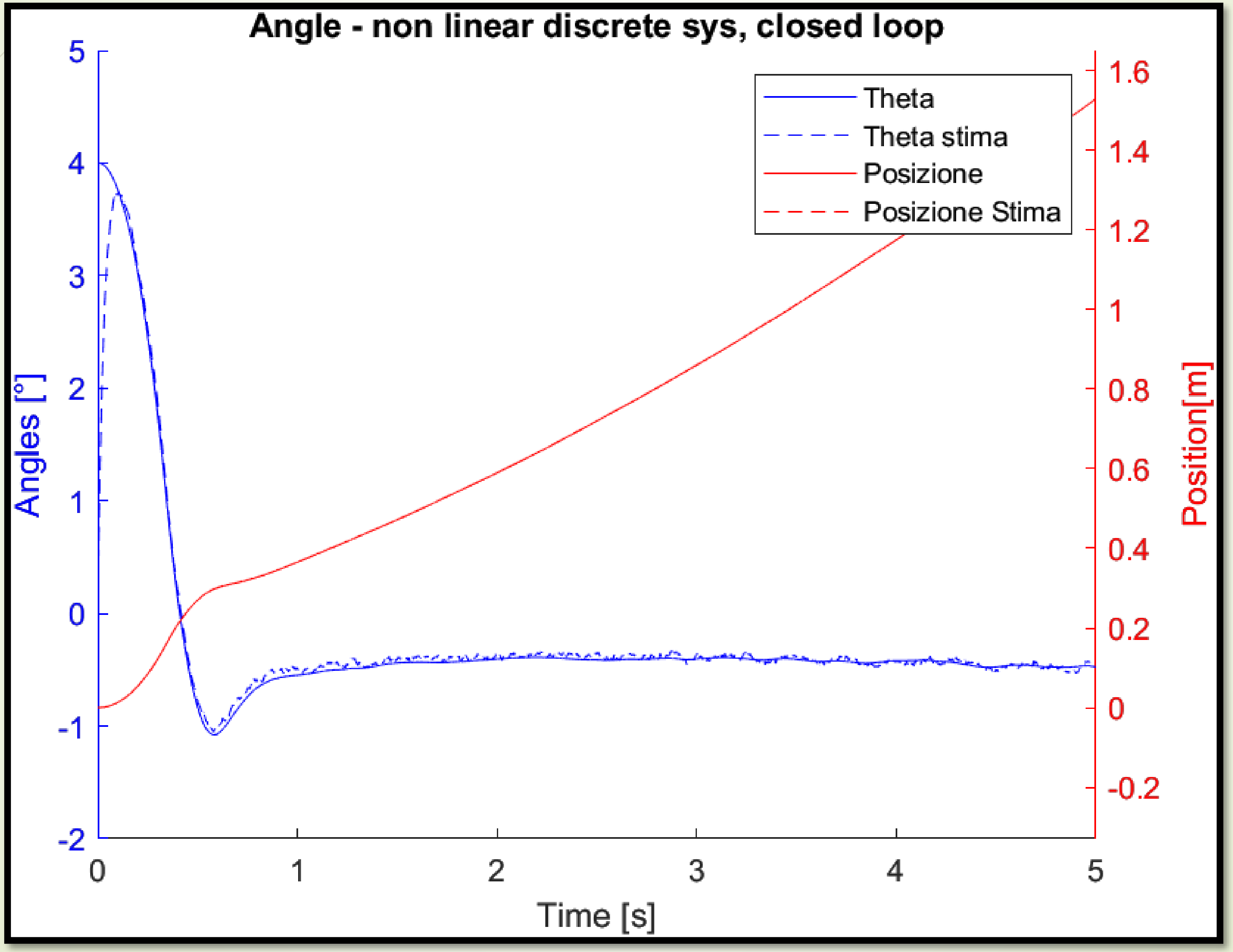
Simulazione, set point in velocità



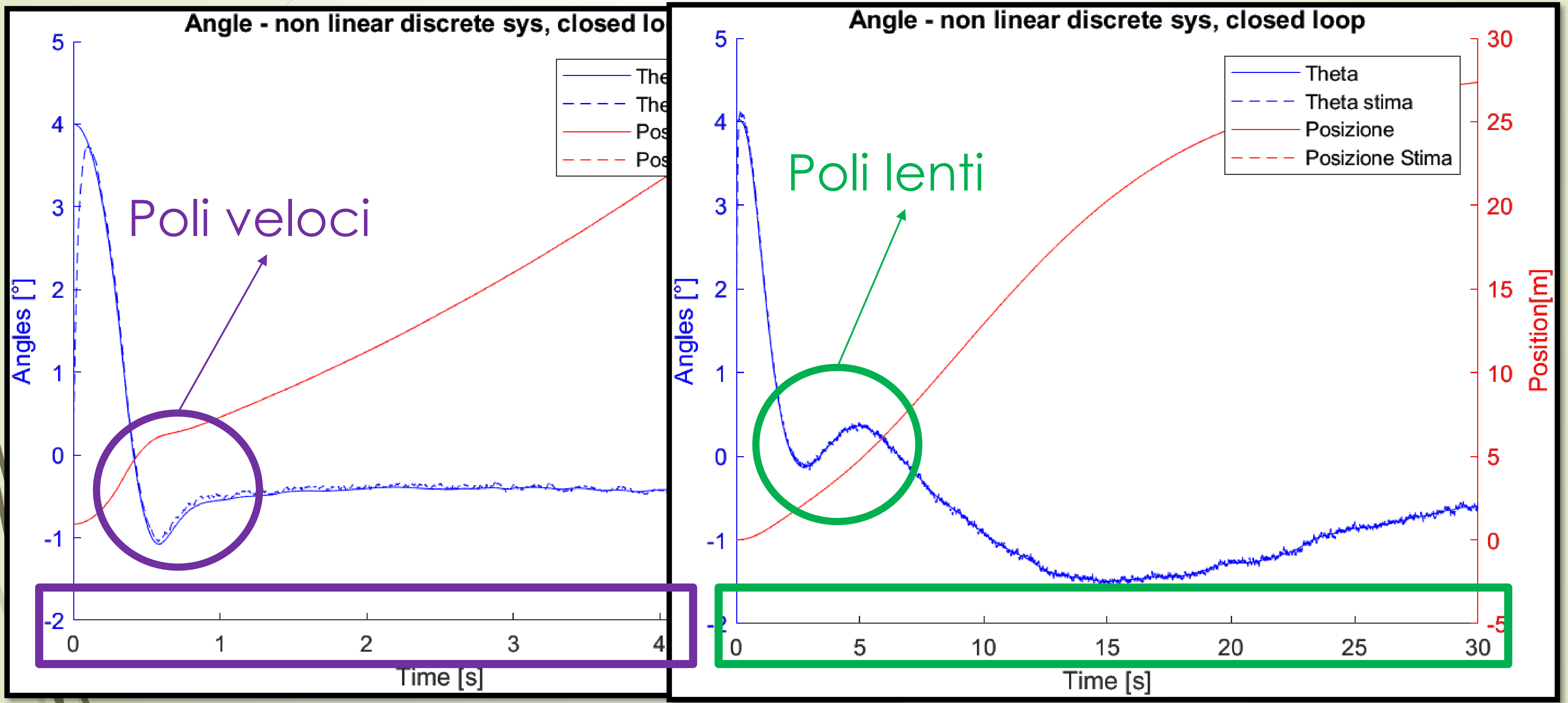
Simulazione, poli più veloci

- Aumentata la corrente limite dal motore da 20A fino a 50A
- Aumentata la frequenza del polo $f_\theta = 0{,}2\ Hz\ \rightarrow 2Hz$
- Poli effettivamente disaccoppiati in frequenza
- Prestazioni nel transitorio migliorate
- Tempo per andare a regime diminuito

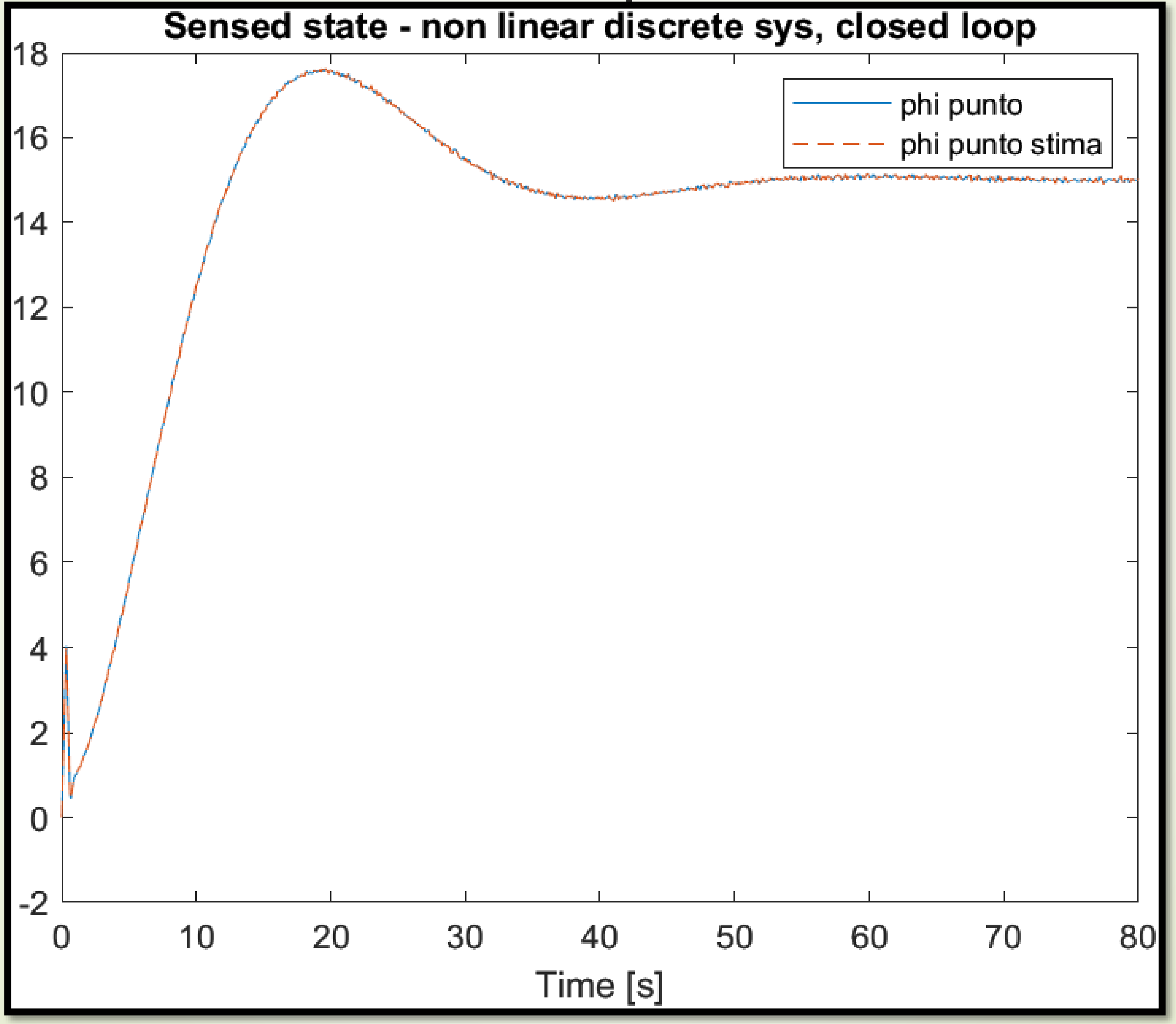
Simulazione, poli più veloci-stato



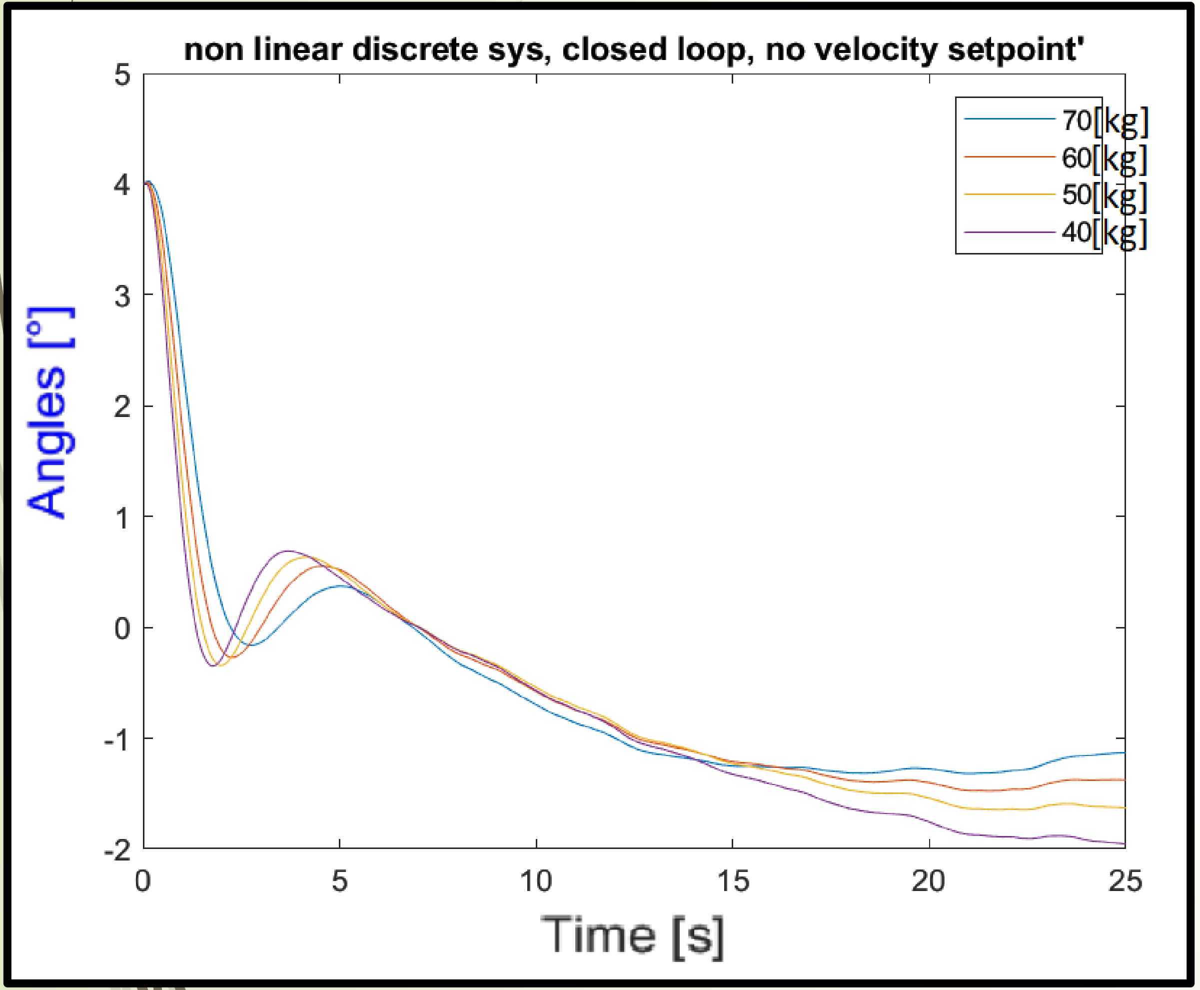
Simulazione - confronto

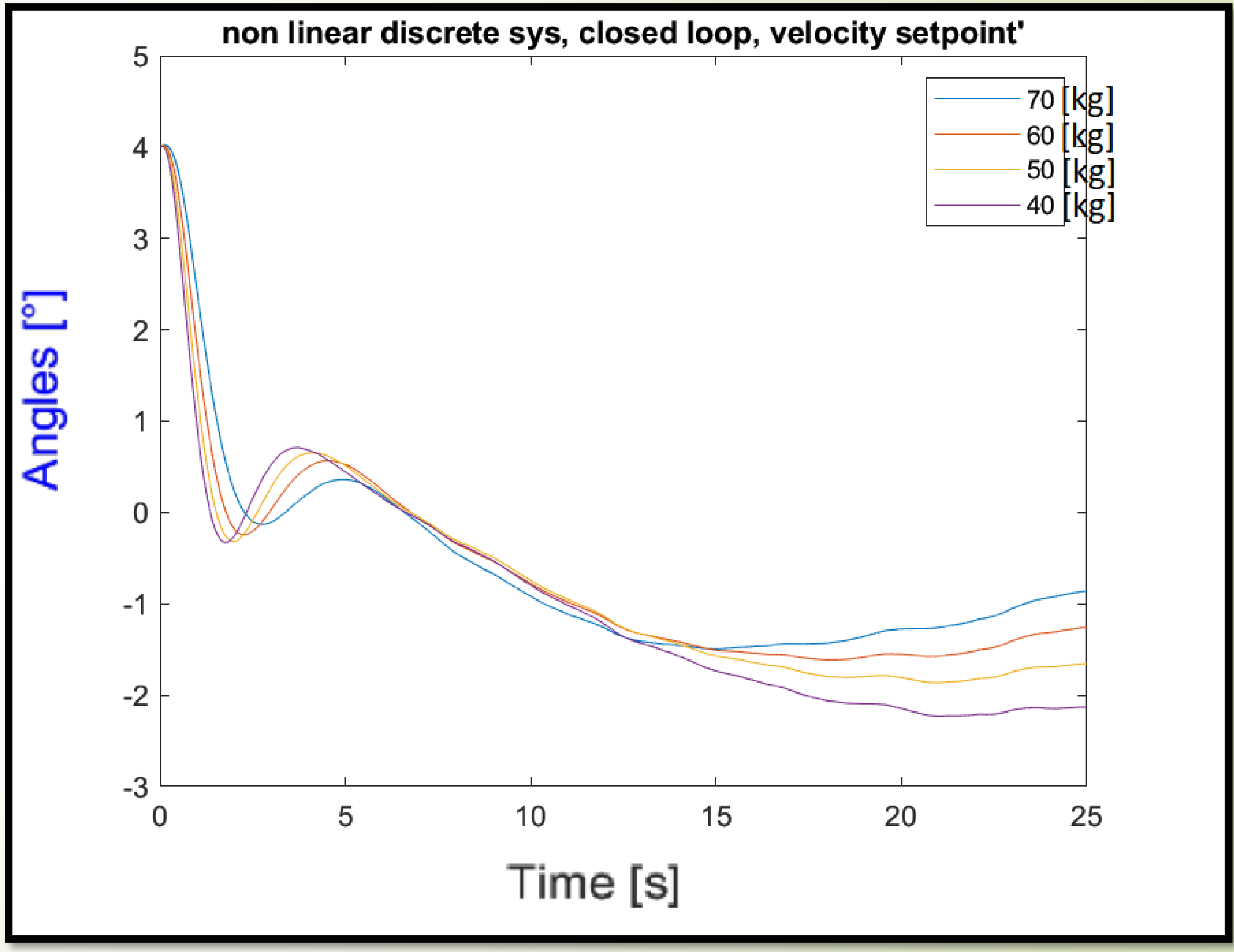


Simulazione, set point in velocità

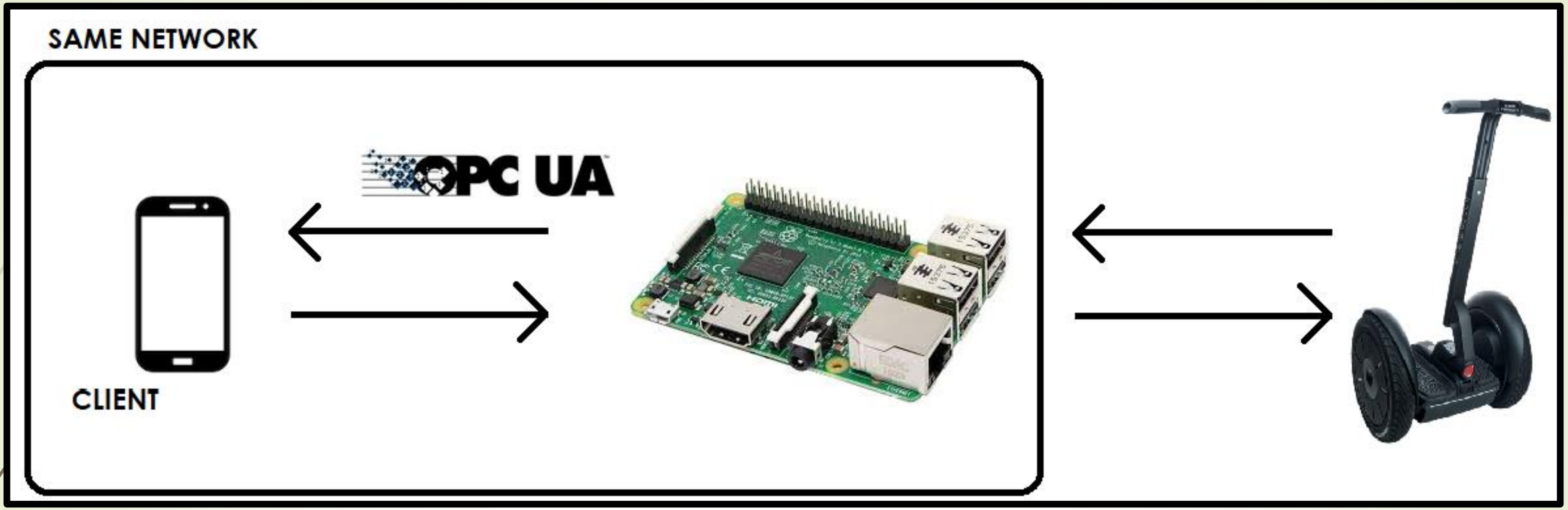


Simulazione, masse diverse





OPC-UA



- Client (su smartphone): Android e/o iOS
- Server (su Raspberry): Python

OPC-UA, specifiche

- Inizializzo il vettore dei gain $K = [K_\phi, K_{\dot{\phi}}, K_\theta, K_{\dot{\theta}}]$ con i parametri salvati su un file permanente in memoria (se questo file non è presente, poiché si tratta del primo avvio del sistema, i parametri saranno inizializzati a 0) e ne crea uno identico temporaneo;
- Ogni modifica successiva ai parametri nel vettore K corrisponde ad una modifica dei valori nel file temporaneo;
- Il codice Python eseguito su Raspberry legge il file temporaneo e utilizza quei valori per calcolare l'azione di controllo da fornire in pasto al sistema;
- Fino a quando l'utente non va a settare il flag per il salvataggio definitivo, tramite l'applicazione smartphone, i parametri non vengono aggiornati sul file temporaneo;
- L'utente può scegliere di modificare anche il file permanente a sua discrezione e, se necessario, può scegliere di fermare l'esecuzione del server (shut down);

Xenomai, real time Raspberry Pi

- È necessario che, una volta scelta T_s, il calcolatore assicuri di terminare la computazione entro e non oltre questa deadline temporale;
- Necessario, quindi, rendere Raspberry Pi un sistema operativo real-time (*RTOS – Real Time Operative System*);
- Non avendo un Raspberry Pi fisicamente disponibile, abbiamo proceduto ad installare una VM con già a bordo il framework Xenomai;
- Xenomai è un framework che fornisce API real-time a sistemi operativi che non lo sono;

Xenomai, Wrapper

```
void wrap() {
        printf("START");
        int counter = 0;
        RTIME period  = 1000000000;
        rt_task_set_periodic(NULL,TM_NOW,period);
        RTIME now = rt_timer_read();
        RTIME before= now-period;
        while(1){
                now = rt_timer_read();
                read_sensor_phi(counter);
                read_sensor_theta(counter);
                calculate_phi_p(counter);
                calculate_theta_p(counter);
                read_k();
                controllore();
                controllore_motore(counter);
                counter = (counter+1)%WINDOW_WIDTH;
                printf("Actual motor torque: %f\n", actual_cm);
                printf("Elapsed time: %f\n",Ts);
                before= now;
        }

        return;
}
```

$T_s(ns)$

Init variabili per la gestione del tempo

Ciclo di lavoro

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos	^Y Prev Page
^X Exit	^R Read File	^\ Replace	^U Uncut Text	^T To Spell	^_ Go To Line	^V Next Page

Xenomai, funzionamento real-time

- Come si vede in figura, i requisiti temporali sono rispettati: ogni ciclo macchina richiede un tempo di esecuzione fisso (in questo caso 1 ms)