ISI report

Francesco Pegonzi, Vittorio Rosellini

A.A. 2023-2024

1 Modello del sistema dinamico

Il sistema in esame è un pendolo inverso montato su un carrello azionato da un motore elettrico. Basandoci sull'articolo fornito con il testo dell'esercizio possiamo ricavare le equazioni del moto, riportate di seguito

$$\ddot{x}_c = \frac{VNW - \dot{x}_c NW^2 + g\,cos(\theta)sin(\theta)R_m P^2 + \dot{\theta}sin(\theta)R_m NP}{R_m(NM - cos^2(\theta)P^2)} \tag{1.1}$$

$$\ddot{\theta} = -\frac{P(g\,sin(\theta)R_m M + cos(\theta)\dot{\theta}^2 sin(\theta)R_m P + cos(\theta)VW - cos(\theta)\dot{x}_c W^2}{R_m(NM - cos^2(\theta)P^2)} \tag{1.2}$$

All'interno di queste equazioni sono riportati i simboli W, P, N, M che verranno discussi in seguito. Le quantità θ, X_c sono raffigurate in figura 1.

Dall'articolo allegato al testo dell'esercizio andando nella sezione *2.2 Linear System Model* viene specificato $u = V$, ovvero che l'unico ingresso del sistema è la tensione ai capi del motore. Procedendo con queste informazioni possiamo ricavare il modello tempo continuo del sistema dinamico e implementarlo in Simulink in modo da validarne il comportamento.

1.1 Scelta dei valori nominali

All'interno dell'articolo non viene fornito alcun valore numerico, alla luce di ciò siamo costretti a decidere i parametri del sistema senza neanche avere un idea di quanto possa essere grande ogni oggetto. Partendo dalla decisione di avere un binario su cui far scorrere il carrello della lunghezza di 10m siamo giunti ad alcuni parametri che ci sembrano plausibili riguardo le dimensioni e le masse dei vari elementi in gioco. Tutti i parametri sono modificabili all'interno del file *init_param.m* che viene automaticamente eseguito all'inizio di ogni simulazione. I parametri del carrello sono stati scelti come segue cercando di mantenere le proporzioni con il disegno fornito nel testo dell'esercizio.

- Massa del pendolo: 10 kg, Distanza del pendolo dal centro del carrello: 3 m
- Massa del carrello: 30 kg
- Massa delle due aste: 10 kg, Semi lunghezze delle aste: 3 m

Per quanto riguarda il motore invece abbiamo deciso di prenderne uno vero così da utilizzare dei parametri, in particolare parliamo di un servo motore che opera in corrente continua con i seguenti parametri:

- Resistenza di armatura: 0.11 Ω
- Costante del motore: 0.23 $\frac{Nm}{A}$

Con il diametro del pignone del carrello ($r = 0.05\,m$) e il rapporto di trasmissione del rotismo epicicloidale $K_g = 4$ possiamo calcolare i parametri W, P, N, M come specificato nell'articolo allegato.

1.2 Validazione

Per validare il sistema sono state utilizzate soprattutto prove con ingresso nullo variando le condizioni iniziali del sistema. Le prove effettuate hanno dato risultati compatibili con il comportamento che ci si aspetterebbe intuitivamente da un sistema del genere, in particolare:

- $x_0 = \vec{0}$ il sistema rimane fermo e non si muove dalle condizioni iniziali
- $\dot{\theta} = 0.1$ $[rad/s]$ il pendolo comincia ad oscillare portandosi dietro per inerzia il carrello il quale si muove sul binario diminuendo progressivamente l'ampiezza delle oscillazioni

- $\theta = \pi + 0.01$ $[rad]$ il sistema si trova nella posizione con la massa ad altezza massima la quale cadendo muove il carrello, in questa configurazione si nota che ogni volta che il pendolo raggiunge la condizione $\dot{\theta} = 0$ l'altezza è sempre minore andando avanti con la simulazione

Il precedente elenco non rappresenta un elenco esaustivo delle prove fatte per validare il sistema, tuttavia le prove più significative sono quelle riportate sopra, le quali, riportano un comportamento ragionevole del sistema.

1.3 Incertezze

Il sistema viene eseguito ogni volta con dei parametri diversi da quelli indicati nella sezione 1.1, in particolare abbiamo aggiunto incertezza su:

- Masse: ogni massa all'interno del sistema si porta dietro un'incertezza di $0.1kg$[1], il sistema tempo continuo viene eseguito con una massa pari alla massa nominale a cui viene sommata l'incertezza moltiplicata per un numero casuale nell'intervallo [-1;1].
- Lunghezze: lo stesso discorso fatto per le masse è valido anche in questo contesto con un incertezza di $0.01m$.
- Condizioni iniziali: le incertezze utilizzate in questo caso sono: $0.1m$ per la coordinata X_c, $0.1rad$ per l'angolo, $0.01m/s$ per la velocità del carrello e $0.001rad/s$ per la velocità angolare.

Gli algoritmi che implementano i filtri invece utilizzano i parametri geometrici e inerziali come specificato nella sezione 1.1. Lo stato iniziale per inizializzare gli algoritmi di filtraggio è quello nominale accompagnato da una matrice di covarianza che contiene al suo interno tutte le incertezze sullo stato.

Per quanto riguarda l'ingresso abbiamo deciso di aggiungere un rumore bianco a media nulla e varianza $0.01V$ al segnale nominale, a frequenza 100Hz.

2 Modello di misura

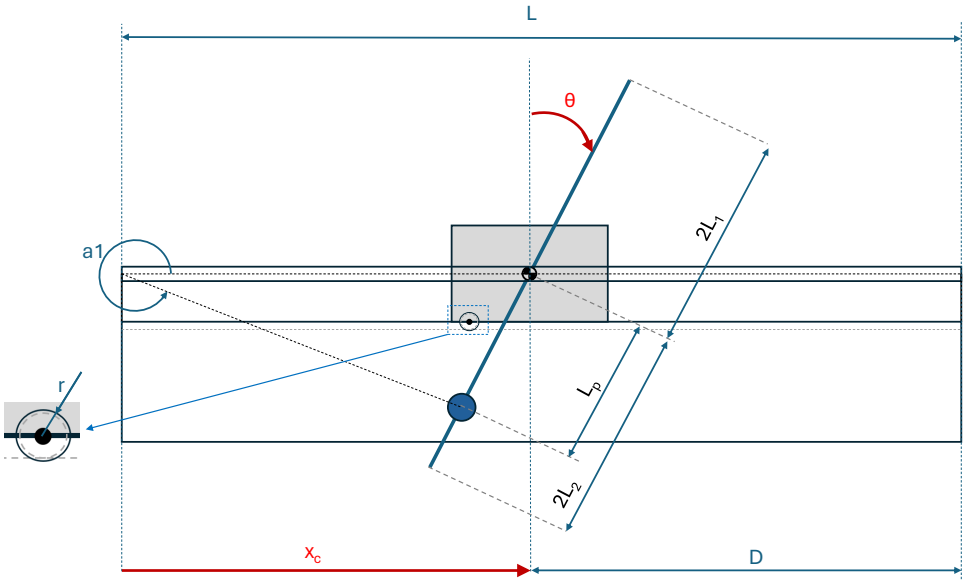

Figura 1: Schema sistema dinamico

Il modello di misura mette in relazione, mediante relazioni geometriche e cinematiche, lo stato con le seguenti grandezze fisiche:

- Distanza D
- Angolo a1

[1]Ipotizzando di aver misurato tutte le grandezze con lo stesso strumento

- Velocità angolare pignone carrellino (ω)

I sensori ritenuti più opportuni per effetuare tali misurazioni sono rispettivamente:

- Sensore laser Autosen AL001
- Telecamera Logitech C270
- Sensore effetto Hall Vishay 981HE

I modelli di misura di ciascun sensore sono stati implementati su simulink in tre blocchi separati. Ciascuno caratterizzato dal proprio periodo di campionamento, coerentemente con le caratteristiche dei sensori presenti sui datasheet.

2.1 Blocco Dsistance-Mesaurement

Tale blocco presenta la funzione che lega stato e distanza D :

$$D = L - x_c \tag{2.1}$$

Il tempo di campionamento di tale blocco è 30ms.

2.2 Blocco Angle-Mesaurement

Tale blocco presenta la funzione che lega stato ed angolo a1 :

$$a1 = \text{atan2}(-Lp \cdot cos(\theta), x_c - Lp \cdot sin(\theta)) \tag{2.2}$$

Il tempo di campionamento di tale blocco è 40ms. La misura dell'angolo a1 nel testo è indicata nell'intervallo $[0, 2\pi]$ è necessario quindi correggere l'equazione 2.2 aggiungendo 2π se l'angolo è negativo. Per semplicità però all'interno del filtro abbiamo utilizzato l'equazione 2.2 riportando prima il valore dell'angolo nell'intervallo $[-\pi, \pi]$ per facilitare la derivata nel filtro EKF.

2.3 Blocco Angular-Speed-Mesaurement

Tale blocco presenta la funzione che lega stato e velocità angolare ω :

$$\omega = \dot{x}_c / r \tag{2.3}$$

Il tempo di campionamento di tale blocco è 10ms.

2.4 Rumori di misura

Il rumore di misura dei sensori è stato modellato come additivo e generato mediante un numero random con distribuzione gaussiana di media 0 e varianza σ^2 dipendente dalle specifiche del sensore.

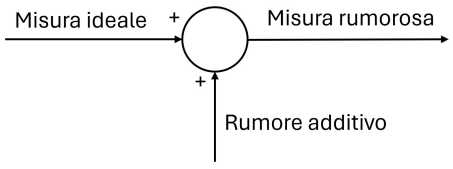

Figura 2: Rumore additivo

2.5 Tempi associati a segnali da sensori

Ad ogni misura dai sensori è stato associato un istante temporale, campionato con frequenza pari a quella del rispettivo sensore, in modo da identificare il tempo a cui tale misurazione è stata effettuata. Questa procedura è stata necessaria per gestire correttamente le misure aggiornate nella successiva fase di filtraggio.

3 Stima dello stato

La stima dello stato viene effettuata mediante il sistema Osservatore, esso è un sistema dinamico che riceve in ingresso:

- V_k ingresso nominale (senza rumore)
- y_k misure dal sistema reale (modello sistema dinamico + modello di misura)

E restituisce come uscita lo stato stimato $\hat{x}_k$.

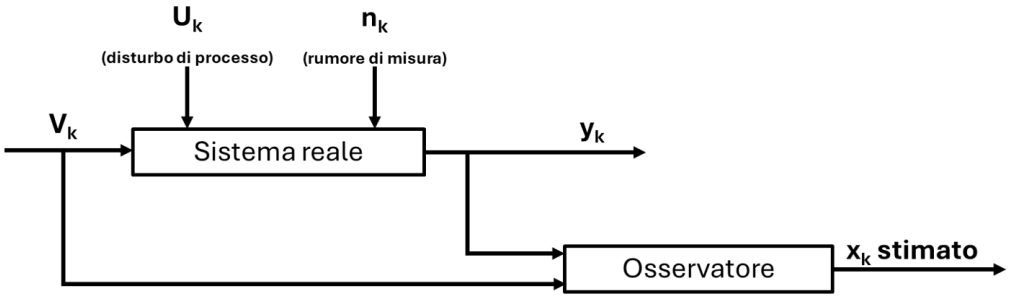

Figura 3: Osservatore

L'osservatore utilizzato in questo elaborato è il filtro di Kalman. Esso utilizza il modello del sistema dinamico e di misura, discretizzati.

3.1 Discretizzazione

Il modello del sistema dinamico ed il modello di misura sono stati discretizzati con Eulero in avanti, la dinamica è stata poi riportata in forma di stato, i risultati sono i seguenti:

- Modello del sistema dinamico

$$x_{c,\mathrm{k+1}} = x_{c,k} + \dot{x}_{c,k}dt \tag{3.1}$$

$$\theta_{\mathrm{k+1}} = \theta_k + \dot{\theta}_k dt \tag{3.2}$$

$$\dot{x}_{c,\mathrm{k+1}} = \dot{x}_{c,k} + \frac{(V_k + U_k)NW - \dot{x}_{c,k}NW^2 + g\,cos(\theta_k)sin(\theta_k)R_mP^2 + \dot{\theta_k}sin(\theta_k)R_mNP}{R_m(NM - cos^2(\theta_k)P^2)}dt \tag{3.3}$$

$$\dot{\theta}_{\mathrm{k+1}} = \dot{\theta}_k + \frac{P(g\,sin(\theta_k)R_mM + cos(\theta_k)\dot{\theta_k}^2 sin(\theta_k)R_mP + cos(\theta_k)(V_k + U_k)W - cos(\theta_k)\dot{x}_{c,k}W^2}{R_m(NM - cos^2(\theta_k)P^2)}dt \tag{3.4}$$

- Modello di misura

$$D_k = L - x_{c,k} + n_{1k}$$
$$a1_k = atan2(-Lp \cdot cos(\theta_k)x_{c,k} - Lp \cdot sin(\theta_k)) + n_{2k}$$
$$\omega_k = \dot{x}_{c,k}/r + n_{3k}$$

La scelta di dt, ovvero del tempo di campionamento del filtro, risulta fondamentale per realizzare una stima soddisfacente. Considerando il sistema in questione, con le relative caratteristiche geometriche ed inerziali, le condizioni iniziali dello stato e i valori di ampiezza e frequenza dell'ingresso, è stato selezionato un tempo di campionamento di 1ms.

3.2 Filtraggio non lineare

Essendo sia il modello del sistema dinamico che il modello di misura, descritti da funzioni non lineari, sarà necessario stimare lo stato utilizzando due varianti del filtro di Kalman per sistemi non lineari:

- Extended Kalman Filter (EKF)
- Unscented Kalman Filter (UKF)

4 EKF

4.1 Descrizione

L'EKF utilizza l'algoritmo di Kalman in forma di correzione/predizione, basandosi su linearizzazioni del modello dinamico e di misura, attorno ad opportune configurazioni dello stato e degli ingressi. Per poterlo implementare sono necessari:

- Modello dinamico del sistema: $x_{k+1} = f_k(x_k, V_k, U_k)$
- Modello di misura : $y_k = h(x_k, n_k)$
- Ingresso deterministico V_k
- Caratteristiche statistiche stato iniziale: $x_0 \sim \ \overline{x}_0, P_0$
- Caratteristiche statistiche disturbo di processo: $U_k \sim \ \overline{U}, Q_k$
- Caratteristiche statistiche rumore di misura: $n_k \sim \ \overline{n}, R_k$

L'algoritmo si struttura nel modo seguente:

1. Inizializzazione:

$$\hat{x}_{1|0} = \overline{x}_0 \tag{4.1}$$
$$P_{1|0} = P_0 \tag{4.2}$$

2. Correzione:

$$H_k = \frac{\partial h_k}{\partial x_k}\Big|_{x_k=\hat{x}_{k|k-1}, n_k=\overline{n}} \tag{4.3}$$
$$M_k = \frac{\partial h_k}{\partial n_k}\Big|_{x_k=\hat{x}_{k|k-1}, n_k=\overline{n}} \tag{4.4}$$
$$e_k = y_k - h_k(\hat{x}_{k|k-1}, \overline{n}) \tag{4.5}$$
$$S_k = H_k P_{k|k-1} H_k^T + M_k R_k M_k^T \tag{4.6}$$
$$L_k = P_{k|k-1} H_k^T S_k^{-1} \tag{4.7}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k e_k \tag{4.8}$$
$$P_{k|k} = (I - L_k C_k) P_{k|k-1} (I - L_k C_k)^T + L_k R_k L_k^T \tag{4.9}$$

3. Predizione:

$$F_k = \frac{\partial f_k}{\partial x_k}\Big|_{x_k=\hat{x}_{k|k}, U_k=\overline{U}, V_k} \tag{4.10}$$
$$D_k = \frac{\partial f_k}{\partial U_k}\Big|_{x_k=\hat{x}_{k|k}, U_k=\overline{U}, V_k} \tag{4.11}$$
$$\hat{x}_{k+1|k} = f_k(\hat{x}_{k|k}, \overline{U}) \tag{4.12}$$
$$P_{k+1|k} = F_k P_{k|K} F_k^T + D_k Q_k D_k^T \tag{4.13}$$

Le matrici Jacobiane nelle equazioni 4.3, 4.4, 4.10 e 4.11 risultano complicate da derivare in forma discreta a mano, per questo abbiamo creato 4 file Matlab che utilizzando il calcolo simbolico derivano le equazioni 3.1, 3.2, 3.3 e 3.4. Le matrici derivate vengono salvate in appositi file di funzione che permettono di calcolare le matrici Jacobiane richiamando le funzioni e passando come argomento lo stato e gli altri parametri necessari.

4.2 Implementazione

L'Extended Kalman Filter è stato implementato su simulink nel subsystem EKF Filter, all'interno del quale è presente la matlab function EKF. Essa si avvale di vari unit delay che consentono di salvare i valori in output e riutilizzarli nell'iterazione successiva, questo aspetto è stato fondamentale per consentire la corretta realizzazione dell'algoritmo di filtraggio. La funzione riceve in ingresso:

- $\hat{x}_{k|k-1}$ stato predetto al passo precedente

- $P_{k-1|k-1}$ matrice di covarianza al passo precedente
- V_k ingresso nominale (senza rumore)
- y_k vettore di misure
- Ts vettore contenente i marker temporali delle misure
- Ts_{old} marker temporali dell'iterazione precedente
- f_{old} flag per le misure ripetute dell'iterazione precedente

In uscita abbiamo:

- $\hat{x}_{k+1|k}$
- $P_{k+1|k}$
- $\hat{x}_{k|k}$
- f flag per le misure ripetute

4.3 Filtro

L'algoritmo all'interno della funzione esegue i seguenti passaggi:

1. Conversione dell'angolo a1 nell'intervallo $[-\pi, \pi]$
2. Gestione degli outliers e delle misure ripetute
3. Correzione
4. Predizione

4.4 Outliers e misure ripetute

Le condizioni per cui la correzione relativa ad una determinata misura venga effettuata sono due:

- $D_m < D_{\text{trashold}}$
  Con $D_m = \sqrt{e_k^T S_k^{-1} e_k}$ distanza di Mahalanobis, che da informazione sulla coerenza dell'innovazione con la covarianza dell'innovazione S_k, ovvero dice se le nuove misure sono attendibili o meno.
  $D\text{trashold} = 5$
- Misura proveniente da sensore aggiornata.
  Considerando le misure da sensori come un vettore y, l'iesima misura y_i risulterà aggiornata se la flag corrispondente f_i sarà 1. La flag f_i sarà uguale ad 1 se il marker temporale relativo alla misura y_i sarà maggiore rispetto al marker della stessa misura all'iterazione precedente dell'EKF.

Nel caso in cui entrambe le condizioni fossero soddisfatte, le componenti della matrice di correzione relative all'innovazione $e(i)$, ovvero $[L_{1\text{i}} \quad L_{2\text{i}} \quad L_{3\text{i}}]^T$, saranno calcolate in accordo con le formule nell'equazione 4.7. Altrimenti sarà un vettore di zeri e quindi la correzione non sarà effettuata.

4.5 Regolarizzazione

Gli algoritmi di regolarizzazione stimano lo stato una volta che sono disponibili tutte le misure, vengono impiegati offline a fine simulazione. Nell'elaborato è stato implementato su Matlab l'algoritmo di Rauch Tung Striebel, che si divide in due fasi:

1. In avanti:
   Uso l'EKF per calcolare $\hat{x}_{k|k}$, $\hat{x}_{k+1|k}$, $P_{k|k}$, $P_{k+1|k}$, F_k
2. All'indietro:

$$C_k = P_{k|k} F_{k+1}^T P_{k+1|k}^{-1} \tag{4.14}$$

$$\hat{x}_{k|n} = \hat{x}_{k|k} + C_k(\hat{x}_{k+1|n} - \hat{x}_{k+1|k}) \tag{4.15}$$

$$P_{k|n} = P_{k|k} + C_k(P_{k+1|n} - P_{k+1|k})C_k^T \tag{4.16}$$

   Con $\hat{x}_{k|n}$ lo stato stimato all'istante k date tutte le misure fino ad n (istante di fine simulazione)

4.6 Risultati

Di seguito è riportato il risultato di una simulazione di 20s con condizioni iniziali $X_{c,0} = 2m$, $\theta_0 = \pi/6$, $\dot{X}_{c,0} = 0$ e $\dot{\theta} = 0$, con ingresso sinusoidale di ampiezza 5V e frequenza 1 Hz.

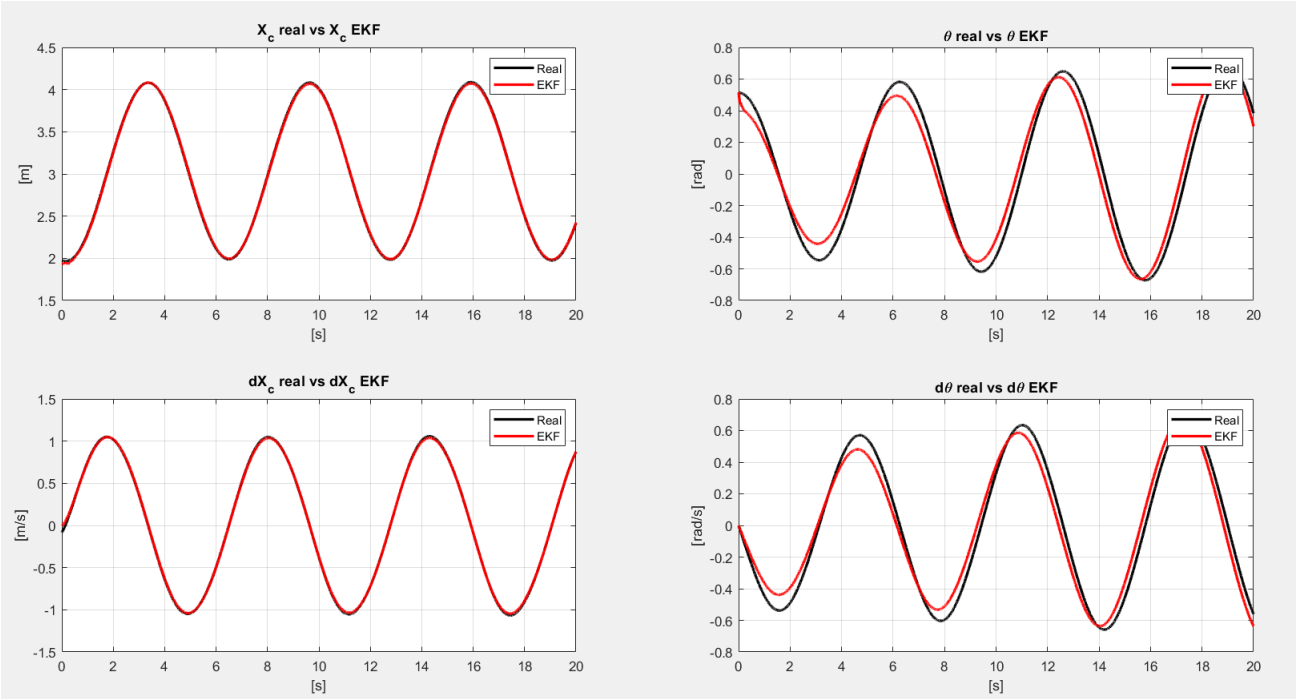

Figura 4: Stato simulato vs stato stimato EKF

Come si vede da figura 4 il filtro riesce a stimare bene xc e $\dot{x}_c$, mentre le stime di θ e $\dot{\theta}$ sono meno accurate. in figura 5 possiamo osservare gli errori di stima.

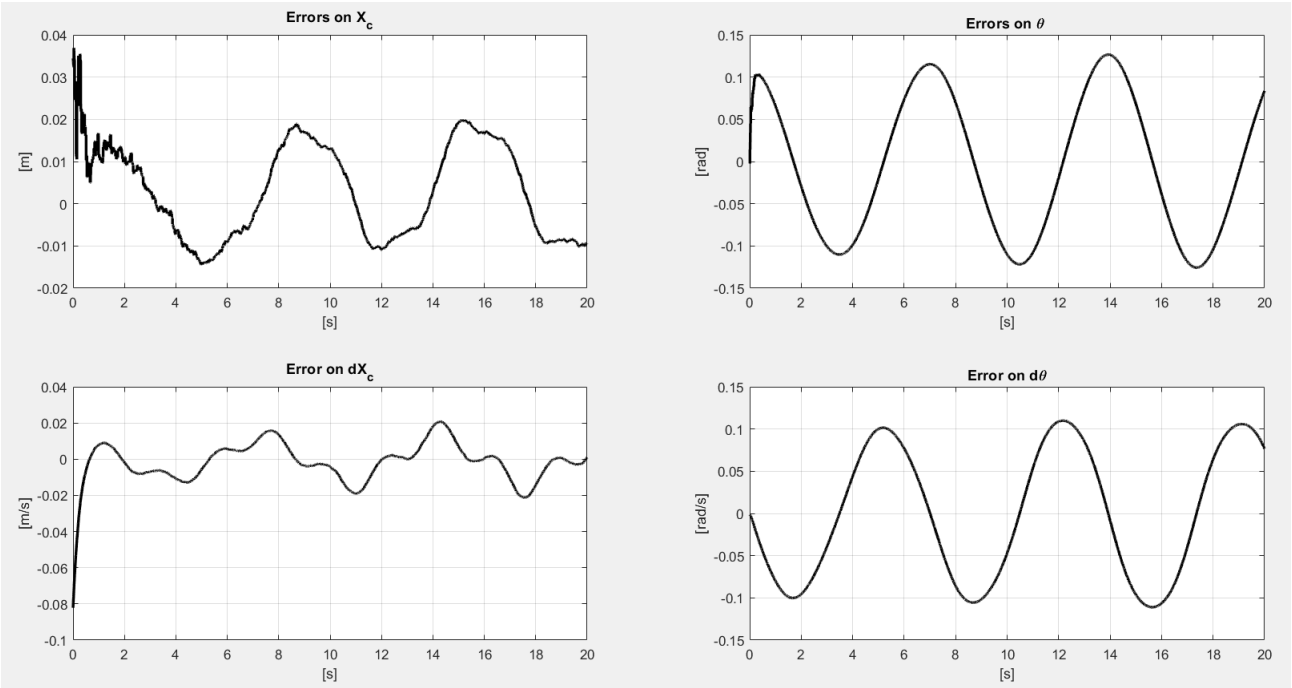

Figura 5: Errore di stima EKF

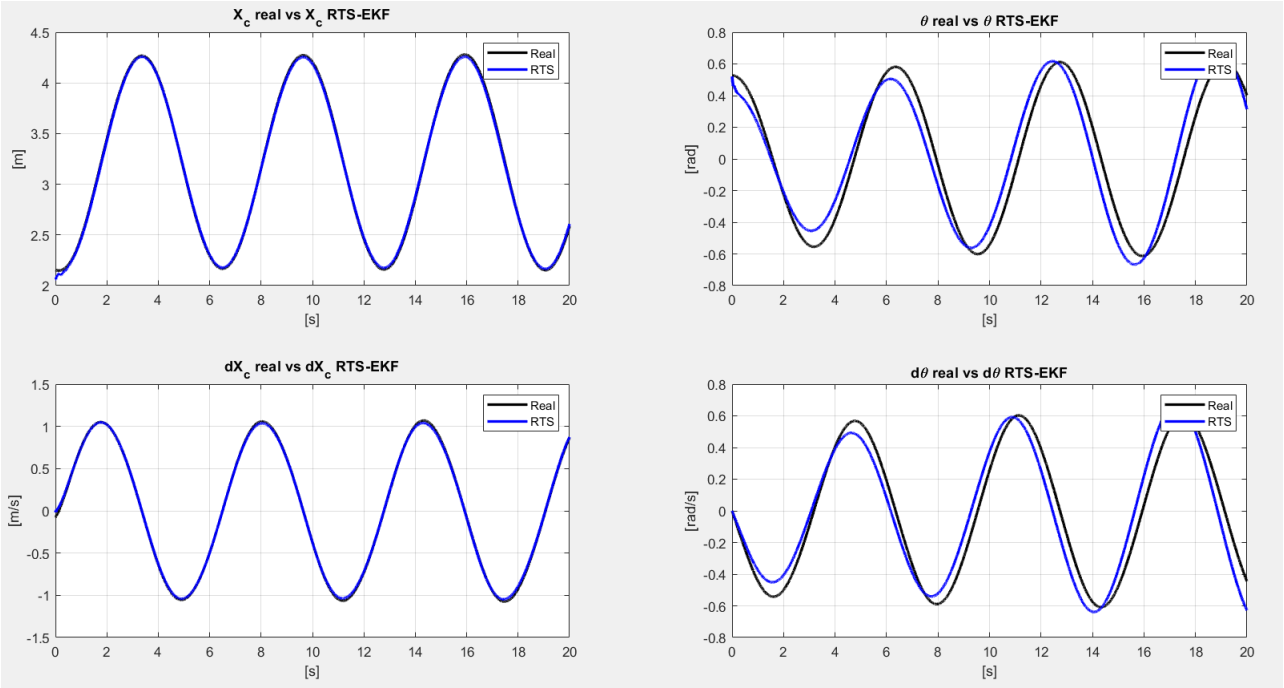

Figura 6: Stato simulato vs stato regolarizzato

5 UKF

5.1 Descrizione

L'UKF utilizza l'algoritmo di Kalman in forma di correzione/predizione, basandosi sulla trasformata Unscented.

5.1.1 Trasformata Unscented

Date:

- Variabile aleatoria $x \sim \ \overline{x}, \Sigma_x$
- Funzione $y = g(x)$

La trasformata Unscented permette di ottenere i momenti $\overline{y}$, Σ_y, Σ_{xy}, fornendo in ingresso $\overline{x}$, Σ_x, $g(\cdot)$: $\left[\overline{y}, \Sigma_y, \Sigma_{xy}\right] = \mathrm{UT}\left[\overline{x}, \Sigma_x, g(\cdot)\right]$. I passaggi sono i seguenti:

1. Scelta dei parametri α, β, k: Sono stati fissati $\alpha = 1$, $\beta = 2$, $k = 0$.
2. Generazione dei sigma points: vengono generati $2n + 1$ sigma points, con n dimensione della variabile aleatoria x.

$$\lambda = \alpha^2(n + k) - n \tag{5.1}$$

$$x_{\pm i} = \begin{cases} \overline{x} \quad \text{se} \quad i = 0 \\ \overline{x} \pm \sqrt{n + \lambda} \cdot \gamma_i \quad \text{se} \quad i = 1,...,\text{n} \end{cases} \tag{5.2}$$

   Con γ_i colonna i-esima della matrice Γ : $\quad \Sigma_x = \Gamma\Gamma^T$

3. Calcolo dei pesi:

$$W_{\pm i} = \begin{cases} \dfrac{\lambda}{n + \lambda} \quad \text{se} \quad i = 0 \\ \dfrac{1}{2(n + \lambda)} \quad \text{se} \quad i = 1,...,\text{n} \end{cases} \tag{5.3}$$

$$W_{\pm i}^{'} = \begin{cases} W_0 + 1 + \alpha^2 + \beta \quad \text{se} \quad i = 0 \\ \dfrac{1}{2(n + \lambda)} \quad \text{se} \quad i = 1,...,\text{n} \end{cases} \tag{5.4}$$

4. Propagazione dei sigma points:

$$y_i = g(x_i) \quad \forall i = \text{-n,...,n} \tag{5.5}$$

5. Calcolo dei momenti di y come stime campionarie:

$$\overline{y} = \sum_{i=-n}^{n} W_i y_i \tag{5.6}$$

$$\Sigma_y = \sum_{i=-n}^{n} W_i^{'}(y_i - \overline{y})(y_i - \overline{y})^T \tag{5.7}$$

$$\Sigma_{xy} = \sum_{i=-n}^{n} W_i^{'}(x_i - \overline{x})(y_i - \overline{y})^T \tag{5.8}$$

5.1.2 Algoritmo UKF

Per poterlo implementare sono necessari:

- Modello dinamico del sistema: $x_{\text{k+1}} = f_k(x_k, V_k, U_k)$
- Modello di misura : $y_k = h(x_k, n_k) = h^*(x_k) + n_k$
- Ingresso deterministico V_k
- Caratteristiche statistiche stato iniziale: $x_0 \sim \ \overline{x}_0, P_0$
- Caratteristiche statistiche disturbo di processo: $U_k \sim \ \overline{U}, Q_k$
- Caratteristiche statistiche rumore di misura: $n_k \sim \ \overline{n}, R_k$

L'algoritmo si struttura nel modo seguente:

1. Inizializzazione:

$$\hat{x}_{1|0} = \overline{x}_0 \tag{5.9}$$

$$P_{1|0} = P_0 \tag{5.10}$$

2. Correzione:

$$\left[\hat{y}_{k|k-1}, \Sigma_{y^*,k}, \Sigma_{xy,k}\right] = \text{UT}\left[\hat{x}_{k|k-1}, P_{k|k-1}, h^*(\cdot)\right] \tag{5.11}$$

$$e_k = y_k - \hat{y}_{k|k-1} \tag{5.12}$$

$$S_k = \Sigma_{y^*,k} + R_k \tag{5.13}$$

$$L_k = \Sigma_{xy,k} S_k^{\text{-1}} \tag{5.14}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k e_k \tag{5.15}$$

$$P_{k|k} = P_{k|k-1} - L_k S_k L_k^T \tag{5.16}$$

3. Predizione:

$$\left[\hat{x}_{k+1|k}, P_{k+1|k}\right] = \text{UT}\left[[\hat{x}_{k|k-1} \quad \bar{U}]^T, \begin{bmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{bmatrix}, f(\cdot,\cdot)\right] \tag{5.17}$$

5.2 Implementazione

Anche il filtro UKF come il filtro EKF è realizzato in Simulink nello stesso file in cui viene simulato il sistema dinamico reale tempo continuo. Per implementarlo abbiamo creato 2 file funzione Matlab che vengono richiamati dal blocco UKF.

Il filtro è composto da un blocco contenente l'algoritmo che riceve in ingresso

- $\hat{x}_{k|k-1}$ stato predetto al passo precedente
- $P_{k|k-1}$ matrice di covarianza al passo precedente

- V_k ingresso nominale (senza rumore)
- y_k vettore di misure con relativi marker temporali

In uscita dal blocco invece abbiamo:

- $\hat{x}_{k+1|k}$
- $P_{k+1|k}$
- $\hat{x}_{k|k}$
- Flag per le misure ripetute

Le prime due quantità vengono memorizzate e utilizzate come ingresso nel filtro al passo successivo.

5.3 Filtro

L'algoritmo all'interno del filtro esegue i seguenti passaggi

1. Controllo se i dati dei sensori sono aggiornati
2. Conversione dell'angolo a1 nell'intervallo $[-\pi, \pi]$
3. Correzione tramite $UT[\hat{x}_{k|k-1}, P_{k|k-1}, h(\cdot)] = [\hat{y}, \Sigma_y, \Sigma_{xy}]$
4. Calcolo dell'innovazione e aggiornamento della covarianza
5. Calcolo della distanza di Mahalanobis
6. Calcolo di $\hat{x}_{k|k}, P_{k|k}$
7. Predizione tramite $UT[\hat{x}_{k|k}, P_{k|k}, V, Q, f(\cdot)] = [\hat{x}_{k+1|k}, P_{k+1|k}]$

Per quanto riguarda la gestione di outliers e misure ripetute vale quello che è riportato nella sezione 4.4.

5.4 Implementazione delle trasformate

Le trasformate UT delle funzioni $h(\cdot), f(\cdot)$ sono state implementate in due file Matlab che vengono richiamati dal file Simulink. La funzione $h(\cdot)$ è una versione vettoriale delle equazioni 2.1, 2.2,2.3 impilate nell'ordine in cui compaiono, la funzione risultante risulta essere del tipo

$$\vec{y}(x,k) = \vec{h}(x,k) + v(\vec{k}) \tag{5.18}$$

ovvero con rumore additivo, è quindi possibile effettuare la trasformata considerando $z = h(x)$ e la trasformata

$$[\bar{z}, \Sigma_z, \Sigma_{xz}] = UT(\bar{x}, \Sigma_x, h(\cdot)) \tag{5.19}$$

in modo da avere un vettore di *sigma points* di 3 componenti (corrispondenti alle 3 uscite del sistema) piuttosto che 6. Per la funzione $f(\cdot)$ descritta nelle equazioni 3.1, 3.2, 3.3 e 3.4 invece bisogna utilizzare la trasformata nella versione completa in quanto la funzione è del tipo $f(x, v)$, non lineare sia nello stato che nel rumore di misura. I *sigma points* da utlizzare nella trasformata sono di dimensione $n = 5$.

5.5 Regolarizzazione

Anche in questo caso l'algoritmo è stato implementato in fase di post-processing su matlab. Si è optato in questo caso per una variante dell'algoritmo di Rauch Tung Striebel strutturato come segue:

1. In avanti:
   Uso l'UKF per calcolare $\hat{x}_{k|k}$, $\hat{x}_{k+1|k}$, $P_{k|k}$, $P_{k+1|k}$ Nonostante non sia necessaria per l'UKF in se, calcolo F_k in maniera analoga all'EKF ma con i valori di stato stimati nell'UKF.
2. All'indietro:
   Come nella sottosezione 4.5

5.6 Risultati

Di seguito è riportato il risultato di una simulazione di 20s con condizioni iniziali $X_{c,0} = 2m$, $\theta_0 = \pi/6$, $\dot{X}_{c,0} = 0$ e $\dot{\theta} = 0$, con ingresso sinusoidale di ampiezza 5V e frequenza 1 Hz.

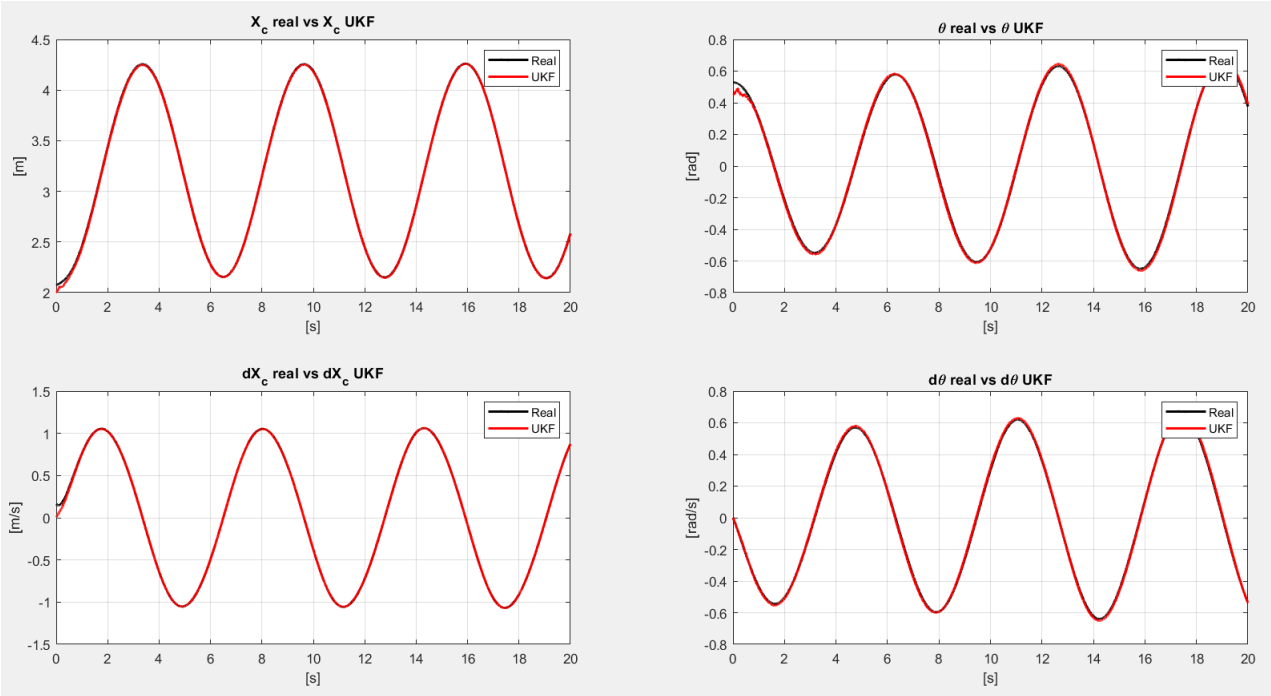

Figura 7: Stato simulato vs stato stimato UKF

Come si vede da figura 7 l'andamento dello stato è riprodotto in maniera fedele dal filtro, in figura 8 possiamo riuscire a quantificare gli errori del filtro.

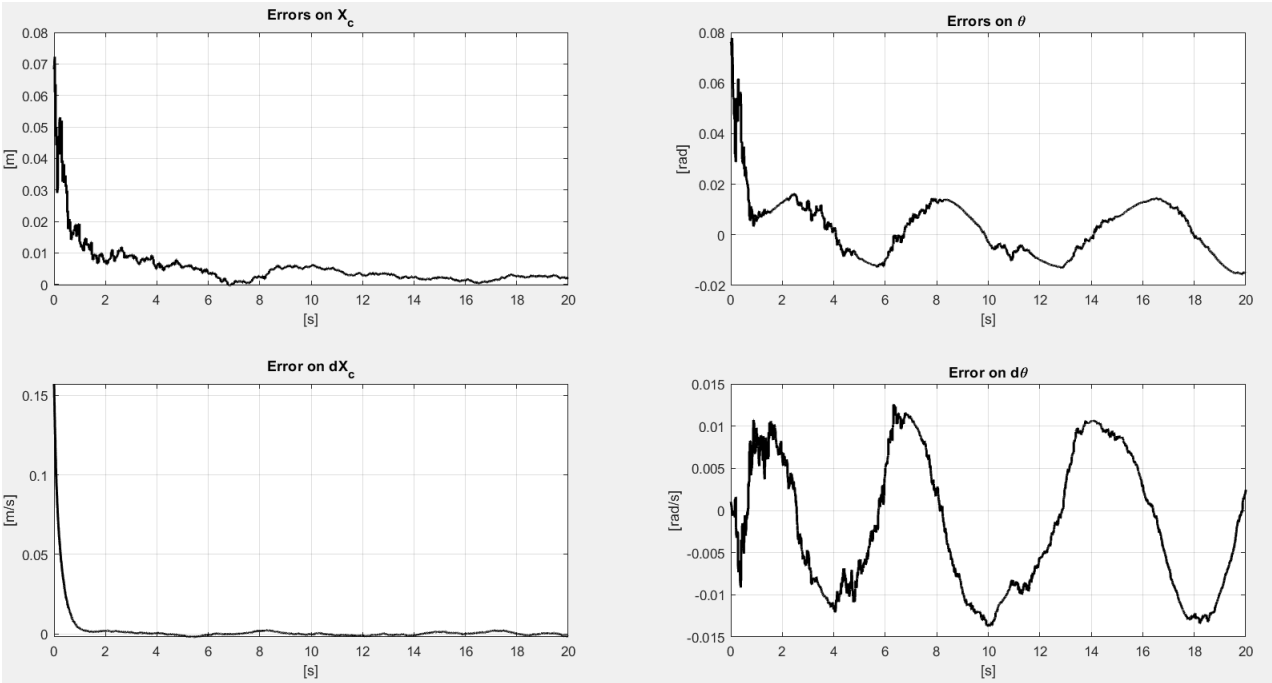

Figura 8: Errore di stima UKF

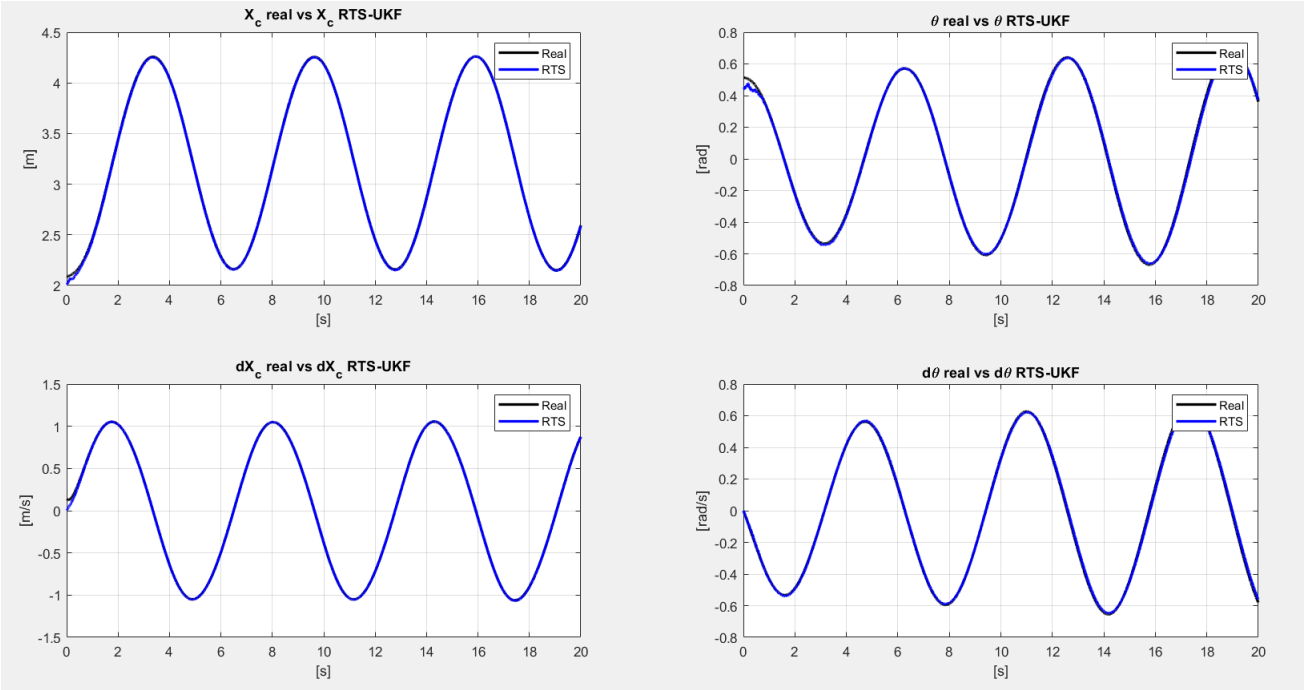

Figura 9: Stato simulato vs stato regolarizzato

6 Conclusioni

In generale si nota che per un sistema molto non lineare come questo il filtro UKF commette errori minori rispetto al filtro EKF. Tramite il codice Matlab è possibile verificare la robustezza degli algoritmi al variare dei parametri e delle incertezze su di esso. Si nota facilmente che l'incertezza sull'angolo e sulla velocità angolare sono difficili da gestire per entrambi i filtri.