**Filtro di Kalman non lineare**

Dato che le equazioni dinamiche del sistema ed il modello di osservazione sono non lineari, facciamo appello al **filtro di Kalman Esteso** ed al **UKF** (Unscented Kalman Filter). Per quanto riguarda il modello di osservazione (la funzione h) facciamo riferimento alle equazioni:

Per quanto riguarda invece le equazioni dinamiche (la funzione f), la struttura è quella della dinamica a tempo discreto, ove la derivata dello stato è calcolata a partire alle equazioni meccaniche del sistema, in termini di accelerazione ed accelerazione angolare:

Da cui ricaviamo le equazioni dinamiche tempo discreto del sistema:

dove è definito come:

**Scelta prove simulative**

Le prove che abbiamo scelto per testare i filtri che andremo a vedere in seguito sono le seguenti:

* **Prova 1**: condizioni iniziali di e una
* **Prova 2**: condizioni iniziali nulle e una (=1500N)
* **Prova 3**: condizioni iniziali nulle e una (A = 1500N, )

In tutte le prove abbiamo considerato che i filtri di Kalman facciano una stima della condizione iniziale della funivia, secondo una densità di probabilità Gaussiana con valor medio la condizione iniziale vera:

**Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente**

Oltretutto i filtri hanno incertezze sulla massa, inerzia e coeff. di attrito:



Immagine che contiene testo, diagramma, Piano, linea

Descrizione generata automaticamente**Filtro di Kalman Esteso (EKF)**

* Come possiamo vedere dalla *Figura 3.1,* il filtro EKF prevede in ingresso le misure dei sensori (con disturbo additivo di tipo Gaussiano) e l’ingresso F anch’esso disturbato.
* Abbiamo aggiunto 2 integratori di tipo discreto così che da poter avere la stima dello stato e la sua varianza all’istante precedente (cioè e ).
* Il tempo di campionamento del filtro è

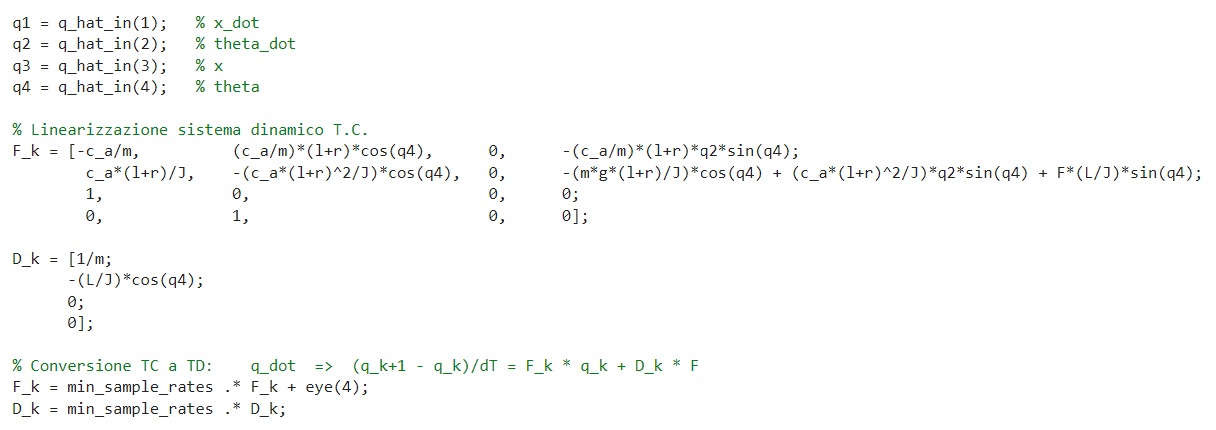
dato dal tempo di campionamento più grande tra i sensori scelti (in questo caso abbiamo che il tempo ).

**Figura 3.1** – Realizzazione EKF tramite Simulink

Infine, l’uscita del filtro è semplicemente la stima dello stato del sistema, cioè , e dove è stato preso come:

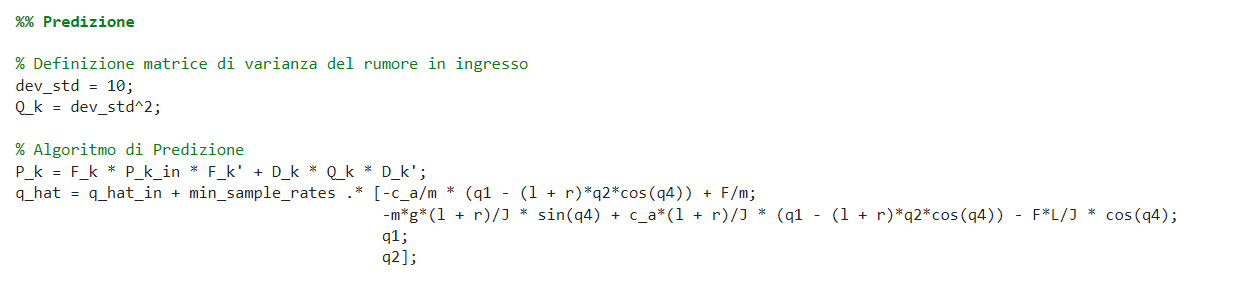
**Predizione EKF**

Ora andiamo a vedere la linearizzazione della dinamica del sistema su cui si basa il filtro EKF; di seguito è riportata la funzione MatLab interna al blocco Simulink della figura *Figura 3.1*:



**Figura 3.2** – Codice per la Linearizzazione della dinamica del sistema (calcolo matrici e )

Da cui segue la parte di predizione del filtro:



**Figura 3.3** – Codice per l’algoritmo di predizione del filtro EKF

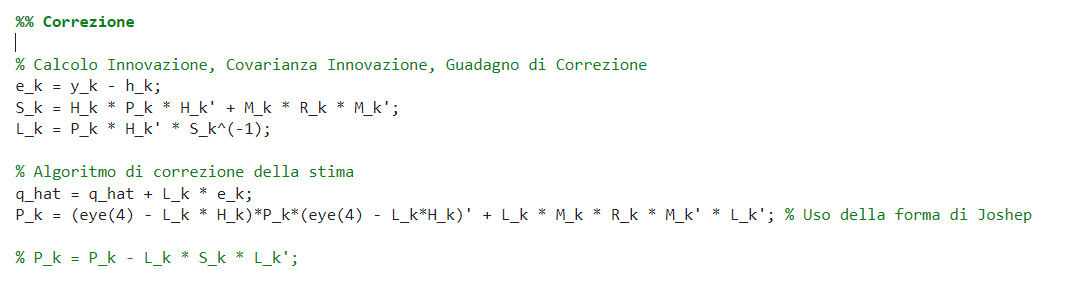
**Correzione EKF**

Ora andiamo a vedere la parte di codice di correzione del filtro, che si trova nello stesso blocco di *Figura 3.1*, dove ci aspettiamo una linearizzazione della dinamica di uscita (sensori) del sistema: Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Figura 3.4** – Codice per la Linearizzazione della dinamica di uscita del sistema (calcolo matrici e )

E analogamente alla predizione, vediamo ora la parte di correzione:



**Figura 3.5** – Codice per l’algoritmo di correzione del filtro EKF

**Osservazioni e Conclusioni EKF**

Durante lo sviluppo del filtro EKF su MatLab, ci siamo accorti che la stima dello stato della funivia funzionava correttamente a parte delle piccole oscillazioni per la stima dello stato ; infatti, mettendo come ingresso una forza costante di e con condizioni iniziali tutte nulle, troviamo i seguenti dati in di simulazione (Prova 2):

**Figura 3.6.1** – Differenza tra stima EKF e lo stato vero del sistema (Prova 2)

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Il comportamento della stima di può essere spiegato dal fatto che il rumore dei sensori diventa rilevante quando abbiamo piccole oscillazioni attorno al punto di equilibrio del sistema, da cui ne deriva un errore maggiore nei picchi dell’oscillazioni (dove ha piccole variazioni confrontabili con i rumori dei sensori).

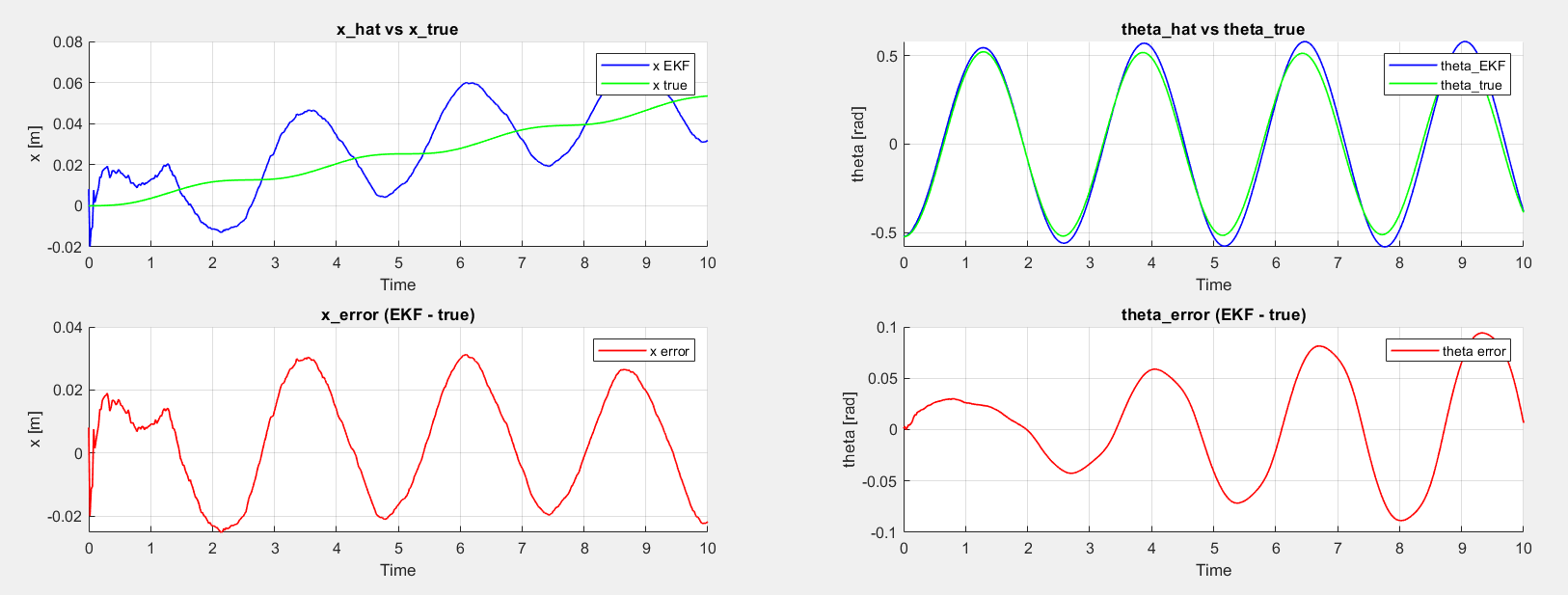
Inoltre, per verificare se c’è accordo tra il modello utilizzato per il filtro e il sistema vero, abbiamo controllato se le innovazioni delle 3 misure dei sensori fossero dei rumori bianchi:

**Figura 3.7** – Innovazione delle misure dei sensori EKF

Immagine che contiene testo, schermata, linea, Diagramma

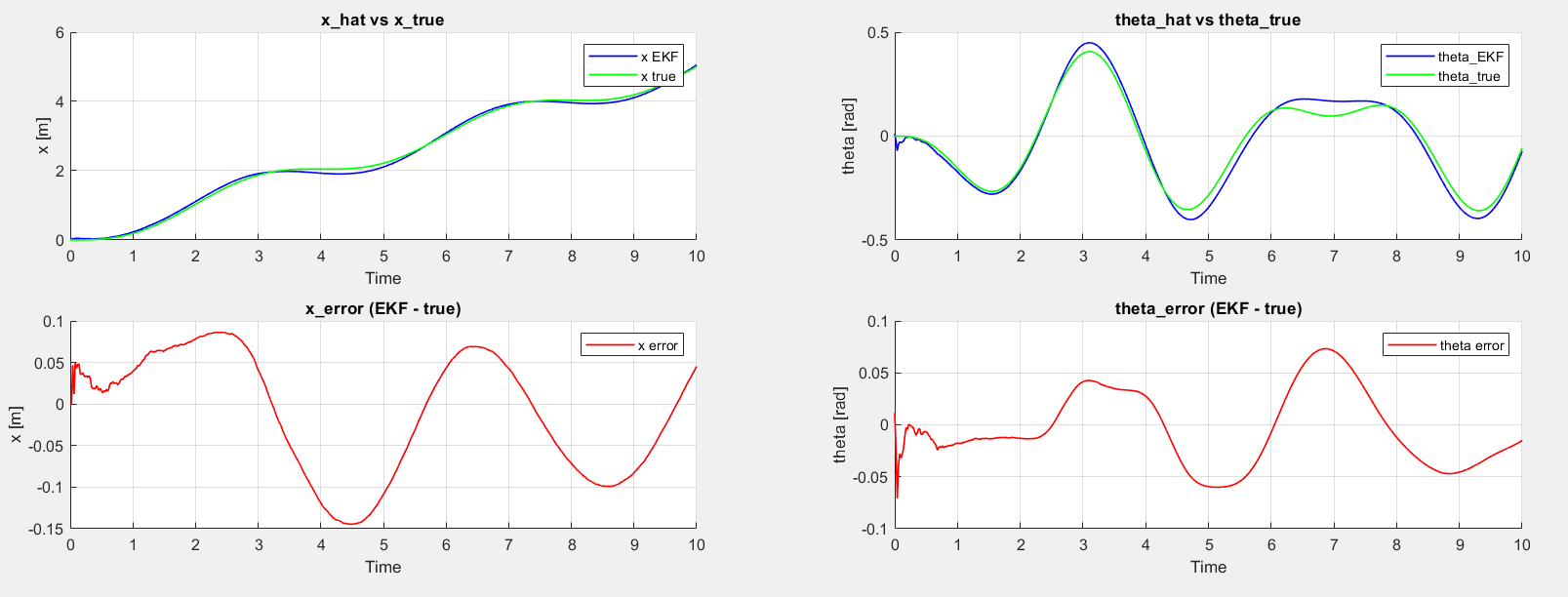
Descrizione generata automaticamente

E come possiamo vedere da Figura 3.7, la condizione di rumore bianco per tutte e tre le misure è soddisfatta.



**Figura 3.6.2** – Differenza tra stima EKF e lo stato vero del sistema (Prova 1)

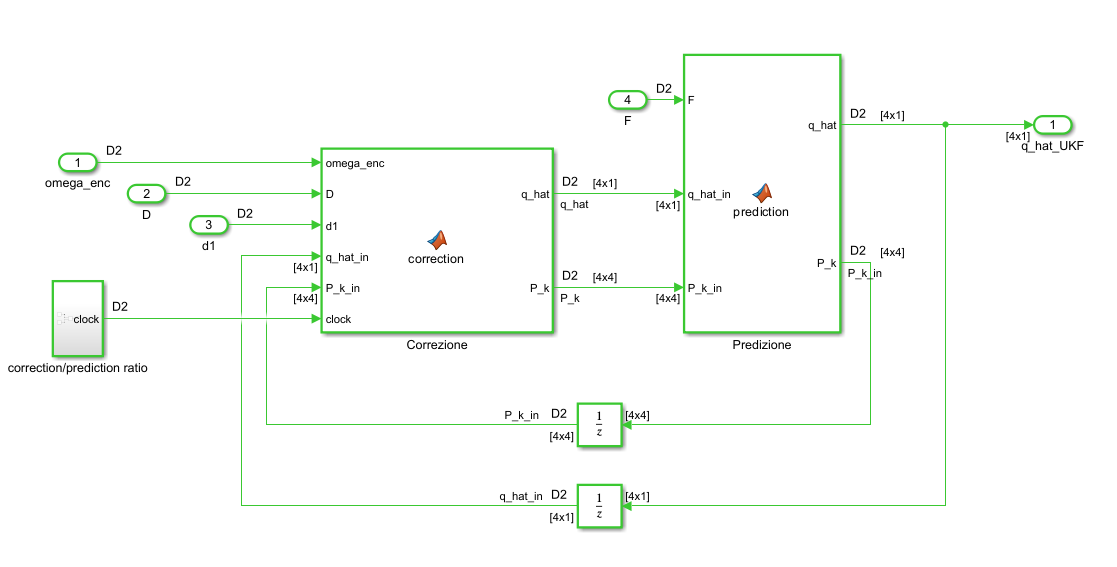
Il comportamento della stima di nella *Figura 3.6.2* può essere spiegato in maniera analoga per la Prova 2 con la stima di : in questo caso le piccole oscillazioni sono per lo stato , e quindi in questo caso i ruoli sono opposti ( ha piccole variazioni confrontabili con i rumori dei sensori).



**Figura 3.6.3** – Differenza tra stima EKF e lo stato vero del sistema (Prova 3)

Per la Prova 3 invece non c’è molto da dire, dato che entrambi gli stati hanno valori non comparabili con gli errori di misura.

**Filtro di Kalman Unscented (UKF)**

****

**Figura 3.8** – Realizzazione UKF tramite Simulink

* Come possiamo vedere dalla *Figura 3.8,* il filtro EKF prevede in ingresso le misure dei sensori (con disturbo additivo di tipo Gaussiano) e l’ingresso F anch’esso disturbato.
* Abbiamo aggiunto 2 integratori di tipo discreto così che da poter avere la stima dello stato e la sua varianza all’istante precedente (cioè e ).
* Il tempo di campionamento del filtro è dato dal tempo di campionamento più grande tra i sensori scelti (in questo caso abbiamo che il tempo ).

Prima di andare a vedere la predizione e correzione per il filtro, vediamo la parte iniziale del codice interno al blocco di figura *Figura 3.8*, dove definiamo le costanti e i parametri dell’UKF:

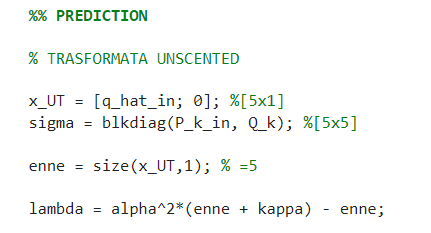
Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Figura 3.9** – Definizione costanti fisiche e parametri del filtro , e

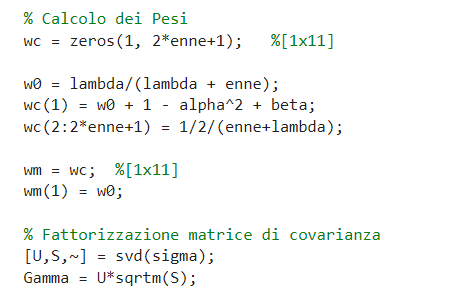
**Predizione UKF**

Partiamo con il vedere la parte di codice in cui applichiamo la trasformata unscented per la predizione dello stato, e quindi l’uso della trasformata per la funzione , cioè la dinamica del sistema:

****

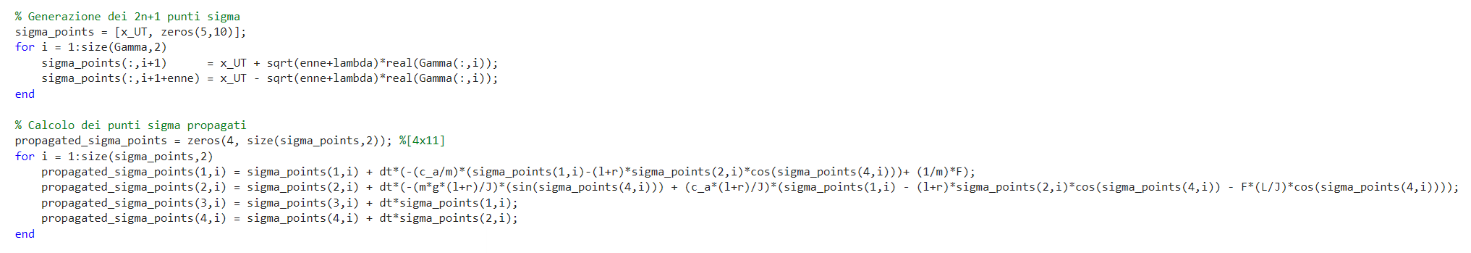
**Figura 3.10** – Definizione parametri UT (variabili di stato + rumore ) e costanti e

Il passaggio successivo è il calcolo dei pesi e la fattorizzazione della matrice di covarianza:



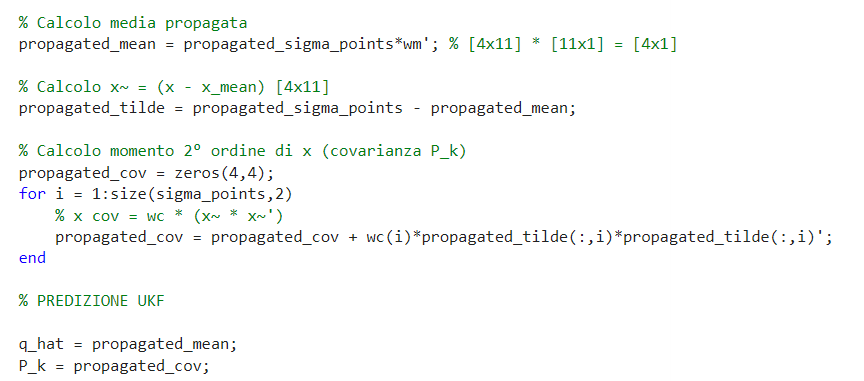
**Figura 3.10** – Calcolo pesi , e fattorizzazione SVD per ricavare la matrice

Da cui segue il calcolo dei punti sigma e l’utilizzo della per il calcolo dei punti sigma propagati (alla dinamica):



**Figura 3.11** – Calcolo punti sigma utilizzati per trovare quelli propagati nella funzione

Infine, ci basta calcolare i momenti del 1° e 2° ordine (media e covarianza dello stato, la cross-covarianza non ci interessa) attraverso i punti sigma propagati:



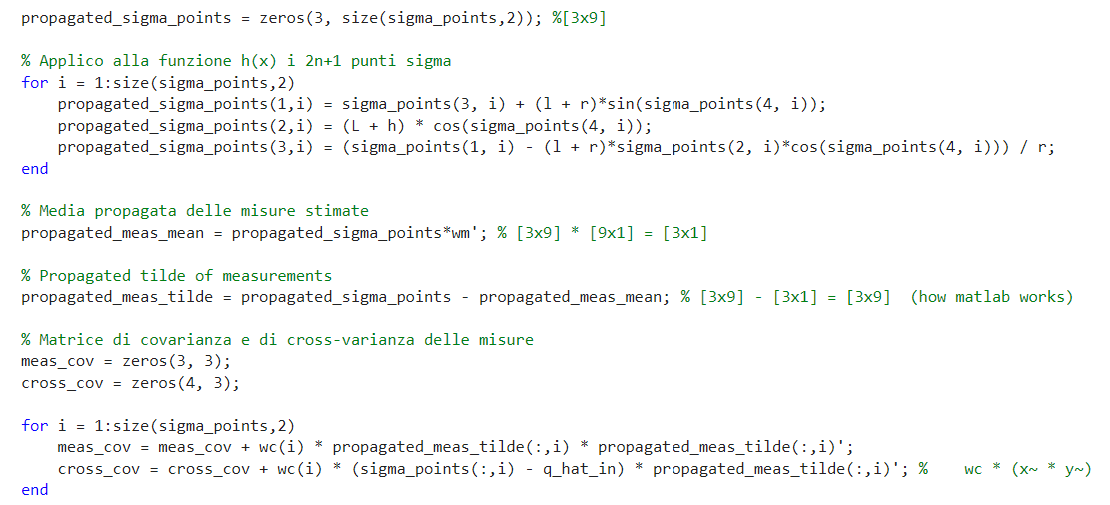
**Figura 3.12** – Predizione dello stato con relativo calcolo della media propagata e covarianza propagata

**Correzione UKF**

Ora vediamo la parte di correzione per l’UKF, che è analoga alla predizione solo con alcune differenze in alcune parti dell’algoritmo UT:

* In ingresso all’algoritmo UT non abbiamo e , ma in questo caso direttamente lo stato predetto e la sua varianza (quindi senza la parte dei rumori)
* Quindi la variabile in questo caso (non 5 come prima per la predizione)
* I calcoli dei pesi e i punti sigma rimangono invariati, tenendo conto delle considerazioni dei punti precedenti

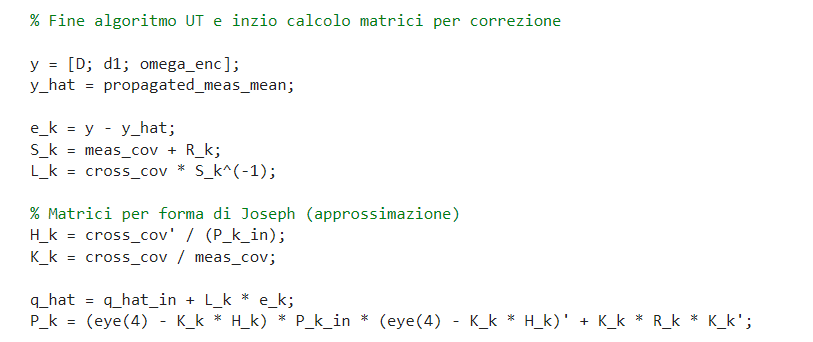
Vediamo dunque la parte dei punti propagati utilizzando la funzione , cioè la funzione del modello di osservazione non lineare (senza rumore):



**Figura 3.13** – Calcolo dei momenti delle misure in uscita tramite i punti sigma propagati

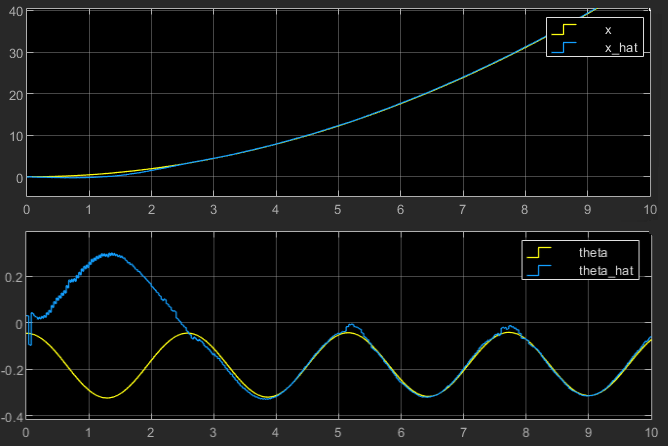
Ora per finire la parte di correzione, bisogna calcolare l’innovazione (con relativa matrice di covarianza) e la matrice di guadagno di correzione:

**Figura 3.14** – Correzione della stima e della relativa matrice di covarianza con un’approssimazione della forma di Joshep



**Osservazioni e Conclusioni UKF**

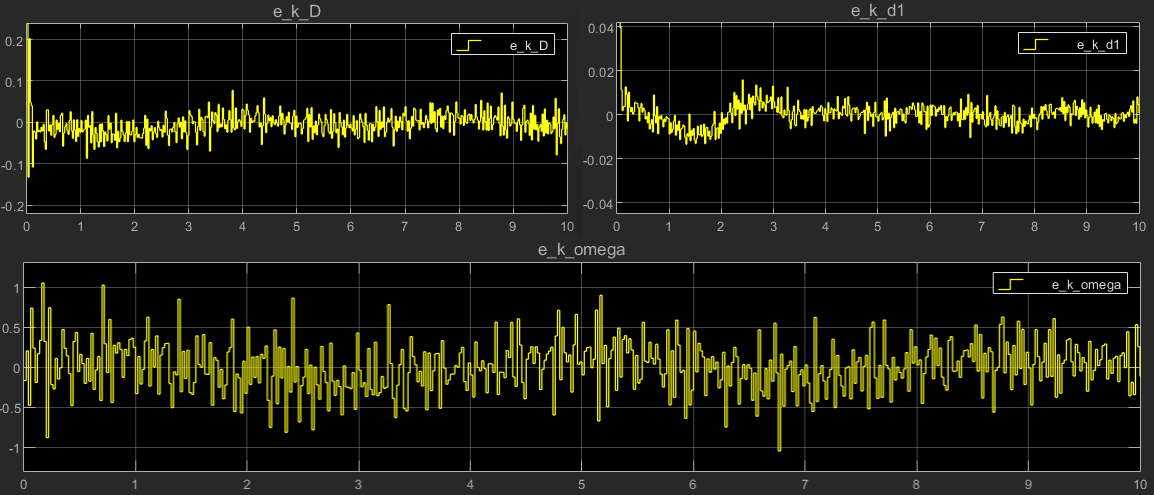
Durante lo sviluppo del filtro UKF su MatLab, ci siamo accorti che la stima per lo stato funziona correttamente, invece per lo stato , in certe condizioni, ha problemi di stima per alcune simulazione; infatti, mettendo come ingresso una forza costante di e con condizioni iniziali tutte nulle, troviamo i seguenti dati in di simulazione (Prova 2):



**Figura 3.15.1** – Differenza tra stima UKF e lo stato vero del sistema (Prova 2)

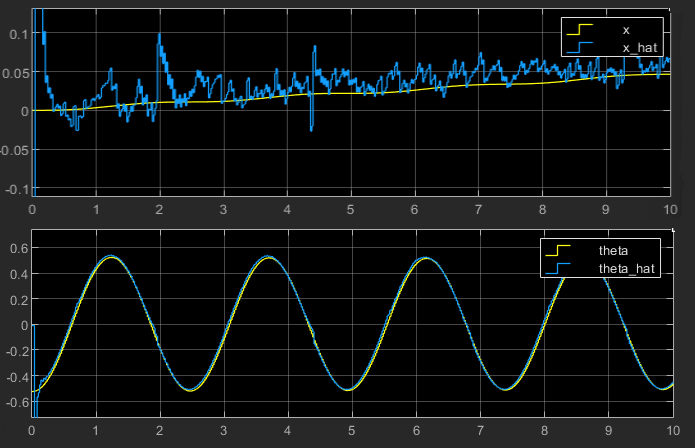
Questo problema può essere dato dal fatto che, durante la fase di correzione, la matrice di covarianza dell’errore di stima viene calcolata con un’approssimazione della vera formula di correzione:

Non abbiamo usato questa formula perché durante le varie simulazioni abbiamo riscontrato problemi di matrice di covarianza che diventavano definite negative e quindi con stime estremamente irregolari (perciò abbiamo cercato una forma tipo quella di Joshep, come nel caso EKF).

Infine, abbiamo valutato la bianchezza del rumore per le misure, per vedere se c’è accordo tra il sistema vero e il modello usato, tramite l’innovazione delle misure:

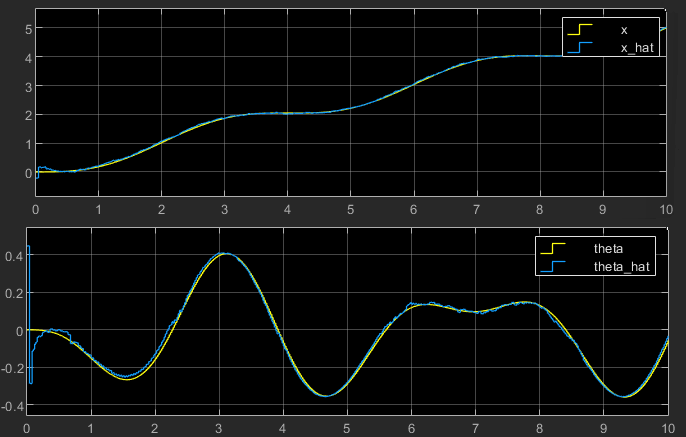
**Figura 3.16** – Innovazione delle misure dei sensori UKF

E come possiamo vedere da *Figura 3.16*, le innovazioni delle misure si comportano come dei rumori bianchi.



**Figura 3.15.2** – Differenza tra stima UKF e lo stato vero del sistema (Prova 1)

Con la Prova 1 invece, abbiamo una stima molto rumorosa per lo stato , che ci torna dato che per quello stato ho piccole variazioni e quindi i rumori di misura sono rilevanti; di contro, la stima per è invece una stima che segue quasi perfettamente la traiettoria vera (vedi *Figura 3.15.2*).



**Figura 3.15.3** – Differenza tra stima UKF e lo stato vero del sistema (Prova 3)

Per la Prova 3, abbiamo analogamente per l’EKF, che non si sono particolari errori nelle stime, come ci aspettavamo date le grandezze degli stati rispetto ai rumori di misura (vedi *Figura 3.15.3*).

**EKF con Smoother**

In questa parte andremo a vedere una variante del filtro EKF con stima regolarizzata a posteriori della simulazione (dato che la stima regolarizzata non può essere fatta in real time, ma solo a posteriori dell’esperimento). Il codice usato per la regolarizzazione della stima è il seguente:

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

**Figura 3.17** – Codice per la Regolarizzazione della stima

Come risultato ci aspettiamo di trovare una stima migliore rispetto a quella dell’EKF visto in precedenza, dato che facciamo una stima dello stato con tutte le misure dell’uscita della simulazione (quindi a parità di simulazione, ho maggiori informazioni).  
Infatti, il risultato ottenuto per la stima di e , comparato con il precedente EKF, è il seguente:

Immagine che contiene linea, diagramma, Diagramma, schermata

Descrizione generata automaticamente

**Figura 3.18** – Confronto EKF e EKF con Smoother nel caso di Prova 3 (10 secondi)

Come anticipato, la stima con lo smoother si avvicina maggiormente ai valori veri degli stati del sistema, anche se non subito evidente (vedi *Figura 3.18*).