



UNIVERSITÀ DI PISA

Progetto di Identificazione dei Sistemi Incerti: Stima dinamica dello stato di un veicolo a guida autonoma (AGV) con EKF e UKF



Francesco Bechelli

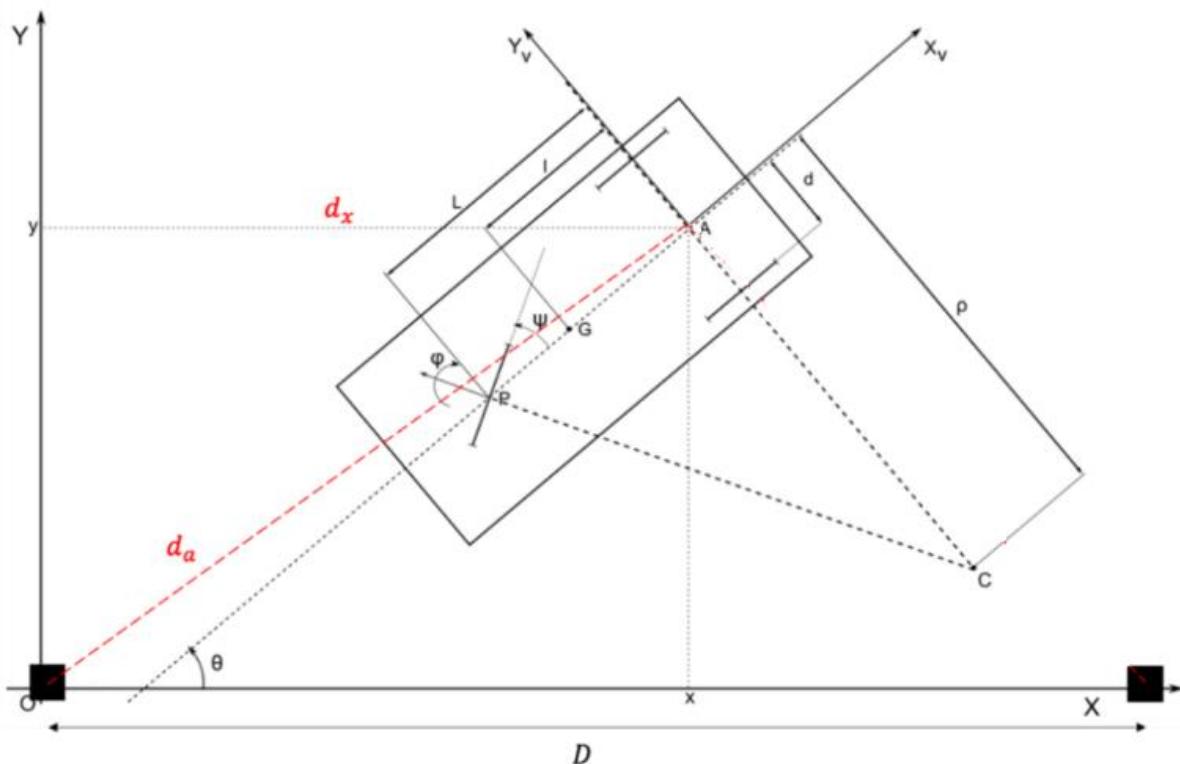
Cesare Palamara

Alberto Regoli

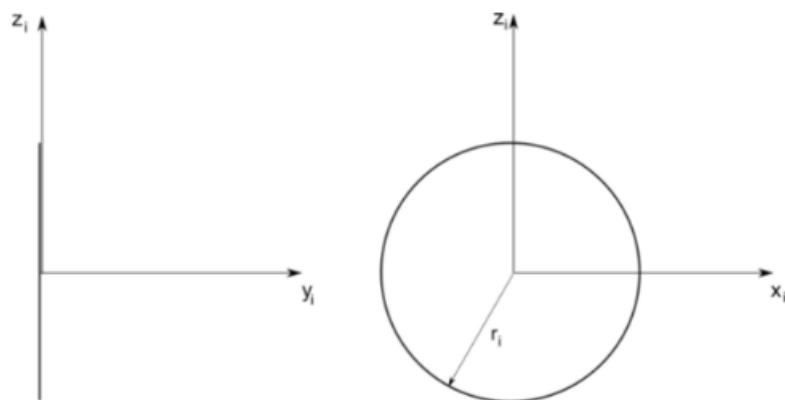
Anno Accademico 2022/2023

Università di Pisa, Ingegneria Robotica e dell' Automazione

Introduzione e descrizione modello



Il sistema in figura rappresenta un veicolo industriale a guida autonoma (AGV) la cui posa nello spazio è indicata dalle coordinate x , y e dalla direzione θ rispetto al sistema di riferimento fisso. L'AGV è dotato di tre ruote, due folli anteriori ed una ruota posteriore attuata per l'angolo di sterzo ψ da una coppia τ_ψ e da una coppia τ_φ per la velocità angolare $\dot{\varphi}$. Per ogni ruota è definito un sistema di riferimento come nella figura sotto (pedice A per ruote anteriori e pedice P per ruota posteriore).



I parametri geometrici necessari per il modello sono indicati nella prima figura.
Si definiscono i parametri inerziali del telaio e delle ruote (massa e momenti di inerzia) .

Parametri geometrici e inerziali del veicolo

```
Muletto_parametri.m + 
8 Mv=352; % [kg] massa telaio
9 ma=26; % [kg] massa ruota anteriore
10 mp=17; % [kg] massa ruota posteriore
11 Lv=2.236; %[m] lunghezza veicolo
12 L=1.53; % [m] passo
13 d=0.425; %[m] metà carreggiata
14 f=0.366; %[m] distanza inizio veicolo-assale anteriore
15 l=(L*mp+0.5*(Lv-f)*Mv)/(mp+2*ma+Mv); % [m] posizione baricentro rispetto assale anteriore
16 ra=0.23; % [m] raggio ruota anteriore
17 rp=0.19; % [m] raggio ruota posteriore
18 sa=0.178; % [m] spessore ruota anteriore
19 sp=0.127; % [m] spessore ruota posteriore
20 Iaz=0.5*ma*ra^2; % [kg*m^2] momento di inerzia asse za ruota anteriore
21 Ipz=0.5*mp*rp^2; % [kg*m^2] momento di inerzia asse zp ruota posteriore
22 Iay=0.5*Iaz+1/12*ma*(sa)^2; %[kg*m^2] momento inerzia asse ya ruota anteriore
23 Ipy=0.5*Ipz+1/12*ma*(sp)^2; %[kg*m^2] momento inerzia asse yp ruota posteriore
24 Igz=1/12*Mv*((2*d)^2+Lv^2); % [kg*m^2] momento di inerzia asse z telaio
```

La scelta dei parametri è stata fatta facendo riferimento a un muletto reale dell'azienda Cesab (www.cesab-forklifts.eu).

Equazioni del modello

Il modello del veicolo è governato da 3 equazioni cinematiche (per la posizione e l'orientazione nel piano) e da 2 dinamiche (legame coppie in ingresso con accelerazioni $\ddot{\psi}$ e $\ddot{\phi}$).

Cinematica

$$\begin{cases} \dot{x} = r_P \dot{\phi} \cos \theta \cos \psi \\ \dot{y} = r_P \dot{\phi} \sin \theta \cos \psi \\ \dot{\theta} = -\frac{r_P \dot{\phi} \sin \psi}{L} \end{cases}$$

Dinamica

$$\begin{cases} \tau_\phi = \left(ar_P^2 \cos^2 \psi + b \left(\frac{r_P}{L} \right)^2 \sin^2 \psi + I_{PY} \right) \ddot{\phi} - I_{PZ} \frac{r_P}{L} \ddot{\psi} \sin \psi + \\ \qquad \left(b \left(\frac{r_P}{L} \right)^2 - ar_P^2 \right) \dot{\phi} \dot{\psi} \sin \psi \cos \psi \\ \tau_\psi = -I_{PZ} \frac{r_P}{L} \ddot{\phi} \sin \psi + I_{PZ} \ddot{\psi} - I_{PZ} \frac{r_P}{L} \dot{\phi} \dot{\psi} \cos \psi \end{cases}$$

$$a = \left(M_V + m_P + 2m_A + 2 \frac{I_{AY}}{r_A^2} \right)$$

$$b = \frac{1}{2} \left(M_V l^2 + m_P L^2 + 2m_A d^2 + I_{GZ} + I_{PZ} + 2I_{AZ} + 2I_{AY} \left(\frac{d}{r_A} \right)^2 \right)$$

$$c = (M_V l + m_P L)$$

a, b e c sono parametri utilizzati per semplificare la scrittura delle equazioni.

Stato del sistema

Si definisce lo stato totale del sistema comprendente i sottosistemi dinamica e cinematica:

```
Muletto_parametri.m
37 %Vettore di stato sistema totale
38 stato=[x; y; theta; psi; psi_dot; phi_dot];
```

Sensori

Nello scenario sono presenti dei sensori per l'acquisizione di dati necessari per il problema d'identificazione:

- Sensore per la velocità di rotazione della ruota attuata $\dot{\phi}$
- Sensore per la posizione dello sterzo ψ
- Sensore per l'ascissa del punto medio assale anteriore x
- Sensore per la distanza tra origine e punto medio assale anteriore d_a

Per la misura della velocità angolare $\dot{\phi}$ utilizziamo un giroscopio della multinazionale Analog Devices Inc (www.analog.com) :

- Tempo di campionamento è di 0.1 secondi
- Rumore bianco additivo di misura a media nulla con covarianza $9 \cdot 10^{-4} (\text{rad/s})^2$

Per la misura dell' angolo di sterzo ψ utilizziamo un encoder dell' azienda Omron (www.industrial.omron.it) :

- Tempo di campionamento è di 0.08 secondi
- Rumore bianco additivo di misura a media nulla con covarianza $4 \cdot 10^{-4} \text{ rad}^2$

Per la misura delle distanze x e d_a utilizziamo un lidar dell' azienda Velodyne Lidar (www.velodynelidar.com) :

- Tempo di campionamento è di 0.05 secondi
- Rumore bianco additivo di misura a media nulla con covarianza $1 \cdot 10^{-4} \text{ m}^2$

Questa scelta di sensori permette di avere delle misure accurate necessarie per un buon effetto della parte di correzione nei filtri EKF e UKF.

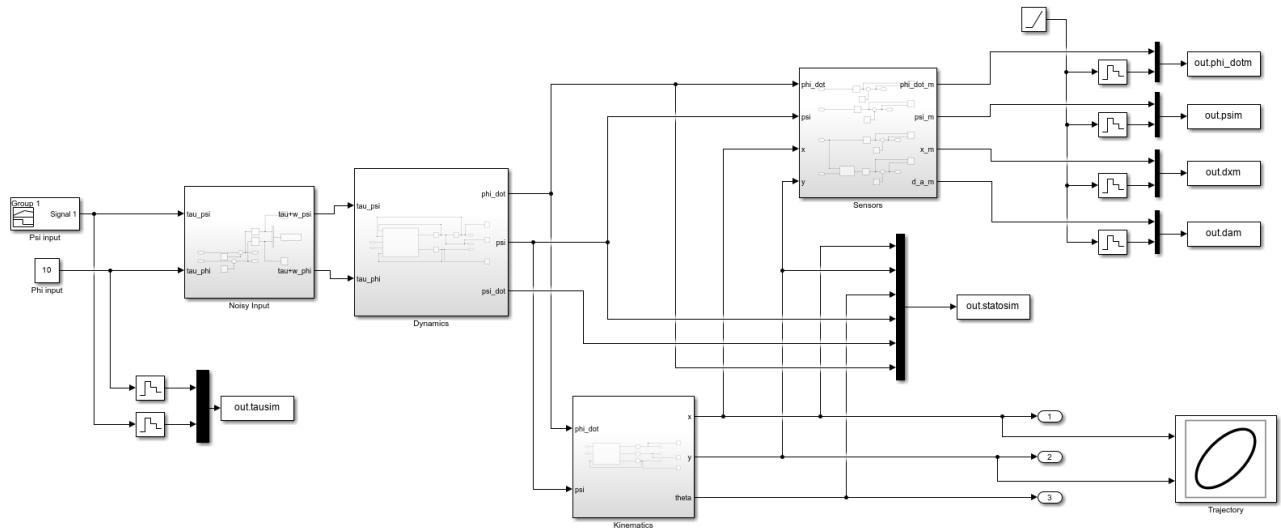
Modello di Osservazione

Per ogni sensore valutiamo la misura come somma fra il valore vero e il rumore di misura:

$$\begin{aligned}\dot{\varphi}_m &= \dot{\varphi} + v_{\dot{\varphi}} \\ \psi_m &= \psi + v_{\psi} \\ x_m &= x + v_x \\ d_{am} &= \sqrt{x^2 + y^2} + v_{d_a}\end{aligned}$$

Modello Simulink

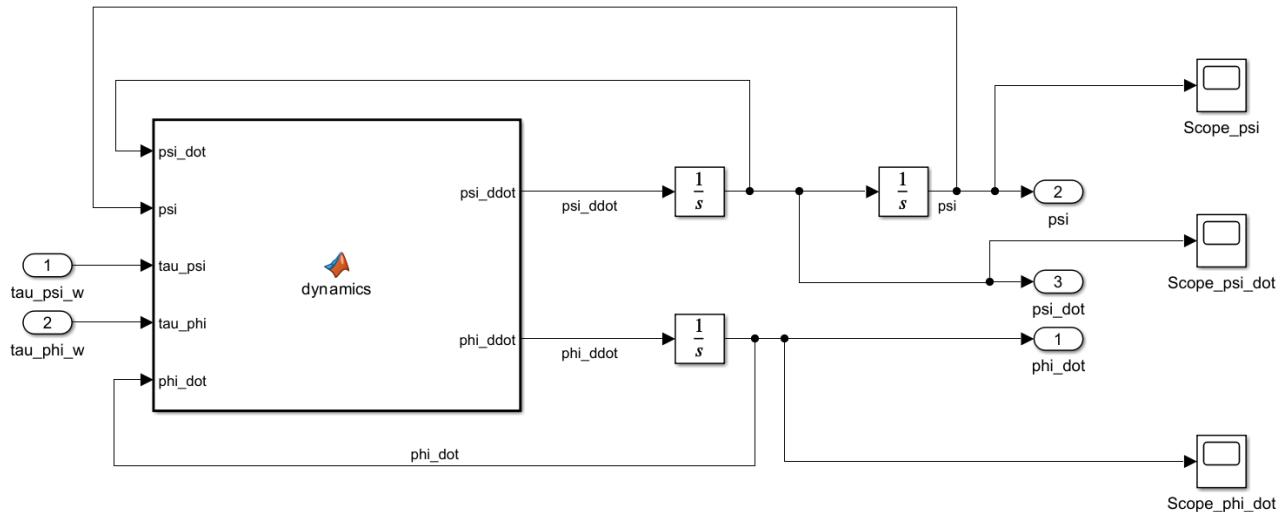
Il modello è stato sviluppato con Matlab e in particolare con l' ambiente Simulink.



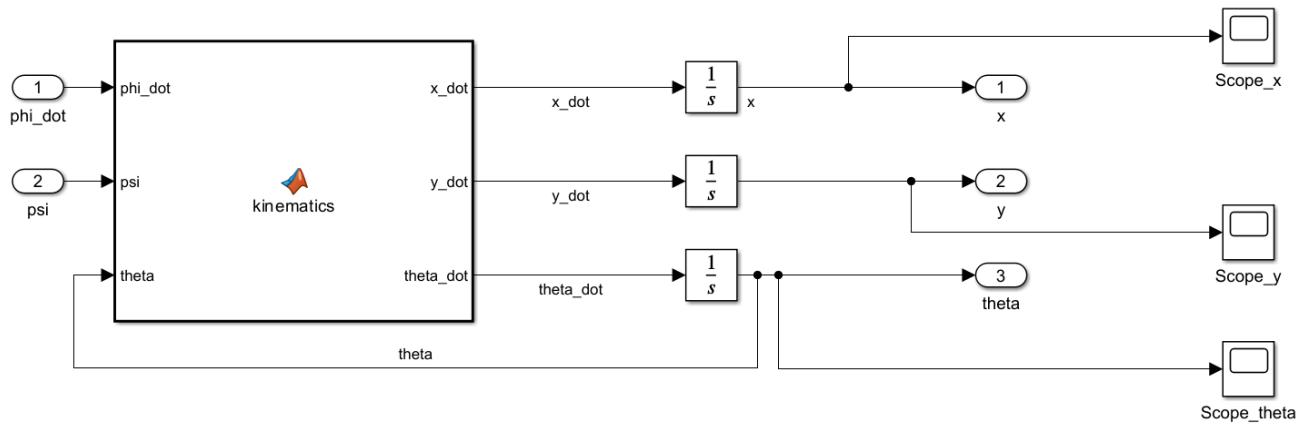
In figura è riportato lo schema completo dell' AGV che comprende ingressi, dinamica, cinematica e sensori.

Esplodendo i sottosistemi:

Dinamica



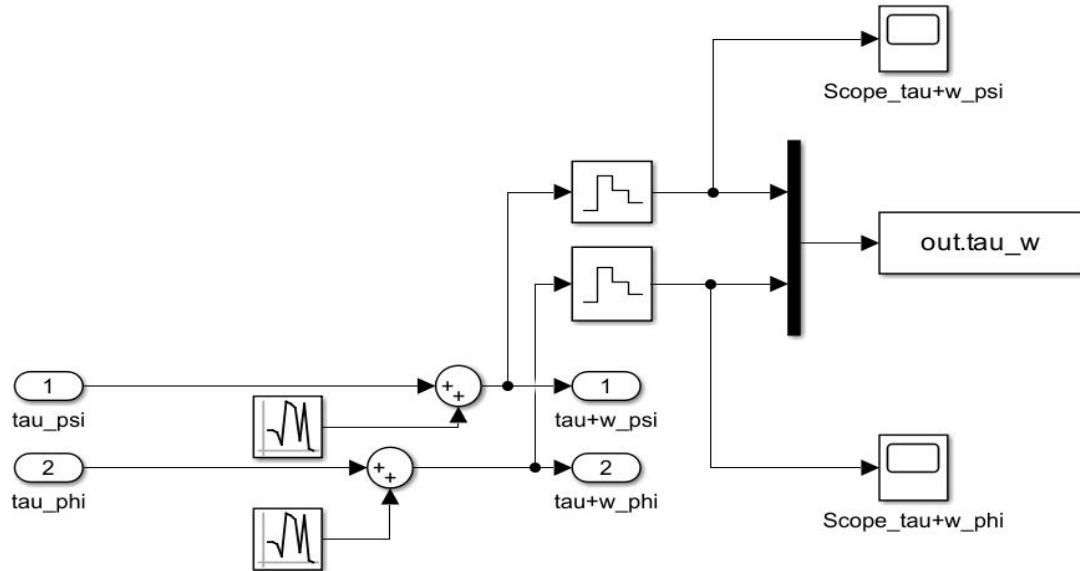
Cinematica



Per questi due sottosistemi vengono utilizzati 3 blocchi:

- Matlab function per scrivere le equazioni cinematiche e dinamiche
- Integrator per risolvere le equazioni differenziali
- Scope per visualizzare i grafici delle soluzioni delle equazioni differenziali

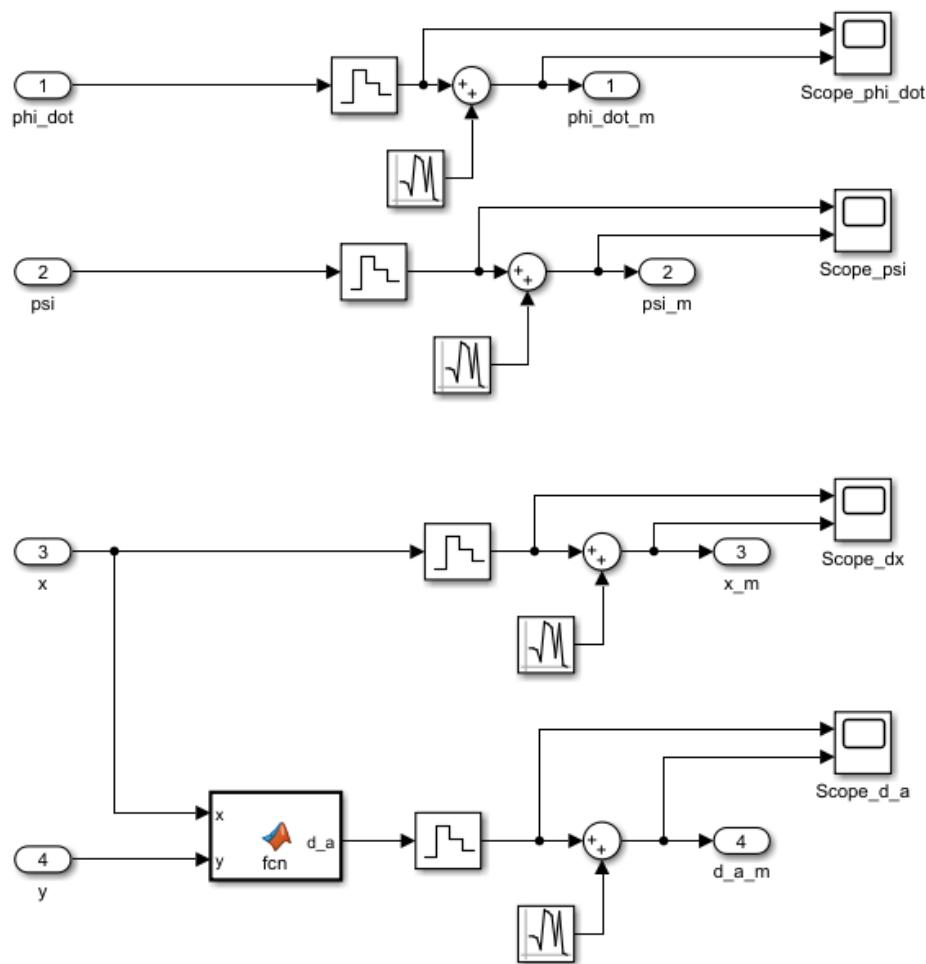
Ingressi



In questo sottosistema si hanno i seguenti blocchi:

- Random Number per simulare il rumore e aggiungerlo agli ingressi nominali
- I valori rumorosi degli ingressi salvati nella variabile “out.tau_w” per le successive simulazioni
- Scope per visualizzare gli ingressi rumorosi

Sensori



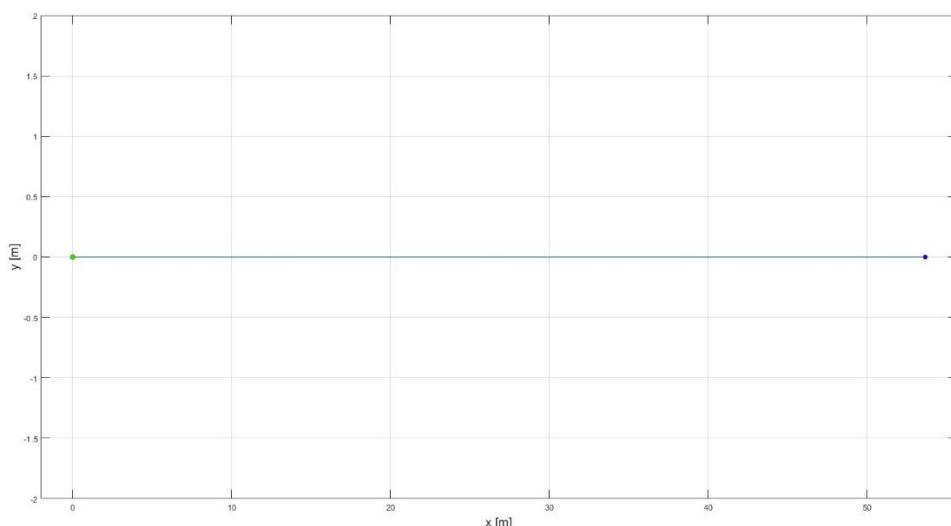
I primi 3 sensori si costruiscono con aggiunta del rumore (blocco Random Number) alle soluzioni delle equazioni differenziali. Invece il quarto presenta anche un blocco Matlab function per fare il modulo di (x, y) oltre al rumore additivo. Tutte le misure vengono poi campionate tramite il blocco “zero order hold”. Il blocco scope visualizza le misure.

Validazione del modello

Si valida il modello considerando varie traiettorie con ingressi noti:

Caso $\tau_\phi = 10 \text{ Nm}$ e $\tau_\psi = 0 \text{ Nm}$

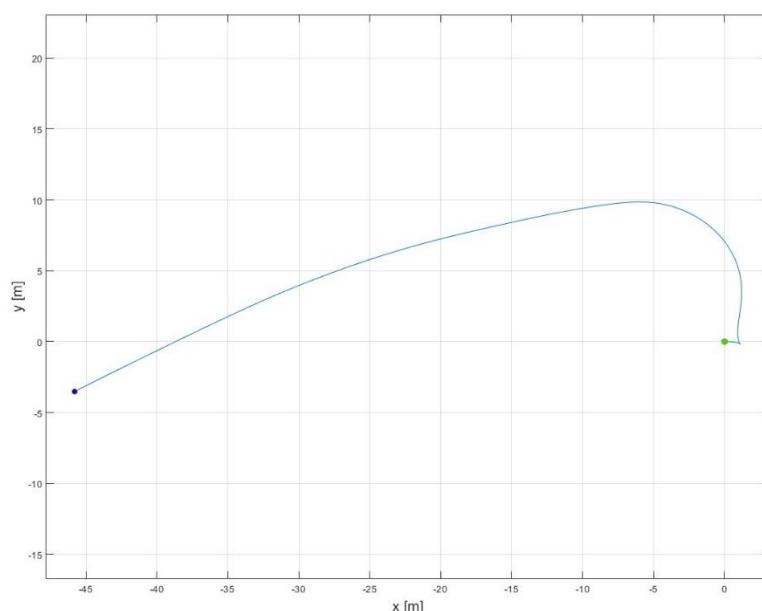
Il veicolo si muove con un moto rettilineo soggetto a $\tau_\phi = 10 \text{ Nm}$ costante, poiché non viene effettuato un controllo sullo sterzo. Non si considera il disturbo sullo sterzo perché è molto sensibile.



Caso $\tau_\phi = 10 \text{ Nm}$ e $\tau_\psi = \pm 0.05 \text{ Nm}$

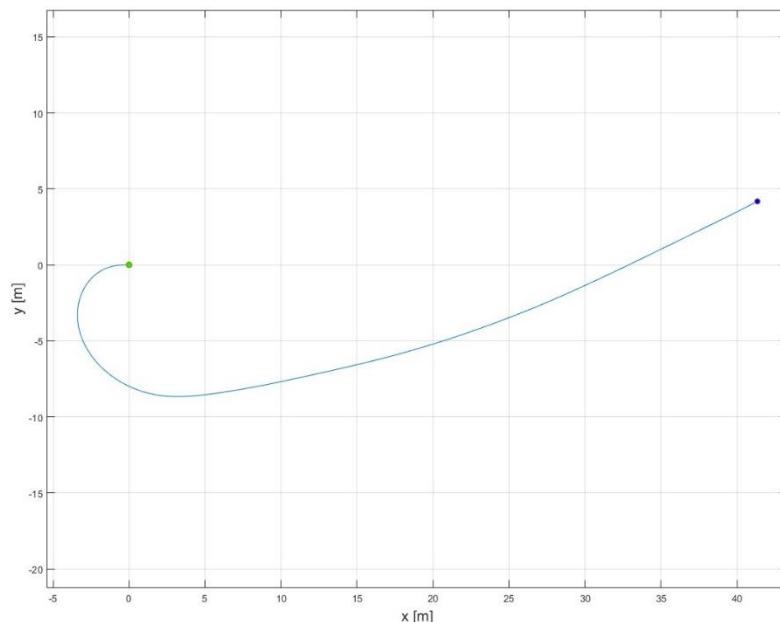
Il controllo dello sterzo τ_ψ assume valore pari a 0.05 Nm per un intervallo di tempo da 0.58 a 1.84 secondi, -0.05 Nm fra 14.3 e 16 secondi (sovraposizione di onde quadre) e nullo altrove.

$\tau_\phi = 10 \text{ Nm}$ costante per tutta la traiettoria. Il veicolo ha un' inversione del senso di marcia dopo la prima azione di sterzo, esegue una curva e infine precede con un moto mediamente rettilineo grazie all'azione di contro sterzo.



Caso $\tau_\phi = -10 \text{ Nm}$ e $\tau_\psi = \pm 0.05 \text{ Nm}$

Rispetto al caso precedente viene cambiato solo il segno di τ_ϕ per dimostrare che l'inversione del senso di marcia è dovuto alla fisica del veicolo.

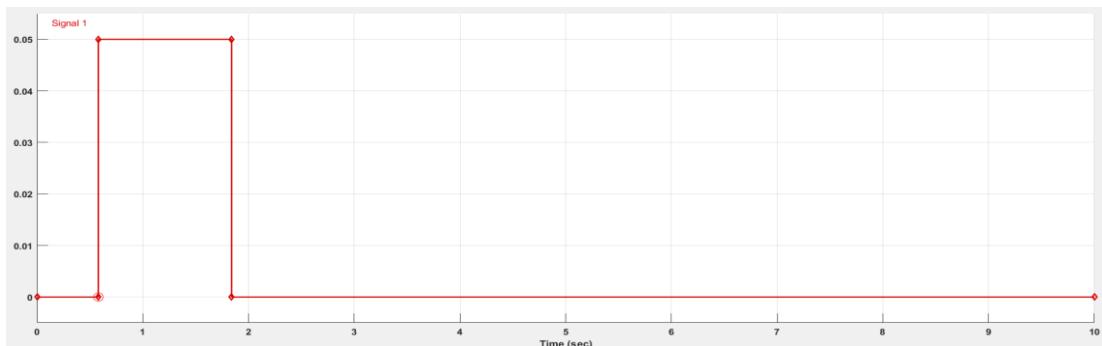


Caso in esame per la stima dinamica

Ingressi

Gli ingressi controllabili sono τ_ϕ e τ_ψ , rispettivamente della velocità e dello sterzo, da considerarsi rumorosi.

- Il controllo della velocità τ_ϕ è costante e pari a 10 Nm per tutto il tempo della simulazione. Il rumore è un rumore bianco additivo con media nulla e covarianza pari a $3,6 \cdot 10^{-3} (\text{Nm})^2$.
- Il controllo dello sterzo τ_ψ assume valore pari a 0.05 Nm per un intervallo di tempo da 0.58 a 1.84 secondi e nullo per il restante tempo della simulazione. Il rumore è sempre un rumore bianco additivo con media nulla e covarianza pari a $1 \cdot 10^{-6} (\text{Nm})^2$.

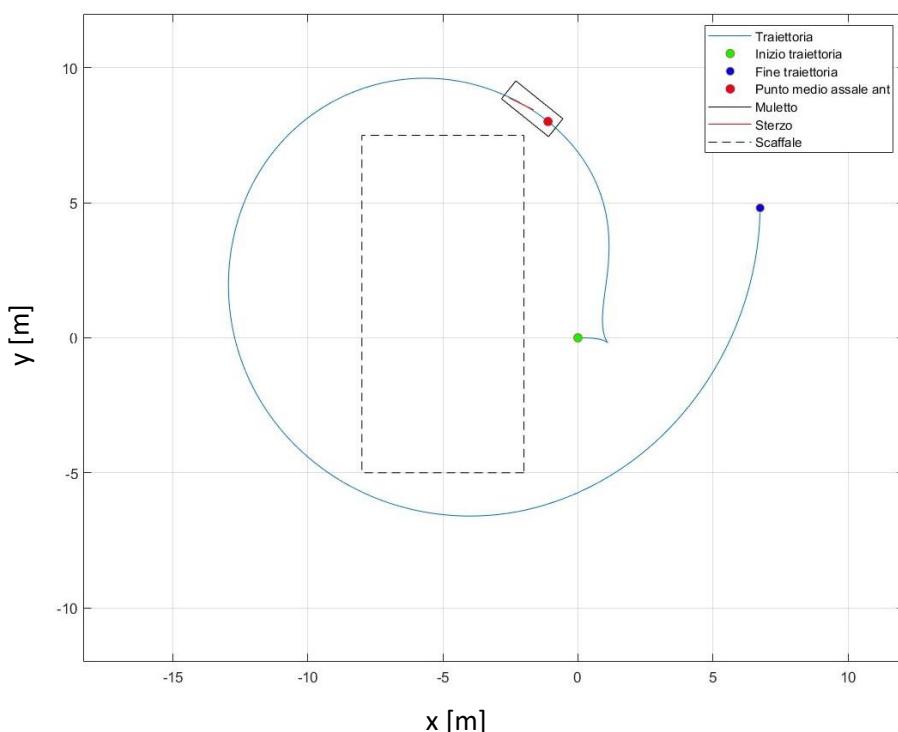


In figura l' andamento della coppia τ_ψ .

La scelta di questi valori numerici per le coppie di ingresso è dovuta alla traiettoria imposta al veicolo e ai limiti cinematici (velocità del veicolo inferiore ai 3 m/s).

In particolare, un controllo fine dello sterzo ci permette di non avere continue inversioni del moto dovute alla fisica del problema.

Traiettoria



L'obiettivo dell' AGV è di girare intorno allo scaffale (dopo aver ritirato un pacchetto) e di raggiungere lo scaffale successivo (punto blu). Il tempo richiesto per compiere la traiettoria è di 30 secondi.

Lo scaffale viene rappresentato da un rettangolo in linea tratteggiata.

La traiettoria, ottenuta con gli ingressi scelti precedentemente, inizia dall'origine degli assi (0,0) metri e finisce in (6.75,4.81) metri. A causa della fisica del problema, per i primi 11.45 secondi il veicolo si muove con l'assale anteriore che precede la ruota posteriore, dopo di che si ha inversione di moto e il restante tempo percorre la traiettoria in retromarcia.

Obiettivi del progetto

- 1) Stima dinamica dello stato del sistema con le seguenti tecniche:
 - EKF (Extended Kalman Filter)
 - EKF con RTS (Extended Kalman Filter con smoother Rauch-Tung-Striebel)
 - UKF (Unscented Kalman Filter)
- 2) Analisi e confronto delle tecniche

Filtro di Kalman

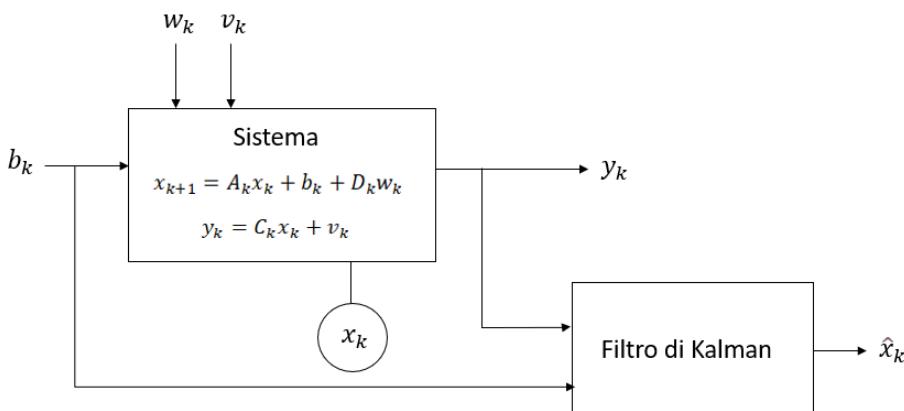
Dato un sistema lineare tempo variante tempo discreto

$$\begin{aligned} x_{k+1} &= A_k x_k + b_k + D_k w_k && \text{(funzione di transizione dello stato)} \\ y_k &= C_k x_k + v_k && \text{(modello di osservazione)} \end{aligned}$$

dove $x_k \in \mathbb{R}^n$ è lo stato del sistema con pdf $(x \sim (x_k, P_k))$, b_k l'ingresso deterministico, y_k l'uscita misurata e w_k e v_k rumori bianchi a media nulla con relative covarianze Q_k e R_k .

Si suppone nota la pdf di x al tempo $t=0$ s: $x \sim (x_0, P_0)$.

Per stimare lo stato del sistema si utilizza il filtro di Kalman, che è il migliore stimatore lineare dello stato nel senso MMSE, ovvero minimizza l'errore quadratico medio di stima ad ogni istante di tempo.



Il filtro di Kalman è costituito da 3 equazioni ricorsive:

$$\begin{aligned} K_k &= A_k P_{k|k-1} C_k^T (R_k + C_k P_{k|k-1} C_k^T)^{-1} \\ \hat{x}_{k+1|k} &= A_k \hat{x}_{k|k-1} + b_k + K_k (y_k - C_k \hat{x}_{k|k-1}) \\ P_{k+1|k} &= A_k P_{k|k-1} A_k^T - A_k P_{k|k-1} C_k^T (R_k + C_k P_{k|k-1} C_k^T) C_k P_{k|k-1} A_k^T + D_k Q_k D_k^T \end{aligned}$$

È possibile, inoltre, esprimere lo stimatore in due fasi per un' implementazione migliore:

Correzione

$$\begin{aligned} e_k &= y_k - C_k \hat{x}_{k|k-1} \\ S_k &= R_k + C_k P_{k|k-1} C_k^T \\ L_k &= P_{k|k-1} C_k^T S_k^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k e_k \quad P_{k|k} = P_{k|k-1} - L_k S_k L_k^T \end{aligned}$$

Predizione

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + b_k \quad P_{k+1|k} = A_k P_{k|k} A_k^T + D_k Q_k D_k^T$$

Il filtro di Kalman è inoltre lo stimatore ottimo nel senso MMSE tra gli stimatori lineari e non lineari se tutte le densità di probabilità considerate sono gaussiane.

EKF(Extended Kalman Filter)

Introduzione teorica

L'EKF è una tecnica di filtraggio applicabile al caso di sistemi non lineari, dove i rumori sono da considerarsi bianchi e incorrelati.

$$\begin{cases} x_{k+1} = f_k(x_k, w_k) \\ y_k = h_k(x_k, v_k) \end{cases}$$

Di fatto, l'Extended Kalman Filter rappresenta il corrispettivo dinamico del metodo di linearizzazione nel caso statico.

Si effettua pertanto una linearizzazione, ovvero una approssimazione al prim'ordine dello sviluppo di Taylor, delle funzioni f e h valutandone i Jacobiani. Il metodo prevede due fasi, una di predizione e una di correzione, che servono per stimare lo stato al passo successivo.

Predizione

Per prima cosa si valutano i Jacobiani F_k e D_k .

$$F_k = \frac{\partial f_k}{\partial x_k} \Big|_{\substack{x_k = \hat{x}_{k|k} \\ w_k = 0}} \quad D_k = \frac{\partial f_k}{\partial w_k} \Big|_{\substack{x_k = \hat{x}_{k|k} \\ w_k = 0}}$$

dove $\hat{x}_{k|k}$ è la stima corretta al tempo k e w_k rumore nullo in media.

Dopodiché si procede al calcolo della matrice di covarianza, valutata come:

$$P_{k+1|k} = F_k \cdot P_{k|k} \cdot F_k^T + D_k \cdot Q_k \cdot D_k^T$$

dove Q_k rappresenta la matrice di covarianza dei disturbi in ingresso.

Correzione

Per la correzione si procede a valutare i Jacobiani H_k e M_k .

$$H_k = \frac{\partial h_k}{\partial x_k} \Big|_{\substack{x_k = \hat{x}_{k|k-1} \\ v_k = 0}} \quad M_k = \frac{\partial h_k}{\partial v_k} \Big|_{\substack{x_k = \hat{x}_{k|k-1} \\ v_k = 0}}$$

dove $\hat{x}_{k|k-1}$ è la stima predetta al tempo k con $k - 1$ osservazioni e v_k rumore nullo in media.

Si valuta l'innovazione come:

$$e_k = y_k - h_k(\hat{x}_{k|k-1}, 0)$$

$$S_k = H_k \cdot P_{k|k-1} \cdot H_k^T + M_k \cdot R_k \cdot M_k^T$$

$$L_k = P_{k|k-1} \cdot H_k \cdot S_k^{-1}$$

dove R_k rappresenta la matrice di covarianza dei disturbi dei sensori.

Infine, si valuta lo stato corretto e la matrice di covarianza come:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k \cdot e_K$$

$$P_k = P_{k|k-1} - L_k \cdot S_K \cdot L_k^T$$

Implementazione EKF

Inizializzazione del filtro

Vengono inizialmente definiti i parametri di inizializzazione dell'algoritmo EKF, seguiti da alcune matrici che conterranno i dati per le iterazioni successive e per il plot dei risultati.

```
EKF_progetto.m
1 %Caricamento dei parametri costruttivi
2 Muletto_parametri;
3
4 %Definizione valori iniziali
5 stato_0=[0 0 0 0 0 0]';
6 I2=eye(2);
7 std_dev_l=0.05;
8 std_dev_ang=pi/18;
9 std_dev_vel=0.05;
10 dev_mat=blkdiag(std_dev_l*I2, std_dev_ang*I2, std_dev_vel*I2);
11 stato_hat0=stato_0+dev_mat*randn(size(stato_0,1),1);
12 P0=dev_mat^2;
13
14 dimensione = floor (T_sample/dt);
15
16 stato_hat=zeros(6,dimensione+1);
17 stato_pred=zeros(6,dimensione+1);
18 stato_corr=zeros(6,dimensione+1);
19 ek=zeros(4,dimensione);
20
21
22
```

Inoltre, si vanno a inizializzare variabili e matrici che saranno utili nella gestione dei sensori con tempi di campionamento diversi. Il loro significato sarà chiaro più avanti, nel paragrafo “Implementazione della Correzione”.

```
EKF_progetto.m
1 %calcolo il numero delle misure che mi arriveranno da ciascun sensore.
2 %mi serve per evitare di prendere una misura di troppo in caso di sensori i
3 %cui tempi di campionamento non siano multipli
4
5 dim_phi_dot = floor(T_sample/dt_phi_dot)+1;
6 dim_psi = floor(T_sample/dt_psi)+1;
7 dim_x = floor(T_sample/dt_dx)+1;
8 dim_da = floor(T_sample/dt_da)+1;
9
10 flag_phi_dot=0;
11 flag_psi=0;
12 flag_x=0;
13 flag_da=0;
14
15
16 %inizializzazione dei parametri per la gestione dei sensori con tempi di
17 %campionamento diversi
18
19 dimy=1;
20 cont_phi_dot=2;
21 cont_psi=2;
22 cont_x=2;
23 cont_da=2;
24 tempo_corr=dt;
25 Vdev=zeros(4,1);
```

Si inizializza il ciclo, composto dalle fasi di predizione e correzione.

```

EKF_progetto.m + 
50
51     %inizializzazione ciclo
52
53     i=0;
54     stato_hat(:,1)=stato_hat0;
55     Pk=P0;
56
57     %ciclo EKF
58     for k=dt:dt:T_sample
59         i=i+1;
60
61

```

Implementazione della predizione

```

EKF_progetto.m + 
63 %
64 %          Predizione          %
65 %
66
67 % ricavo i valori dello stato al passo i-esimo (servono per
68 % risolvere le matrici simboliche Fk, Dk, ecc)
69
70 x= stato_hat(1,i);
71 y= stato_hat(2,i);
72 theta= stato_hat(3,i);
73 psi= stato_hat(4,i);
74 psi_dot= stato_hat(5,i);
75 phi_dot= stato_hat(6,i);
76
77 %ricavo i valori degli ingressi al passo i-esimo
78 tau_phi= out.tausim(i,1);
79 tau_psi= out.tausim(i,2);
80
81 %definisco nulli i rumori su ingressi
82 w_phi=0;
83 w_psi=0;

```

Dopo una prima fase, nella quale vengono raccolte le variabili utili per il calcolo, si procede valutando le matrici F_k e D_k , ovvero i Jacobiani della funzione di transizione dello stato rispetto allo stato stesso \hat{x}_k e rispetto al rumore w_k .

I Jacobiani vengono calcolati a partire dalla funzione di transizione dello stato discretizzata. Per la discretizzazione utilizziamo il metodo di Eulero in avanti.

$$f_{disc} = x + dt \cdot f$$

Le matrici F_k e D_k sono state inizialmente calcolate simbolicamente in uno script Matlab, dopodiché il risultato ottenuto è stato copiato nel codice dell'EKF, dove la valutazione delle due matrici avviene ad ogni ciclo. Questa scelta implementativa è stata dettata da una questione computazionale. Infatti, questo metodo risulta estremamente più veloce rispetto al calcolo delle matrici tramite funzioni Matlab come la funzione “subs”.

```

EKF_progetto.m + 
84
85
86 %calcolo Fk al passo i-esimo
87
88 Fk =...
89 [1 0 -dt*phi_dot*rp*cos(psi)*sin(theta)
90 0 1 dt*phi_dot*rp*cos(psi)*cos(theta)
91 0 0 1
92 0 0 0
93 0 0 0 dt*(((- 2*a*cos(psi)*sin(psi)*L^2*rp^2 + 2*b*cos(psi)*sin(psi)*rp
94 0 0 0
95
96
97 %calcolo Dk al passo i-esimo
98
99 Dk =...
100 [
101 0
102 0
103 0
104 (L*dt*rp*sin(psi))/(Ipy*L^2 - Ipz*rp^2*sin(psi)^2 + b*rp^2*sin(psi)^2 + L^2*a*rp^2*cos(psi)^2), (dt*(a*cos
105 (L^2*dt)/(Ipy*L^2 - Ipz*rp^2*sin(psi)^2 + b*rp^2*sin(psi)^2 + L^2*a*rp^2*cos(psi)^2),
106

```

Sebbene il metodo sia più veloce a livello computazionale, le matrici hanno una struttura complessa e pertanto non riescono ad essere visualizzate in una foto.

La stessa cosa viene fatta anche per il calcolo dello stato predetto, il quale avviene ad ogni passo valutando lo stato attraverso le funzioni costruttive dell'AGV, ovvero le prime tre derivate dalle equazioni della cinematica e le ultime tre delle equazioni della dinamica, che rappresentano la funzione di transizione dello stato.

Anche in questo caso la valutazione avviene sulla funzione discretizzata.

```

EKF_progetto.m + 
107
108 % calcolo lo stato predetto i+1-esimo
109
110 stato_hat(:,i+1)=...
111 [
112
113
114
115 psi_dot + dt*((tau_psi + w_psi + (Ipy*phi_dot*psi_dot*rp*cos(psi))/L)*(a*cos(psi)^2*L^2*rp^2 + Ipy*L^2 + b*sin(psi)^2*rp^2
116 phi_dot + dt*((L^2*(tau_phi + w_phi + phi_dot*psi_dot*cos(psi)))/L) + dt*((L^2*(tau_phi + w_phi + phi_dot*psi_dot*cos(psi)))/L)
117

```

Si conclude poi la fase di predizione con l'aggiornamento della matrice P_k .

```

EKF_progetto.m + 
119
120 % aggiorno Pk
121 Pk=Fk*Pk*Fk'+Dk*Q*Dk';
122
123 %aggiornamento di variabili che servono per lo smoother
124 stato_pred(:,i+1)=stato_hat(:,i+1);
125 P_pred(:,:,i+1)=Pk;
126 Fstore(:,:,:,i+1)=Fk;
127
128

```

Implementazione della correzione

```

EKF_progetto.m + %
131 %%%%%%
132 % Correzione %
133 %%%%%%
134
135
136 % ricavo i valori dello stato al predetto (servono per
137 % risolvere le matrici simboliche Hk, Mk, ecc)
138
139 x= stato_hat(1,i+1);
140 y= stato_hat(2,i+1);
141 theta= stato_hat(3,i+1);
142 psi= stato_hat(4,i+1);
143 psi_dot= stato_hat(5,i+1);
144 phi_dot= stato_hat(6,i+1);
145
146 %definisco nulli i rumori sui sensori
147 v_phi_dot=0;
148 v_psi=0;
149 v_dx=0;
150 v_d_a=0;
151

```

La prima fase è simile alla predizione, ovvero si vanno a raccogliere i dati utili per il calcolo in variabili più leggibili.

Si procede poi con il calcolo delle matrici H_k e M_k , ottenute con lo stesso procedimento utilizzato per F_k e D_k .

```

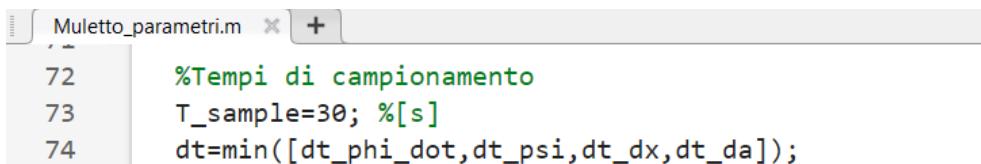
EKF_progetto.m + %
152
153 %calcolo Hk e hk
154 Hk =...
155 [ 0 0 0 0; 0 0 1 0; 0 0 0 0; 0 0 0 0];
156
157
158 x/(x^2 + y^2)^(1/2) y/(x^2 + y^2)^(1/2) 0 0 0 0];
159
160
161 hk=[phi_dot+v_phi_dot; psi_dot+v_psi; x+v_dx; sqrt(x^2+y^2)+v_d_a];
162
163

```

Prima di generare M_k , che nel nostro sistema risulta essere una matrice identità, è però necessaria la valutazione delle misure arrivate dai sensori. È infatti possibile che non tutti possiedano tempi di campionamento uguali e quindi non è detto che all'istante k arrivino tutte le misure simultaneamente. Questo influisce sulla dimensione di M_k e H_k che variano in funzione delle misure arrivate ad ogni istante di tempo k .

Le scelte progettuali legate alla gestione di sensori con tempi di campionamento diversi sono le seguenti.

Tra i vari sensori che ha il sistema, uno di questi è più veloce e si prende come riferimento per la scelta del tempo di campionamento di tutta la simulazione.



```

Mulletto_parametri.m
72 %Tempi di campionamento
73 T_sample=30; %[s]
74 dt=min([dt_phi_dot,dt_psi,dt_dx,dt_da]);

```

Una variabile, chiamata “tempo_corr”, tiene conto del tempo trascorso dall’inizio del filtraggio e viene utilizzata per verificare se la misura di un sensore è arrivata o meno.

Nello specifico si confronta il tempo di arrivo della misura che deve essere ancora letta dal sensore con il tempo corrente, e se il primo risulta essere più piccolo allora questa viene presa e gestita.

Per tenere traccia dell’istante in cui una nuova misura viene generata, le misure vengono raccolte in un vettore di due elementi, contenenti l’uscita del sensore e l’istante di campionamento.

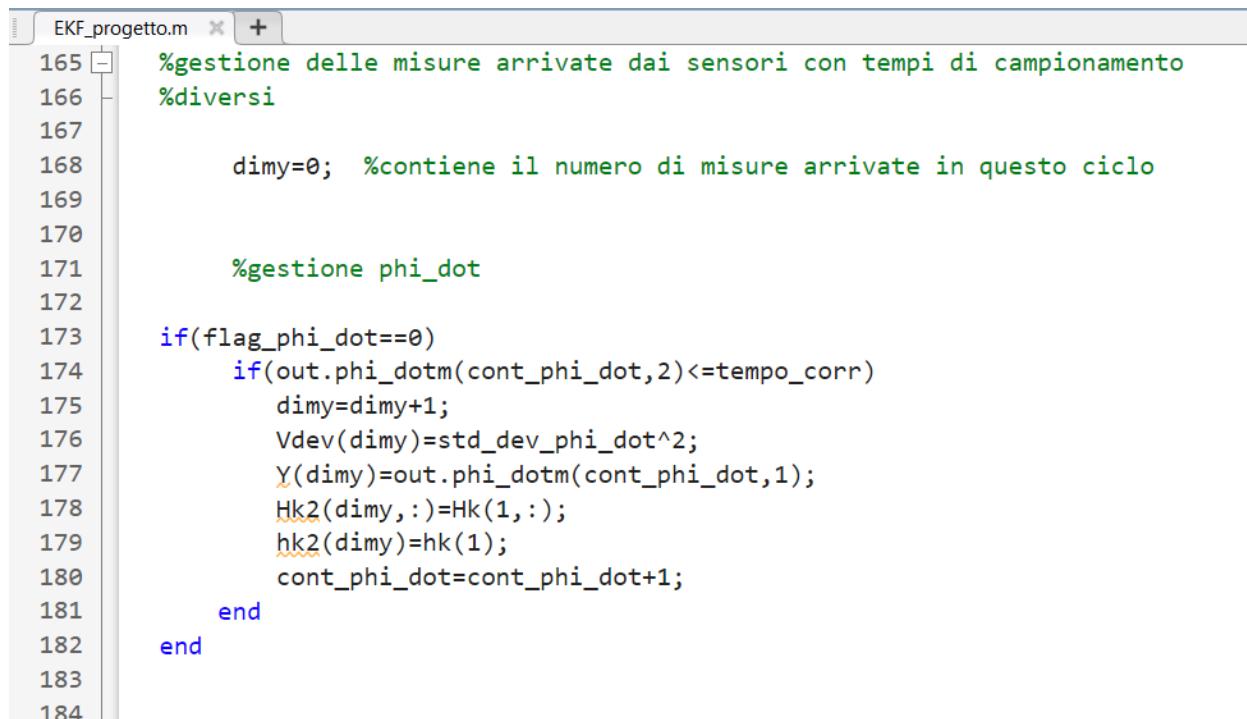
Un contatore, definito per ogni sensore, viene incrementato ogni volta che una misura viene impiegata per la correzione e viene utilizzato per scorrere il dataset delle misure relative.

Con questa scelta progettuale si ha che la funzione h_k e le matrici M_k , R_k ed H_k vengono ricostruite dinamicamente ad ogni ciclo, assumendo una dimensione coerente con il numero di misure effettivamente arrivate in quell’istante, salvato nella variabile “dimy”.

Si può notare che almeno un sensore produrrà sempre una misura, cioè quello più veloce, il cui tempo di campionamento è stato scelto come tempo di scansione, mentre si dovranno gestire più sensori se nel lasso di tempo tra una scansione e l’altra un altro sensore ha generato un’uscita.

Nel codice sono presenti anche dei flag, che sono stati inseriti per evitare problemi relativi allo scorrimento dei dataset delle misure, in particolare per l’ultima lettura.

L’implementazione di questo metodo è la seguente:



```

EKF_progetto.m
165 %gestione delle misure arrivate dai sensori con tempi di campionamento
166 %diversi
167
168 dimy=0; %contiene il numero di misure arrivate in questo ciclo
169
170
171 %gestione phi_dot
172
173 if(flag_phi_dot==0)
174     if(out.phi_dotm(cont_phi_dot,2)<=tempo_corr)
175         dimy=dimy+1;
176         Vdev(dimy)=std_dev_phi_dot^2;
177         Y(dimy)=out.phi_dotm(cont_phi_dot,1);
178         Hk2(dimy,:)=Hk(1,:);
179         hk2(dimy)=hk(1);
180         cont_phi_dot=cont_phi_dot+1;
181     end
182 end
183
184

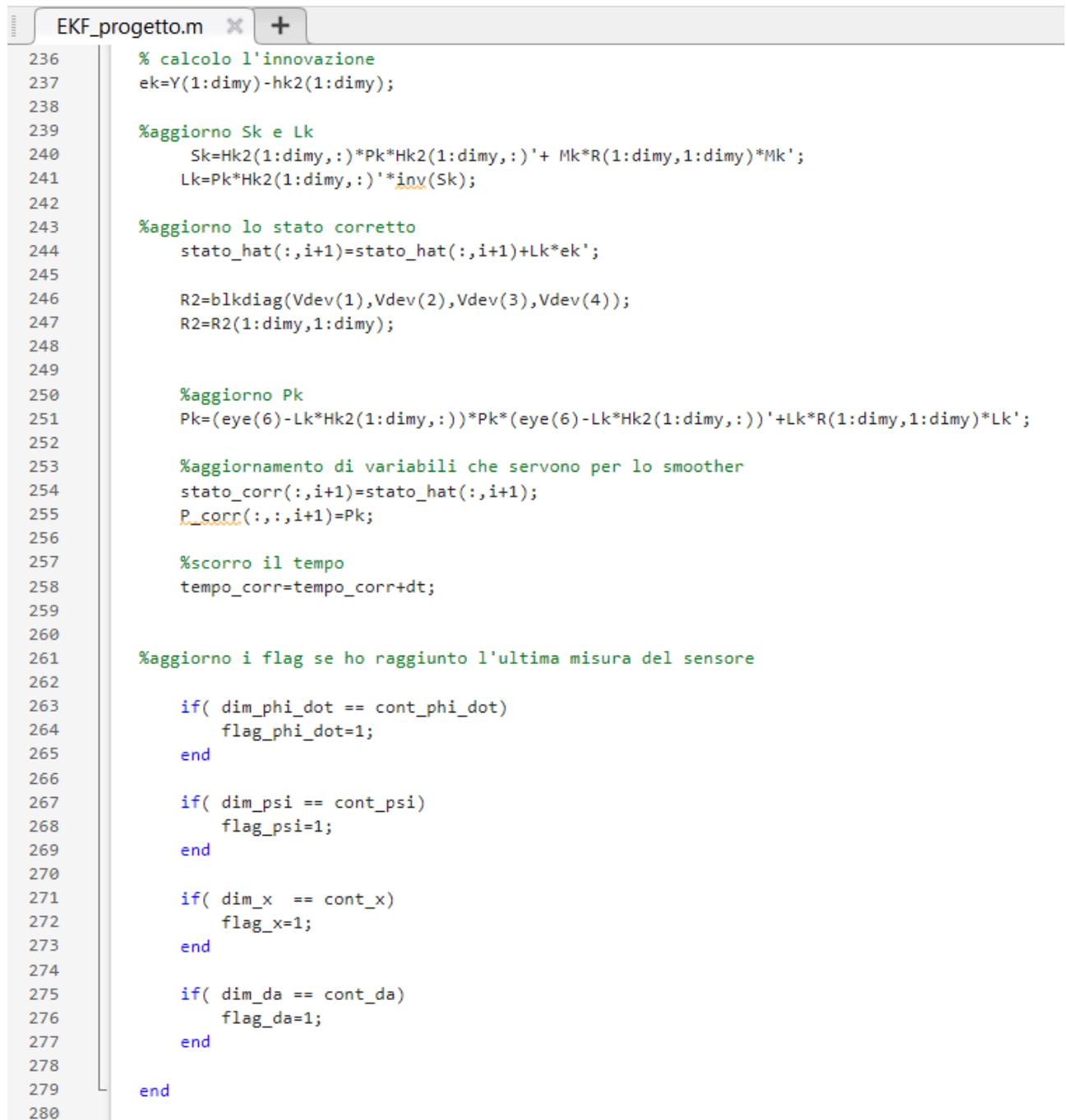
```

```

185     %gestione psi
186
187     if(flag_psi==0)
188         if(out.psim(cont_psi,2)<=tempo_corr)
189             dimy=dimy+1;
190             Vdev(dimy)=std_dev_psi^2;
191             Y(dimy)=out.psim(cont_psi,1);
192             Hk2(dimy,:)=Hk(2,:);
193             hk2(dimy)=hk(2);
194             cont_psi=cont_psi+1;
195         end
196     end
197
198
199     %gestione x
200
201     if(flag_x==0)
202         if(out.dxm(cont_x,2)<=tempo_corr)
203             dimy=dimy+1;
204             Vdev(dimy)=std_dev_dx^2;
205             Y(dimy)=out.dxm(cont_x,1);
206             Hk2(dimy,:)=Hk(3,:);
207             hk2(dimy)=hk(3);
208             cont_x=cont_x+1;
209         end
210     end
211
212
213     %gestione da
214
215     if(flag_da==0)
216         if(out.dam(cont_da,2)<=tempo_corr)
217             dimy=dimy+1;
218             Vdev(dimy)=std_dev_da^2;
219             Y(dimy)=out.dam(cont_da,1);
220             Hk2(dimy,:)=Hk(4,:);
221             hk2(dimy)=hk(4);
222             cont_da=cont_da+1;
223         end
224     end
225
226
227
228 %calcolo Mk
229 %Mk è di fatto una matrice identità, ciò rende più semplici le cose, perché
230 %ad ogni ciclo basta costruirla della dimensione del numero di sensori che
231 %hanno prodotto una misura
232
233 Mk = eye(dimy);
234

```

Si conclude poi la fase di correzione andando a calcolare e_k , S_k , L_k , lo stato corretto (“stato_corr”) e l’aggiornamento di P_k .



```

EKF_progetto.m + 

236 % calcolo l'innovazione
237 ek=Y(1:dimy)-hk2(1:dimy);
238
239 %aggiorno Sk e Lk
240 Sk=Hk2(1:dimy,:)*Pk*Hk2(1:dimy,:)' + Mk*R(1:dimy,1:dimy)*Mk';
241 Lk=Pk*Hk2(1:dimy,:)'*inv(Sk);
242
243 %aggiorno lo stato corretto
244 stato_hat(:,i+1)=stato_hat(:,i+1)+Lk*ek;
245
246 R2=blkdiag(Vdev(1),Vdev(2),Vdev(3),Vdev(4));
247 R2=R2(1:dimy,1:dimy);
248
249
250 %aggiorno Pk
251 Pk=(eye(6)-Lk*Hk2(1:dimy,:))*Pk*(eye(6)-Lk*Hk2(1:dimy,:))'+Lk*R(1:dimy,1:dimy)*Lk';
252
253 %aggiornamento di variabili che servono per lo smoother
254 stato_corr(:,i+1)=stato_hat(:,i+1);
255 P_corr(:,:,i+1)=Pk;
256
257 %scorro il tempo
258 tempo_corr=tempo_corr+dt;
259
260
261 %aggiorno i flag se ho raggiunto l'ultima misura del sensore
262
263 if( dim_phi_dot == cont_phi_dot)
264   flag_phi_dot=1;
265 end
266
267 if( dim_psi == cont_psi)
268   flag_psi=1;
269 end
270
271 if( dim_x == cont_x)
272   flag_x=1;
273 end
274
275 if( dim_da == cont_da)
276   flag_da=1;
277 end
278
279 end
280

```

Valutazione dei risultati EKF

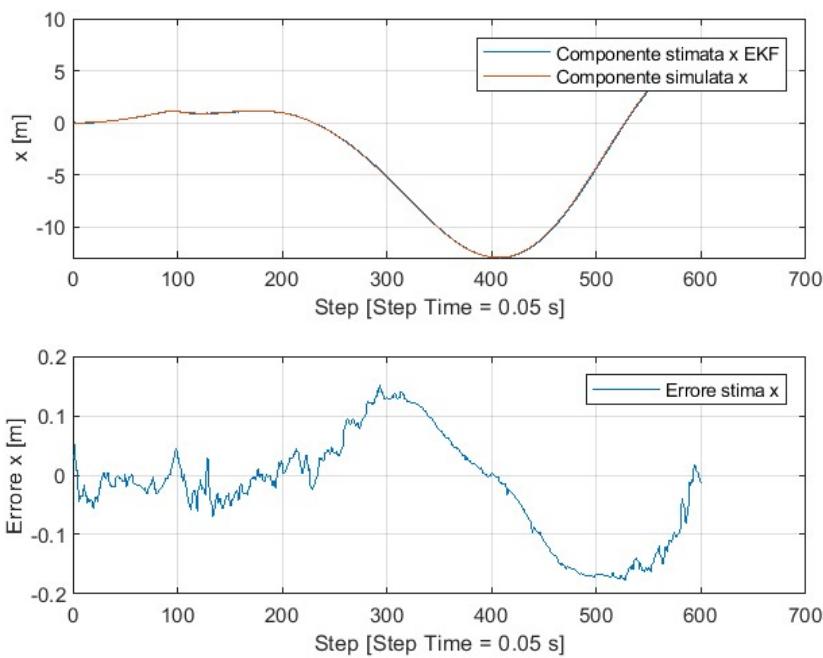
Dopo aver simulato il sistema per 30 secondi, le misure raccolte dai sensori sono state utilizzate dal filtro EKF per stimare la traiettoria dello stato. Si analizzano prima le singole componenti.

X

La componente dello stato x viene stimata bene fin dalle prime iterazioni. Infatti, si nota che l'errore di stima non supera mai i 20 cm, che considerando le dimensioni del nostro veicolo e della lunghezza della traiettoria sono accettabili.

Col passare delle iterazioni le incertezze sullo stato iniziale si riducono, infatti l'errore di stima si attesta a valori prossimi allo 0, come si vede anche dal grafico della traiettoria dove la stima e la simulazione praticamente si sovrappongono.

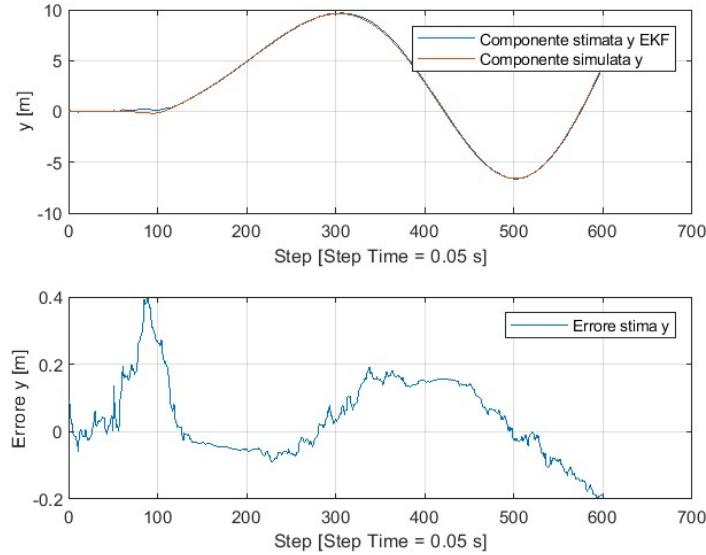
La quantità stimata dopo 10 secondi fornisce errori accettabili, ovvero dell' ordine dei centimetri, ma non nulli. L'errore medio della stima di x è nullo come ci si aspetta.



Y

Le considerazioni che possono essere fatte su y sono analoghe a quelle per la componente x .

La differenza sta nella dimensione degli errori, specialmente nelle prime fasi di filtraggio, dove gli errori sono dell'ordine di 0.4 m. La giustificazione a questo risultato può essere data dal fatto che la componente y non viene misurata direttamente, ma viene ricostruita a partire dalle uscite dei sensori di distanza che restituiscono le grandezze x e d_a . Di conseguenza è naturale che l'errore che si presenta nella stima di y sia più grande di quello visto per x , a cui è associato un sensore apposito.



θ

La componente θ dello stato risulta essere quella maggiormente afflitta da errori di stima.

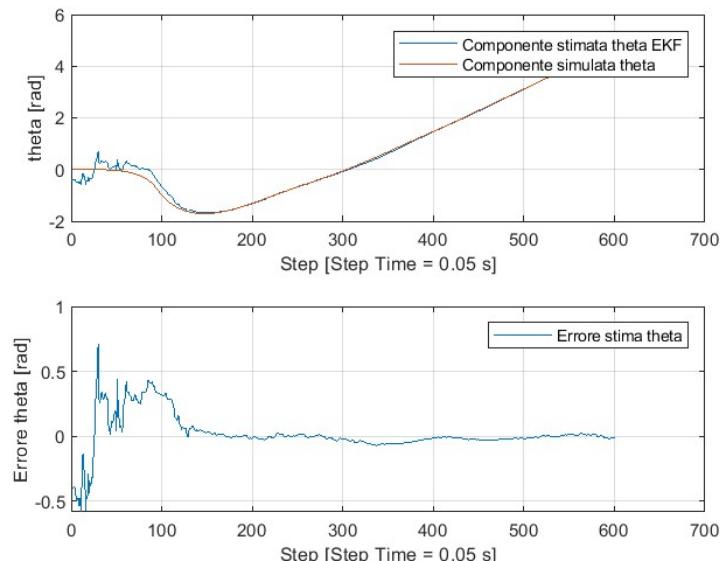
Si può notare infatti come nelle prime fasi la componente stimata sia molto diversa da quella vera, discostandosi anche fino a 0.7 rad, cioè circa 40 °.

La situazione migliora notevolmente quando il filtro è a regime, dove si vede che infatti l'errore si attesta a un valore prossimo a 0 e la traiettoria vera viene seguita bene.

Questo fenomeno è giustificabile con il fatto che, non avendo a disposizione un sensore per la misura diretta di θ , le misure delle altre componenti non riescono a correggerla indirettamente nei primi passi della simulazione .

A regime invece, quando le componenti misurate dai sensori vengono stimate quasi perfettamente, allora anche le componenti non direttamente misurabili come θ vengono corrette.

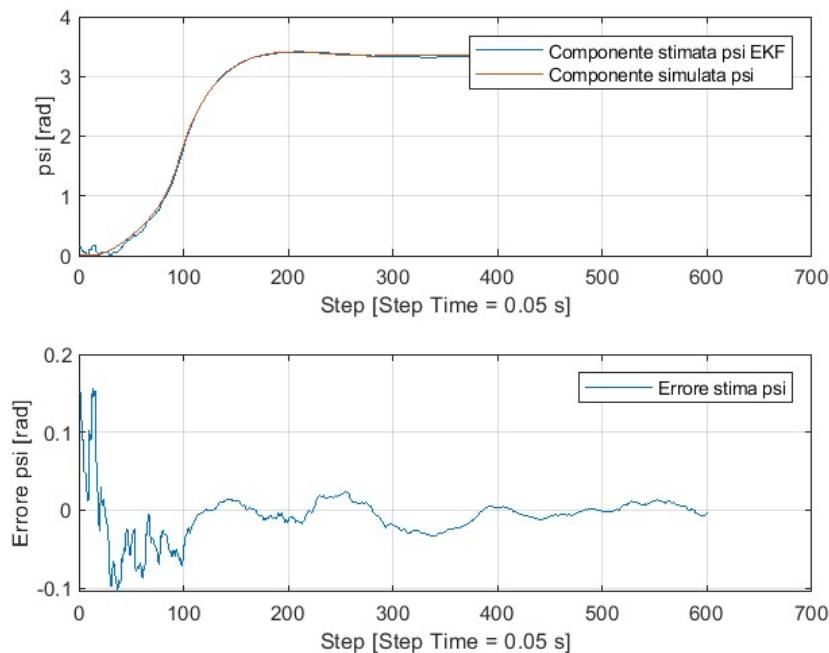
Per questa componente si raggiunge il regime dopo 7.5 secondi.



Ψ

Per quanto riguarda la componente ψ , c'è da dire che anche su di essa il filtro si comporta bene; infatti, il filtro va a convergenza in circa 10 secondi, portando l'errore di stima ad un valore prossimo a 0.

C'è da dire che rispetto alla componente $\dot{\phi}$, il valore di ψ soffre di più l'incertezza sullo stato iniziale. Infatti, nelle prime fasi di filtraggio, l'errore tocca anche picchi di 0.15 rad, ovvero all'incirca 8.6°. Pur non essendo un valore troppo grande provoca un'incertezza non trascurabile che fortunatamente viene ridotta significativamente dopo poche iterazioni.

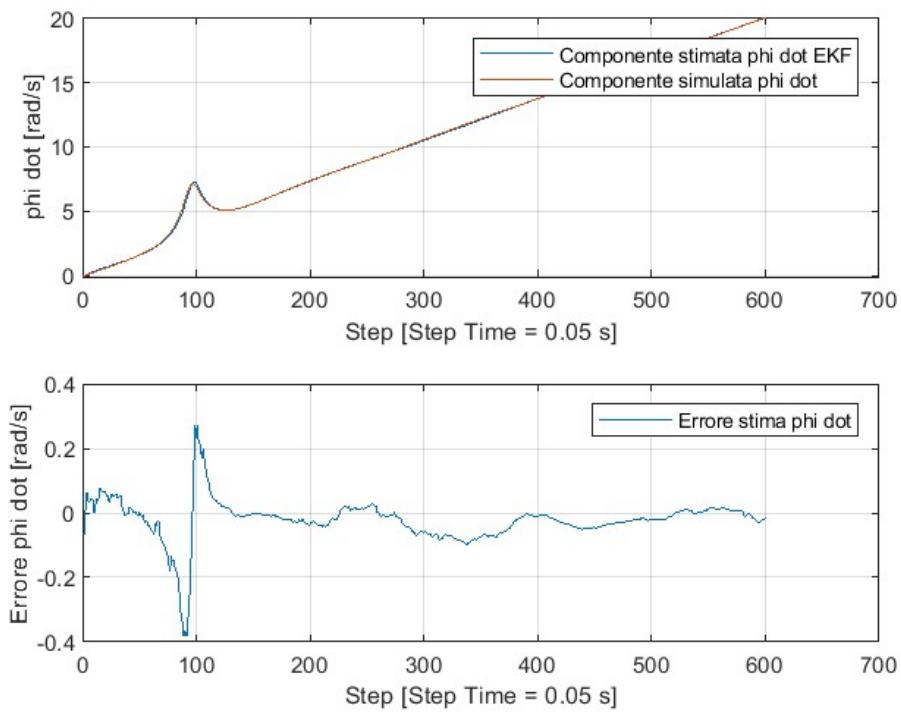


$\dot{\phi}$

La componente $\dot{\phi}$ risulta essere stimata bene dal filtro. Infatti, guardando il grafico relativo alla traiettoria stimata e quella simulata, si vede come le due siano pressoché identiche.

Questo è ancora più evidente nel grafico dove viene tracciato l'errore di stima, dove si vede che tolta una prima fase di convergenza, l'errore sulla componente non si discosta di molto da zero. La fase di convergenza dura circa 10 secondi.

Tra l'altro il grafico presenta un picco in corrispondenza della fase di inversione di marcia dell'AGV, la quale risulta critica per la stima. Tuttavia, anche in questo caso, l'errore non supera i 0.4 rad/s.

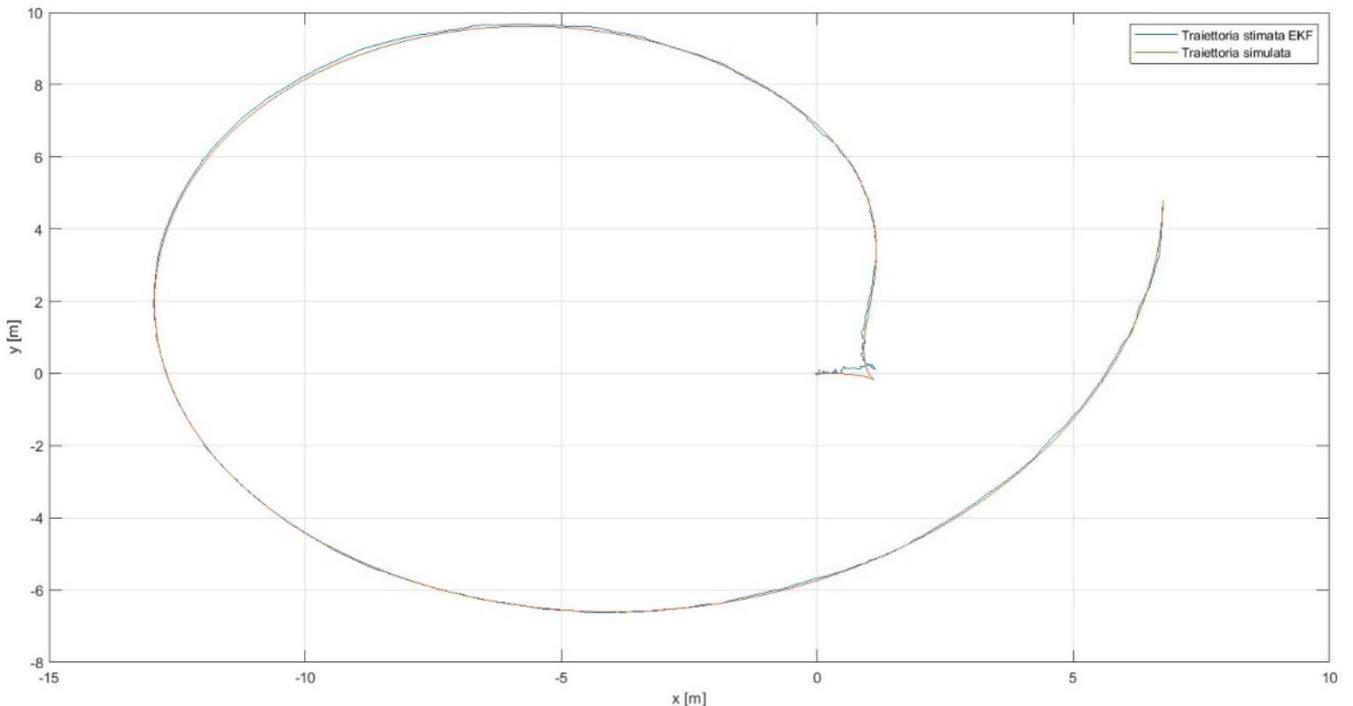


Analisi della traiettoria

Il grafico sottostante riassume i risultati ottenuti, in particolare per quanto riguarda il movimento nel piano dell'AGV.

Nel plot viene riportata la traiettoria compiuta dal muletto simulando il suo movimento con gli ingressi dati, e viene messa a confronto con la traiettoria stimata tramite il filtro EKF.

Come si può vedere, fatto salvo per una prima fase influenzata dalle incertezze sullo stato iniziale, la stima si comporta bene, andando a convergere con la traiettoria simulata.



Smoother RTS (Rauch-Tung-Striebel)

Introduzione teorica

Lo smoother è una tecnica di regolarizzazione della traiettoria composta da due fasi, una “in avanti” e una “all’indietro”, e si basa sulla ricostruzione dello stato utilizzando non solo i dati ai passi precedenti, ma anche a quelli successivi

$$\hat{x}_{k|n} \text{ con } n \geq k$$

Fase “in avanti”

Corrisponde di fatto all’applicazione di un Kalman Filter, o nel nostro caso un EKF, dal quale raccogliamo questi dati:

$$\hat{x}_{k|k}, \hat{x}_{k+1|k}, P_{k+1|k}, P_{k|k}, F_k$$

Fase “all’indietro”

Nella quale si effettua la regolarizzazione vera e propria:

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

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

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

Implementazione dello smoother

```
RTS_EKF_progetto.m
1 % Smoother per EKF
2 % Inizializzazione smoother
3 clc
4
5 stato_smooth(:,size(stato_hat,2))=stato_corr(:,size(stato_hat,2));
6 P_smooth(:, :, size(stato_hat,2))=P_corr(:, :, size(stato_hat,2));
7
8 % Ciclo smoother
9
10 for k=size(stato_hat,2)-1:-1:2
11
12     Ck=P_corr(:, :, k)*Fstore(:, :, k+1)'\inv(P_pred(:, :, k+1));
13
14     stato_smooth(:, k)=stato_corr(:, k)+Ck*(stato_smooth(:, k+1)-stato_pred(:, k+1));
15     P_smooth(:, :, k)=P_corr(:, :, k)+Ck*(P_smooth(:, :, k+1)-P_pred(:, :, k+1))*Ck';
16 end
17
```

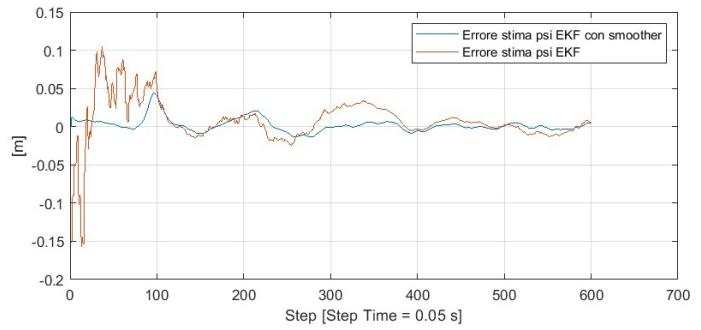
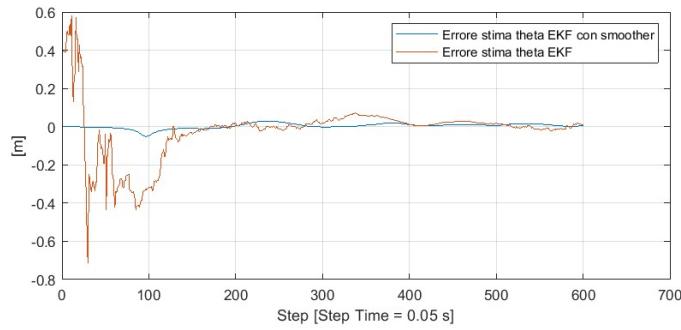
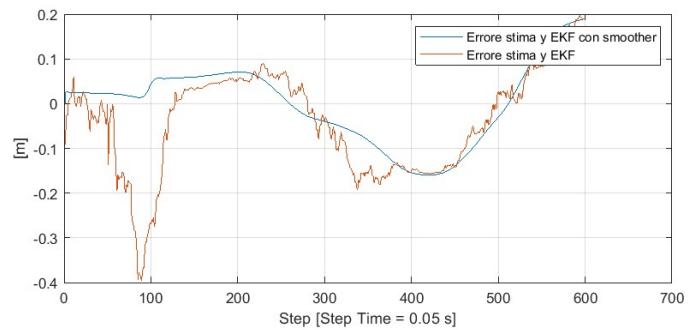
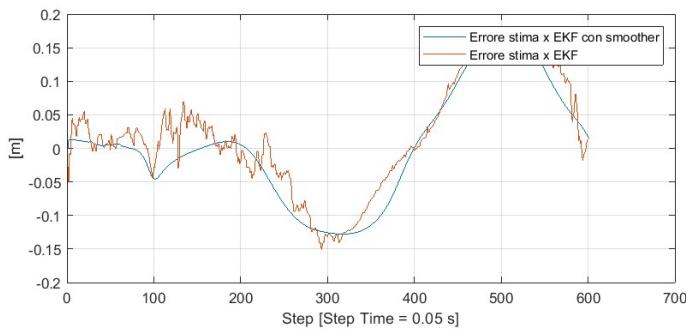
Per l’implementazione dello smoother usiamo un “ciclo for” con l’ indice k che scorre i valori nel tempo del vettore dello stato stimato partendo dal tempo finale andando a ritroso.

Il ciclo rappresenta solamente la fase “all’indietro”.

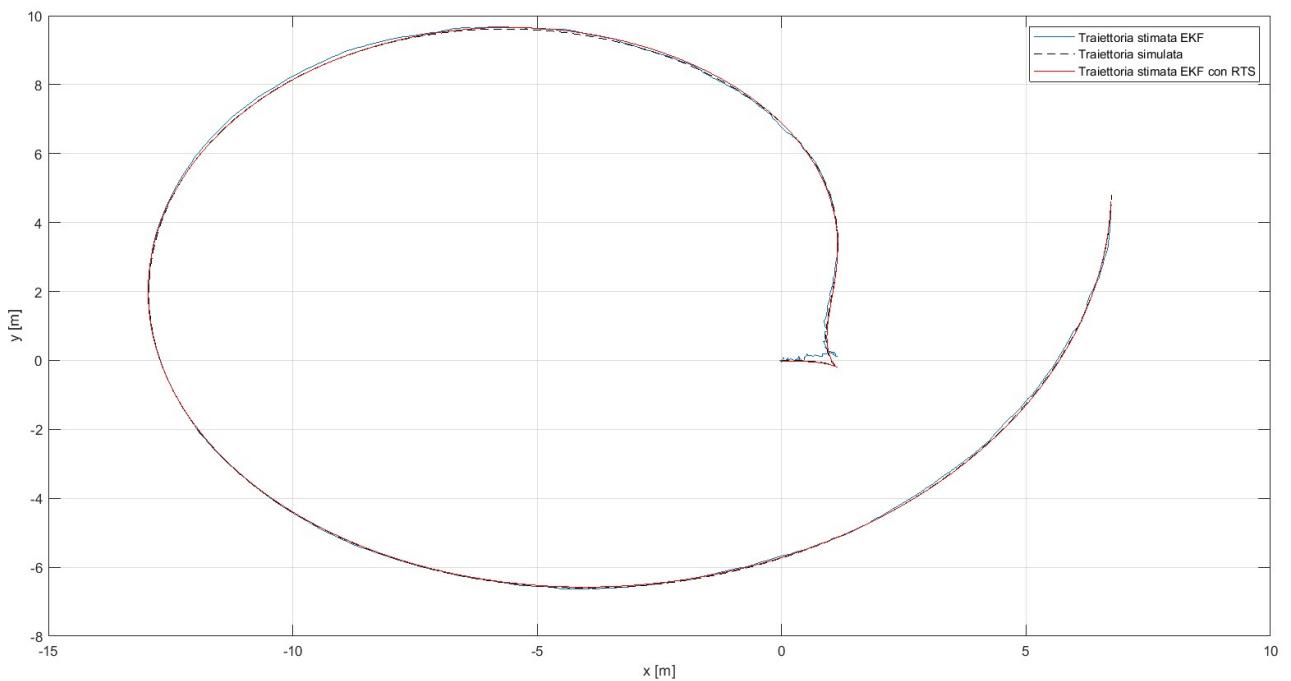
Valutazione dei risultati

Dopo l'applicazione dello smoother, la traiettoria risulta regolarizzata, ovvero vengono eliminati i picchi della stima che si manifestavano con l'applicazione del solo EKF. Inoltre, grazie alla fase "all'indietro", si riducono anche gli errori di stima nei primi passi di filtraggio.

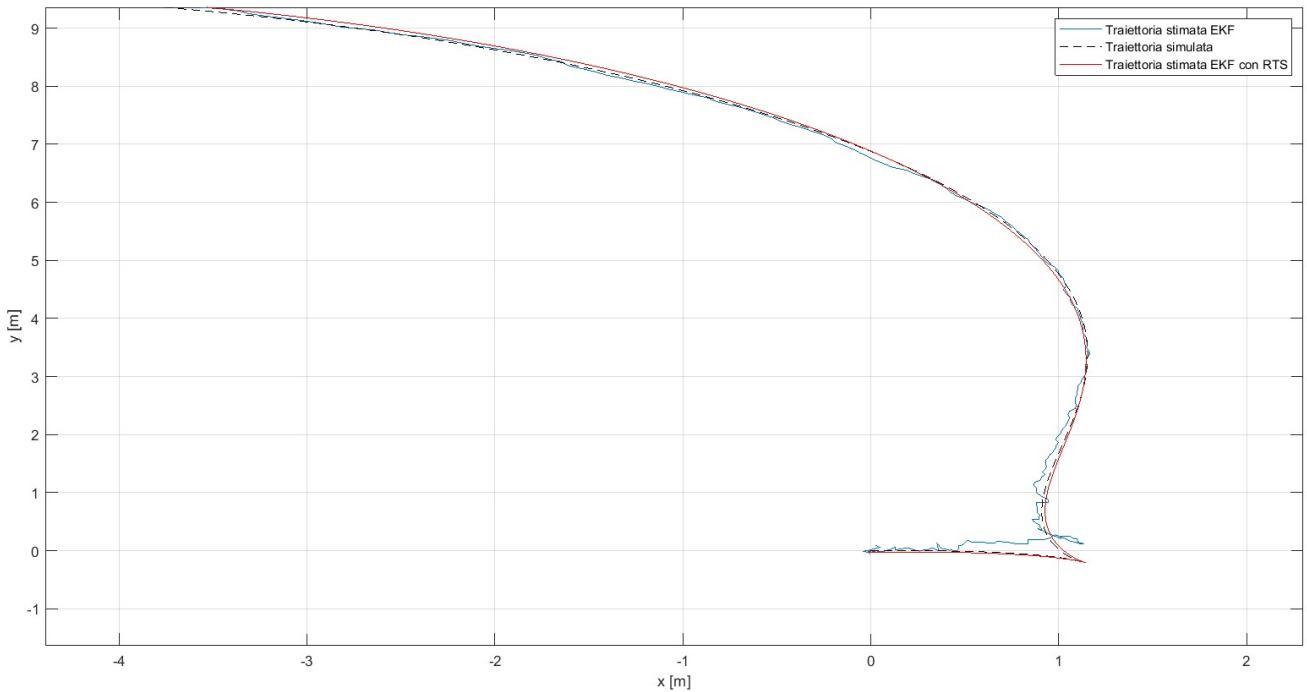
Possiamo vedere che per le 4 componenti rappresentate (x, y, θ, ψ) si ha errori di stima inferiori a quelli ottenuti dal solo EKF e si raggiunge la condizione di regime più velocemente per gli angoli.



La traiettoria ottenuta con lo smoother è quasi coincidente con quella simulata per la maggior parte dell' esperimento.



Si riporta di seguito un dettaglio della traiettoria ai primi istanti della simulazione per apprezzare meglio l'effetto della regolarizzazione.



UKF(Unscented Kalman Filter)

Un'altra variante del filtro di Kalman applicato a sistemi non lineari è il filtro UKF che, come suggerisce il nome stesso, è basato sulla trasformata Unscented.

Trasformata Unscented (caso generale)

Data una variabile aleatoria $Z = f(X)$, dove X è anch'essa variabile aleatoria, e $f(\cdot)$ è una funzione non lineare, la trasformata Unscented permette di ottenere un'approssimazione dei momenti del 1° e del 2° ordine (varianza e cross-covarianza rispetto a X) di Z a partire dai momenti di X .

Il vantaggio della trasformata Unscented rispetto alla linearizzazione con Taylor è che non vanno calcolate le derivate delle funzioni il gioco, che nel caso non lineare possono risultare molto complesse.

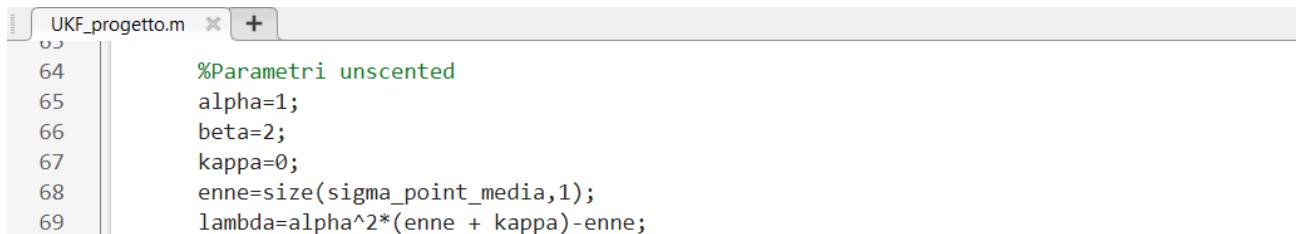
$$[\bar{z}, \Sigma_z, \Sigma_{xz}] = UT(\bar{x}, \Sigma_x, f(\cdot))$$

Procedura operativa

Di seguito verrà mostrato il codice Matlab relativo alla trasformata Unscented applicata in fase di predizione, ma l'implementazione è analoga in fase di correzione.

Il primo passo è la generazione dei "sigma-points", i quali sono un set di campioni di X , scelti in modo tale che la media e la varianza campionaria applicate a questi ultimi coincidano con quelle di X .

Il numero di sigma-points generati è pari a $2 * N + 1$, dove N è la dimensione della variabile X ; questi punti vengono scelti in modo tale da essere simmetrici rispetto alla media di X , cosa che permette di semplificare alcuni passaggi. Per fare ciò si parte da un set di parametri: *alpha*, *beta* e *kappa*, i cui valori dipendono dalla tipologia delle variabili aleatorie; questi ultimi vengono poi combinati in un ulteriore parametro *lambda* = $(\text{alpha}^2) * (N + \kappa) + N$.

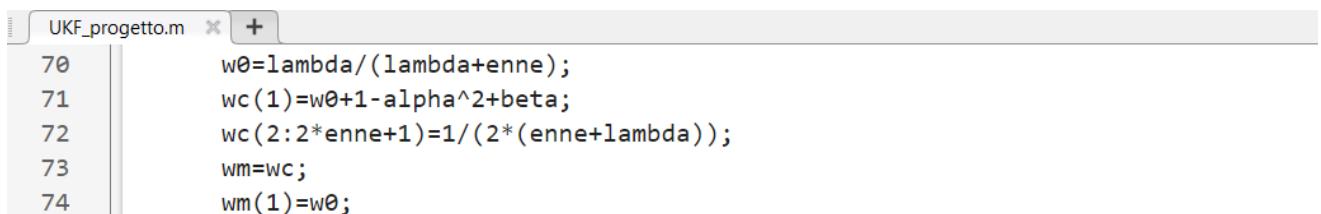


```

UKF_progetto.m
64 %Parametri unscented
65 alpha=1;
66 beta=2;
67 kappa=0;
68 enne=size(sigma_point_media,1);
69 lambda=alpha^2*(enne + kappa)-enne;

```

Vengono calcolati i pesi w_i che andranno a moltiplicare i termini all'interno della media e della varianza campionaria di Z .



```

UKF_progetto.m
70 w0=lambda/(lambda+enne);
71 wc(1)=w0+1-alpha^2+beta;
72 wc(2:2*enne+1)=1/(2*(enne+lambda));
73 wm=wc;
74 wm(1)=w0;

```

I pesi w_m (media campionaria) e w_c (varianza campionaria) hanno tutti lo stesso valore, tranne il termine centrale.

$$W_m(i) = W_c(i) = \frac{1}{2(n + lambda)} \text{ con } i \neq 0$$

$$W_m(0) = \frac{lambda}{n + lambda}$$

$$W_c(0) = W_m(0) + 1 + alpha^2 + beta$$

Si procede determinando una fattorizzazione della matrice di covarianza di X

($\Sigma_x = \text{GAMMA} * \text{GAMMA}^\top$, con GAMMA ottenuta a partire dalla SVD di Σ_x); a questo punto i sigma-points possono essere generati nel seguente modo:

```
UKF_progetto.m
75
76     %Generazione sigma points
77     [U,S,~]=svd(Pcov);
78     GAMMA=U*S^(1/2);
79     sigma_points(:,1)=sigma_point_media;
80
81     for i1=1:enne
82         sigma_points(:,i1+1)=sigma_point_media+sqrt(enne+lambda)*GAMMA(:,i1);
83         sigma_points(:,i1+1+enne)=sigma_point_media-sqrt(enne+lambda)*GAMMA(:,i1);
84     end
```

I sigma-points vengono quindi propagati utilizzando la funzione $f(\cdot)$, creando un nuovo set di punti Z_i , grazie a quali vengono approssimate media, covarianza e cross-covarianza di Z.

```
UKF_progetto.m
88
89     %passaggio necessario per evitare di sbagliarsi con gli indici
90     x = sigma_points(1,i1);
91     y = sigma_points(2,i1);
92     theta = sigma_points(3,i1);
93     psi = sigma_points(4,i1);
94     psi_dot = sigma_points(5,i1);
95     phi_dot = sigma_points(6,i1);
96     w_phi = sigma_points(7,i1);
97     w_psi = sigma_points(8,i1);
98
99     tau_psi = out.tausim(i,2);
100    tau_phi = out.tausim(i,1);

102
103    %propagazione
104    propag_sigma_points(1,i1)= x + dt*phi_dot*rp*cos(psi)*cos(theta);
105    propag_sigma_points(2,i1)= y + dt*phi_dot*rp*cos(psi)*sin(theta);
106    propag_sigma_points(3,i1)= theta - (dt*phi_dot*rp*sin(psi))/L;
107    propag_sigma_points(4,i1)= psi + dt*psi_dot;
108    propag_sigma_points(5,i1)= psi_dot + dt*((tau_psi + w_psi + (Ipz*phi_dot*psi_dot*rp*cos(psi))/L)*(a*
109    propag_sigma_points(6,i1)= phi_dot + dt*((L^2*(tau_phi + w_phi + phi_dot*psi_dot*cos(psi)*sin(psi)*(a
```

```

119
120    %Media propagata
121    propag_media=zeros(6,1);
122    propag_tilde=zeros(6,2*enne+1);
123    propag_var=zeros(6);
124
125    for i1=1:(2*enne+1)
126        propag_media=propag_media+wm(i1)*propag_sigma_points(:,i1);
127    end

```



```

135
136    % Calcolo della differenza tra sigma points propagati e la media propagata
137    for i1=1:(2*enne+1)
138        propag_tilde(:,i1)=propag_sigma_points(:,i1)-propag_media;
139    end
140
141    %Varianza Propagata
142    for i1=1:(2*enne+1)
143        propag_var=propag_var+wc(i1)*propag_tilde(:,i1)*propag_tilde(:,i1)';
144    end
145    stato_ukf(:,i1+1)=propag_media;
146    Pk=propag_var;

```

Fase di predizione

Essendo la dinamica dell'AGV funzione sia dello stato $x_{k|k}$ che del rumore w_k , per poter applicare la trasformata Unscented si considera una variabile aleatoria congiunta che comprende sia $x_{k|k}$ che w_k ; questa variabile avrà come valor medio il vettore delle medie di stato e rumore e come covarianza una matrice diagonale a blocchi con le covarianze di stato e del rumore, che per ipotesi sono indipendenti. La trasformata Unscented avrà quindi come ingresso media e covarianza della variabile congiunta e funzione di transizione di stato dell'AGV, e come uscita lo stato predetto $x_{k+1|k}$ la covarianza predetta $P_{k+1|k}$ (non serve calcolare il momento incrociato).

$$[\hat{x}_{k+1|k}, P_{k+1|k}] = UT \left(\begin{bmatrix} \hat{x}_{k|k} \\ 0 \end{bmatrix}, \begin{bmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{bmatrix}, f_k(\cdot) \right)$$

Fase di correzione

Per quanto riguarda la correzione, il rumore di misura è additivo, quindi non è necessario costruire una variabile congiunta che tiene conto anche di esso; si può applicare la trasformata Unscented direttamente prendendo come ingresso lo stato e la matrice di covarianza calcolati in fase di predizione e il modello di osservazione $h(\cdot)$, ottenendo come uscita la misura predetta $y_{k|k-1}$, la matrice di covarianza associata S_k (alla quale andrà sommata la covarianza del rumore), e la matrice di covarianza $P_{k|k-1}^{xy}$.

$$[\hat{y}_{k|k-1}, S_k, P_{xy\ k|k-1}] = UT \left(\hat{x}_{k|k-1}, P_{k|k-1}, h_k(\cdot) \right)$$

```
UKF_progetto.m
281 %Varianza e Covarianza
282 propag_varH=zeros(dimy);
283 propag_xcovH=zeros(6,dimy);
284
285 for i2=1:(2*enne+1)
286     propag_varH=propag_varH+wc(i2)*propag_tildeH(:,i2)*propag_tildeH(:,i2)';
287     propag_xcovH=propag_xcovH+wc(i2)*(sigma_pointsH(:,i2)-sigma_point_mediaH)*propag_tildeH(:,i2)';
288 end
```

Con la stessa procedura impiegata nel filtro EKF si determinano quali sensori forniscono una misura all'iterazione corrente, così da ridimensionare coerentemente le uscite della trasformata Unscented. Infine, si calcolano l'innovazione e_k e il guadagno di correzione L_k , e si vanno a correggere lo stato e la matrice di covarianza determinati in fase di predizione.

$$\begin{aligned}e_k &= y - \hat{y}_{k|k-1} \\L_k &= P_{xy\ k|k-1} \cdot S_k^{-1} \\\hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k \cdot (y - \hat{y}_{k|k-1}) \\P_{k|k} &= P_{k|k-1} - L_k \cdot S_k \cdot L_k^T\end{aligned}$$

```
UKF_progetto.m
200
201 %gestione phi_dot
202
203 if(flag_phi_dot==0)
204     if(out.phi_dot<=tempo_corr)
205         dimy=dimy+1;
206         Vdev(dimy)=std_dev_phi_dot^2;
207         Y(dimy)=out.phi_dot;
208         propag_mediaH(dimy)=propag_mediaH(1);
209         propag_tildeH(dimy,:)=propag_tildeH(1,:);
210         cont_phi_dot=cont_phi_dot+1;
211         e(dimy)=Y(dimy)-propag_mediaH(dimy);
212     end
213 end
214
215 %gestione psi
216
217 if(flag_psi==0)
218     if(out.psim<=tempo_corr)
219         dimy=dimy+1;
220         Vdev(dimy)=std_dev_psi^2;
221         Y(dimy)=out.psim;
222         propag_mediaH(dimy)=propag_mediaH(2);
223         propag_tildeH(dimy,:)=propag_tildeH(2,:);
224         cont_psi=cont_psi+1;
225         e(dimy)=atan2(sin(Y(dimy)-propag_mediaH(dimy)),cos(Y(dimy)-propag_mediaH(dimy)));
226     end
227 end
```

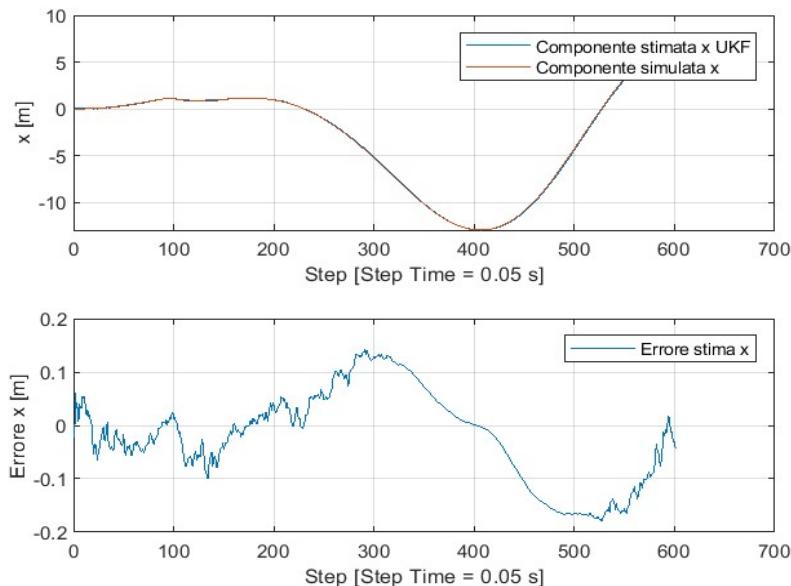
```
UKF_progetto.m
273
274 S=propag_varH;
275 S=S+R2;
276 LH=propag_xcovH*inv(S);
277
278 %aggiorno lo stato corretto e Pk
279 stato_ukf(:,i+1)=stato_ukf(:,i+1)+LH*e(1:dimy)';
280 Pk=Pk-LH*S*LH';
```

Valutazione dei risultati

Verranno ora analizzati i risultati ottenuti con il filtro UKF.

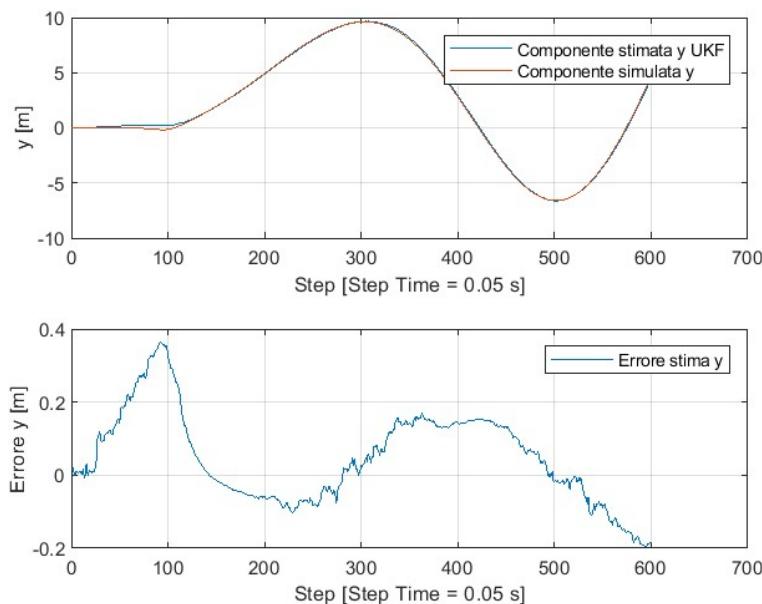
X

Dal grafico si può osservare che la stima della componente X dello stato risulta coerente con i dati ottenuti dalla simulazione dell'AGV. Nonostante non converga a zero, l'errore massimo commesso dallo stimatore UKF in valore assoluto è minore di 20 cm, che se confrontato con le dimensioni del veicolo e con la lunghezza della traiettoria risulta accettabile.



Y

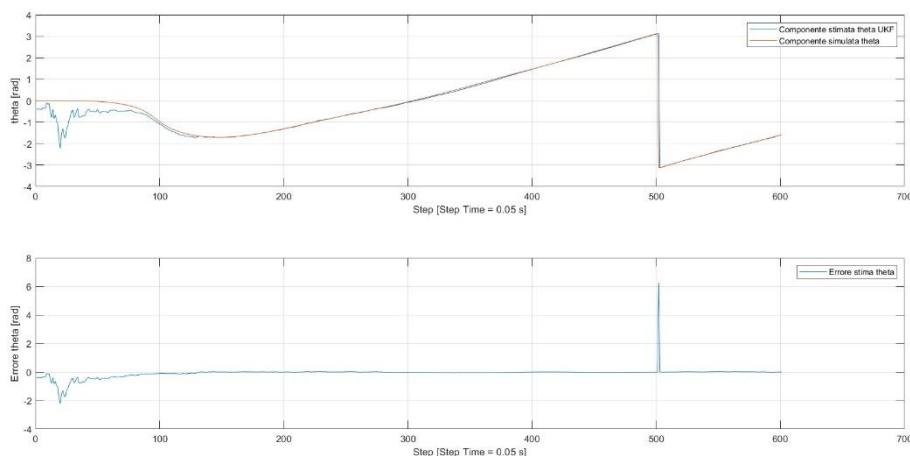
Le stesse considerazioni possono essere fatte sulla componente Y dello stato; in questo caso l'errore massimo di stima si trova nella parte iniziale del grafico (circa 5 secondi) e non supera i 40 cm. Questo comportamento è spiegato dalla mancanza di un sensore diretto e dall'influenza della componente θ .



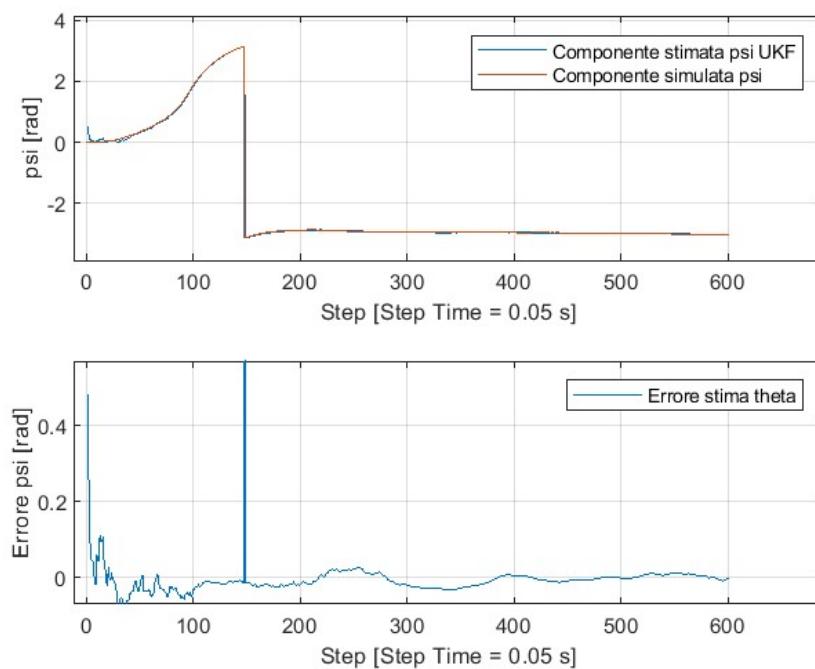
θ

Come è stato possibile osservare nel caso dello stimatore EKF, anche qui il grafico di θ presenta un tratto iniziale afflitto da errori maggiori, per poi migliorare con il proseguire della stima. La presenza di questo errore su θ potrebbe spiegare come mai la stima di Y risulti meno precisa nella fase iniziale: se si osserva la funzione di transizione dello stato, sia X che Y dipendono, oltre che da ψ e $\dot{\phi}$, da θ ; tuttavia, mentre X viene corretta usando la misura diretta di un sensore, Y viene corretta in maniera indiretta, e di conseguenza risente maggiormente dell'errore di θ .

Rispetto al caso EKF, inoltre, qui è presente una discontinuità intorno ai 25 s; questa discontinuità è dovuta all'utilizzo della funzione "atan2", utilizzata nel calcolo delle medie degli angoli per eliminare i problemi dovuti alla periodicità.

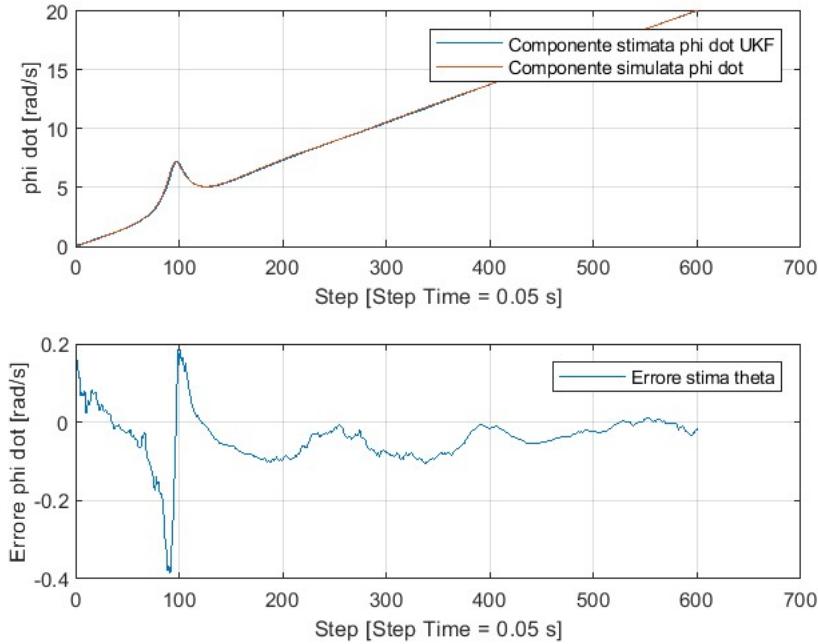
 **Ψ**

A differenza di θ , la stima di ψ risulta molto coerente con la simulazione, con un errore massimo (escludendo il picco in corrispondenza della discontinuità dovuta all'utilizzo di "atan2") di 0.5 rad (30°) nella parte iniziale del grafico.



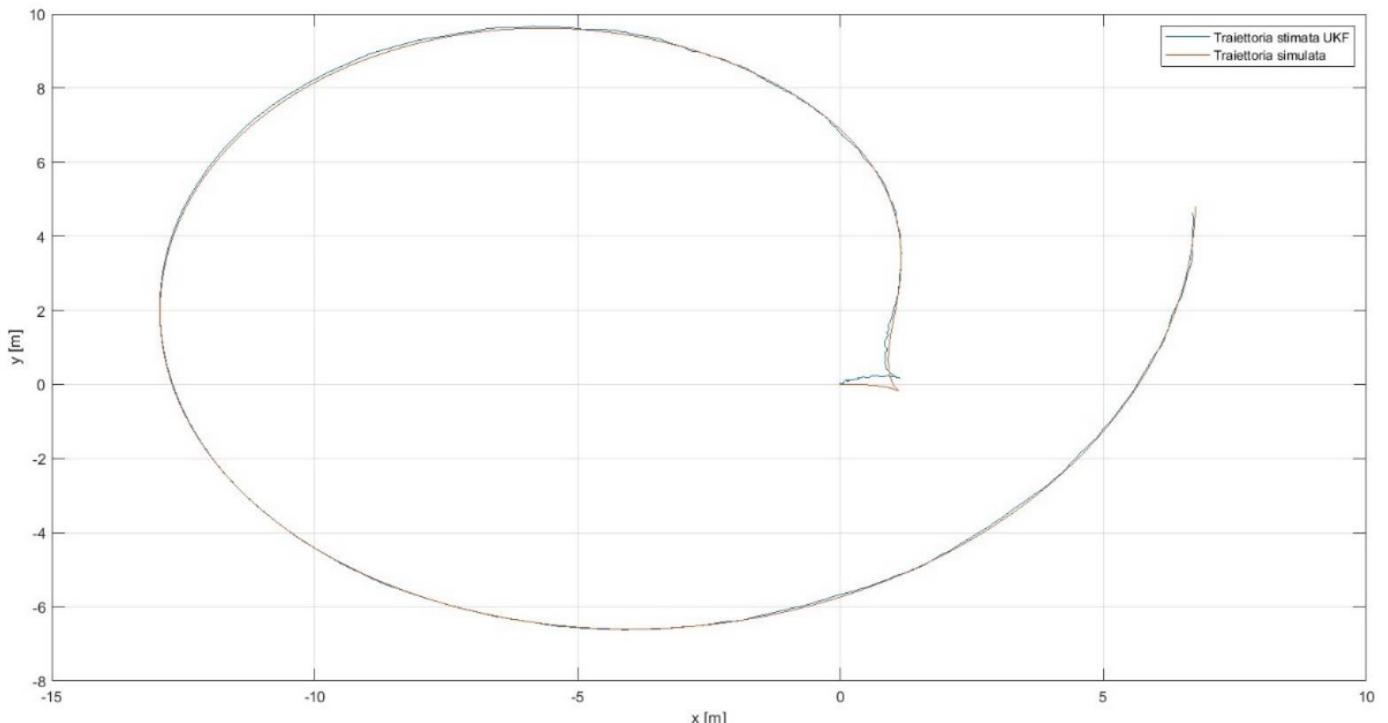
$\dot{\phi}$

Anche nel caso di $\dot{\phi}$, la stima risulta coerente con la simulazione. Analogamente al filtro EKF, l'errore massimo si ha in corrispondenza dell'inversione del senso di marcia della ruota ed è pari a 0.4 rad/s



Analisi della traiettoria

In questo grafico sono messe a confronto la traiettoria stimata con il filtro UKF e la traiettoria "reale" del veicolo. Facendo riferimento a quanto visto nei grafici di X e Y, si può osservare l'effetto degli errori nella parte iniziale della traiettoria stimata. Dopo di che le traiettorie vanno a coincidere.



Confronto EKF e UKF

Si confronta i filtri utilizzati per vedere il migliore nel caso studiato.

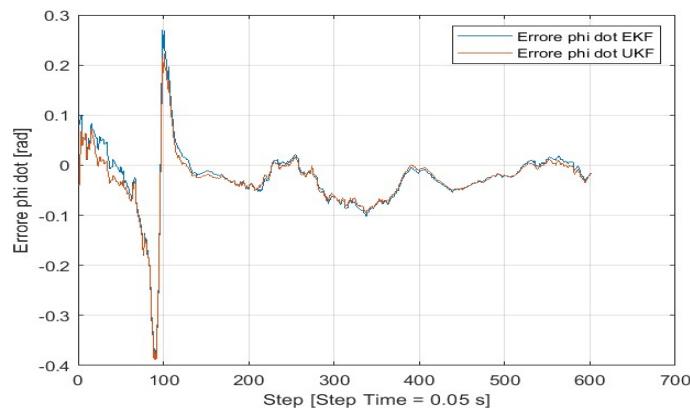
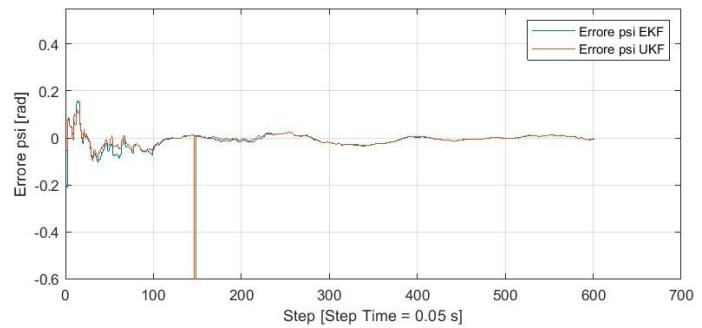
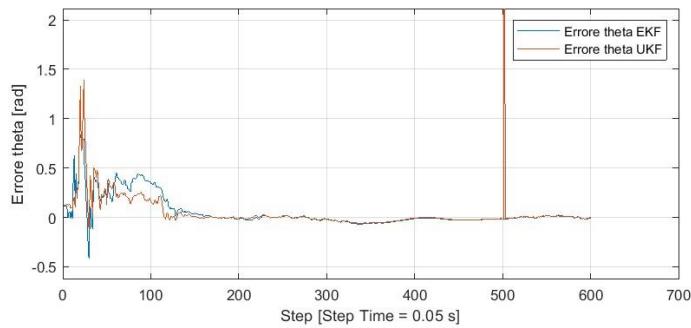
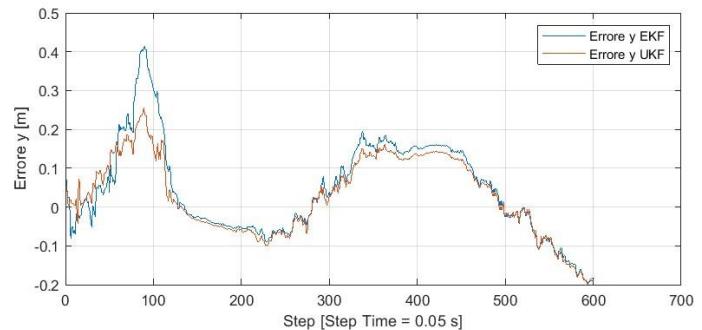
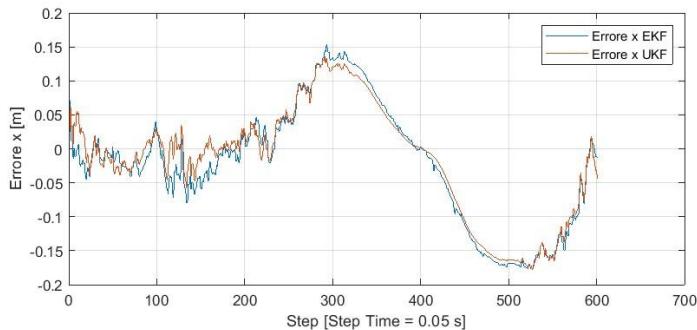
Errore di stima

Come si può osservare dai grafici, le stime ottenute dal filtro EKF e dal filtro UKF presentano errori molto simili in tutte le componenti dello stato.

Le uniche differenze che possiamo notare sono le seguenti:

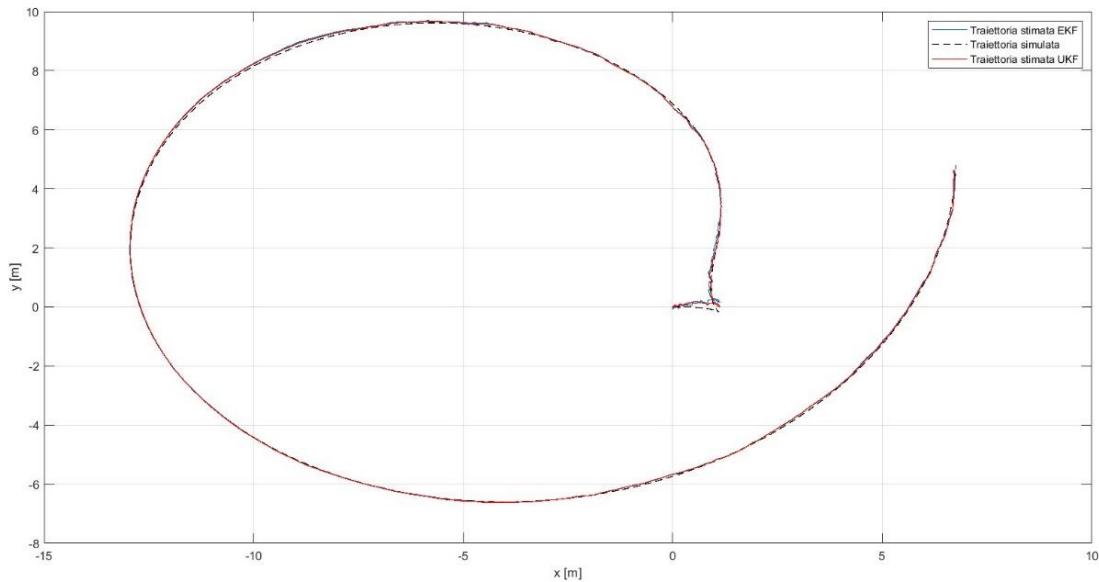
- stima di y nelle prime iterazioni, dove l'UKF si comporta generalmente meglio
- stima delle componenti angolari (θ e ψ), dove sono presenti i picchi, che non sono errori veri e propri, dovuti all'utilizzo della funzione "atan2".

Quindi per avere una migliore stima della traiettoria conviene utilizzare un filtro UKF, che fornisce errori minori sulle prime 3 componenti dello stato.

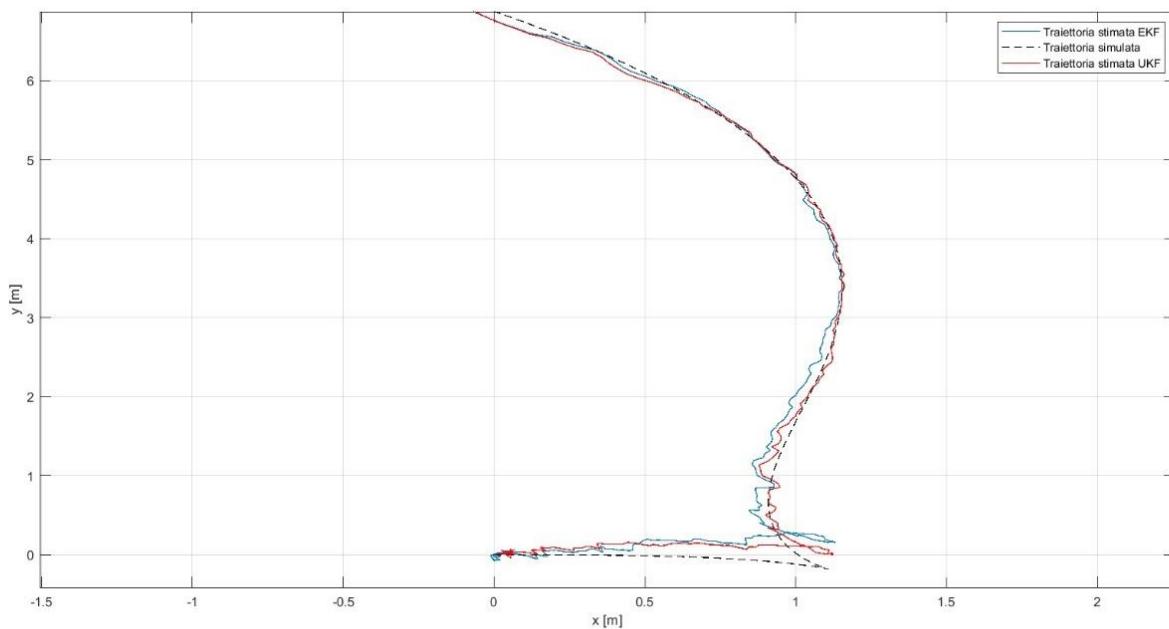


Traiettoria

Per riassumere si riporta il grafico della traiettoria simulata e le traiettorie stimate per rimarcare gli errori di stima.



Si riporta di seguito un dettaglio della traiettoria ai primi istanti della simulazione per apprezzare meglio l'effetto le differenze fra i filtri.

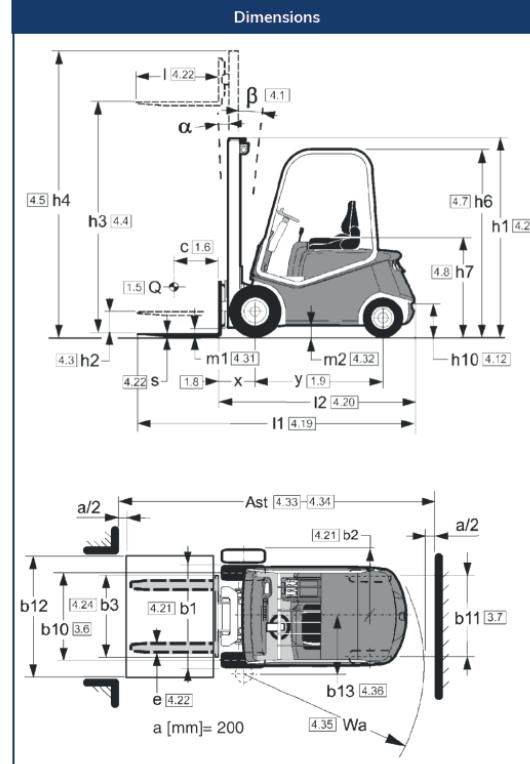


Non viene fatto il confronto fra UKF e EKF con smoother perché la stima del secondo è più raffinata a causa del maggior numero di osservazioni e perché non è un filtro che può essere implementato in tempo reale.

Tabelle e datasheet

Muletto CESAB BLITZ 420

Specifications			Dimensions
1.1	Casa costruttore	CESAB	
1.2	Modello	BLITZ 420	
1.3	Gruppo propulsore: elettrico (batteria), diesel, benzina, GPL	elettrico	
1.4	Guida: a mano, a piedi, in piedi, seduto	seduto	
1.5	Portata	Q (kg)	
1.6		2000	
1.7	Barcenetro	c (mm)	
1.8	Distanza carico	x (mm)	4.1
1.9		y (mm)	4.2
2.1	Interasse	1532	
2.2	Peso	kg	352 (a)
2.3	Carico sugli assali con carico ant./post.	kg	4940 / 580
2.4	Carico sugli assali senza carico ant./post.	kg	1820 / 1700
3.1	Gommatura: C=Cushion, SE=Superalastic, PN=Pneumatici, G=Gommati	C - SE	
3.2	Dimensioni gommatura anteriore	457x178 - 200/50-10	
3.3	Dimensioni gommatura posteriore	381x127 - 16x68	
3.5	Ruote: numero ant./post. (x = motrice)	2x / 2	
3.6	Careggiastra anteriore	b10 (mm)	843 - 866
3.7	Careggiastra posteriore	b11 (mm)	863 - 843
4.1	Brando: avanti / indietro	α / β (gradi)	2°30' / 6'
4.2	Altezza minimo ingombro	h1 (mm)	2160
4.3	Alzata libera	h2 (mm)	80
4.4	Corsa di sollevamento	h3 (mm)	3170
4.5	Altezza massimo ingombro	h4 (mm)	3720
4.7	Altezza protezione condutture	h6 (mm)	1950
4.8	Altezza sedile	h7 (mm)	888
4.12	Altezza gancio	h10 (mm)	630
4.19	Lunghezza totale	l1 (mm)	3235,5 (b)
4.20	Lunghezza incluso dorso forche	l2 (mm)	2235,5 (b)
4.21	Lunghezza totale	b1/b2 (mm)	1006 - 1066/N0
4.22	Dimensioni forche	s/e/l (mm)	35 x 120 x 1000
4.23	Piastra porta forche DIN 15173, classe / tipo A, B		II A
4.24	Lunghezza piastra porta forche	b3 (mm)	900
4.31	Altezza libera sotto il montante, a carico	m1 (mm)	100
4.32	Altezza libera telai al centro, a carico	m2 (mm)	90
4.33	Corridio di stivaggio con pallet 1000 x 1200 inforc. 1200	Ast (mm)	3564
4.34	Corridio di stivaggio con pallet 800 x 1200 inforc. 800	Ast (mm)	3761
4.35	Raggio di curvatura	Wa (mm)	1995



Encoder Ratings and Specifications

Item	Model	E6C3-CWZ5GH	E6C3-CWZ3EH	E6C3-CWZ3XH
Power supply voltage		12 VDC -10% to 24 VDC +15%, ripple (p-p): 5% max.	5 VDC -5% to 12 VDC +10%, ripple (p-p): 5% max.	
Current consumption*1		100 mA max.		
Resolution (pulses/rotation)		100, 200, 300, 360, 500, 600, 720, 800, 1,000, 1,024, 1,200, 1,500, 1,800, 2,000, 2,048, 2,500, 3,600		
Output phases		Phases A, B, and Z ⁵	Phases A, \overline{A} , B, \overline{B} , Z, and \overline{Z}	
Output configuration		Complementary outputs ²	Voltage output (NPN output)	Line driver output ³
		Output voltage: VH = Vcc - 3 V min. (IO = 30 mA) VL = 2 V max. (IO = -30 mA)	Output resistance: 2 k Ω Output current: 35 mA max. Residual voltage: 0.7 V max.	AM26LS31 equivalent Output current: High level: IO = -10 mA Low level: IS = 10 mA Output voltage: VO = 2.5 V min. VS = 0.5 V max.
Output capacity		Output current: ± 30 mA		
Maximum response frequency ⁴		125 kHz (65 kHz when using phase Z reset)		
Phase difference between outputs		$90^\circ \pm 45^\circ$ between A and B ($1/4 T \pm 1/8 T$)		
Rise and fall times of output		1 μ s max. (Cable length: 2 m, Output current: 30 mA)	1 μ s max. (Cable length: 2 m, Output current: 35 mA)	1 μ s max. (Cable length: 2 m, IO: -10 mA, IS: 10 mA)
Starting torque		10 mN·m max. at room temperature, 30 mN·m max. at low temperature		
Moment of inertia		2.0×10^{-6} kg·m ² max.; 1.9×10^{-6} kg·m ² max. at 500 P/R max.		
Shaft loading Radial	Radial	80 N		
Shaft loading Thrust	Thrust	50 N		
Maximum permissible speed		5,000 r/min		
Protection circuits		Power supply reverse polarity protection, Output load short-circuit protection		---
Ambient temperature range		Operating: -10 to 70°C (with no icing), Storage: -25 to 85°C (with no icing)		
Ambient humidity range		Operating/Storage: 35% to 85% (with no condensation)		
Insulation resistance		20 M Ω min. (at 500 VDC) between current-carrying parts and case		
Dielectric strength		500 VAC, 50/60 Hz for 1 min between current-carrying parts and case		
Vibration resistance		Destruction: 10 to 500 Hz, 150 m/s ² or 2-mm double amplitude for 11 min 3 times each in X, Y, and Z directions		
Shock resistance		Destruction: 1,000 m/s ² 3 times each in X, Y, and Z directions		
Degree of protection		IEC 60529 IP65, in-house standards: oilproof		
Connection method		Pre-wired Models (Standard cable length: 1 m)		
Material		Case: Aluminum, Main unit: Aluminum, Shaft: SUS303		
Weight (packed state)		Approx. 300 g		
Accessories		Instruction manual Note: Coupling, mounting bracket and hex-head spanner are sold separately.		

Giroscopio

SPECIFICATIONS

All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed. $T_A = -40^\circ\text{C}$ to $+105^\circ\text{C}$, $V_s = AV_{CC} = V_{DD} = 5 \text{ V}$, $V_{RATIO} = AV_{CC}$, angular rate = $0^\circ/\text{sec}$, bandwidth = 80 Hz ($C_{OUT} = 0.01 \mu\text{F}$), $I_{OUT} = 100 \mu\text{A}$, $\pm 1 \text{ g}$, unless otherwise noted.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSITIVITY¹					
Measurement Range ²	Clockwise rotation is positive output Full-scale range over specifications range	± 300	6	6.48	$^\circ/\text{sec}$
Initial and Over Temperature	-40°C to $+105^\circ\text{C}$	5.52	± 2		$\text{mV}^\circ/\text{sec}$
Temperature Drift ³	Best fit straight line		0.1		%
Nonlinearity					% of FS
NULL⁴					
Null	-40°C to $+105^\circ\text{C}$	2.2	2.5	2.8	V
Linear Acceleration Effect	Any axis		0.1		$^\circ/\text{sec/g}$
NOISE PERFORMANCE					
Rate Noise Density	$T_A \leq 25^\circ\text{C}$		0.05		$^\circ/\text{sec}/\sqrt{\text{Hz}}$
FREQUENCY RESPONSE					
Bandwidth ⁴		0.01		2500	Hz
Sensor Resonant Frequency		12	14.5	17	kHz
SELF-TEST⁵					
ST1 RATEOUT Response	ST1 pin from Logic 0 to Logic 1	-650	-450	-250	mV
ST2 RATEOUT Response	ST2 pin from Logic 0 to Logic 1	250	450	650	mV
ST1 to ST2 Mismatch ⁵		-5		+5	%
Logic 1 Input Voltage		3.3			V
Logic 0 Input Voltage				1.7	V
Input Impedance	To common	40	50	100	kΩ
TEMPERATURE SENSOR¹					
V_{OUT} at 25°C	Load = $10 \text{ M}\Omega$	2.35	2.5	2.65	V
Scale Factor ⁶	@ 25°C , $V_{RATIO} = 5 \text{ V}$		9		mV°/C
Load to V_s			25		kΩ
Load to Common			25		kΩ
TURN-ON TIME	Power on to $\pm 1/2^\circ/\text{sec}$ of final			50	ms
OUTPUT DRIVE CAPABILITY					
Current Drive	For rated specifications			200	μA
Capacitive Load Drive				1000	pF
POWER SUPPLY					
Operating Voltage (V_s)		4.75	5.00	5.25	V
Quiescent Supply Current		3.5	4.5		mA
TEMPERATURE RANGE					
Specified Performance		-40		+105	°C

¹ parameter is linearly ratiometric with V_{RATIO} .² The maximum range possible, including output swing range, initial offset, sensitivity, offset drift, and sensitivity drift at 5 V supplies.³ From $+25^\circ\text{C}$ to -40°C or from $+25^\circ\text{C}$ to 105°C .⁴ Adjusted by external capacitor, C_{OUT} . Reducing bandwidth below 0.01 Hz does not reduce noise further.⁵ Self-test mismatch is described as $(ST2 + ST1)/(ST2 - ST1)/2$.⁶ For a change in temperature from 25°C to 26°C , V_{TEMP} is ratiometric to V_{RATIO} . See the Temperature Output and Calibration section for more details.

Lidar

Specifications	
Laser:	<ul style="list-style-type: none"> • Class 1 - eye safety • 905 nm wavelength • Time of flight distance measurement with Calibrated Reflectivities • Measurement range 1m to typically 80–100m
Sensor:	<ul style="list-style-type: none"> • 32 laser/detector pairs • $+10.67$ to -30.67 degrees field of view (vertical) • 360° field of view (horizontal) • 10 Hz frame rate (user selectable, 5-20Hz) • Operating temperature -10° to $+60^\circ\text{ C}$ • Storage temperature -40° to 105° C • Accuracy: <2 cm (one sigma at 25 m) • Angular resolution (vertical) 1.33° • Integrated web server for easy configuration
Mechanical:	<ul style="list-style-type: none"> • Power: 12V @ 1 Amps • Operating voltage: 9-32 VDC • Weight: HDL-32E = 1kg (2.2lbs) ; Cables = 0.3kg (0.62lbs) • Dimensions: 5.9" height x 3.4" diameter • Shock: 500 m/sec² amplitude, 11 msec duration • Vibration: 5 Hz to 2000 Hz, 3G rms • Environmental Protection: IP67
Output:	<ul style="list-style-type: none"> • Up to 700,000 points/second • 100 Mbps Ethernet connection • UDP packets containing <ul style="list-style-type: none"> - distances - calibrated reflectivities - rotation angles • Orientation - internal MEMS accelerometers and gyros for six-axis motion correction • GPS time-synchronized with included GPS Receiver

Bibliografia

- “System identification: theory for user”, Lennart Ljung (Prentice Hall)
- “Estimation with applications to tracking and navigation”, Bar-Shalom et al. (Wiley & Sons)
- Appunti del corso di Identificazione di Sistemi Incerti (anno accademico 2022/2023)

Sitografia

- Dimensioni muletto: www.cesab-forklifts.eu
- Giroscopio: www.analog.com
- Encoder: www.industrial.omron.it
- Lidar: www.velodynelidar.com

Programmi di calcolo

- Matlab
- Ambiente Simulink in Matlab
- Documentazione di Matlab: www.mathworks.com/help/matlab/

Codici matlab

- Cartella Google Drive per codici Matlab:
<https://drive.google.com/drive/folders/1GLNnIBhzqIZai16j6lTuSRAoLToOykhU?usp=sharing>