UNIVERSITÀ DI PISA

Progetto di Sistemi Subacquei:

Deep PU.R.P.L.E. Wall-Following

Navigation System

Professori:

Andrea Caiti

Riccardo Costanzi

Candidati:

Franca Corradini

Ilaria Rosati

Marco Borraccino

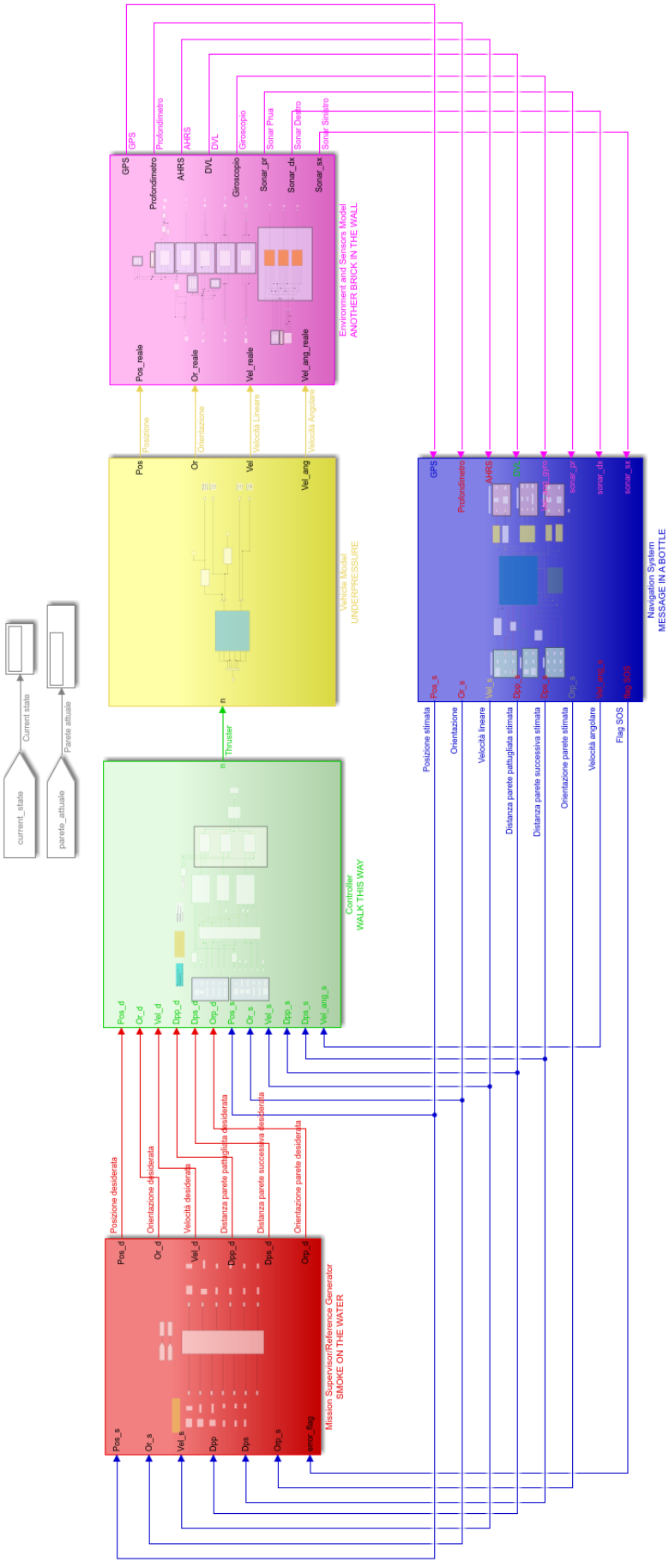
INDICE

1. INTRODUZIONE

L'obiettivo del progetto **Wall Following** consiste nell'implementazione tramite software di un sistema AUV (*Autonomous Underwater Vehicle*) che effettui il pattugliamento delle quattro pareti laterali di un bacino rettangolare, di cui sono noti gli spigoli. Il lavoro è stato organizzato tramite la suddivisione della mission in cinque gruppi, ognuno dei quali si è occupato della realizzazione di un task specifico. Dopo molta indecisione il team ha anche stabilito il nome dell'AUV "Deep PU.R.P.L.E. " (PUffer Robot Poll Levee Evaluation). Un nome che racchiude in sé tutte le caratteristiche dell'AUV: la forma sferica ricorda quella di un pesce palla, ma anche quella di pallone da calcio, le operazioni in profondità che deve effettuare, e infine Purple che richiama il colore della squadra che gioca nello Stadio Artemio Franchi, le cui coordinate corrispondono a quelle dove è localizzato il bacino rettangolare da ispezionare.

Lo scopo finale è dunque quello di effettuare una simulazione sul software Matlab nella quale l'AUV effettui senza alcun ausilio esterno, vengono infatti solo passati i dati di riferimento tramite il file *missionC.m*, il pattugliamento del bacino.

Nello specifico il nostro gruppo (C5) si è occupato dell'implementazione del **Navigation System** (figura successiva), necessario per la stima della posizione e dell'orientazione durante il pattugliamento. Tale blocco elabora i dati provenienti dall'*Environment and Sensors Model* tramite un filtro e restituisce ai *Mission Supervisor/Reference Generator* e *Controller* la posizione e orientazione stimate insieme ad altri segnali ausiliari.



Schema dell'implementazione generale su Simulink

2. FASE DI INIZIALIZZAZIONE: LAVORO SVOLTO CON TUTTO IL TEAM

Durante le prime settimane il lavoro principale svolto dal team è consistito nella definizione e formalizzazione dei parametri generali di modello, dei segnali di interfaccia tra i vari blocchi, i sensori aggiuntivi e la codifica degli stati. Quest'ultima non verrà trattata in questo report in quanto non utilizzata nel nostro progetto.

MODELLO

Durante la fase di inizializzazione il team ha definito le caratteristiche fisiche del modello.
Non avendo direzioni privilegiate è stata scelta una geometria sferica per avere un ridotto coefficiente di attrito in ogni direzione. Il veicolo ha massa pari a 100 kg ed è stato scelto neutro, per questo motivo la densità è stata considerata pari a 1030 kg/m^3, come quella dell'acqua. Per quanto riguarda il numero degli attuatori, si è scelto di utilizzarne due per ogni direzione così da poter controllare tutti i gradi di libertà (Fig.0).

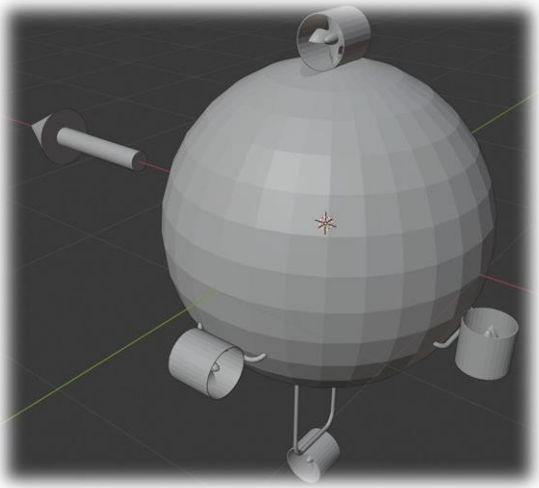


*Figura 1: modello fisico di **DEEP PU.R.P.L.E (PUffer Robotic Pool Levee Evaluation)***

Il centro di massa è stato spostato in basso rispetto al centro di simmetria, lungo l'asse di *heave* (z) così da garantire una coppia che influisca positivamente sulla stabilità di peso del veicolo. Essendo il centro di galleggiamento unicamente dipendente dalla geometria del veicolo è stato posto nel centro di simmetria.
L'origine della terna body è stata scelta coincidente con il centro di massa e allineata agli assi di simmetria. La terna NED, invece, ha origine nel corner C, posto in basso a sinistra nel bacino da perlustrare.

SENSORI AGGIUNTIVI

Per lo svolgimento del task è stato scelto di utilizzare: *GPS*, *profondimetro*, *IMU*, *AHRS* e *DVL*. Per questioni legate a misure ridondanti e di finalità di missione all'interno della *IMU* saranno presi come dati solo quelli del giroscopio escludendo le misure di accelerometro e magnetometro.

Dato che lo scopo della missione è un pattugliamento a distanza costante dalle pareti, come sensori aggiuntivi sono stati scelti tre *sonar* a singolo fascio, posizionati sulla prua, sul lato destro e sul lato sinistro del veicolo.
La scelta di questa tipologia di sensore è legata al suo utilizzo per la misura delle distanze in ambito subacqueo. Un'alternativa sarebbe stata quella di scegliere un sonar multi-beam, questa opzione è stata però scartata in quanto porterebbe ad avere un carico computazionale elevato e non necessario ai fini del task, oltre a un costo maggiore e non giustificato dai requisiti del progetto.
Il numero di sonar a singolo fascio è pari a tre in quanto nella configurazione di cui sopra è possibile ricevere sempre informazioni di almeno due pareti contemporaneamente (con una portata adeguata degli stessi). Un numero minore di sonar potrebbe portare a ricevere dati da una sola parete, senza quindi avere l'informazione relativa all'avvicinamento della parete frontale. Un numero maggiore porterebbe a ridondanze nel calcolo della posizione relativa rispetto al bacino.
Qui di seguito viene riportata la tabella con i vari input e output del *Navigation System*:

Misura	**Dimensione**	**Grandezza fisica**	**Sistema di riferimento**
INPUT FROM SENSORS MODEL			
GPS [lat, long, flag]	$3x1$	$[deg, deg, (0,1)]$	ECEF
profondimetro	$1x1$	$[m]$	Scalar
AHRS: [R,P,Y] (Roll,Pitch,Yaw)	$3x1$	$[rad, rad, rad]$	NED
DVL [vx, vy, vz]	$3x1$	$[m/s, m/s, m/s]$	Body
Vel_ang_gyro [va_R, va_P, va_Y]	$3x1$	$[rad/s, rad/s, rad/s]$	Body
sonar_prua	$1x1$	$[m]$	Scalar
sonar_dx	$1x1$	$[m]$	Scalar
sonar_sx	$1x1$	$[m]$	Scalar
OUTPUT TO CONTROLLER			
Pos_S [lat, long, depth]	$3x1$	$[deg, deg, m]$	ECEF
Or_S [R,P,Y] (Roll,Pitch,Yaw)	$3x1$	$[rad, rad, rad]$	NED
Vel_S [vx, vy, vz]	$3x1$	$[m/s, m/s, m/s]$	Body
Dpp_S	$1x1$	$[m]$	Scalar
Dps_S	$1x1$	$[m]$	Scalar
OUTPUT TO MISSION SUPERVISOR			
Pos_S [lat, long, depth]	$3x1$	$[deg, deg, m]$	ECEF
Or_S [R,P,Y] (Roll,Pitch,Yaw)	$3x1$	$[rad, rad, rad]$	NED
Vel_S [vx, vy, vz]	$3x1$	$[m/s, m/s, m/s]$	Body
Dpp_S	$1x1$	$[m]$	Scalar
Dps_S	$1x1$	$[m]$	Scalar
Vel_ang [va_R, va_P, va_Y]	$3x1$	$[rad/s, rad/s, rad/s]$	Body
Orp_S [Yaw]	$1x1$	$[rad]$	Scalar
SOS	$1x1$	$(0,1)$	Scalar

Tabella 1: input e output del blocco Navigation System

Simboli e Abbreviazioni usati nella Tabella 1

GPS = posizione in terna ECEF data dal GPS, il flag verrà usato come variabile di controllo per segnalare il funzionamento del GPS stesso (1= funzionante, 0 = non funzionante).

profondimetro = profondità data dal profondimetro riferita alla superficie dell'acqua (positiva sotto il livello dell'acqua).

DVL = velocità data dal DVL.

sonar_prua = distanza dalla parete data dal sonar sulla prua.

sonar_dx = distanza dalla parete data dal sonar sul lato destro.

sonar_sx = distanza dalla parete data dal sonar sul lato sinistro.

Vel_ang_gyro = velocità angolare fornita dal giroscopio.

Pos _S = stima della posizione.

Or_S = stima dell'orientazione (***Eulero XYZ***).

Vel _S = stima della velocità lineare.

Dpp_S = distanza dalla parete pattugliata.

Dps_S = distanza dalla parete successiva.

Orp _S = orientazione rispetto alla parete in pattugliamento.

SOS = flag di segnalazione di situazione di emergenza per fare abortire la missione (0 = no_abort, 1 = abort).

Si riportano i valori delle varianze e frequenze dei sensori utilizzati:

SENSORE	**VARIANZA σ^2**			**FREQUENZA**
GPS	$3\ [m^2]$			$1\ Hz$
Profondimetro	$0.2\ [m^2]$			$10\ Hz$
DVL	$0.012^2\ [(m/s)^2]$			$5\ Hz$
AHRS	$0.0087\ [rad^2]$ Roll	$0.0087\ [rad^2]$ Pitch	$0.0017\ [rad^2]$ Yaw	$10\ Hz$
Sonar	$0.05\ [m^2]$			$2\ Hz$
Giroscopio	$2.5133 \cdot 10^{-6}\ [(rad/s)^2]$			$10\ Hz$

Tabella 2: Varianze dei sensori utilizzati

3. LETTERATURA SULL' EXTENDED KALMAN FILTER TD

Data la natura non lineare del sistema analizzato è stato scelto un filtro EKF tempo discreto per la stima della posizione dell'AUV. Tale filtro è uno dei più utilizzati nell'affrontare questa tipologia di problemi nonostante i limiti che lo caratterizzano e che verranno successivamente esposti.

L'EKF si basa sulle seguenti ipotesi:

- ω_k e ν_k sono rumori *gaussiani*, a *media nulla*, *bianchi* (ogni realizzazione è indipendente da quella precedente), e *indipendenti* tra di loro. Nello specifico:
  $$\omega_k \sim N(0, Q_k) \qquad\qquad \nu_k \sim N(0, R_k)$$
  ovvero distribuzioni gaussiane con media nulla e matrice di covarianza Q_k e R_k rispettivamente.
- I rumori sono indipendenti dallo stato iniziale.

Tali ipotesi vengono rispettate dato che il blocco *Environment and Sensors Model* ha elaborato i rumori dei segnali provenienti dai sensori rispettando tali condizioni.

Di seguito vengono riportate le equazioni del modello generale.

Sia dato il seguente sistema tempo discreto:

$$\begin{cases} x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}) \\ z_{k-1} = h(x_{k-1}, \nu_{k-1}) \end{cases}$$

dove:

- *f, h* sono funzioni non lineari
- $\omega_{k-1} = \text{rumore di processo}$
- $\nu_{k-1} = \text{rumore di misura}$

Il filtro EKF applicato a tale sistema è composto da due elementi fondamentali:

- il predittore, che valuta la posizione sulla scorta delle informazioni allo stato precedente;
- il correttore, che corregge la posizione sulla base delle informazioni attuali.

PREDIZIONE

$$\hat{x}_{k|k-1} = f\big(\hat{x}_{k-1|k-1}, u_{k-1}, 0\big)$$

$$P_{k|k-1} = F_{k-1} P_{k-1|k-1} {F_{k-1}}^T + L_{k-1} Q_k {L_{k-1}}^T$$

- $\hat{x}_{k|k-1}$ stato stimato all'istante k sfruttando le informazioni note fino all' istante k-1
- $P_{k|k-1}$ matrice di covarianza associata a $\hat{x}_{k|k-1}$,calcolata all' instante k sfruttando le informazioni note fino all' istante k-1
- $F_{k-1} = \frac{df}{dx}\Big|_{\substack{u_{k-1} \\ \hat{x}_{k|k-1} \\ \omega_{k-1}=0}}$

- $L_{k-1} = \frac{df}{d\omega}\Big|_{\substack{u_{k-1} \\ \hat{x}_{k|k-1} \\ \omega_{k-1}=0}}$

CORREZIONE

$$y_k = z_k - h(\hat{x}_{k|k-1})$$

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

$$K_k = P_{k|k-1} {H_k}^T {S_k}^{-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

- y_k innovazione, con z_k misure effettuate all'istante k.
- S_k matrice di covarianza associata all'innovazione y_k.
- $H_k = \frac{dh}{dx}\Big|_{\hat{x}_{k|k-1}}$
- $M_k = \frac{dh}{dv}\Big|_{\hat{x}_{k|k-1}}$
- K_k guadagno di Kalman.
- $\hat{x}_{k|k}$ stato stimato all'istante k sfruttando le informazioni note fino all' istante k.
- $P_{k|k}$ matrice di covarianza associata a $\hat{x}_{k|k}$,calcolata all' instante k sfruttando le informazioni note fino all' istante k.

I limiti sopra citati che caratterizzano il suddetto filtro sono:

- Non è un filtro ottimo
- È sensibile alle condizioni iniziali
- La matrice P ottenuta è una stima di quella reale. Spesso viene sottostimata.

4. STIMA DELLA POSIZIONE TRAMITE FILTRO EKF TD

Per il nostro progetto è stata applicata la struttura precedentemente definita del filtro EKF alle condizioni operative del AUV per stimare la sua posizione istantanea. Di seguito viene descritta in dettaglio l'implementazione del filtro

Generalità

Nella fase di predizione si sfruttano DVL e AHRS (la cui misura della direzione nord si ipotizza non corrotta da campi magnetici locali), mentre in quella di correzione profondimetro, GPS e sonar. Il GPS viene utilizzato solamente in superfice, quando la profondità supera i $0.4\ m$ la correzione viene effettuata con i sonar. Per determinare il funzionamento del GPS o dei sonar viene sfruttata la terza componente del GPS (*Tabella 1*) durante la fase di inizializzazione. Il profondimetro viene utilizzato durante tutta la missione. La *Tabella 2* elenca i vettori adoperati nel filtro:

Segnali	Descrizione	Frequenza	Dimensione	Simbolo
Posizione	Posizione stimata in terna NED (origine in spigolo C)	$10\ Hz$	$3x1$	$\overrightarrow{\eta_1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
Orientazione	Valori da AHRS	$10\ Hz$	$3x1$	$\overrightarrow{\eta_2} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}$
Velocità lineare	Valori da DVL	$5\ Hz$	$3x1$	$\overrightarrow{\upsilon_1}$
Velocità angolare	Valori da giroscopio	$10\ Hz$	$3x1$	$\overrightarrow{\upsilon_2}$
Misure	Valori dai sensori*	$*$	$6x1$	$\vec{z}$
Errore associato all'errore di modello	Trascurabile (motivazioni date successivamente)		$3x1$	$\overrightarrow{\omega_m}$
Errore associato alla velocità lineare	Legato al DVL		$3x1$	$\overrightarrow{\omega_{\upsilon_1}}$
Errore associato all'orientazione	Legato all'AHRS		$3x1$	$\overrightarrow{\omega_{\eta_2}}$
Errore associato al vettore delle misure	Legato ai sensori	$*$ (vedi testo successivo)	$6x1$	$\overrightarrow{\omega_z}$
Matrice di rotazione Jacobiana	Per la trasformazione di vettori da terna body a terna NED. Viene calcolata con la frequenza del filtro.	$10Hz$	$3x3$	$J(\overrightarrow{\eta_2})$

Tabella 3: Vettori usati nell'EKF

Il vettore delle misure contiene le misure più recenti provenienti dai vari sensori:

$$\vec{z} = \begin{pmatrix} GPS_{(xNorth)} \\ GPS_{(yEast)} \\ depth_{profondimetro} \\ d_{sonar_{prua}} \\ d_{sonar_{dx}} \\ d_{sonar_{sx}} \end{pmatrix} \qquad \overrightarrow{\omega_z} = \begin{pmatrix} \overrightarrow{\omega_{GPS_{(xNorth)}}} \\ \overrightarrow{\omega_{GPS_{(yEast)}}} \\ \overrightarrow{\omega_{depth_{profondimetro}}} \\ \overrightarrow{\omega_{d_{sonar_{prua}}}} \\ \overrightarrow{\omega_{d_{sonar_{dx}}}} \\ \overrightarrow{\omega_{d_{sonar_{sx}}}} \end{pmatrix}$$

Le frequenze (*) di questi sensori sono:

- GPS $1Hz$
- Profondimetro $10\ Hz$
- Sonar $2\ Hz$

Per il filtro è stato scelto un **tempo di campionamento** di $\boldsymbol{\Delta T = 0.1\ secondi}$:

- la predizione viene effettuata ad una frequenza di $10\ Hz$.
- la correzione viene invece valutata ogni volta che arriva una nuova misura da almeno uno dei sensori.

Il modello del sistema da noi analizzato ha la seguente forma di stato:

$$\begin{cases} \dot{\overrightarrow{\eta_1}} = J_{(\overrightarrow{\eta_2} + \overrightarrow{\omega_{\eta_2}})}\ (\overrightarrow{\upsilon_1} + \vec{\omega}_{\vec{\upsilon}_1}) \\ \vec{z} = \overrightarrow{\eta_1} + \vec{\omega}_z \end{cases}$$

Dovendo implementare un algoritmo di calcolo che utilizzi il filtro, si deve passare dalla trattazione continua del fenomeno a quella discreta. Quindi approssimiamo la derivata con il rapporto incrementale:

$$\dot{\overrightarrow{\eta_1}} = \frac{\overrightarrow{\eta_1^k} - \overrightarrow{\eta_1^{k-1}}}{\Delta T}$$

Si ottiene il sistema tempo discreto:

$$\begin{cases} \overrightarrow{\eta_1^k} = \overrightarrow{\eta_1^{k-1}} + \Delta T\, J_{\left(\overrightarrow{\eta_2^{k-1}} + \overrightarrow{\omega_{\eta_2}^{k-1}}\right)}\ \left(\overrightarrow{\upsilon_1}^{k-1} + \overrightarrow{\omega_{\upsilon_1}}^{k-1}\right) + \overrightarrow{\omega_m}^{k-1} \\ \overrightarrow{z^{k-1}} = \overrightarrow{\eta_1^{k-1}} + \overrightarrow{\omega_z}^{k-1} \end{cases}$$

$\vec{\omega}_m$ viene trascurato in quanto $\vec{\omega}_m \ll \vec{\omega}_{sensori}$ (ovvero $\overrightarrow{\omega_{\upsilon_1}}$ e $\overrightarrow{\omega_{\eta_2}}$). Infatti il tempo di campionamento del passo di predizione è scelto in modo tale da essere inferiore o pari a quelli dei sensori.

$$\begin{cases} \overrightarrow{\eta_1^k} = \overrightarrow{\eta_1^{k-1}} + \Delta T\, J_{\left(\overrightarrow{\eta_2^{k-1}} + \overrightarrow{\omega_{\eta_2}^{k-1}}\right)}\ \left(\overrightarrow{\upsilon_1}^{k-1} + \overrightarrow{\omega_{\upsilon_1}}^{k-1}\right) \\ \overrightarrow{z^{k-1}} = \overrightarrow{\eta_1^{k-1}} + \overrightarrow{\omega_z}^{k-1} \end{cases}$$

Inizializzazione dell'algoritmo

Le condizioni iniziali sono state poste pari ai primi dati forniti dal GPS e dal profondimetro, in quanto l'EKF è sensibile alle loro variazioni. La matrice P_0 è stata inizializzata con le varianze del GPS e del profondimetro:

$$\eta_0 = \begin{bmatrix} GPS_{(xNorth)} \\ GPS_{(yEast)} \\ depth_{profondimetro} \end{bmatrix}_{|k=1} \qquad P_0 = \begin{bmatrix} \sigma^2_{GPS} & 0 & 0 \\ 0 & \sigma^2_{GPS} & 0 \\ 0 & 0 & \sigma^2_{depth} \end{bmatrix}$$

Determinazione del passo di predizione

Assumendo un tempo di campionamento $\Delta T = 0.1[s]$ il passo di predizione è dato da:

$$\overrightarrow{\hat{\eta}_1^{k|k-1}} = f\left(\overrightarrow{\hat{\eta}_1^{k-1|k-1}}, \overrightarrow{\upsilon_1{}^{k-1}}, \overrightarrow{\eta_2^{k-1}}, 0\right) = \overrightarrow{\hat{\eta}_1^{k-1|k-1}} + \Delta T \, J_{\left(\overrightarrow{\eta_2^{k-1}}\right)} \overrightarrow{\upsilon_1{}^{k-1}}$$

$$P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}{}^T + L_{k-1} Q_k L_{k-1}{}^T$$

- La matrice Q contiene le varianze dei segnali forniti da AHRS e DVL, che sono state passate dal gruppo di *Environment and Sensors Model*:

$$Q = diag(\sigma^2_{AHRS_{roll}} \quad \sigma^2_{AHRS_{pitch}} \quad \sigma^2_{AHRS_{yaw}} \quad \sigma^2_{DVL} \quad \sigma^2_{DVL} \quad \sigma^2_{DVL})$$

- La matrice F è calcolata come segue:

$$F_{k-1} = \begin{bmatrix} \frac{df(1)}{d\eta_{1(x)}} & \frac{df(1)}{d\eta_{1(y)}} & \frac{df(1)}{d\eta_{1(z)}} \\ \frac{df(2)}{d\eta_{1(x)}} & \frac{df(2)}{d\eta_{1(y)}} & \frac{df(2)}{d\eta_{1(z)}} \\ \frac{df(3)}{d\eta_{1(x)}} & \frac{df(3)}{d\eta_{1(y)}} & \frac{df(3)}{d\eta_{1(z)}} \end{bmatrix}_{\substack{\overrightarrow{\upsilon_1{}^{k-1}} \\ \overrightarrow{\hat{\eta}_1^{k|k-1}} \\ \overrightarrow{\omega_{k-1}}=0}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Nota: $\overrightarrow{\omega_{k-1}} = \begin{pmatrix} \overrightarrow{\omega_{\eta_2}{}^{k-1}} \\ \overrightarrow{\omega_{\upsilon_1}{}^{k-1}} \end{pmatrix}$ (vettore $6x1$)

- Per il calcolo della matrice L:

$$L_{k-1} = \begin{pmatrix} \frac{df(1)}{d\overrightarrow{\omega(1)_{\eta_2}}} & \frac{df(1)}{d\overrightarrow{\omega(2)_{\eta_2}}} & \frac{df(1)}{d\overrightarrow{\omega(3)_{\eta_2}}} & \frac{df(1)}{d\overrightarrow{\omega(1)_{\upsilon_1}}} & \frac{df(1)}{d\overrightarrow{\omega(2)_{\upsilon_1}}} & \frac{df(1)}{d\overrightarrow{\omega(3)_{\upsilon_1}}} \\ \frac{df(2)}{d\overrightarrow{\omega(1)_{\eta_2}}} & \frac{df(2)}{d\overrightarrow{\omega(2)_{\eta_2}}} & \frac{df(2)}{d\overrightarrow{\omega(3)_{\eta_2}}} & \frac{df(2)}{d\overrightarrow{\omega(1)_{\upsilon_1}}} & \frac{df(2)}{d\overrightarrow{\omega(2)_{\upsilon_1}}} & \frac{df(2)}{d\overrightarrow{\omega(3)_{\upsilon_1}}} \\ \frac{df(3)}{d\overrightarrow{\omega(1)_{\eta_2}}} & \frac{df(3)}{d\overrightarrow{\omega(2)_{\eta_2}}} & \frac{df(3)}{d\overrightarrow{\omega(3)_{\eta_2}}} & \frac{df(3)}{d\overrightarrow{\omega(1)_{\upsilon_1}}} & \frac{df(3)}{d\overrightarrow{\omega(2)_{\upsilon_1}}} & \frac{df(3)}{d\overrightarrow{\omega(3)_{\upsilon_1}}} \end{pmatrix}_{\substack{\overrightarrow{\upsilon_1{}^{k-1}} \\ \overrightarrow{\hat{\eta}_1^{k|k-1}} \\ \overrightarrow{\omega_{k-1}}=0}}$$

Il calcolo di questa matrice è stato implementato tramite *Matlab* (vedi pag. 64).

Determinazione del passo di correzione

$$\overrightarrow{e^k} = \overrightarrow{z^k} - \vec{h}\left(\overrightarrow{\hat{\eta}_1^{k|k-1}}, J\right)$$

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

$$K_k = P_{k|k-1} {H_k}^T {S_k}^{-1}$$

$$\overrightarrow{\hat{\eta}_1^{k|k}} = \overrightarrow{\hat{\eta}_1^{k|k-1}} + K_k e_k$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Dove:

- $\overrightarrow{e^k}$ è l'innovazione
- $\overrightarrow{z^k}$ sono le misure aggiornate fino all'istante k.
- La matrice R contiene le varianze dei segnali forniti da GPS, profondimetro e sonar ricavate dai rispettivi datasheet:

$$R = diag([\sigma^2_{GPS} \quad \sigma^2_{GPS} \quad \sigma^2_{profondimetro} \quad \sigma^2_{sonar} \quad \sigma^2_{sonar} \quad \sigma^2_{sonar}])$$

- Calcolo della matrice M:

$$M_k = \left.\frac{dh(\eta_1, J)}{d\omega_z}\right|_{\overrightarrow{\hat{\eta}_1^{k|k-1}}} = I_{6x6}$$

Da notare che, essendo h dipendente dalla matrice J e quindi dall'orientazione del veicolo misurata direttamente dal sensore AHRS, nel vettore $\overrightarrow{\omega_z}$ dovrebbero essere considerate anche le componenti di $\overrightarrow{\omega_{\eta_2}}$. In questa trattazione si assume tale errore trascurabile rispetto a quello degli altri sensori. Per una trattazione più accurata si rimanda alla sezione *13. Conclusioni e spunti implementativi per migliorare ulteriormente le prestazioni.*

- Calcolo matrice H:

il vettore $h(\hat{\eta}_{k|k-1}, J)$, di dimensione $6x1$, contiene le misure virtuali ovvero le misure che si otterrebbero dai sensori GPS, profondimetro e i tre sonar se questi non fossero affetti da rumore. Tale vettore è dato da:

$$\vec{h}(\hat{\eta}_{k|k-1}, J) = \begin{pmatrix} \hat{\eta}_{k|k-1(x)} \\ \hat{\eta}_{k|k-1(y)} \\ \hat{\eta}_{k|k-1(z)} \\ h_{sonar\_prua}(\hat{\eta}_{k|k-1}, J) \\ h_{sonar\_dx}(\hat{\eta}_{k|k-1}, J) \\ h_{sonar\_sx}(\hat{\eta}_{k|k-1}, J) \end{pmatrix}$$

l'innovazione viene quindi calcolata come segue:

$$\overrightarrow{e_k} = \overrightarrow{z_k} - \vec{h}(\hat{\eta}_{k|k-1}, J) = \begin{pmatrix} GPS_{(xNorth)} \\ GPS_{(yEast)} \\ depth_{profondimetro} \\ d_{sonar_{prua}} \\ d_{sonar_{dx}} \\ d_{sonar_{sx}} \end{pmatrix}_k - \begin{pmatrix} \hat{\eta}_{k|k-1(x)} \\ \hat{\eta}_{k|k-1(y)} \\ \hat{\eta}_{k|k-1(z)} \\ h_{sonar_{prua}}(\hat{\eta}_{k|k-1}, J) \\ h_{sonar_{dx}}(\hat{\eta}_{k|k-1}, J) \\ h_{sonar_{sx}}(\hat{\eta}_{k|k-1}, J) \end{pmatrix}$$

Per la posizione e la profondità quindi si utilizza lo stato stimato ricavato nel passo di predizione, invece per i sonar si utilizza una funzione che simula il loro comportamento ideale.

Essendo nota la geometria del bacino (che approssimativamente si può descrivere come un parallelepipedo rettangolo) e sapendo che le pareti sono verticali, si possono calcolare le equazioni dei piani da esse descritte a partire dagli spigoli. Il sistema da risolvere è:

$$\begin{cases} ax_1 + by_1 + cz_1 + d = 0 \\ ax_2 + by_2 + cz_2 + d = 0 \end{cases}$$

Scegliendo il coefficiente $a = 1$ e dato che la parete è verticale, cioè $c = 0$, otteniamo:

$$\begin{cases} a = 1 \\ b = \dfrac{(x_1 - x_2)}{(y_2 - y_1)} \\ c = 0 \\ d = -x_1 - bx_2 \end{cases}$$

A questo punto sono definite tutte le equazioni dei piani e lo scopo è di calcolare la distanza dalla posizione stimata alla parete colpita dal sonar lungo la direzione di quest'ultimo.

Vengono definiti i versori dei sonar in terna body:

$$v_{sonar_{prua}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad v_{sonar_{dx}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad v_{sonar_{sx}} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

moltiplicandoli per la matrice di rotazione J si ricava il versore dei sonar in terna NED.

$$v_{sonar_{NED}} = J\, v_{sonar} = \begin{bmatrix} x_{sonarNED} \\ y_{sonarNED} \\ z_{sonarNED} \end{bmatrix}$$

Perciò l'equazione parametrica in t della retta con direzione pari a quella del versore e passante per il punto pari alla posizione stimata è:

$$\begin{cases} x = \hat{\eta}_{1(x)} + x_{sonarNED} \cdot t \\ y = \hat{\eta}_{1(y)} + y_{sonarNED} \cdot t \\ z = \hat{\eta}_{1(z)} + z_{sonarNED} \cdot t \end{cases}$$

Mettendo a sistema tale retta con l'equazione del piano possiamo calcolarci il valore del parametro t. Dal momento che il vettore direzione considerato è un versore, tale parametro è proprio la distanza dalla nostra posizione alla parete lungo la direzione del sonar.

$$\begin{cases} ax + by + cz + d = 0 \\ x = \hat{\eta}_{1(x)} + x_{sonarNED} \cdot t \\ y = \hat{\eta}_{1(y)} + y_{sonarNED} \cdot t \\ z = \hat{\eta}_{1(z)} + z_{sonarNED} \cdot t \end{cases} \qquad t = \left| \frac{-(\hat{\eta}_{1(x)} - b\hat{\eta}_{1(y)} + d)}{by_{sonarNED} + x_{sonarNED}} \right| = h_{sonar}$$

Questa procedura viene utilizzata per calcolare la misura virtuale di ciascun sonar.

La forma generale della matrice H è:

$$H_k = \frac{d\vec{h}(\eta_1, J)}{d\eta}\bigg|_{\hat{\eta}_{k|k-1}} = \frac{d}{d\eta}\begin{pmatrix} \eta_{1(x)} \\ \eta_{1(y)} \\ \eta_{1(z)} \\ h_{sonar\_prua}(\eta_{1(x)}, J) \\ h_{sonar\_dx}(\eta_{1(y)}, J) \\ h_{sonar\_sx}(\eta_{1(z)}, J) \end{pmatrix}\Bigg|_{\hat{\eta}_{k|k-1}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{\partial h_{sonar}}{\partial \eta_{1(x)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(y)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(z)}} \\ \frac{\partial h_{sonar}}{\partial \eta_{1(x)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(y)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(z)}} \\ \frac{\partial h_{sonar}}{\partial \eta_{1(x)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(y)}} & \frac{\partial h_{sonar}}{\partial \eta_{1(z)}} \end{pmatrix}$$

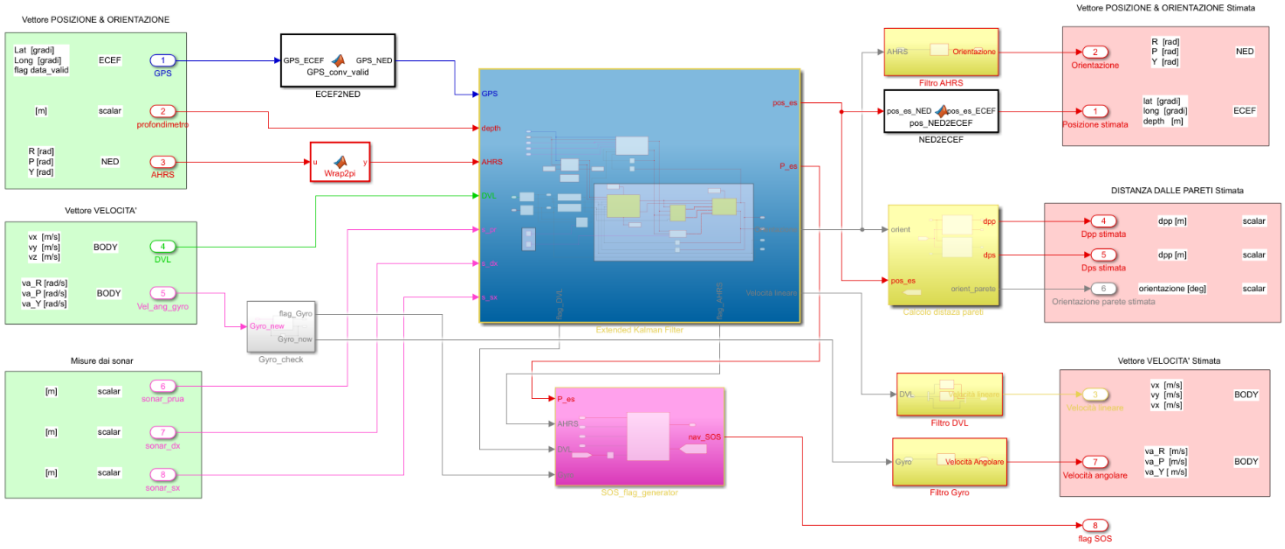
Naturalmente le derivate parziali di $h_{sonar}(\hat{\eta}_{k|k-1}, J)$ dipendono dallo stato, dall'orientazione e quindi da J, dal versore del sonar preso in esame e dai coefficienti del piano intercettato dal sonar. Le espressioni di tali elementi nella matrice sono:

$$\frac{\partial h_{sonar}}{\partial \eta_{1(x)}} = \frac{sgn(\eta_{1(x)} + b\eta_{1(y)} + d)}{|by_{sonarNED} + x_{sonarNED}|}$$
$$\frac{\partial h_{sonar}}{\partial \eta_{1(y)}} = \frac{b \cdot sgn(\eta_{1(x)} + b\eta_{1(y)} + d)}{|by_{sonarNED} + x_{sonarNED}|}$$
$$\frac{\partial h_{sonar}}{\partial \eta_z} = 0$$

5. STIMA DI ORIENTAZIONE VELOCITÀ LINEARE E ANGOLARE

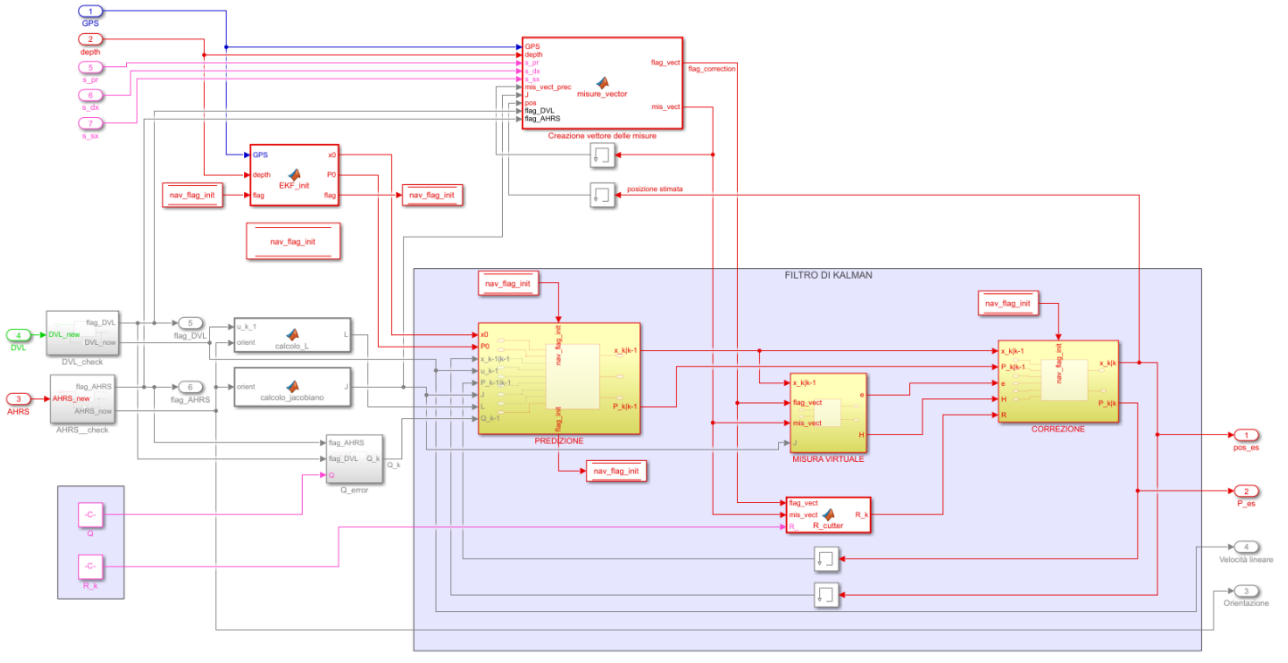
Le misure ricevute dall'AHRS, dal DVL e dal giroscopio sono state trattate applicando in uscita un filtro a media campionaria per ridurre le oscillazioni ad alta frequenza e migliorare la qualità delle misure stesse.

BLOCCO SYSTEM NAVIGATION



Blocco di Navigation System in Simulink

EXTENDED KALMAN FILTER



EKF in Simulink

6. DESCRIZIONE DEI VARI BLOCCHI SU SIMULINK

Blocco misure_vector

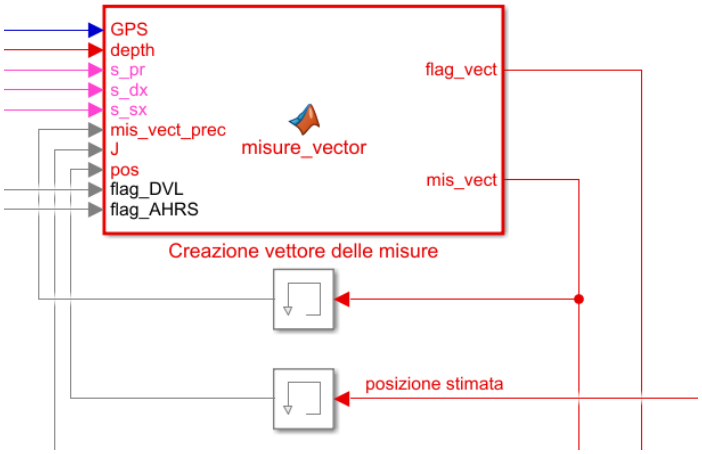


Figura 1: Blocco misure_vector in Simulink

Scopo: realizzazione del vettore delle misure virtuali $\vec{h}(\hat{\eta}_{k|k-1}, J)$.

In ingresso a questo blocco sono presenti tutti i sensori necessari per la fase di correzione del filtro mentre in uscita sono generati due vettori: *mis_vect* e *flag_vect* entrambi di dimensione $7x1$. *mis_vect* è un vettore colonna contenente tutti i valori dei sensori:

$$mis_{vect} = \begin{pmatrix} GPS_{(xNorth)} \\ GPS_{(yEast)} \\ GPS_{(flag)} \\ depth_{profondimetro} \\ d_{sonar_{prua}} \\ d_{sonar_{dx}} \\ d_{sonar_{sx}} \end{pmatrix}$$

flag_vect include i flag relativi ad ogni misura, necessari a discriminarla valida o meno.

Questi due vettori vengono calcolati alla stessa frequenza del filtro; dal momento che la frequenza dei segnali dei sensori è inferiore o pari ad essa, se in ingresso al sistema arrivano due valori successivi uguali, tale misura viene considerata non valida perché ritenuta non aggiornata. Ad esempio, dato che questo blocco viene eseguito in simulazione ogni $0.1s$ mentre il segnale del GPS è aggiornato ogni secondo, solo una delle dieci misure viene ritenuta valida nell'arco di un intervallo di un secondo.

Questa logica è usata per tutti i sensori usati nella correzione; nel caso specifico dei sonar sono state aggiunte delle restrizioni per aumentare le prestazioni del filtro. Da prove sperimentali e per il modo in cui è stato implementato il sonar dal blocco *Environment and Sensors Model*, si è osservato un aumento dell'errore sulla posizione stimata nelle seguenti situazioni:

- Quando la misura del sonar è elevata: da prove sperimentali è stato notato in questo caso che piccole variazioni di angolo di yaw comportano un segnale molto rumoroso. Per limitare

l'errore che si crea, è stata stimata una soglia massima oltre la quale la misura non è più considerata valida.

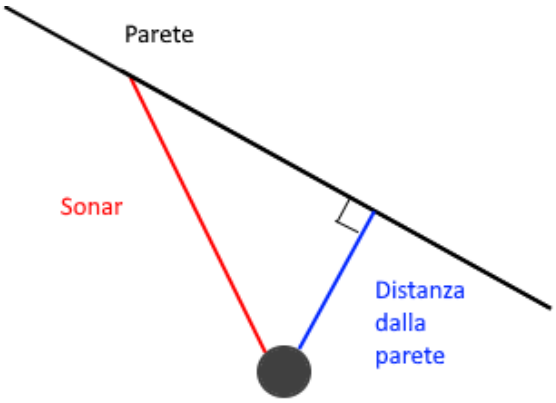


Figura 2: rappresentazione del sonar e della sua distanza dalla parete

Per la scelta della soglia è stato supposto il seguente ragionamento: assumendo la distanza dalla parete fissa in fase di pattugliamento, consideriamo il seguente triangolo rettangolo *(fig.3).* La misura del sonar diventa critica nel caso in cui l'angolo tra sonar e parete sia eccessivamente acuto. Posto che sia $20°$:

$$distanza_{sonar} = \frac{distanza_{parete}}{\cos(\alpha)}\Big|_{\alpha_{max}=90-20°} \cong distanza_{parete} \cdot 3$$

In seguito a varie prove sperimentali si è verificato che tale soglia determina un netto miglioramento delle prestazioni.

- Quando il fascio del sonar è vicino agli spigoli del bacino: Il sonar *singolo beam* è stato implementato dal gruppo *Environment and Sensors Model* come un fascio di rette raggruppate in un cono, quindi la misura passataci viene calcolata pesando adeguatamente e sommando le distanze AUV-parete ottenute lungo le direzioni delle singole rette. Di conseguenza se il fascio del sonar incontra uno degli spigoli del bacino la misura risulta essere non corretta.
  Per valutare la vicinanza allo spigolo viene calcolata l'intersezione tra la retta lungo la direzione del versore del sonar e il piano della parete che tale retta interseca. Calcolando la norma della differenza tra le coordinate dello spigolo e quelle dell'intersezione si valuta la distanza effettiva. Se quest'ultima è inferiore a una soglia allora la misura viene scartata. (i calcoli geometrici effettuati sono gli stessi usati per le *misure virtuali, pag.14-15*)

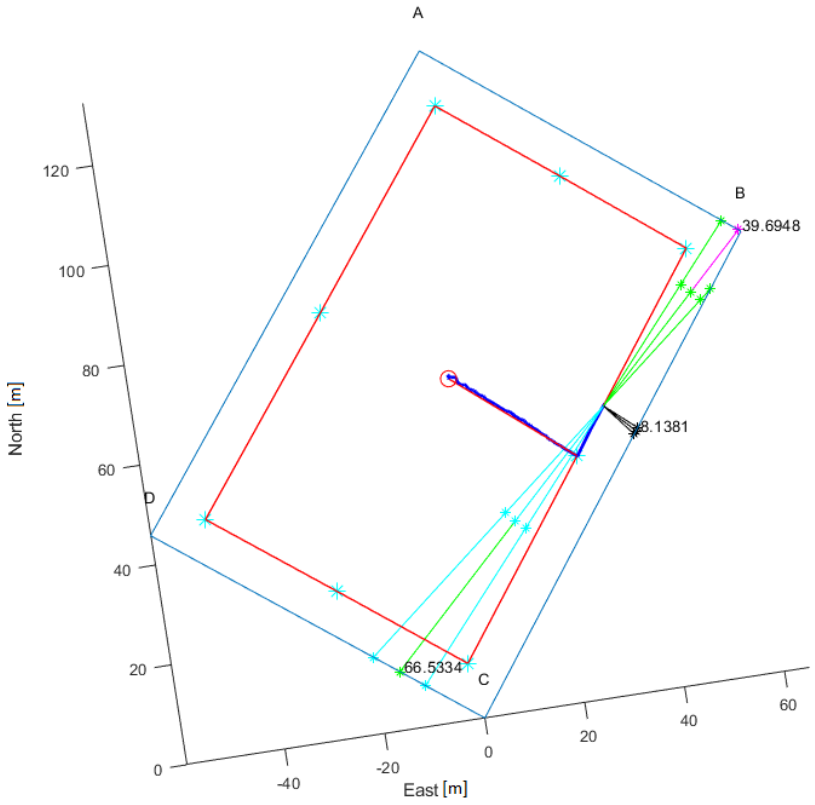


Figura 3: esempio dei fasci conici dei sonar quando puntano verso uno spigolo

- Nella fase di rotazione dell'AUV, che avviene in prossimità degli spigoli del bacino alla fine del pattugliamento di una parete, si osserva una repentina variazione delle misure che comporta un incremento dell'errore sulla posizione stimata. Di conseguenza è stato necessario introdurre una soglia sulla massima differenza ammissibile tra due misure consecutive provenienti dal medesimo sonar.

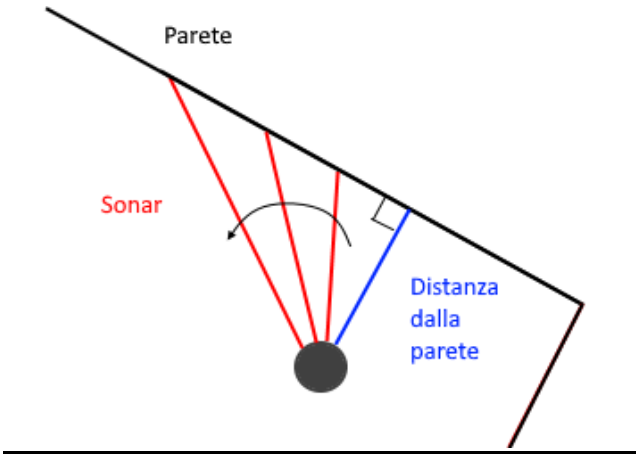


Figure 5: Fase di rotazione dell'AUV in prossimità degli spigoli

Questo problema è anche presente quando il fascio del sonar è quasi parallelo alla parete, come in Fig. 6.

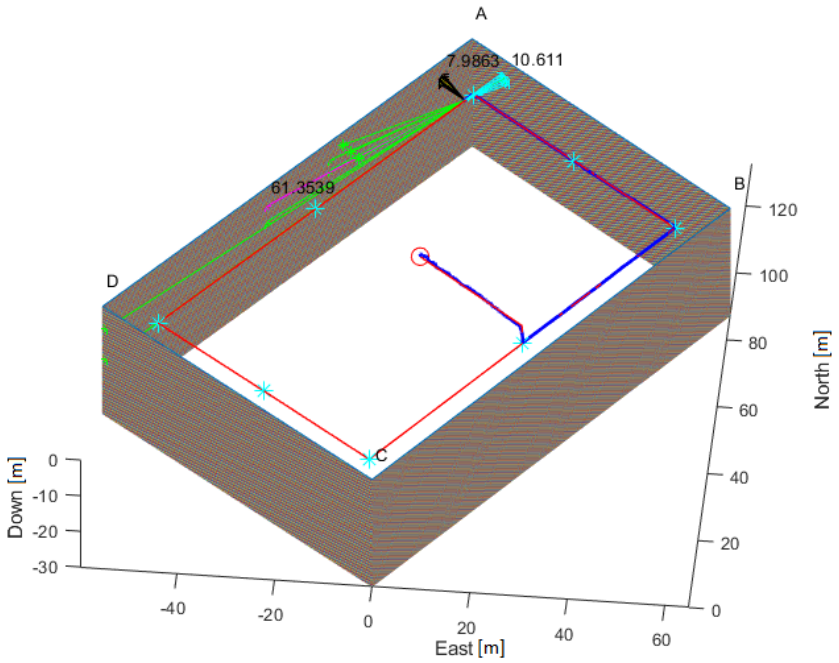


Figura 6: Fase di rotazione dell'AUV con fascio del sonar parallelo alla parete

In tutte queste situazioni le informazioni derivanti dalle misure dei sonar sono marginali se il DVL e l'AHRS sono sempre funzionanti. In caso contrario, se uno o entrambi di questi sensori smettessero di inviare dati, la stima della posizione in profondità dovrebbe essere basata solo sui sonar, ragion per cui in ingresso a questo blocco ci sono anche i *flag_DVL* e *flag_AHRS* che comunicano eventuali malfunzionamenti.

Inizializzazione

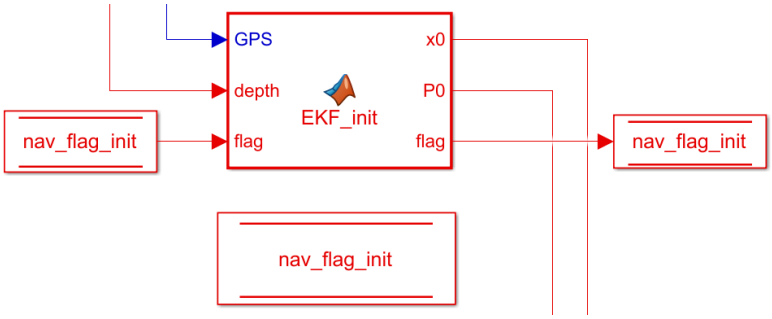


Figura 4: blocco di inizializzazione in Simulink

In questa funzione viene eseguita l'inizializzazione del filtro, cioè è generato il primo stato η_{1_0} e la prima matrice di covarianza della stima P_0. Questo blocco sfrutta il flag *nav_flag_init:* esso viene posto inizialmente a 0 (valore per il quale né la predizione, né la correzione vengono eseguite), dopo la prima assegnazione il flag viene aggiornato a 1. A questo punto il filtro viene fatto partire.

Calcolo matrice L e matrice di rotazione J

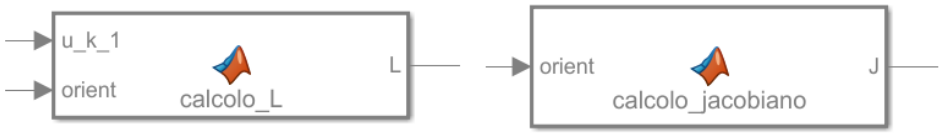


Figura 5: Blocchi in Simulink dove si effettua il calcolo della matrice L e J

In questo blocco è implementato il calcolo numerico delle matrici L e J nota l'orientazione e la velocità lineare.

Calcolo matrice R

Figura 6: Blocco di calcolo della matrice R in Simulink

Questo blocco si occupa di andare a eliminare le rispettive righe e colonne della matrice R in corrispondenza dei valori dei sensori di GPS, profondimetro e sonar qualora essi non siano validi o aggiornati e quindi non rientrino nel passo di correzione.

Calcolo matrice Q

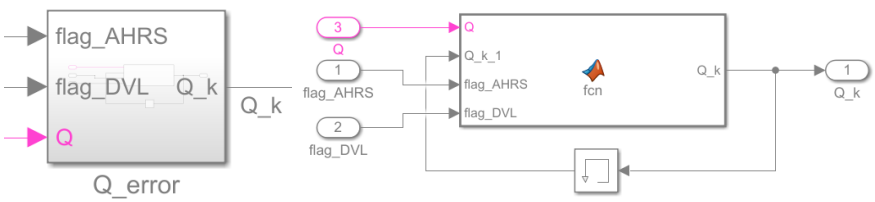


Figura 7: Blocchi in Simulink di calcolo della matrice Q

Tramite questa funzione viene incrementata adeguatamente Q nel caso in cui si abbia il malfunzionamento del DVL e dell'AHRS. Operativamente viene aggiunto alla matrice Q un fattore proporzionale alle covarianze dei segnali le cui misure non sono valide.

$$Q_k = Q_{k-1} + K_{DVL} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{DVL} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{DVL} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{DVL} \end{bmatrix} + K_{AHRS} \cdot \begin{bmatrix} \sigma^2_{AHRS_{roll}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{AHRS_{pitch}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{AHRS_{yaw}} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

I valori di K_{DVL} e K_{AHRS} sono stati scelti empiricamente (*pag. 66*)

Predizione

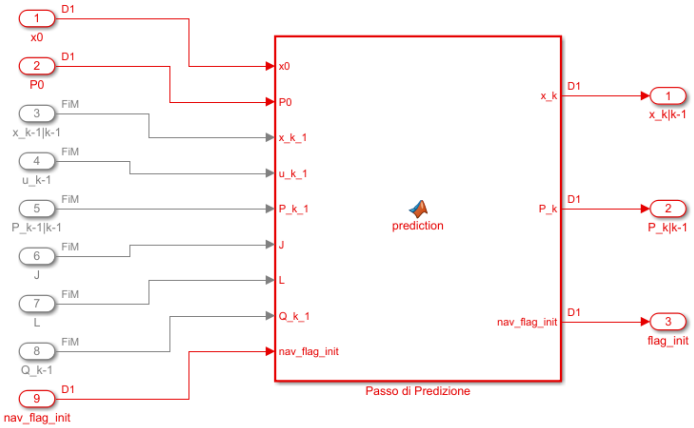


Figura 8: Blocco di predizione in Simulink

Tale blocco è implementato con una sola funzione che gestisce il passo di predizione del filtro di Kalman.

Qui inoltre è gestito *nav_flag_init* (flag di inizializzazione): dopo la prima inizializzazione tale flag viene posto ad un valore di default e non verrà più usato fino alla fine della simulazione.

Misure Virtuali

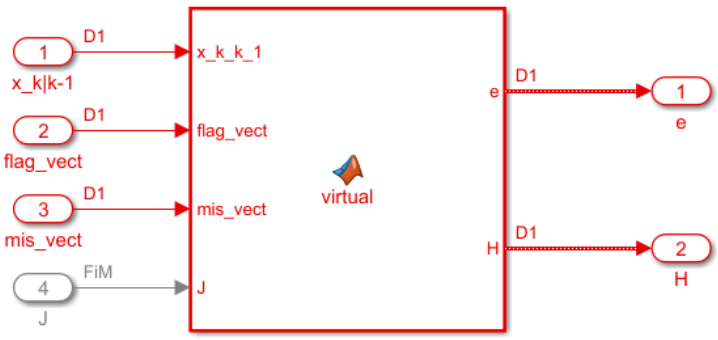


Figura 9: Blocco delle misure virtuali in Simulink

In tale blocco avviene il calcolo dell'innovazione $\vec{e}$ e della matrice H che vengono passate al blocco di correzione. In ingresso abbiamo il vettore delle misure (*mis_vect*) e il vettore dei flag associato a tali misure (*flag_vect*).

Tramite un controllo sui flag, vengono creati i vettori $\vec{z}$ e $\overrightarrow{h\left(\widehat{\eta_1}_{k|k-1}, J\right)}$ e la matrice H_k andando a trascurare le righe relative alle misure non considerate nella fase di correzione.

Il vettore delle misure $\vec{z}$ viene creato impilando le componenti di *mis_vect* che sono valide per la correzione, ciò viene fatto con le informazioni contenute in *flag_vect.*

La correzione deve essere fatta solo sulle misure che in quell'istante sono disponibili, quindi nell'ultima parte della funzione sia $\vec{e}$ che H vengono ridotte di dimensione eliminando le righe non necessarie.

Correzione

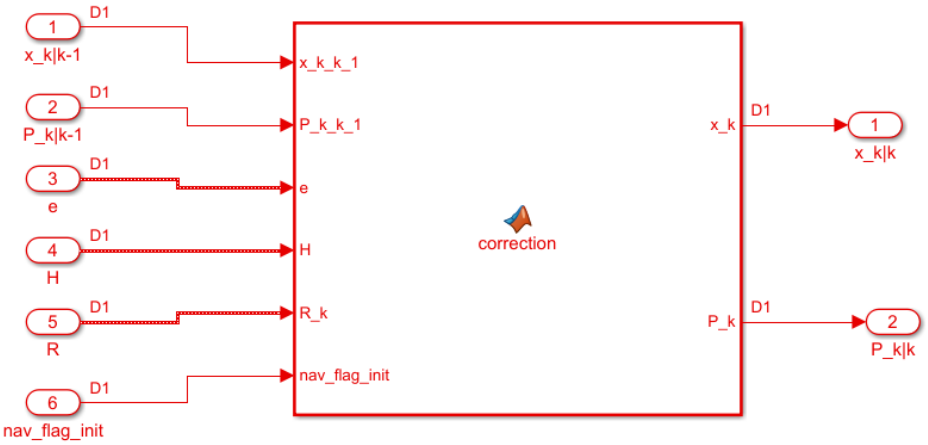


Figura 10: Blocco di correzione in Simulink

Il blocco di correzione implementa le equazioni del passo di correzione una volta calcolata l'innovazione $\vec{e}$, la matrice H e la matrice R.

La correzione viene eseguita solo se il filtro è stato inizializzato, informazione passata dal flag *nav_flag_init*.

Conversione ECEF-NED e NED-ECEF

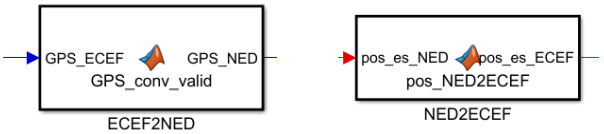


Figura 11: Blocchi di conversione delle misure da terna ECEF a terna NED e viceversa in Simulink

Nel blocco ECEF2NED viene effettuata la conversione delle misure del GPS, che ci vengono fornite in latitudine e longitudine, in terna NED attraverso la funzione Matlab *geodetic2ned*. Come terza componente di questo vettore viene restituito il flag di validità della misura, il quale, discriminando

se la misura fornita sia o meno valida, ne determina anche la sua conversione. In accordo con tutto il team, è stato utilizzato come sferoide di riferimento per la Terra il ***wgs84Ellipsoid***.

Il blocco NED2ECEF attua l'operazione inversa del blocco precedente applicata alla posizione stimata.

Blocco calcolo distanza dalla parete e orientazione relativa

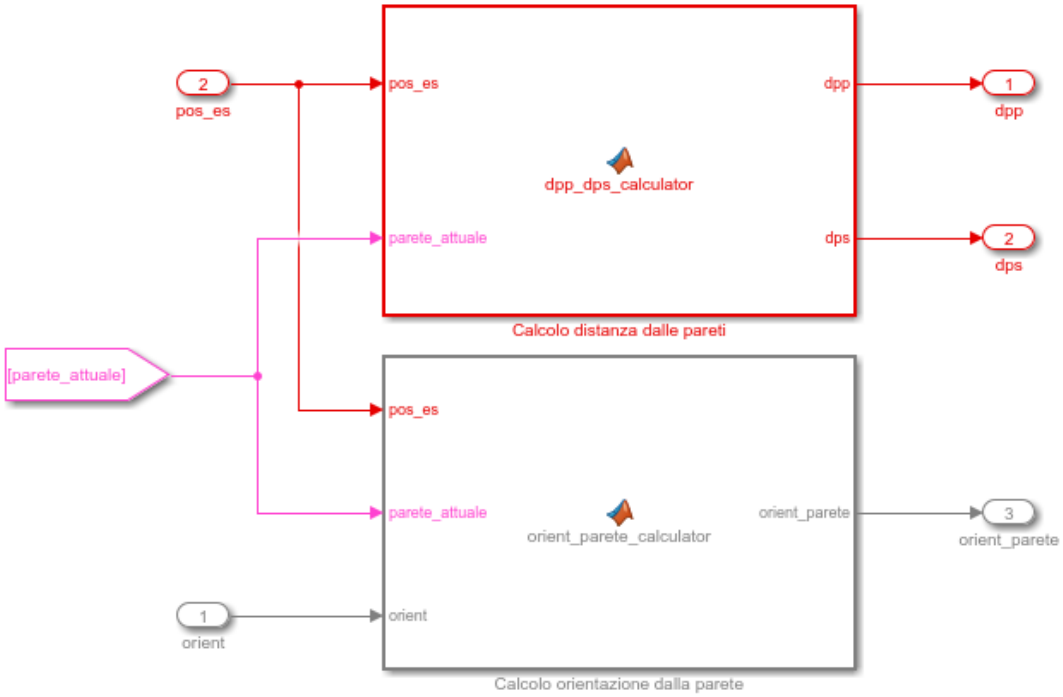


Figura 12: Blocchi in Simulink di calcolo della distanza dalla parete e di orientazione relativa

In questo blocco viene calcolata la distanza dalla parete di pattugliamento e l'orientazione relativa. Per quanto riguarda la prima, una volta nota la parete attuale di riferimento dipendente dallo stato, si calcola geometricamente la distanza minima tra la posizione stimata e il piano della parete. Data la nostra posizione stimata in terna NED $\widehat{\eta_1}_{k|k} = [\eta_{1_x}, \eta_{1_y}, \eta_{1_z}]^T$ e l'equazione del piano in esame $ax + by + cz + d = 0$, si applica la formula della distanza nota in algebra:

$$dist = \frac{\left|a\eta_{1_x} + b\eta_{1_y} + c\eta_{1_z} + d\right|}{\sqrt{a^2 + b^2 + c^2}}$$

Per l'orientazione relativa rispetto alla parete invece, è necessario calcolare l'angolo relativo tra il versore perpendicolare della parete e il versore dell'asse x della terna body.

Tale orientazione varia da $[-180°, 180°]$ ed è positiva per rotazioni orarie e negativa per rotazioni antiorarie, in accordo con la convenzione dato che l'asse z in terna NED è verso il basso.

Per il calcolo si utilizza il versore ortogonale al piano in terna NED definito come:

$$v_{plane_{NED}} = \frac{[a, b, c]^T}{\sqrt{a^2 + b^2 + c^2}}$$

Il versore che determina la direzione $surge$ in body, riportata in terna NED, si calcola ruotando dell'angolo di yaw il versore $[1 \quad 0 \quad 0]^T$, cioè:

$$v_{surge_{NED}} = {R_z}^T [1 \quad 0 \quad 0]^T$$

L'angolo acuto α tra la retta definita da $v_{surge_{NED}}$ e il piano è dato da:

$$\alpha = \arcsin\left(\frac{|v_{plane_{NED}} \cdot v_{surge_{NED}}|}{\|v_{plane_{NED}}\| \cdot \|v_{surge_{NED}}\|}\right)$$

In questo modo dobbiamo però discriminare se l'angolo è positivo o negativo.

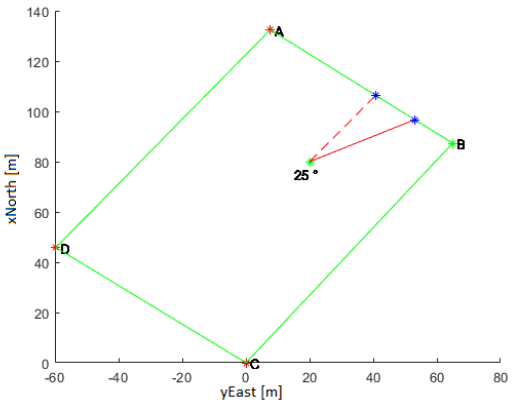


Figura 13: caso in cui l'angolo acuto è positivo

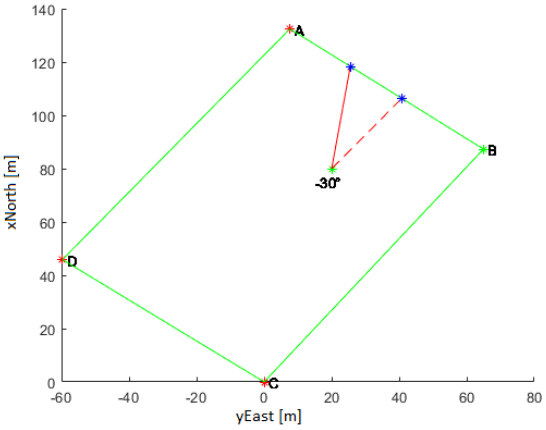


Figura 14: caso in cui l'angolo acuto è negativo

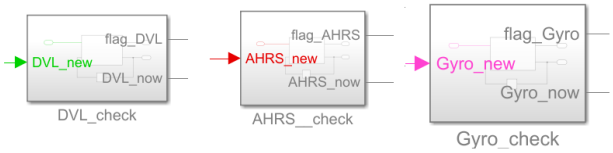
Per ovviare a questo problema si trovano i valori dell'intersezione sul piano lungo la direzione $v_{surge_{NED}}$ e si valuta la sua posizione rispetto agli spigoli del bacino.

Per il calcolo delle intersezioni abbiamo utilizzato i seguenti sistemi calcolando i parametri t e t', e quindi le intersezioni stesse.

$$\begin{cases} a\eta_{1_x} + b\eta_{1_y} + c\eta_{1_z} + d = 0 \\ x = \eta_{1_x} + v_{surge_{NED_x}} \cdot t \\ y = \eta_{1_y} + v_{surge_{NED_y}} \cdot t \\ z = \eta_{1_z} + v_{surge_{NED_z}} \cdot t \end{cases}$$

$$\begin{cases} a\eta_{1_x} + b\eta_{1_y} + c\eta_{1_z} + d = 0 \\ x = \eta_{1_x} + v_{plane_{NED_x}} \cdot t' \\ y = \eta_{1_y} + v_{plane_{NED_y}} \cdot t' \\ z = \eta_{1_z} + v_{plane_{NED_z}} \cdot t' \end{cases}$$

Gestione misure errate DVL, AHRS e giroscopio



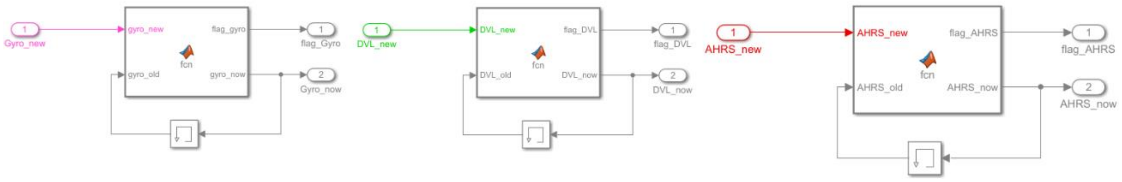


Figura 15: Blocchi in Simulink di gestione in caso di misure errate fornite da DVL, AHRS e giroscopio

In fase di implementazione si è considerato il caso in cui le misure di DVL, AHRS e giroscopio non siano valide per un'ignota causa. Dato che per la maggior parte della missione tali misure sono quasi constanti nel tempo, tranne nei cambi di traiettoria, può essere ragionevole, in caso si verifichi un malfunzionamento di questi sensori, usare le misure dei tempi precedenti al posto di non considerarle affatto. Tale situazione viene segnalata con un flag per ogni sensore e inviata al blocco del calcolo della matrice Q, per aumentare la covarianza della stima al passo di predizione, al blocco SOS che comunica l'aborto della missione se il tempo in cui le misure non sono valide supera una soglia prestabilita e al blocco *misure_vect* che nel caso di guasto a DVL e AHRS elimina le soglie di validità sulle misure dei sonar.

Blocco SOS

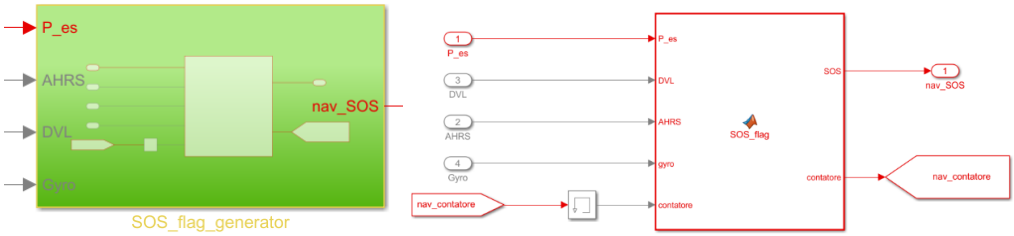


Figura 16: Blocco di segnalazione di aborto missione in Simulink

Viene lanciato un FLAG di SOS con conseguente termine della missione, in tre casi:

1. Se GPS, sonar e profondimetro, che abbiamo preferito trattare complessivamente, non restituiscono una misura valida, la matrice di covarianza P aumenta in quanto non viene effettuata la correzione. Il flag di SOS viene posto pari a 1 se la traccia della matrice P supera una certa soglia, che da prove di *trail&error* è stato stimato essere pari a $10\ [m^2]$. La P da noi implementata riporta a regime un errore sull'ordine del cm, questo è dovuto all'elevata precisione del DVL e del breve tempo di simulazione.
2. Se il DVL è guasto e restituisce NaN per un intervallo di tempo superiore a $5\ s$.
3. Se l'AHRS non funziona e restituisce una misura pari a NaN per un intervallo di tempo superiore a $5\ s$.

Per tenere traccia del tempo trascorso da quando i sensori hanno iniziato a fornire il valore NaN, è stato implementato un contatore. Se il contatore supera appunto $5\ secondi$ viene abortita la missione.

Filtri a media campionaria per DVL, AHRS e giroscopio

Figura 17: Filtri applicati al DVL, AHRS E giroscopio in Simulink

Per rendere la misura fornita dall'AHRS meno rumorosa, è stato implementato un filtro discreto a media campionaria sugli ultimi sette campioni di cui vengono pesati maggiormente quelli più recenti. Anche per le prime due componenti del DVL e per il giroscopio è stato inserito un filtro a media campionaria sui campioni più recenti, rispettivamente sugli ultimi 4 e 3 campioni. I coefficienti scelti tramite prove di *trial&error* sono:

- $coeff_{AHRS} = [0.2, 0.2, 0.2, 0.1, 0.1, 0.1, 0.1]$
- $coeff_{DVL} = [0.3, 0.3, 0.2, 0.2]$
- $coeff_{Gyro} = [0.9, 0.05, 0.05]$

Questi blocchi sono stati implementati dopo le prime prove a ciclo chiuso sotto richiesta esplicita da parte del gruppo *Controller*.

7. PROVE EFFETTUATE CON DATI REALI

Come primo test per effettuare la validità del filtro, sono stati utilizzati dei dati di GPS, profondimetro e DVL derivanti da prove di ricerca reali.

Dopo averli resi compatibili con la nostra simulazione, i primi risultati sono quelli sotto riportati.

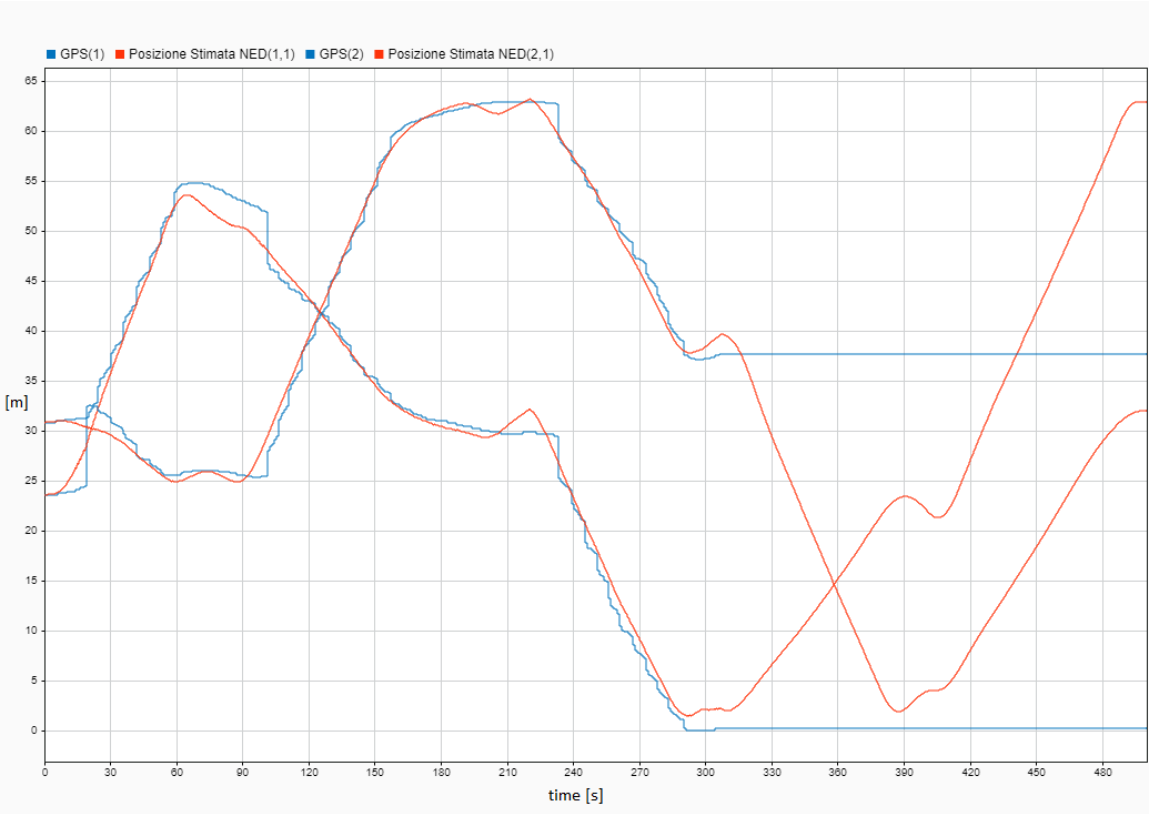


Figura 18: confronto tra posizione stimata e dati provenienti da GPS

Da questo grafico si evince che le prime due componenti della posizione stimata seguono adeguatamente i dati del GPS fino all'istante in cui quest'ultimi sono disponibili (fino a circa $300\ s$). La stima successivamente si basa solo su AHRS e DVL, che invece risultano essere sempre disponibili.

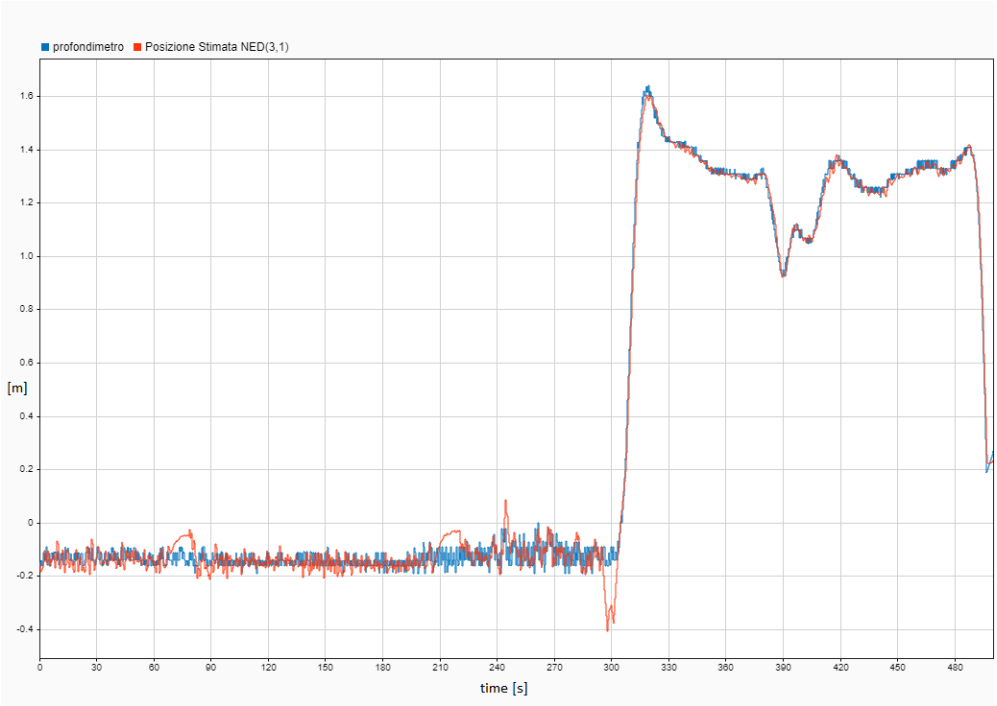


Figura 19: confronto tra profondità stimata e dati provenienti dal profondimetro

Il grafico riporta la terza componente della posizione stimata e il segnale che viene dal profondimetro. I picchi presenti sulla stima sono dovuti a variazioni del segnale del DVL che vengono pesate maggiormente in fase di predizione.

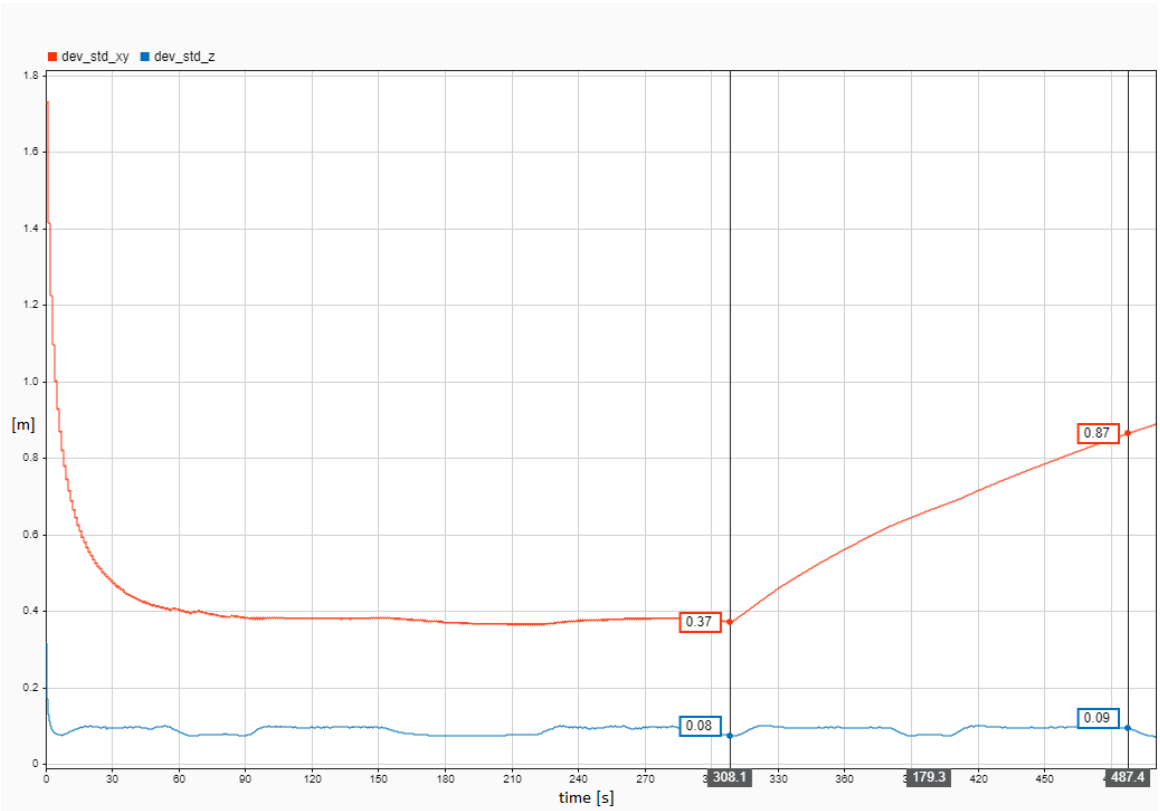


Figura 20: componenti della matrice di covarianza P

In Fig.23 è riportato l'andamento delle deviazioni standard, crescenti in fase di sola predizione.

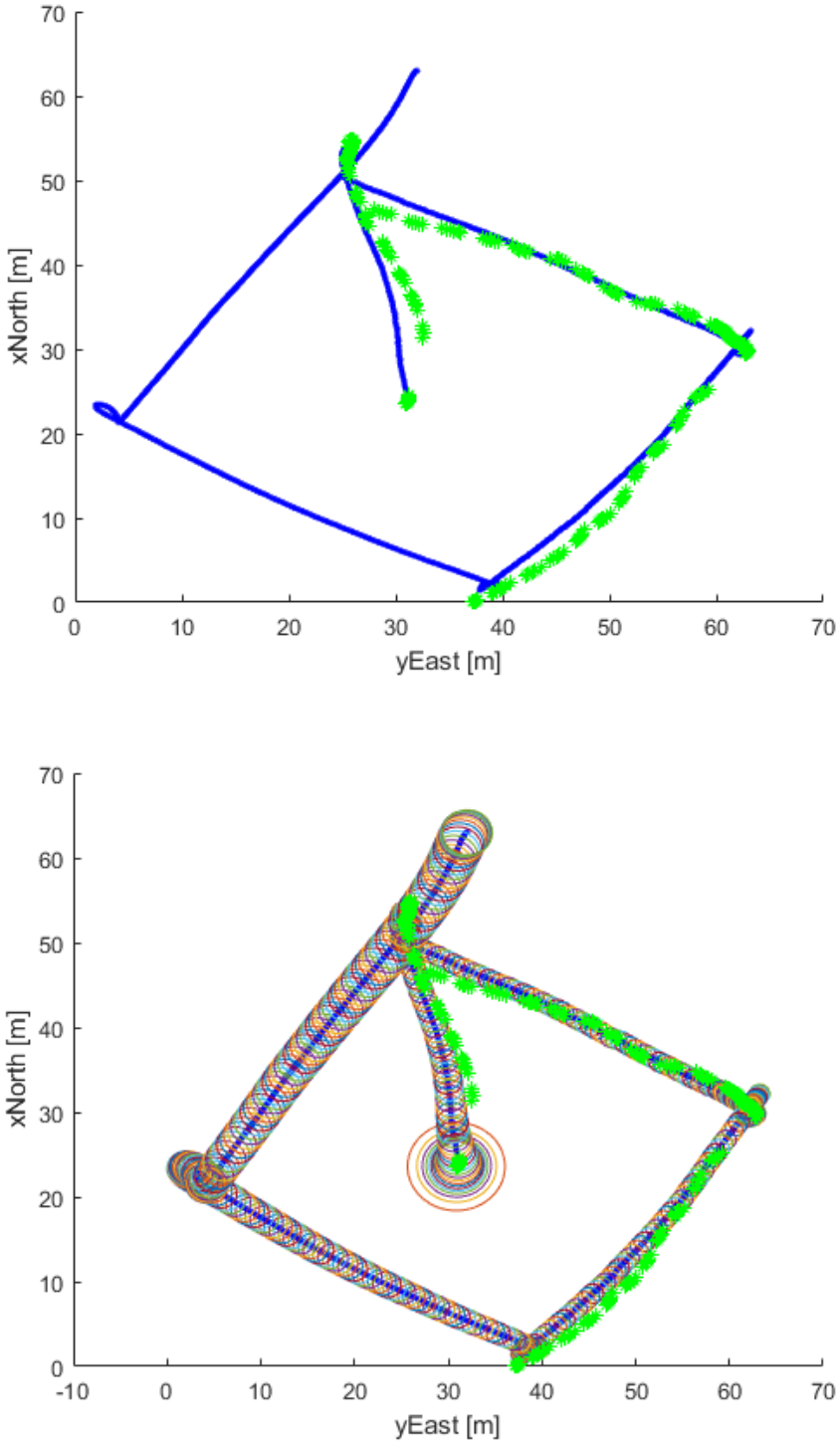


Figure 21: confronto tra la traiettoria stimata dell'AUV (linea blu) e i dati del GPS (punti verdi). Sotto viene riportata la stessa figura con l'aggiunta degli ellissoidi d'incertezza (semiassi pari a tre volte la deviazione standard)

Nelle figure precedenti sono confrontate la traiettoria stimata dell'AUV e i dati del GPS. In Fig.24 è riportata l'ellisse d'incertezza per ogni posizione calcolata con semiassi pari a tre volte la deviazione standard. Dall'analisi delle figure si evince che la posizione stimata segue il GPS, ma risente anche dell'influenza di DVL e AHRS (la misura di questi sensori è maggiormente pesata rispetto al GPS, in quanto ad esse è associata una deviazione standard minore).

8. FASE FINALE: INTEGRAZIONE TRA I VARI BLOCCHI E SIMULAZIONE TOTALE

In seguito all'elaborazione dei singoli blocchi, affrontata individualmente dai gruppi, ne è stata fatta l'integrazione all'interno del template complessivo. Dopo una fase di confronto necessaria all'individuazione e rimozione di errori legati al codice e alla temporizzazione, sono state effettuate delle prove per il settaggio di alcune soglie. In questa sezione vengono riportati i plot dei test maggiormente significativi nell'analisi dei risultati.

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}4\ m/s$

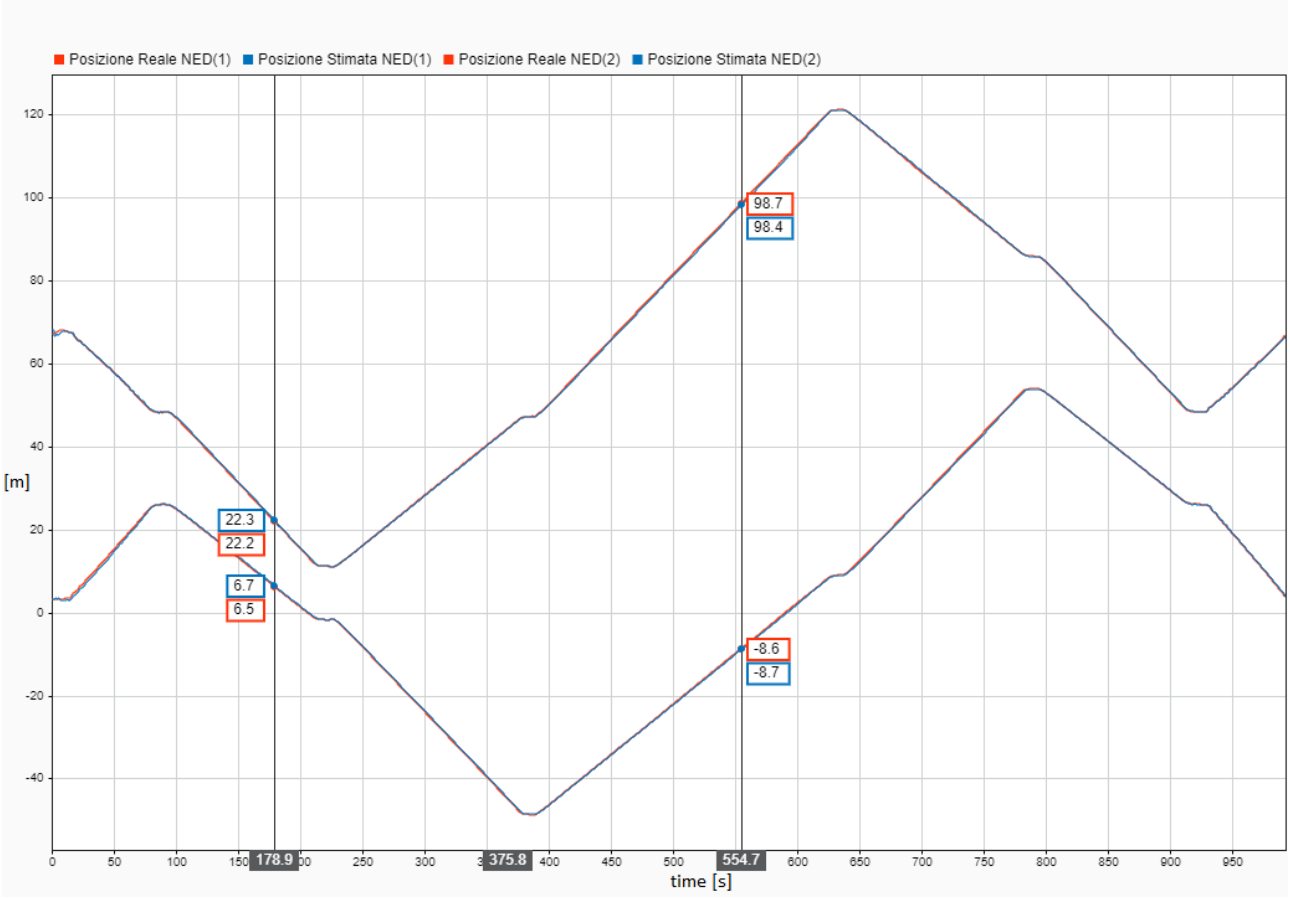


Figura 22A: confronto tra posizione stimata e reale (ottenuta in uscita dal Vehicle Model)

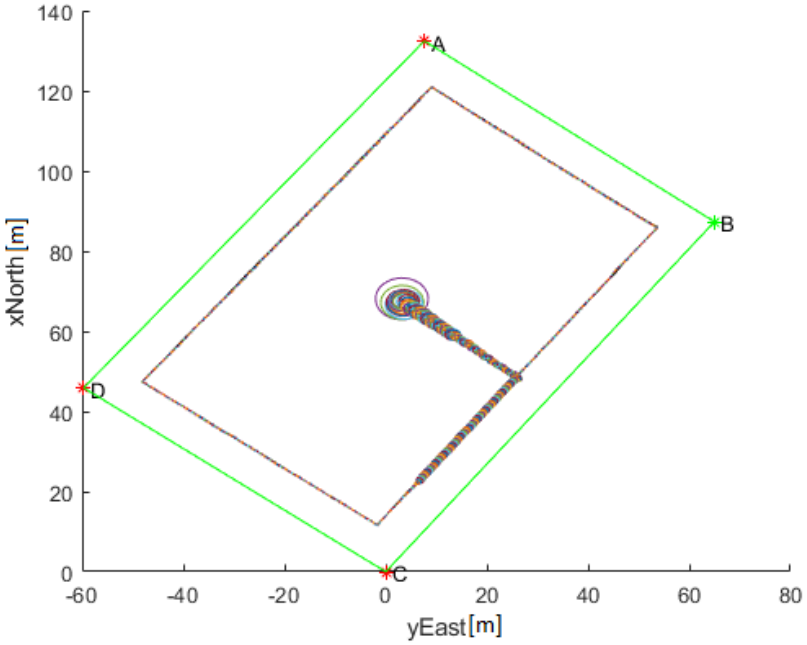


Figura 25B: traiettoria dell'AUV nel bacino con ellissi d'incertezza (semiassi pari a tre volte la deviazione standard)

Figura 23: prime due componenti dell'errore assoluto della posizione stimata riferita alla posizione reale (ottenuta in uscita dal Vehicle Model)

In Fig. 26 è riportato l'errore tra la posizione stimata lungo x e y in terna NED con quella reale (in uscita dal blocco V*ehicle Model)*. Si osserva che l'errore in valore assoluto a regime sulle singole componenti non supera i $\pm 40\ cm$ durante il pattugliamento. Errori più elevati sono solo presenti nei primi istanti durante la fase di inizializzazione.

Nelle Fig.27 e Fig.28 è riportato il confronto tra profondità stimata e reale e la loro differenza. In questo caso la variazione non supera i $\pm 15\ cm$.

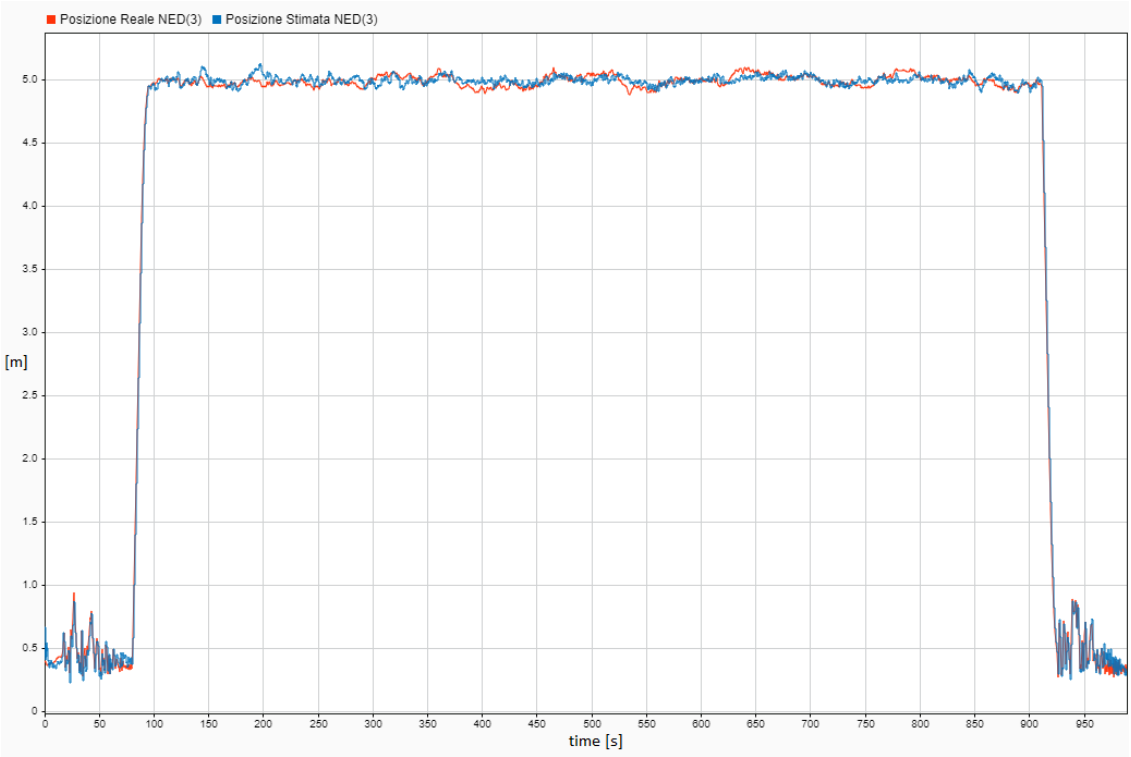


Figura 24: confronto tra profondità stimata e profodità reale(ottenuta in uscita dal Vehicle Model)

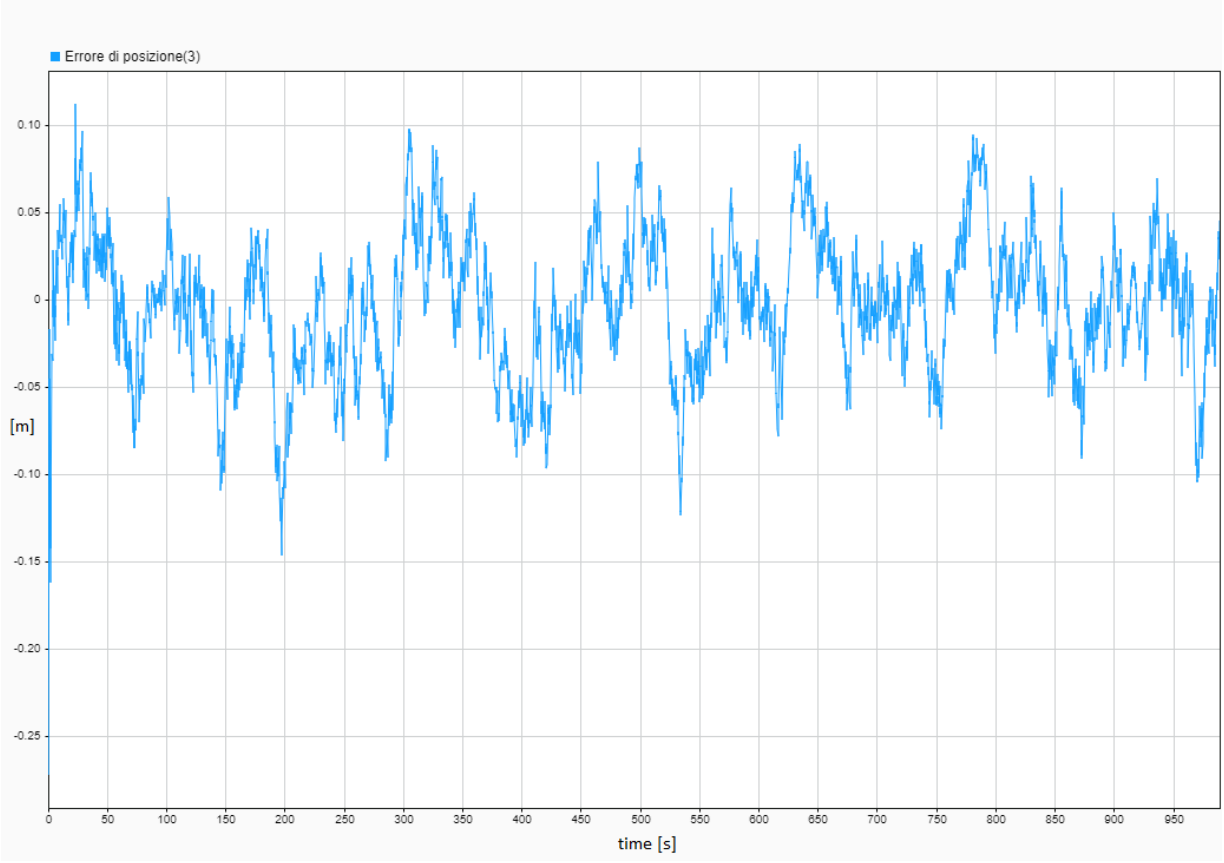


Figura 25: Errore assoluto della profondità riferita alla profondità reale (ottenuta in uscita dal Vehicle Model)

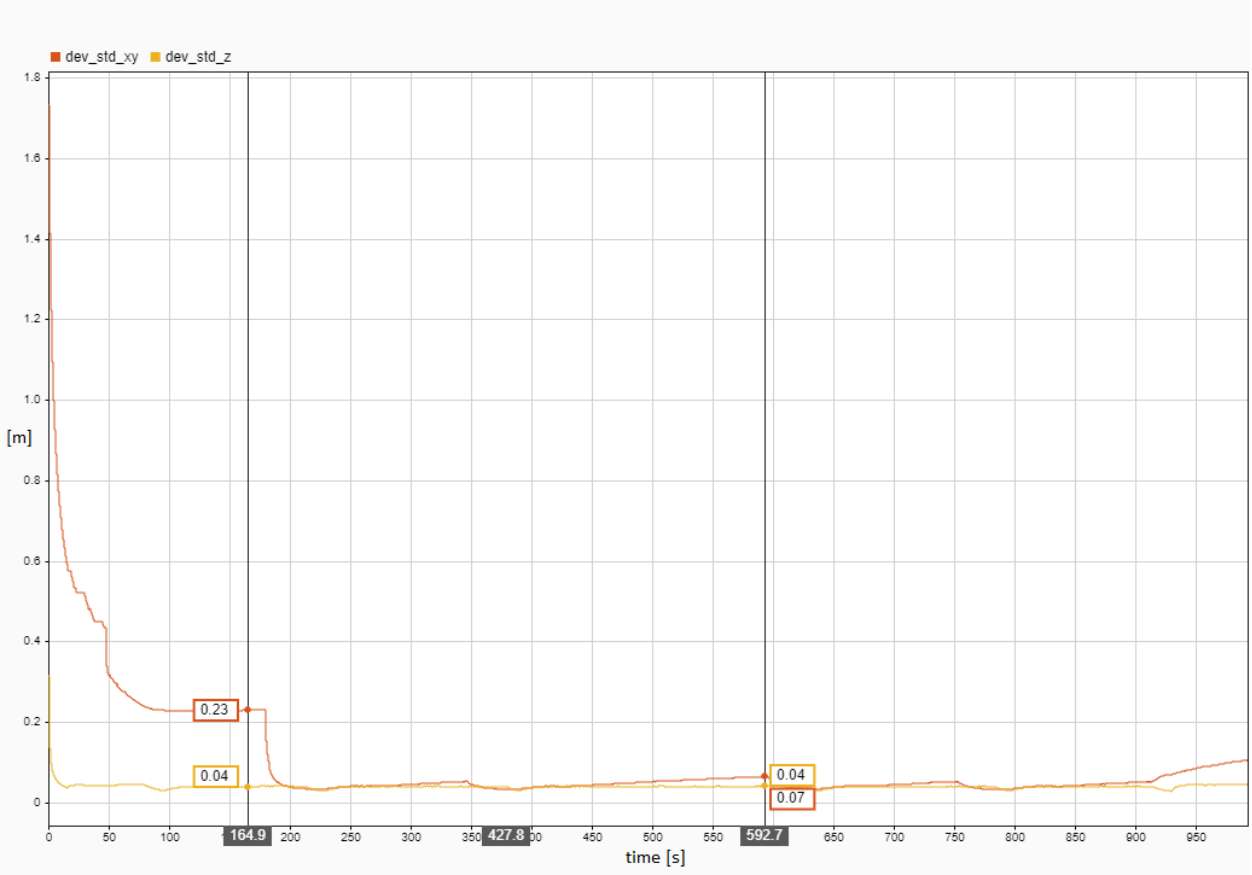


Figura 26: deviazioni standard legate alla posizione stimata su piano x-y (rosso) e legata alla profondità (giallo)

Si può notare infine l'andamento della deviazione standard della posizione stimata: nell'istante di inizializzazione del filtro vengono presi solo i dati del GPS, per questo motivo nella P_0 sono stati inseriti come primi valori le varianze del GPS. Il filtro poi inizia a elaborare dati di GPS, DVL e AHRS, con conseguente diminuzione della deviazione standard che si assesta a circa $20\,cm$, valore giustificato dall'alta precisione di questi ultimi due sensori. Successivamente durante il pattugliamento in profondità, il solo utilizzo dei sonar aumenta notevolmente l'accuratezza della stima ($\sigma_{std} \approx 5\,mm$), poiché sono sensori con varianza inferiore. Le interferenze dei fasci dei sonar sulla superficie dell'acqua possono considerevolmente alterarne la misura, ragion per cui il corretto funzionamento di questi sensori è apprezzabile quando l'AUV si trova a profondità adeguate, dipendente dalla lunghezza del fascio e quindi dalla distanza dalla parete osservata.

L'accuratezza della stima sulla profondità invece è stabile dato che sono sempre utilizzate le misure fornite dal profondimetro.

9. SIMULAZIONI PER LA SCELTA DELLE SOGLIE SUI SONAR

Come precedentemente riportato, sono presenti tre diversi casi in cui le misure dei sonar non vengono utilizzate per il passo di correzione. Di seguito sono trattati dei casi esplicativi che mostrano il miglioramento dovuto all'introduzione della soglia in termini di diminuzione significativa dell'errore di stima. L'analisi sul benificio della soglia viene effettuata mantenedo le altre due attive.

Soglia 1: limite sulla distanza misurata dal sonar

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$10°$	$0{,}4\ m/s$

Vengono qua riportati i grafici dei sonar senza soglia sulla distanza misurata e con le altre soglie in funzione. Dall'andamento del sonar sinistro e il flag relativo (colore arancione) si osserva che la misura rumorosa viene usata o meno per il passo di correzione.

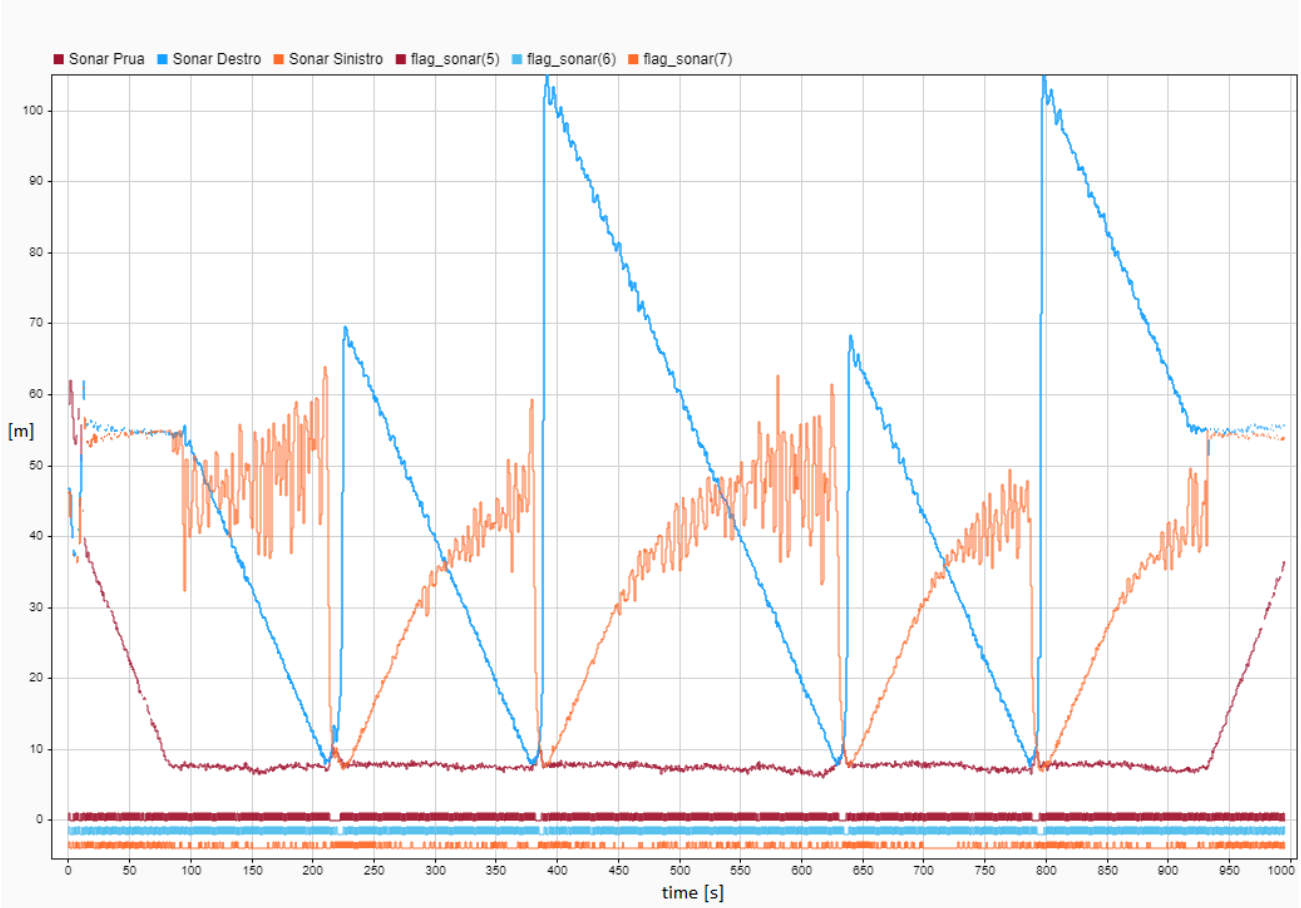


Figura 27: grafico senza soglia 1 sulla distanza misurata, flag di funzionamento dei sonar in basso (flag=1 indica l'utilizzo del sensore nella correzione)

Figura 28: grafico degli errori assoluti sulla posizione stimata in assenza della soglia 1

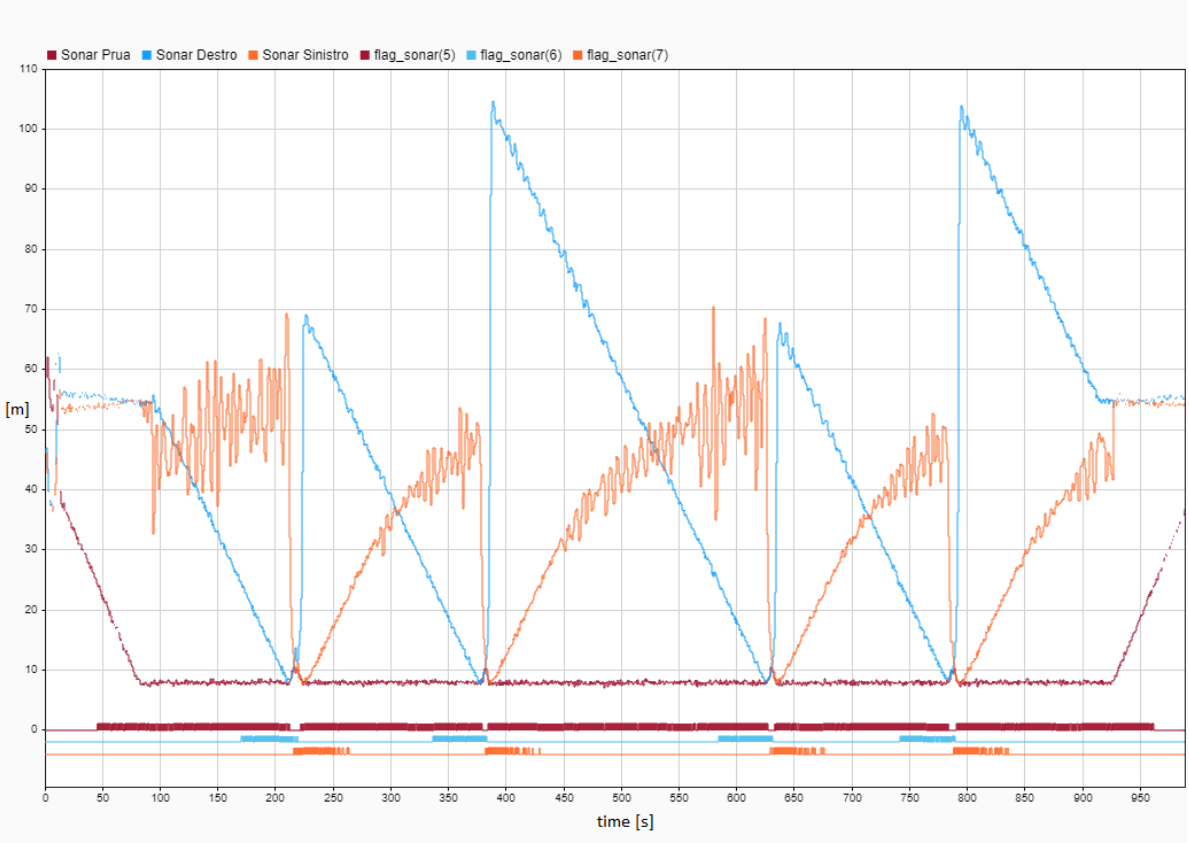


Figura 29: grafico con la soglia 1 sulla distanza misurata, flag di funzionamento dei sonar in basso (flag=1 indica l'utilizzo del sensore nella correzione).

Figura 30: grafico degli errori assoluti sulla posizione stimata nel caso con la soglia 1 sulla distanza misurata

Si nota una netta diminuzione dell'errore di stima, dovuto allo scarto della misura del sonar se supera la soglia di distanza prefissata.

Soglia 2: limite sulla variazione tra due misure successive

Vengono illustrati i risultati ottenuti dalla prova effettuata con i seguenti dati.

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	5°	$0{,}4\ m/s$

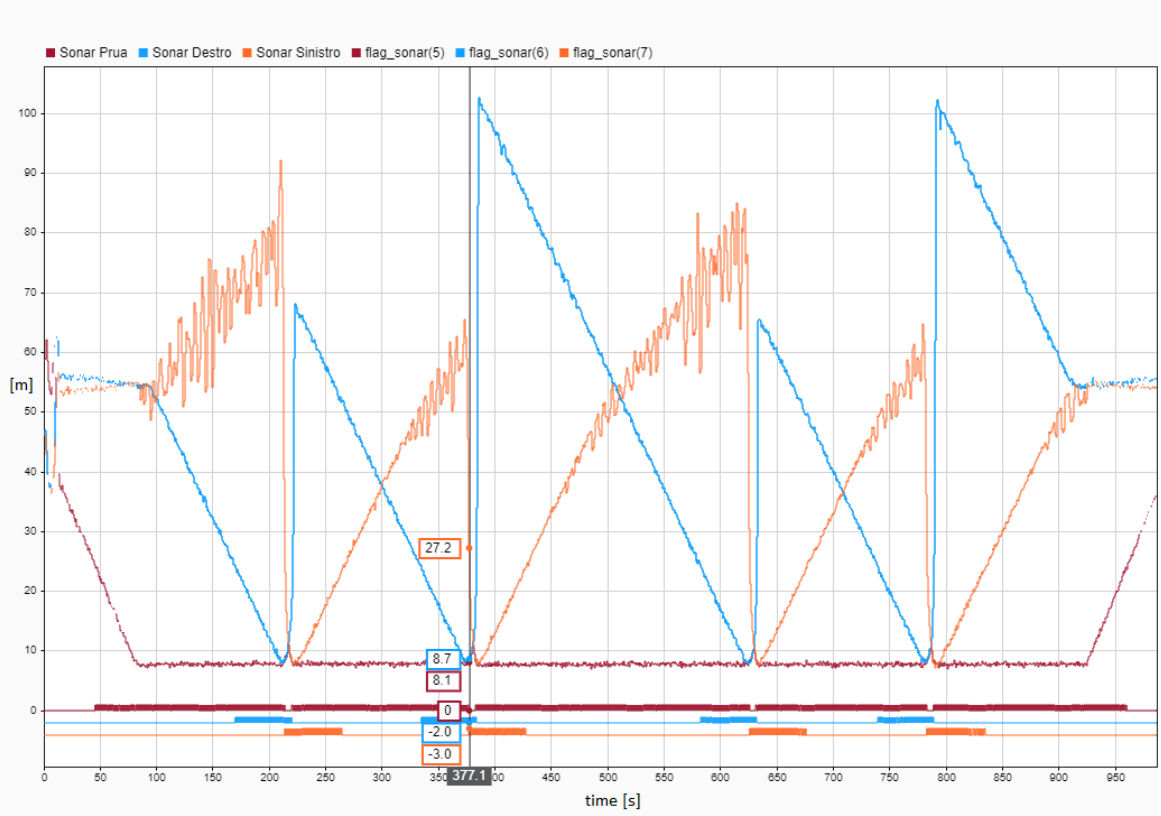


Figura 31: grafico senza soglia sulla variazione, flag di funzionamento dei sonar in basso (flag=1 indica l'utilizzo del sensore nella correzione)

Figura 32: grafico degli errori assoluti sulla posizione stimata in assenza della soglia

All'istante $377.1\ s$ si ha proprio il caso in cui la misura cambia repentinamente per il sonar sinistro (Fig.17 linea arancione), e si osserva infatti un aumento di circa $15\ cm$ dell'errore sulla stima di posizione lungo la seconda componente. La soglia scelta da prove sperimentali è di $0{,}8\ m$.

Soglia 3: limite sulla vicinanza dallo spigolo

La distanza minima tra un qualsiasi spigolo del bacino e l'intersezione sonar-parete è stata posta a $5\ metri$, oltre tale valore la misura non viene considerata nella fase di correzione.

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}4\ m/s$

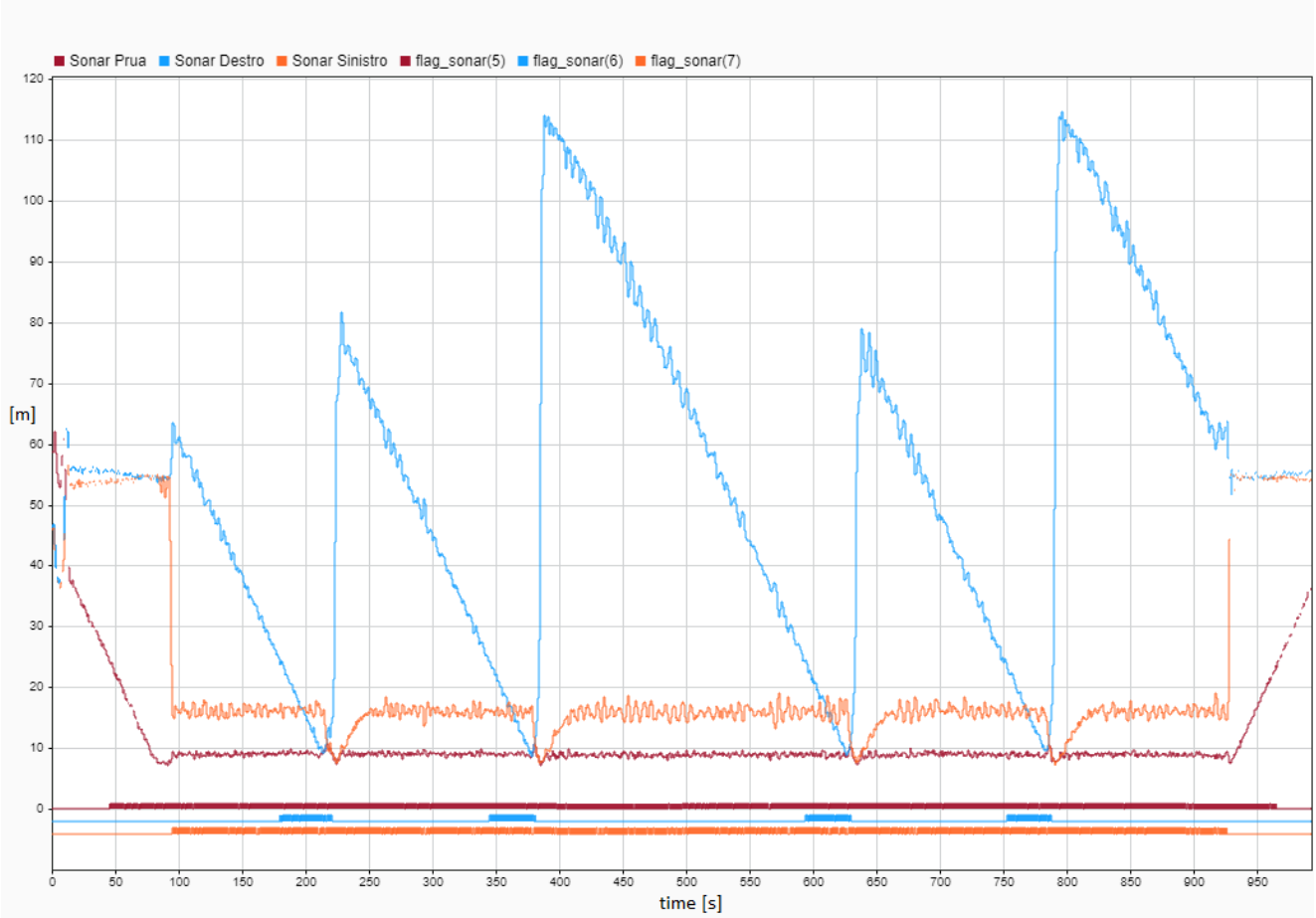


Figura 33: grafico senza soglia sulla vicinanza allo spigolo, flag di funzionamento dei sonar in basso (flag=1 indica l'utilizzo del sensore nella correzione)

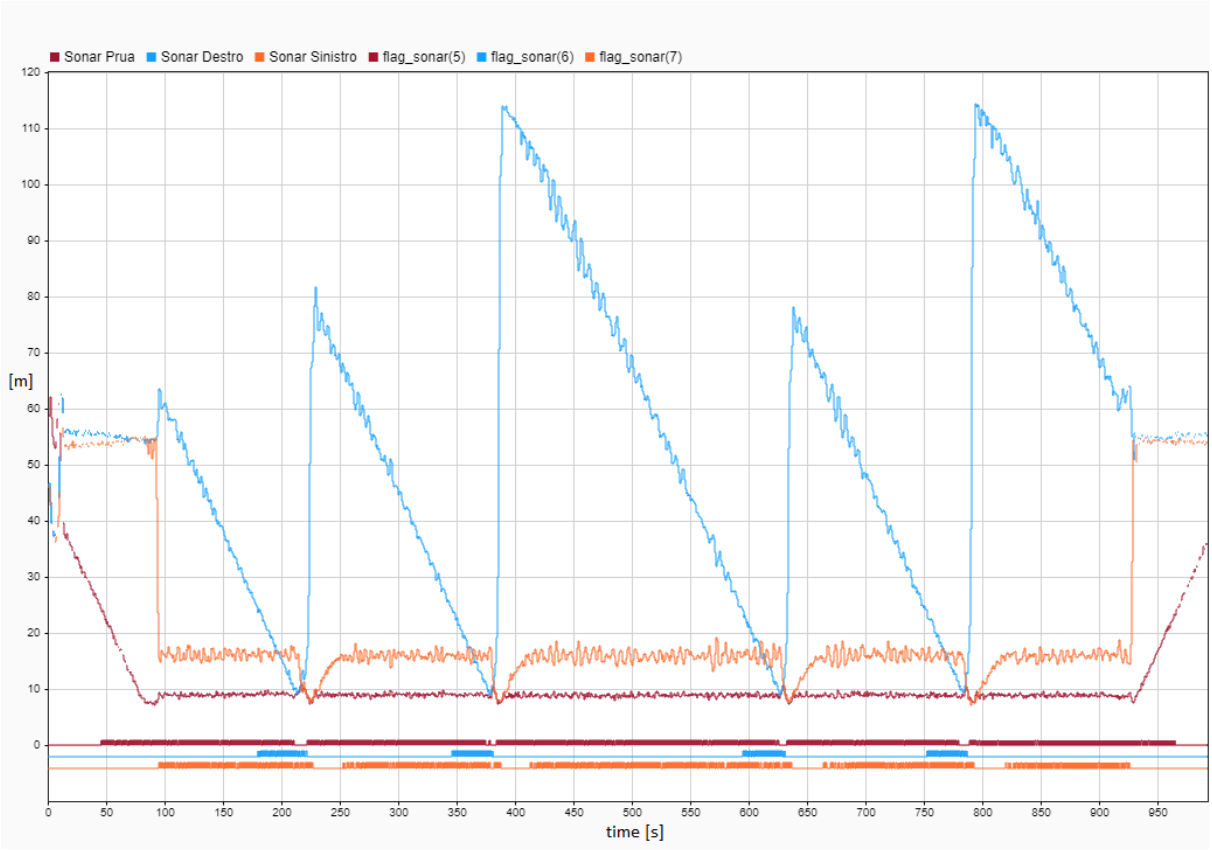


Figura 34: grafico con la soglia sulla vicinanza allo spigolo, valori dei flag in basso

Figura 35:grafico dell'errore di posizione lungo la componente x nei casi con e senza soglia. La linea rossa rappresenta l'errore in assenza della soglia.

Come si vede dalla Fig.39 l'errore assoluto sulla prima componente della posizione stimata migliora con l'aggiunta della soglia (linea verde).
Stessa osservazione si può fare per la seconda componente in Fig.40.

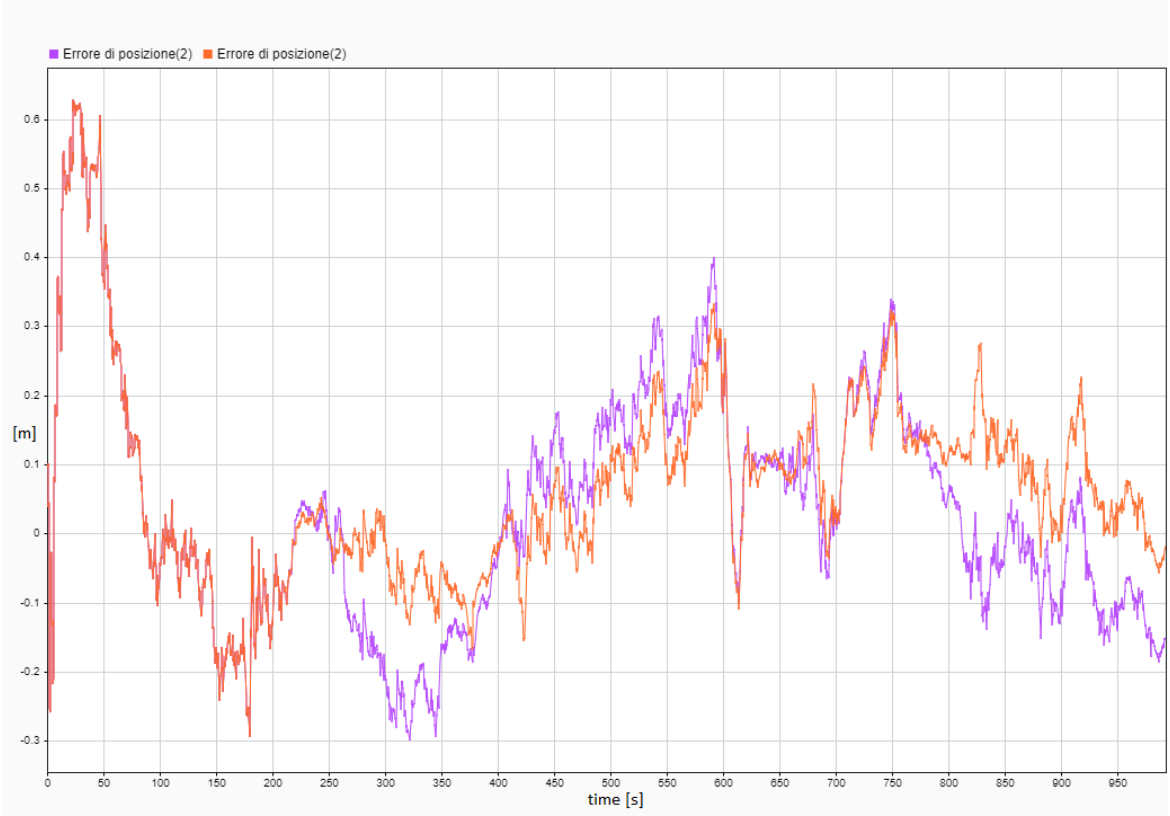


Figura 36: confronto tra l'errore di posizione lungo y, nei casi in cui la soglia sia o meno attiva. La linea viola rappresenta l'errore senza la presenza della soglia.

L'introduzione di queste soglie limita però in alcuni casi l'utilizzo dei sonar, come nel caso:

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$90°$	$0{,}4\ m/s$

Se l'orientazione relativa rispetto alla parete è pari a $90°$, il sonar destro come è ben evidente dalla Fig.41 non è funzionate, in quanto supera la soglia 1. Nonostante ciò, l'assenza del sonar non influenza significativamente l'errore di posizione rispetto al caso in cui tutti i sonar sono funzionanti.

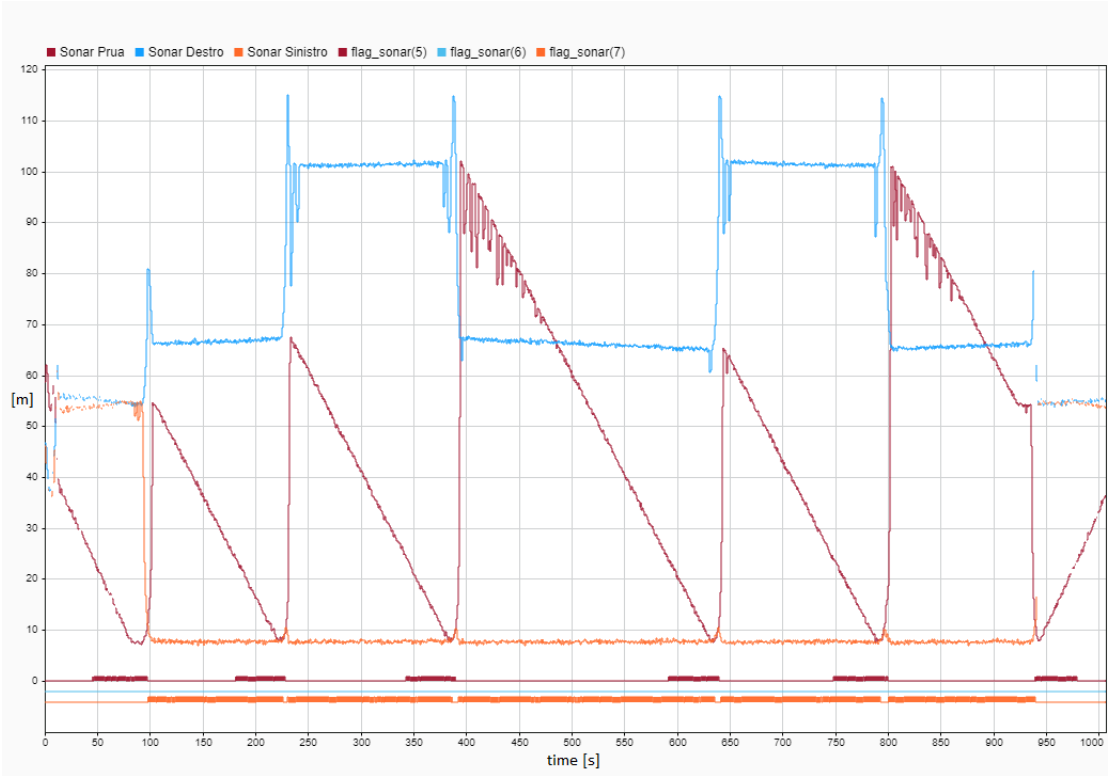


Figura 37: sonar destro inutilizzato, come si nota dal valore del flag (azzurro) corrispondente posto sempre a zero

Figura 38: errore della posizione stimata in assenza del sonar destro

Per il tipo di trattazione fatta, se il valore di un sonar risulta non rispettare una delle tre soglie allora viene scartato. Una possibile variazione consiste nell'effettuare comunque la correzione con tale misura, con un peso inferiore rispetto alle altre (vedi *13. Conclusioni e spunti implementativi per il miglioramento delle prestazioni*).

10. SIMULAZIONE SENZA SONAR

In questa sezione presentiamo i risultati ottenuti da una simulazione in cui **non** sono state effettuate correzioni con i sonar. I dati utilizzati sono:

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}4\ m/s$

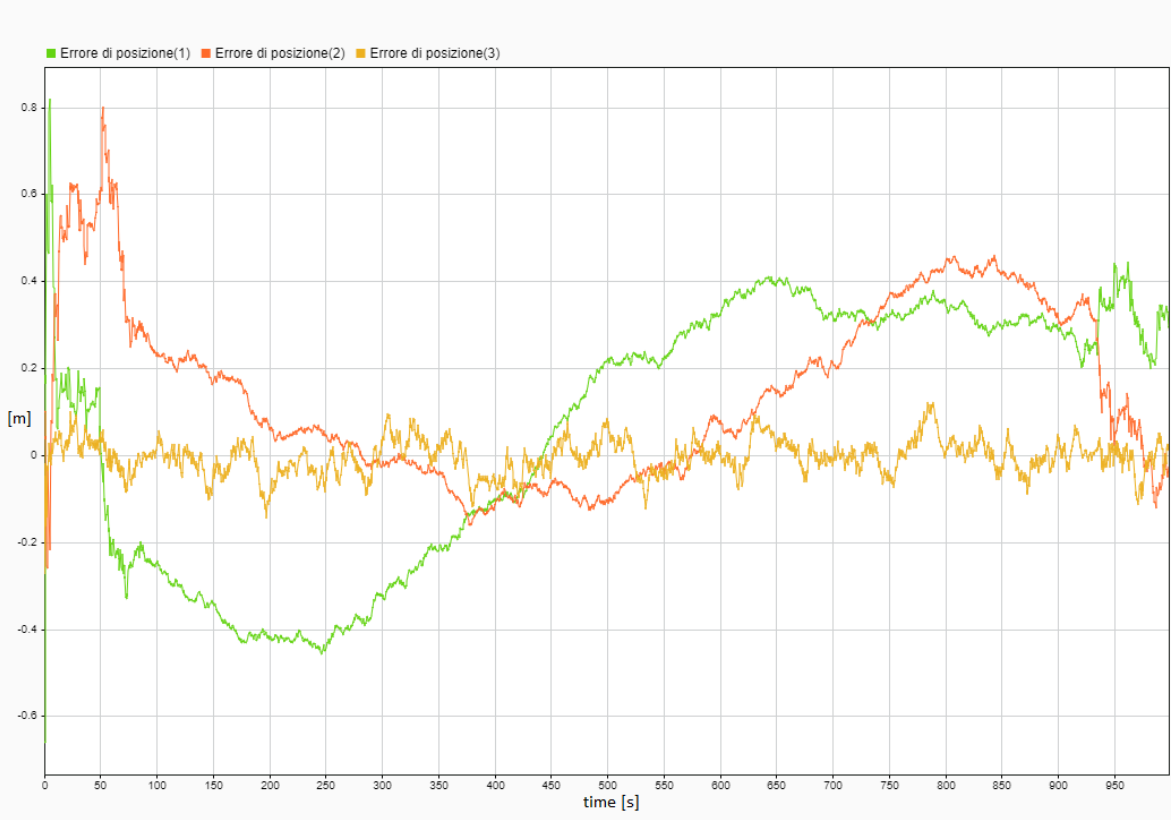


*Figura 42: errori assoluti delle tre componenti della posizione stimata **in assenza** di correzione con i sonar*

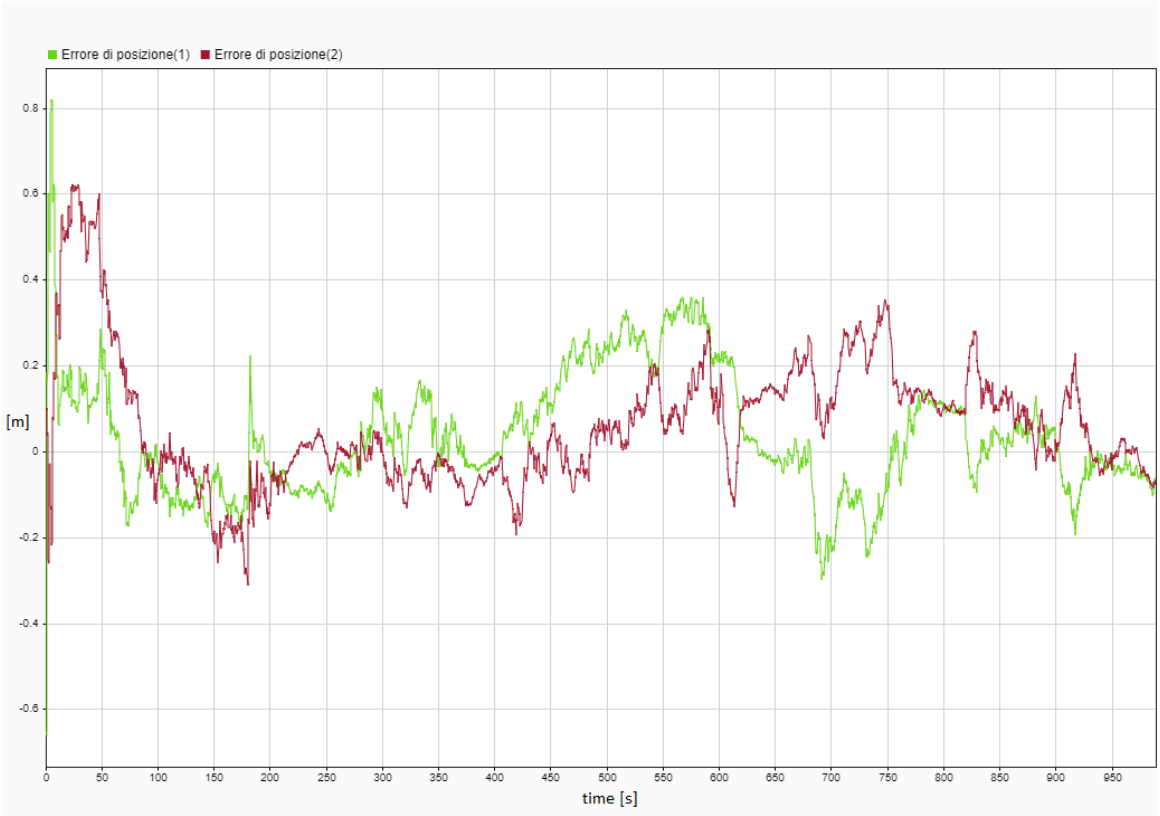


*Figura 43: errori assoluti delle prime due componenti della posizione stimata **con** la correzione dei sonar*

Dal confronto degli errori assoluti delle prime due componenti della posizione stimata si evince un notevole beneficio dall'utilizzo dei sonar nella fase di correzione.

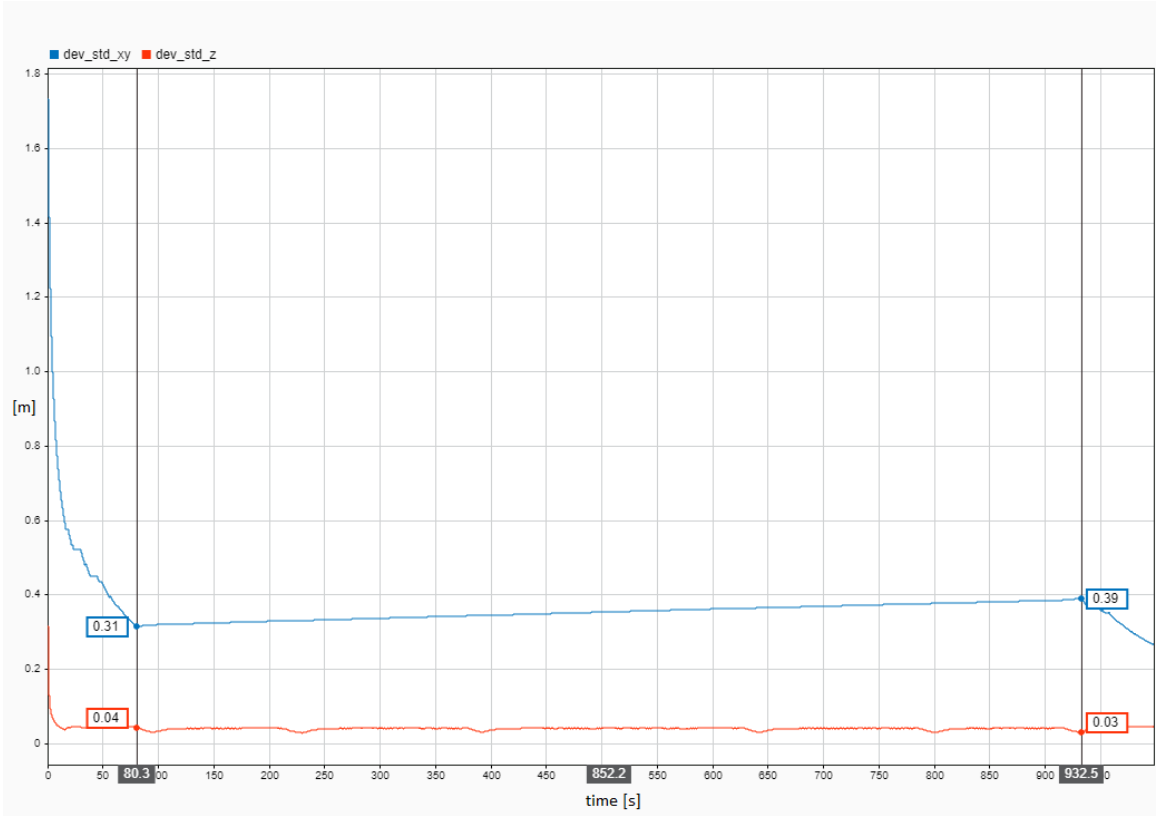


Figura 394: deviazioni standard della posizione stimata in assenza di correzione con i sonar

In assenza dei sonar si nota un incremento costante delle deviazioni standard (da $0.31\ m$ a $0.39\ m$), che non risulta essere eccessivo grazie all'elevata precisione del DVL. Alla fine del pattugliamento si abbassa ulteriormente perché siamo nella fase di riemersione e si riprende la correzione con il GPS.

Se viene aumentata la velocità di pattugliamento dell'AUV si nota l'esistenza di una correlazione tra l'aumento di velocità di pattugliamento dell'AUV e l'incremento della deviazione standard *(Fig.46)* e dell'errore assoluto in *Fig.45*.

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	**Velocità**
BC	$5\ m$	$8\ m$	Orario	58°	$\mathbf{1\ m/s}$

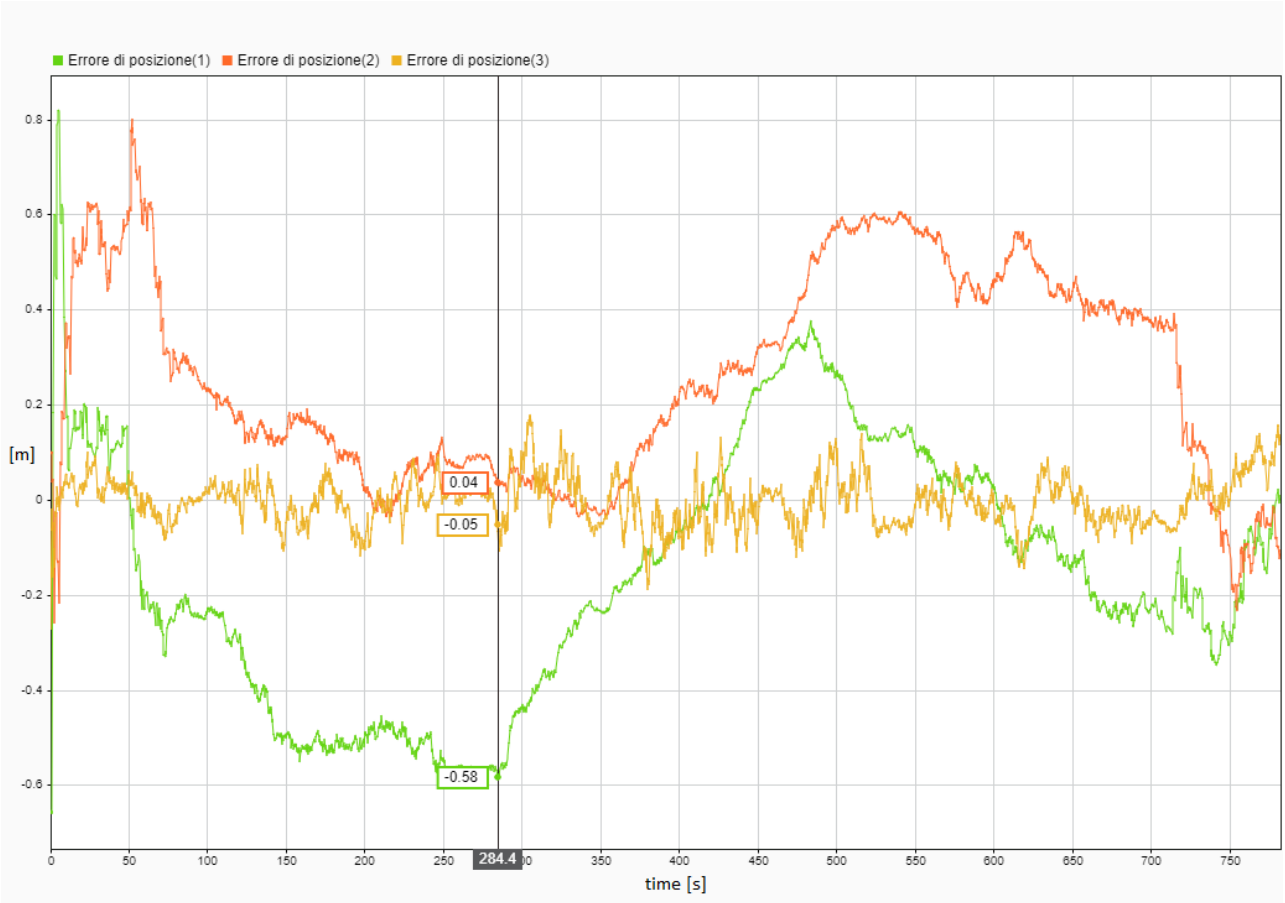


Figura 45: errori assoluti delle tre componenti della posizione stimata senza correzione dei sonar con velocità $\mathbf{1\ m/s}$

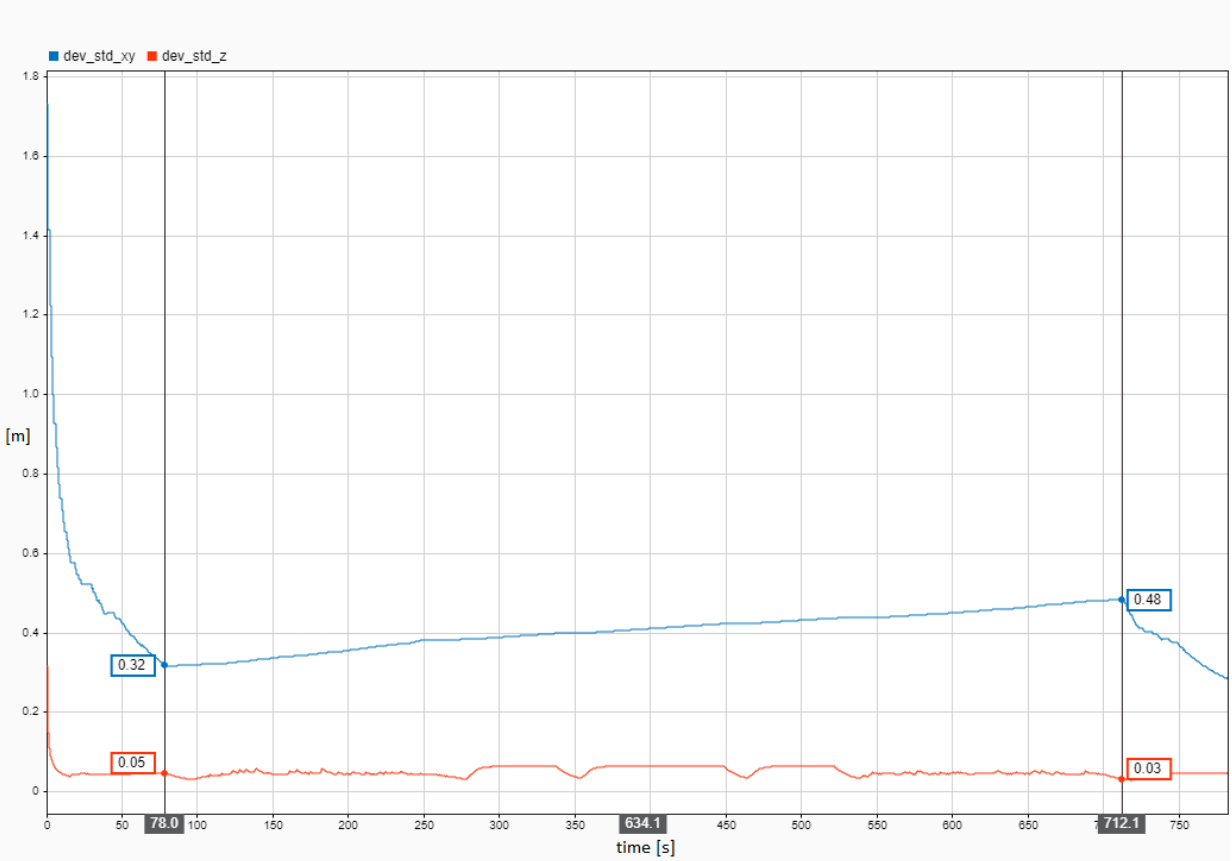


Figura 46: deviazioni standard della stima in assenza di correzione con i sonar con velocità $\mathbf{1}\ \boldsymbol{m/s}$

11. GRAFICI SUI FILTRI A MEDIA CAMPIONARIA

Oltre al filtro di Kalman usato per la stima della posizione, sono stati inseriti dei filtri a media campionaria in uscita ai valori di AHRS, DVL e giroscopio.

Presentiamo a titolo di esempio alcuni grafici relativi a AHRS e DVL ottenuti dalla simulazione effettuata con questi parametri:

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}4\ m/s$

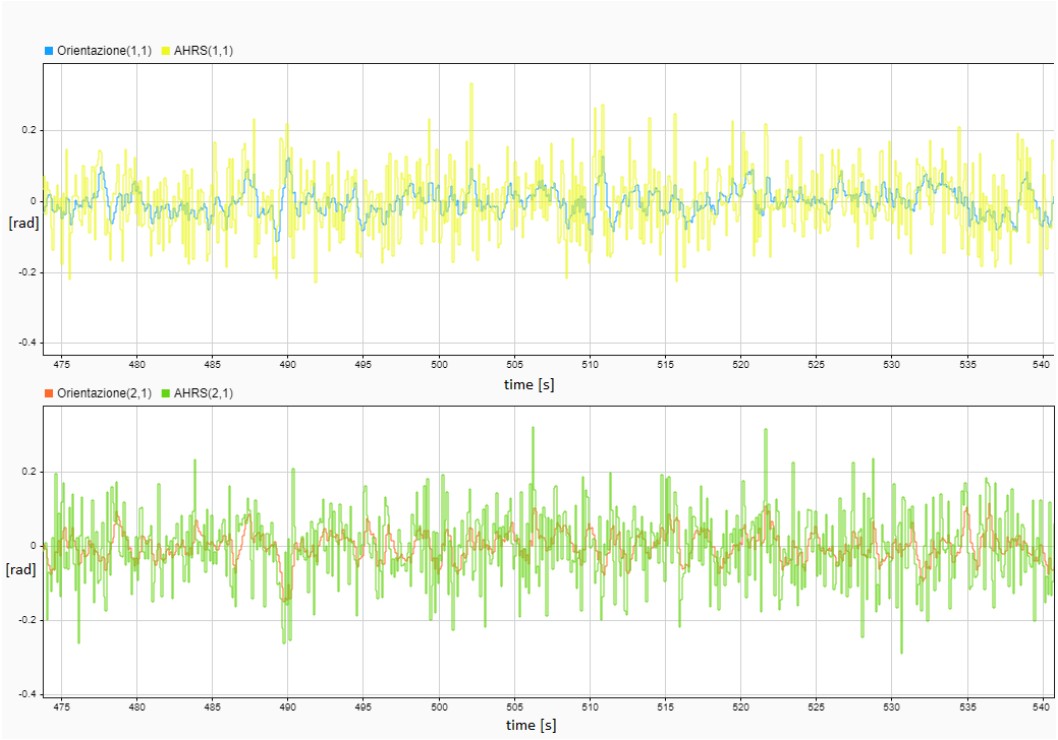


Figura 47: confronto delle prime due componenti dell'AHRS con e senza il filtro (in giallo e verde le misure senza filtro, in azzurro e rosso le altre)

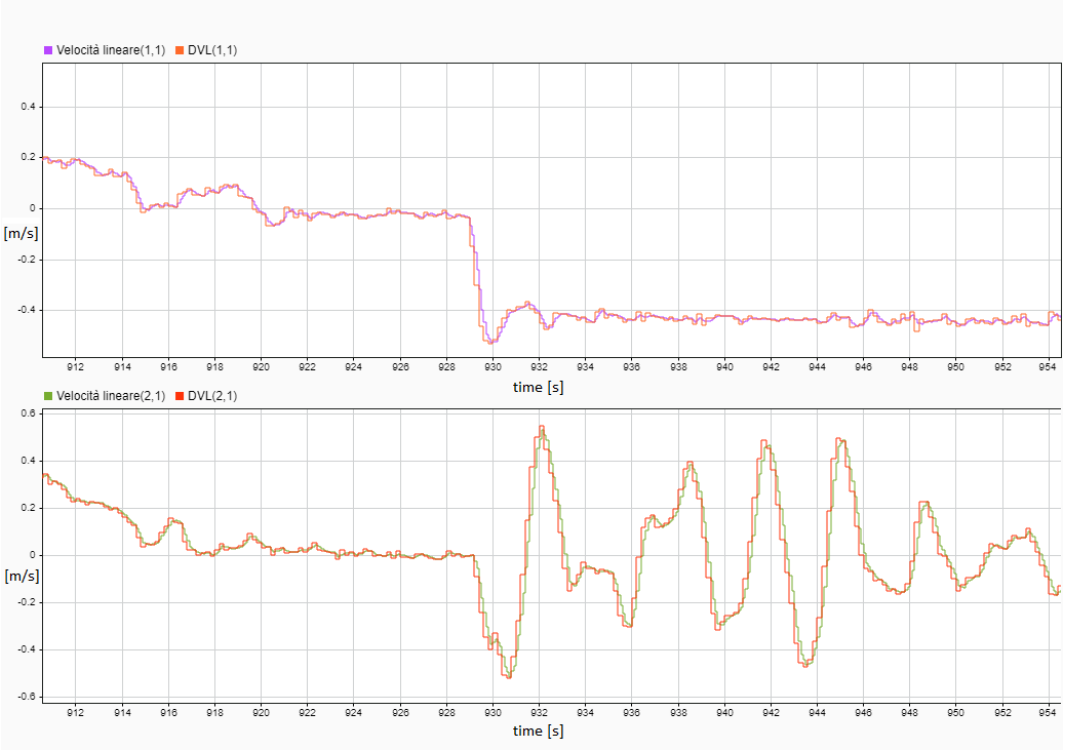


Figura 48: confronto delle prime due componenti del DVL con e senza il filtro (DVL è la misura senza filtro, velocità lineare è la misura con il filtro)

Mentre risulta evidente il miglioramento della misura dell'AHRS dopo l'applicazione del filtro, sembrerebbe non essere riscontrato nel DVL. Ciò nonstante da prove sperimentali effuate insieme al blocco *Controller,* si è osservato un aumento delle prestazioni in quest'ultimo blocco.

12. SIMULAZIONI NEL CASO DI DVL O AHRS NON FUNZIONANTI

Come detto precedentemente, la predizione sfrutta le informazioni provenienti da DVL e AHRS, che vengono passati come ingressi nel nostro modello. Un malfunzionamento di questi due sensori comporterebbe quindi l'inefficienza del filtro; per ovviare a questa situazione è stato deciso, in prima approssimazione, di mantenere l'ultimo valore valido fornito dai sensori e utilizzarlo per gli istanti successivi. Il deterioramento dell'informazione viene rappresentato attraverso un aumento della matrice Q pari a un fattore di 50 per il DVL e di 500 nel caso di AHRS.

Sono state fatte una serie di simulazioni per analizzare il comportamento del filtro in caso di guasto per un lasso di tempo finito sia del DVL che dell'AHRS. I parametri usati per tutte le seguenti prove sono i seguenti:

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}6\ m/s$

DVL non funzionante: durante pattugliamento della parete

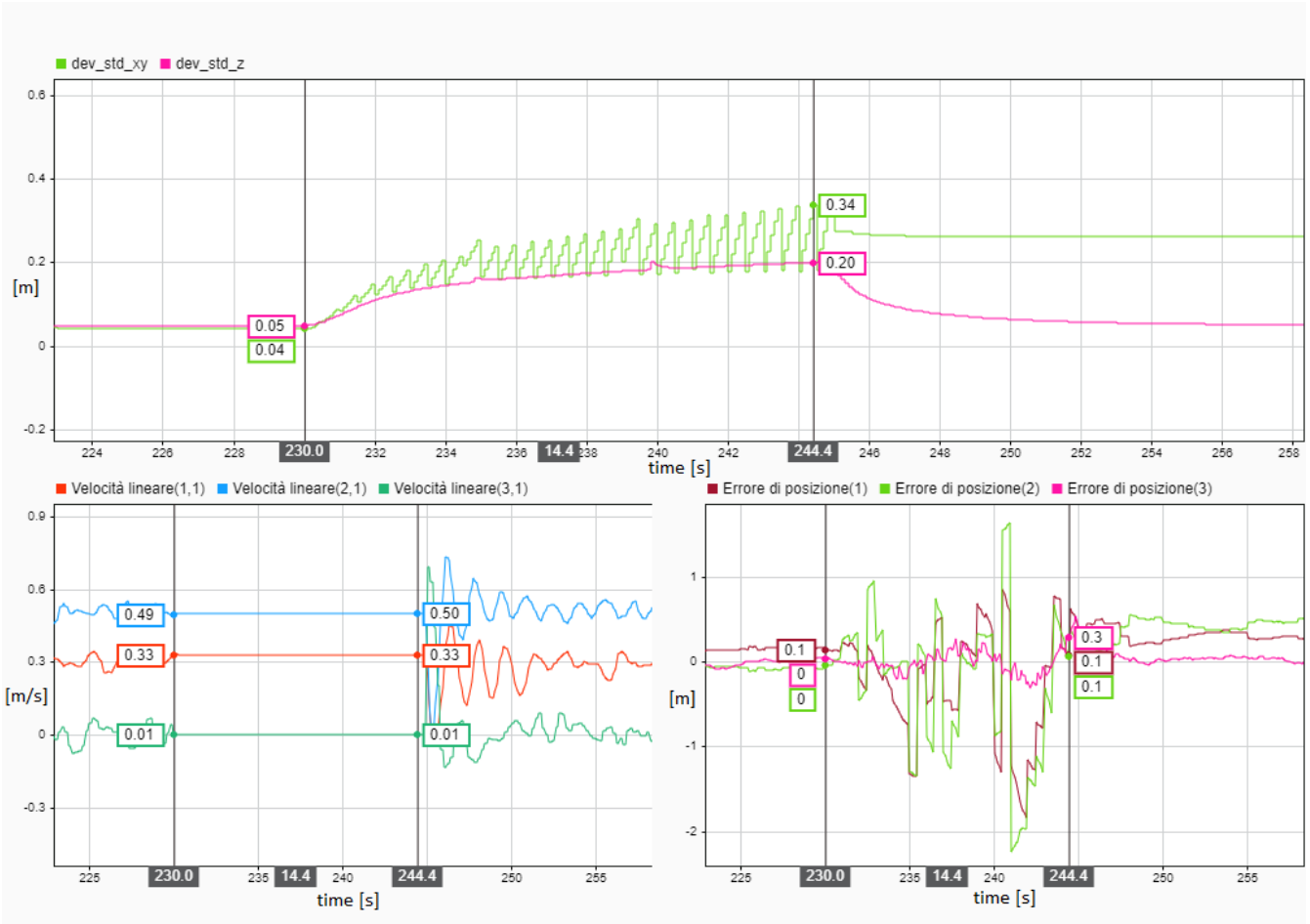


Figura 49: Grafici della deviazione standard, velocità lineare ed errore di posizione della stima in caso di rottura del DVL da $230\ s$ a $245\ s$.

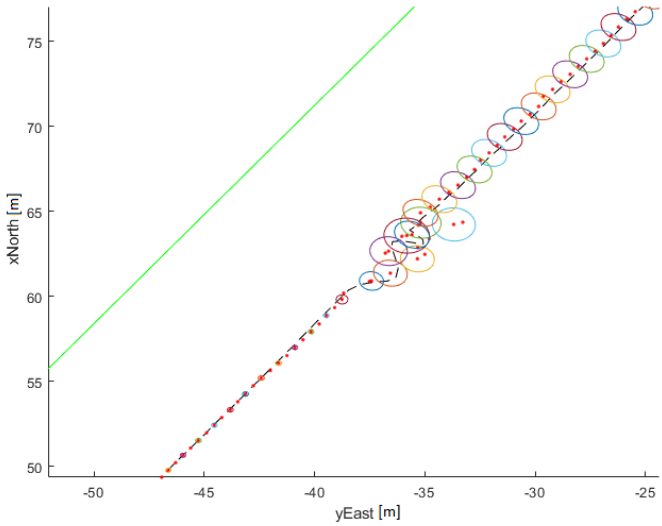


Figura 50: confronto tra posizione stimata (punti rossi) e posizione reale (tratti neri), con ellissi d'incertezza (i semiassi sono tre volte le deviazioni standard).

In questa simulazione si è ipotizzato un malfunzionamento del DVL durante la fase di pattugliamento di una delle pareti. Si nota in corrispondenza di tali istanti l'aumento della deviazione standard e dell'errore della posizione stimata. Grazie all'azione correttiva dei sonar, la stima non sembra subire un deterioramento significativo.

DVL non funzionante: durante rotazione a fine pattugliamento parete

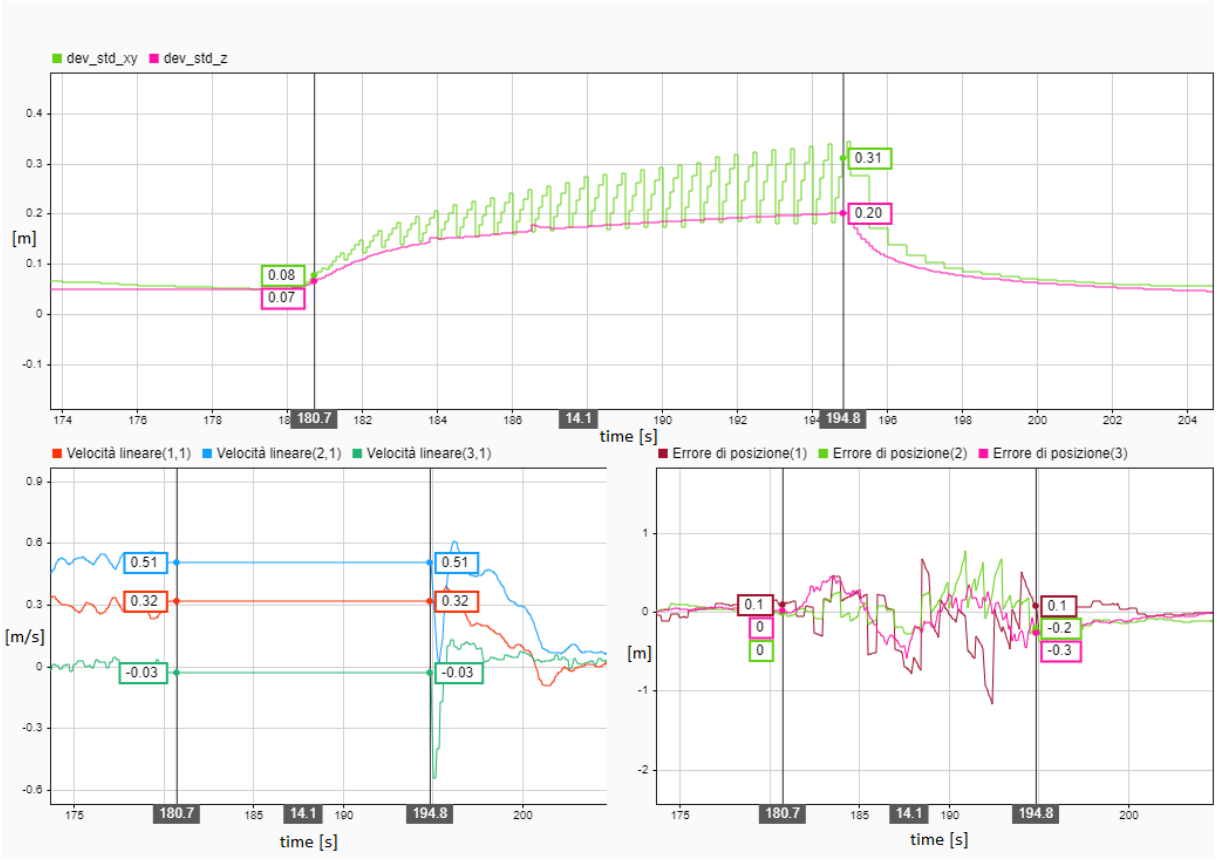


Figura 51: Grafici della deviazione standard, velocità lineare ed errore di posizione della stima in caso di rottura del DVL da 180s a 195s.

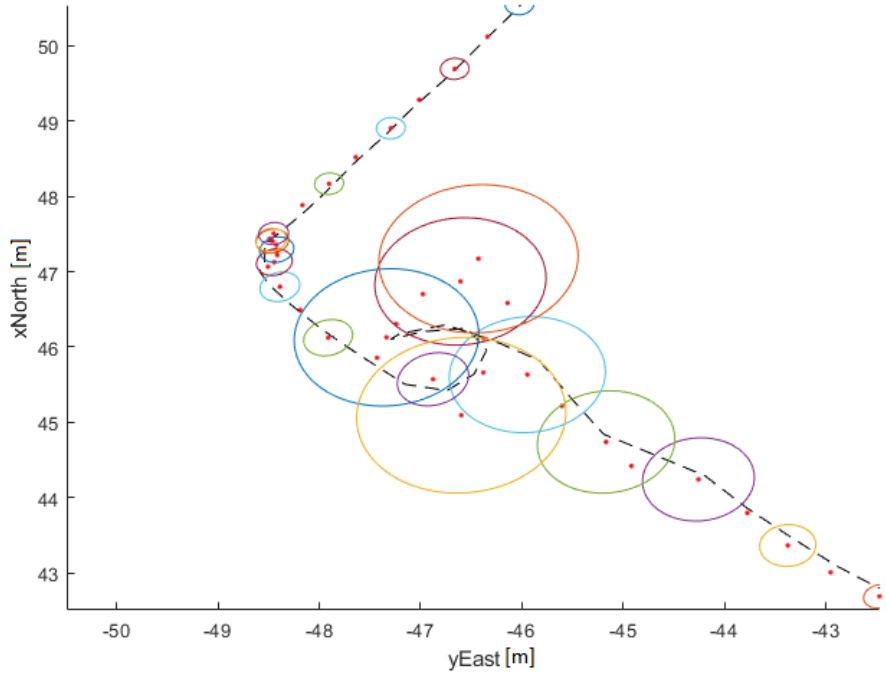


Figura 52: confronto tra posizione stimata (punti rossi) e posizione reale (tratti neri), con ellissi d'incertezza (i semiassi sono tre volte le deviazioni standard).

Se il malfunzionamento del DVL avviene in prossimità di uno spigolo del bacino, quindi nella fase di rallentamento e rotazione del veicolo per cambiare la parete pattugliata, si ha un netto peggioramento delle prestazioni del filtro. Nonostante ciò, queste sono sufficienti per evitare una collisione con la parete.

Possiamo infatti notare nella *Fig.52* che l'AUV segue una particolare traiettoria dovuta all'azione del controllo e al fatto che il sistema non passa al pattugliamento della parete successiva fino a quando la misura della velocità non ritorna valida. L'azione simultanea dei vari blocchi del sistema integrale risolve questo problema, nonostante la stima ne risulti compromessa.

AHRS non funzionante: durante pattugliamento della parete

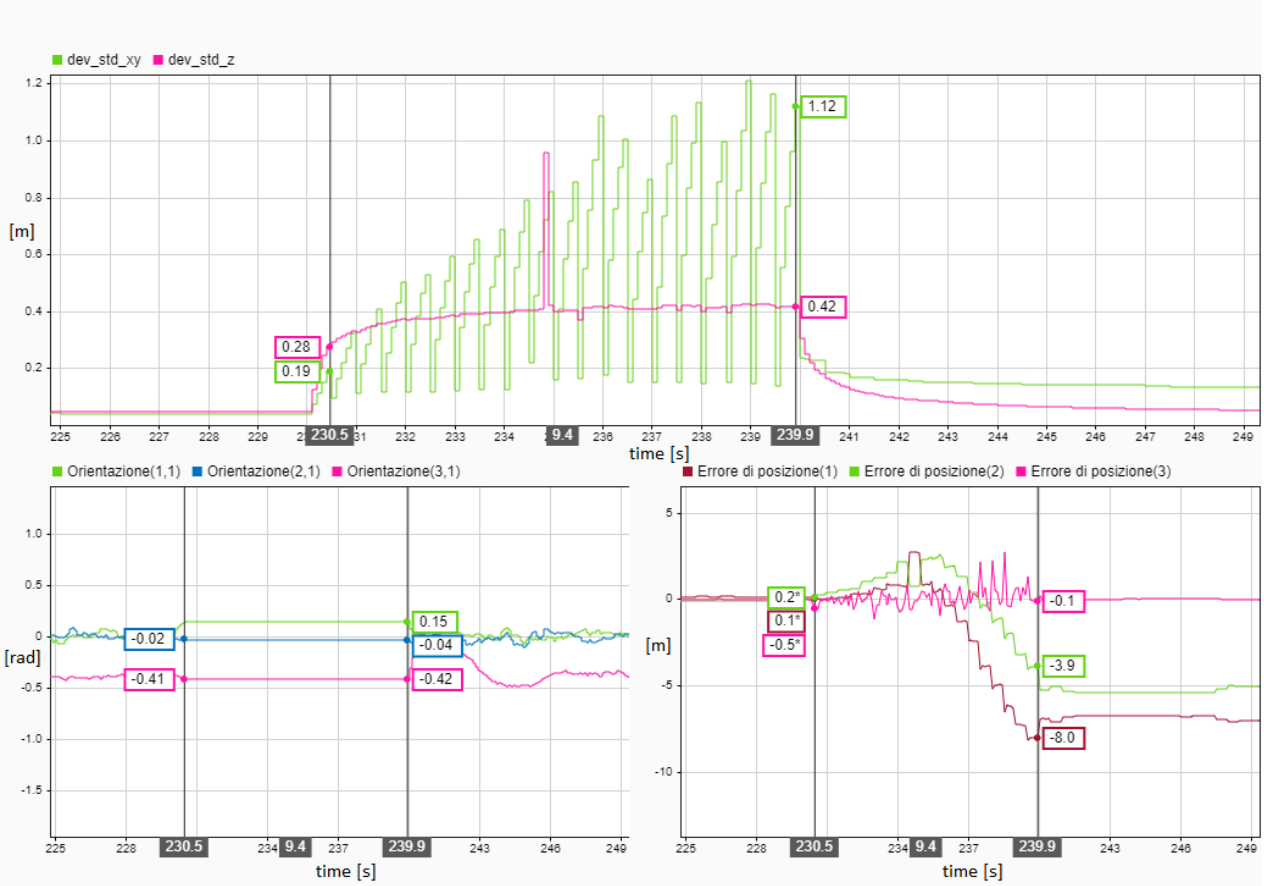


Figura 53: Grafici della deviazione standard, orientazione ed errore di posizione della stima in caso di rottura dell'AHRS da 230s a 240s.

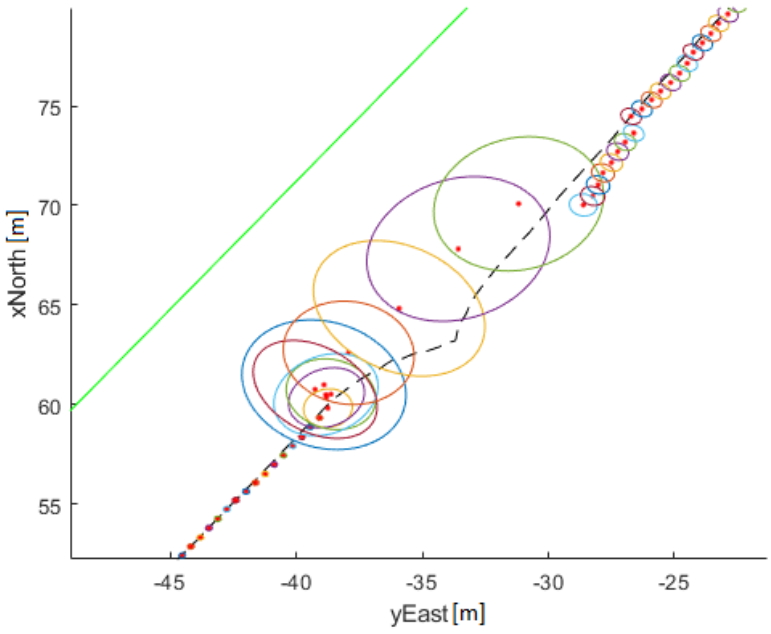


Figura 54: confronto tra posizione stimata (punti rossi) e posizione reale (tratti neri), con ellissi d'incertezza (i semiassi sono tre volte le deviazioni standard) in caso di rottura dell'AHRS da 230s a 240s.

Il malfunzionamento dell'AHRS invece è più dannoso per il comportamento del filtro, in quanto l'orientazione viene utilizzata sia per la fase di predizione che per tutti i calcoli geometrici delle misure virtuali dei sonar. Per tale ragione la deviazione standard associata alla posizione stimata ne risente maggiormente, ma anche la correzione con i dati provenienti dai sonar è falsata. In questo caso l'errore con cui viene prevista la posizione dell'AUV può essere superiore alla distanza reale dell'AUV stesso con la parte, portandolo alla collisione con la parete.

AHRS non funzionante: durante rotazione a fine pattugliamento parete

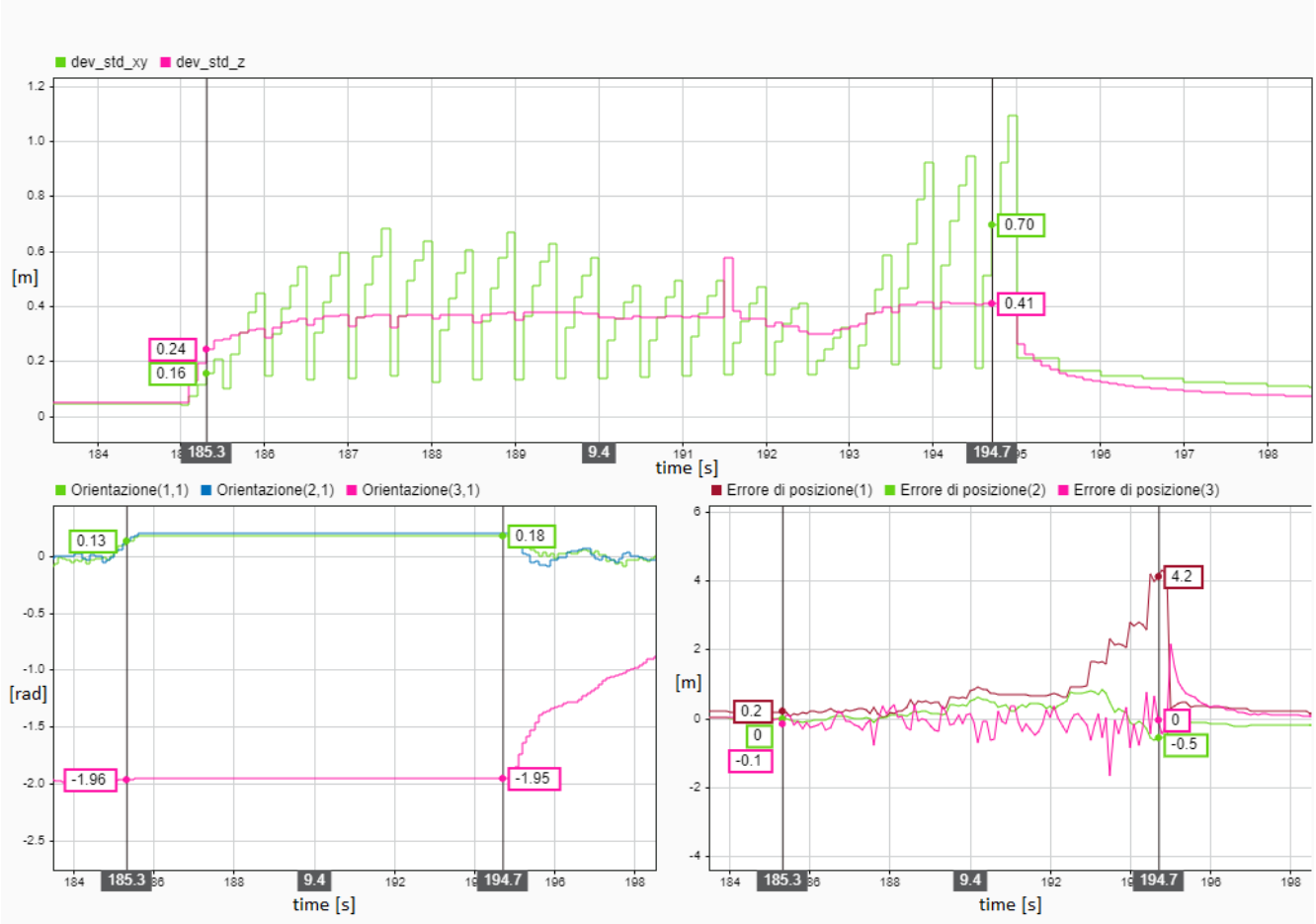


Figura 55: Grafici della deviazione standard, orientazione ed errore di posizione della stima in caso di rottura dell'AHRS da $185s$ *a* $195s$.

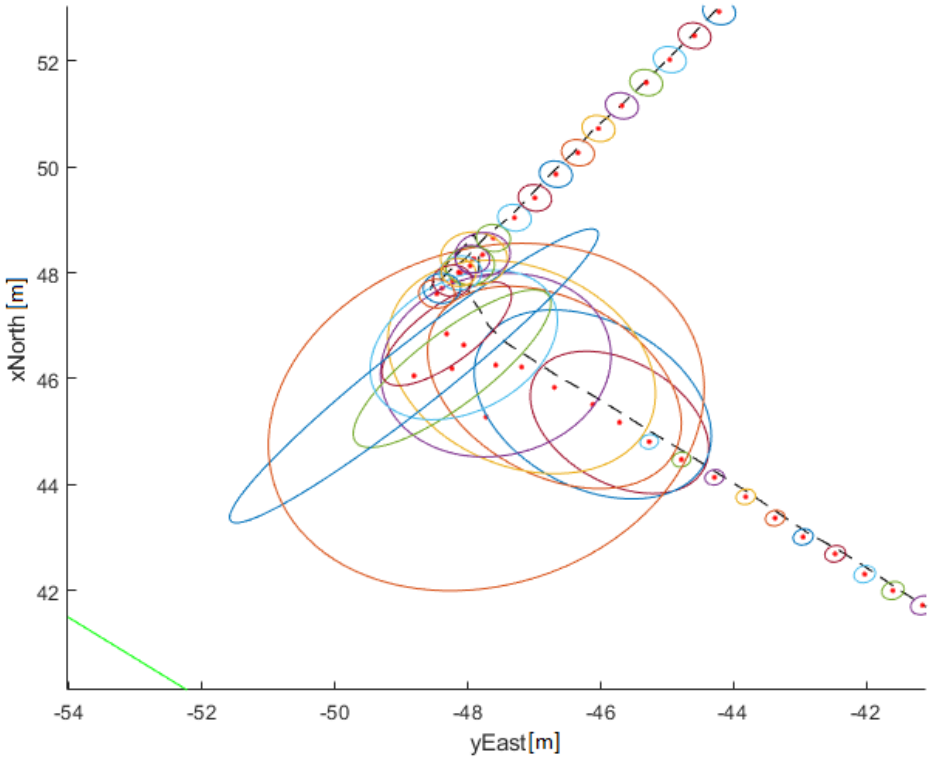


Figura 56: confronto tra posizione stimata (punti rossi) e posizione reale (tratti neri), con ellissi d'incertezza (i semiassi sono tre volte le deviazioni standard).

Anche in questa situazione, mantenere l'ultimo valore d'orientazione valido nel tempo blocca l'aggiornamento del *current_state* del sistema e quindi l'esecuzione della missione quando ci troviamo in corrispondenza della manovra in uno degli angoli. Appena riprende il funzionamento dell'AHRS e avviene la manovra nell'angolo, l'errore di stima riprende l'andamento usuale. In questo secondo caso l'errore assoluto di posizione risulta essere più contenuto in quanto siamo a velocità quasi nulla.

Le deviazioni standard x-y rappresentate in *Fig.49, 51, 53* e *55* presentano un andamento a dente di sega: infatti nella fase di predizione la σ_{xy} aumenta, mentre in quella di correzione diminuisce repentinamente grazie alle informazioni provenienti dai sonar, a frequenza di $2\ Hz$.

È bene puntualizzare che in queste simulazioni il tempo di rottura dei sensori ($10 \div 15\ secondi$) è ben maggiore di $5\ secondi$, tempo oltre il quale il filtro comunica uno stato di SOS al blocco *Mission Supervisor* per il termine della missione. Da varie prove è emerso che in particolari situazioni $10\ secondi$ possono portare il filtro a sbagliare irrimediabilmente la stima, ma nonostante ciò si è ritenuto opportuno analizzare anche questi casi di singolarità.

13. CONCLUSIONI E SPUNTI IMPLEMENTATIVI PER IL MIGLIORAMENTO DELLE PRESTAZIONI

A valle delle simulazioni effettuate e riportate precedentemente, si può concludere che l'implementazione realizzata del filtro EKF TD ha prestazioni adeguate ed in linea con le necessità del progetto complessivo fintanto che si ha il corretto funzionamento del DVL e dell'AHRS. Si nota infatti che dopo la fase di inizializzazione le deviazioni standard e gli errori assoluti sulla posizione stimata non superano i seguenti valori massimi:

Deviazioni standard	*Sensori per la correzione*	*Valori massimi*
σ_{xy}	GPS (superficie)	$\approx \mathbf{0,23}\ \boldsymbol{m}$
	Sonar (immersione)	$\approx \mathbf{0,07}\ \boldsymbol{m}$
σ_z	profondimetro	$\approx \mathbf{0,05}\ \boldsymbol{m}$

Componente posizione stimata	*Errore assoluto massimo*
x	$\pm\mathbf{0,40}\ \boldsymbol{m}$
y	
z	$\pm\mathbf{0,10}\ \boldsymbol{m}$

Tabella 4: Deviazioni standard ed errori assoluti massimi della posizione stimata

Inoltre, l'applicazione delle tre soglie sul funzionamento dei sonar ha aumentato le prestazioni soprattutto nelle fasi di rotazione in prossimità degli spigoli del bacino.

Abbiamo però notato delle situazioni critiche che possono influenzare negativamente il corretto funzionamento del filtro, e perciò, in una successiva implementazione, potrebbero essere inserite le seguenti migliorie che ne incrementerebbero la robustezza previsionale:

- aggiungere allo stato stimato la velocità lineare. Tale modifica porterebbe il filtro a basare la propria stima di posizione sulla velocità *stimata,* e non su quella ricevuta dal DVL. In questo modo in caso di guasti, la stima della velocità resterebbe comunque valida essendo basata sulle misure degli altri sensori e l'errore della posizione stimata rispetto a quella reale risulterebbe essere più contenuto;
- considerare la rottura dell'AHRS anche nel calcolo delle misure virtuali dei sonar pesandole di meno nella fase di correzione (variando la matrice R_k), o comunque inserire un fattore nella propagazione della covarianza della stima che tenga conto della precisione dei dati di orientazione. Quest'ultima soluzione comporterebbe una matrice M_k diversa dall'identità, e quindi il peso dell'errore di misura avrebbe un'influenza variabile nel tempo basata sull'incertezza dei dati di orientazione. La M_k avrebbe quindi la seguente forma:

$$M_k = \frac{dh(\eta_1, J)}{d\omega_z}\Big|_{\widehat{\eta}_1^{k|k-1}} \qquad \text{con} \quad \overrightarrow{\omega_z} = \begin{pmatrix} \overrightarrow{\omega_{GPS\,(xNorth)}} \\ \overrightarrow{\omega_{GPS\,(yEast)}} \\ \overrightarrow{\omega_{depth_{profondimetro}}} \\ \overrightarrow{\omega_{d_{sonar_{prua}}}} \\ \overrightarrow{\omega_{d_{sonar_{dx}}}} \\ \overrightarrow{\omega_{d_{sonar_{sx}}}} \\ \overrightarrow{\omega_{\eta_2}}(1) \\ \overrightarrow{\omega_{\eta_2}}(2) \\ \overrightarrow{\omega_{\eta_2}}(3) \end{pmatrix}$$

14. SIMULAZIONE CON SENSORI PIÙ ACCURATI

In quest'ultimo capitolo si riportano i grafici ottenuti dalla simulazione con varianze di sensori più accurati:

SENSORE	**VARIANZA σ^2**		
GPS	$3\ [m^2]$		
Profondimetro	$0.2^2\ [m^2]$		
DVL	$0.012^2\ [(m/s)^2]$		
AHRS	$0.0087^2\ [rad^2]$ Roll	$0.0087^2\ [rad^2]$ Pitch	$0.0017^2\ [rad^2]$ Yaw
Sonar	$0.05\ [m^2]$		
Giroscopio	$2.5133 \cdot 10^{-6}\ [(rad/s)^2]$		

Tabella 5: Varianze dei sensori utilizzati

La prova è stata effettuata con gli stessi parametri di missione del primo caso preso in esame:

Parametri di missione

Parete iniziale	Profondità	Distanza dalla parete	Verso di ispezione	Orientazione relativa	Velocità
BC	$5\ m$	$8\ m$	Orario	$30°$	$0{,}4\ m/s$

Sono sotto riportati i grafici dell'errore assoluto della posizione stimata e della deviazione standard.

Figura 57: errore di posizione di posizione delle componenti x e y (blu e rosso) e z (verde)

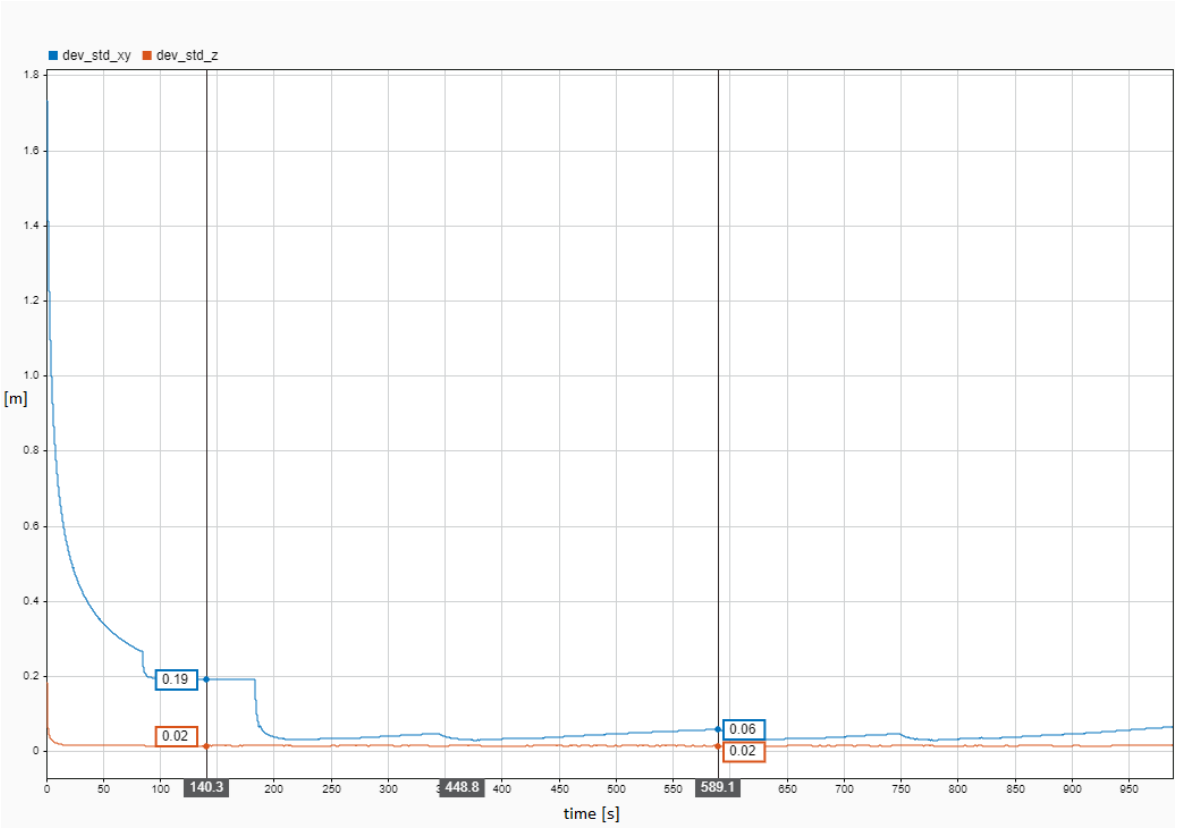


Figura 58: deviazioni standard legate alla posizione stimata su piano x-y (blu) e legata alla profondità (rosso)

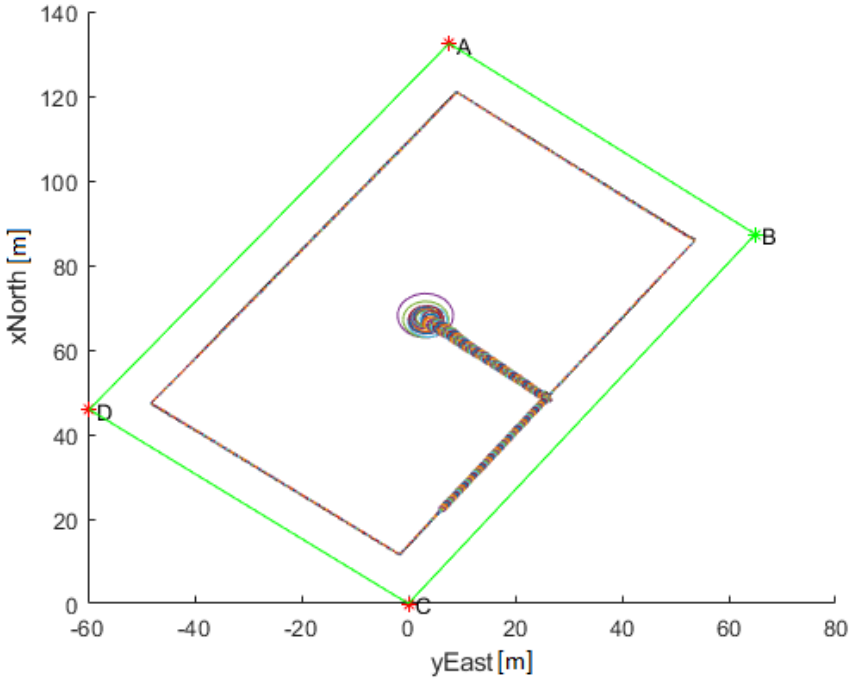


Figura 59: plot 2D della traiettoria dell'AUV nel bacino con ellissi d'incertezza (i semiassi sono tre volte le deviazioni standard)

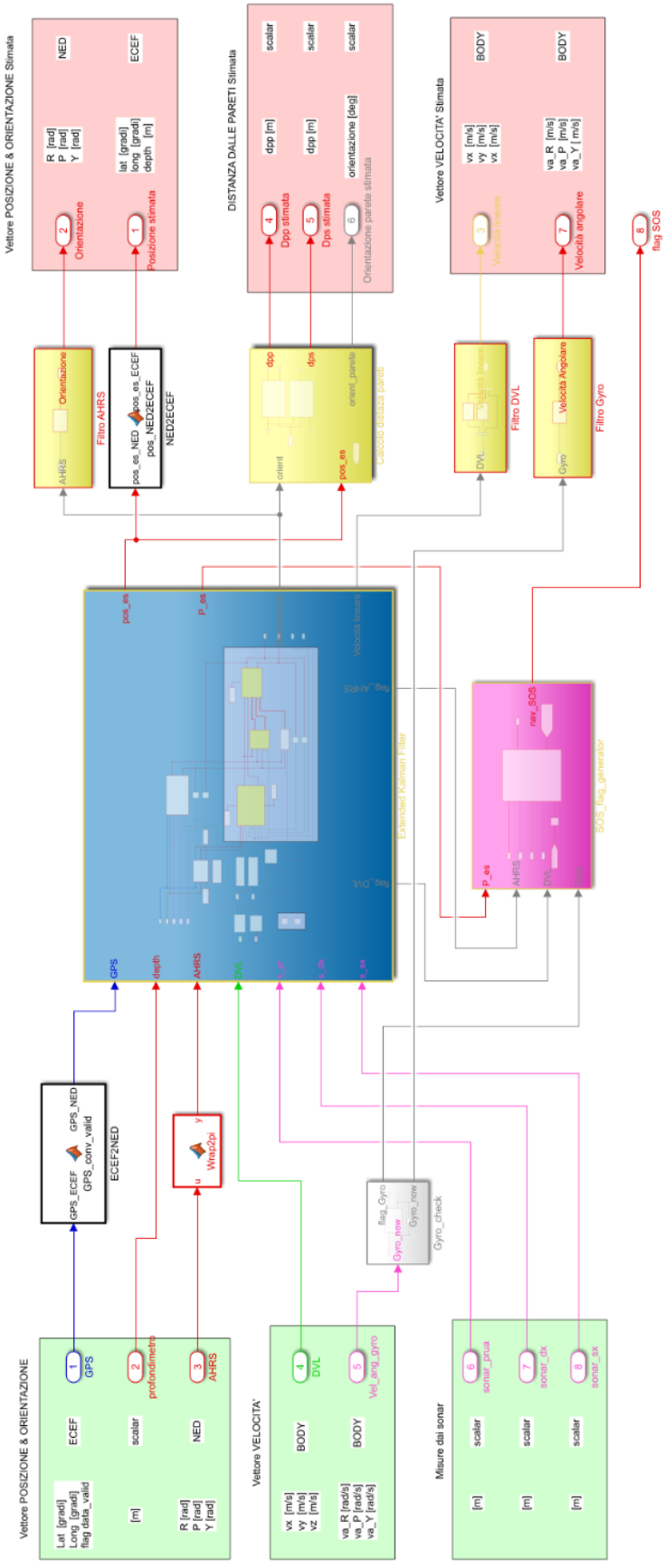
Come prevedibile, con dei sensori più accurati si ha un miglioramento delle prestazioni:

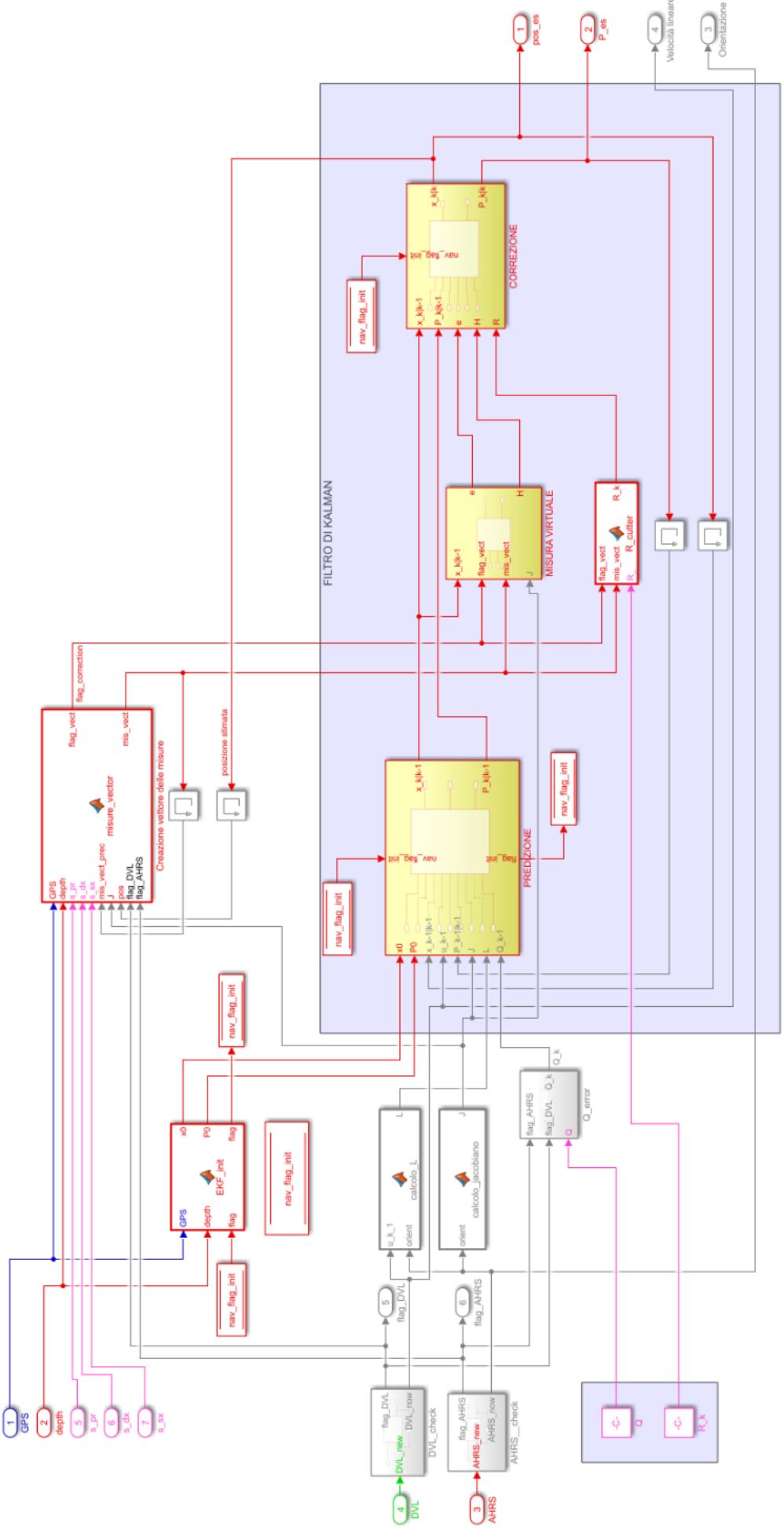
Deviazioni standard	*Sensori per la correzione*	*Valori massimi*
σ_{xy}	GPS (superficie)	$\approx \mathbf{0,19}\ \boldsymbol{m}$
	Sonar (immersione)	$\approx \mathbf{0,06}\ \boldsymbol{m}$
σ_z	profondimetro	$\approx \mathbf{0,02}\ \boldsymbol{m}$

Componente posizione stimata	*Errore assoluto massimo*
x	$\pm\mathbf{0,30}\ \boldsymbol{m}$
y	
z	$\pm\mathbf{0,02}\ \boldsymbol{m}$

Tabella 6: Deviazioni standard ed errori assoluti massimi della posizione stimata

Si nota infatti un decremento dell'errore assoluto del $\mathbf{25}\%$ rispetto agli assi x e y e del $\mathbf{60}\%$ su z.


Vettore POSIZIONE & ORIENTAZIONE Stimata
R [rad]
P [rad]
Y [rad]
NED
Orientazione
lat [gradi]
long [gradi]
depth [m]
ECEF
Posizione stimata
DISTANZA DALLE PARETI Stimata
dpp [m]
scalar
Dpp stimata
dps [m]
scalar
Dps stimata
orientazione [deg]
scalar
Orientazione parete stimata
Vettore VELOCITA' Stimata
vx [m/s]
vy [m/s]
vx [m/s]
BODY
Velocità lineare
va_R [m/s]
va_P [m/s]
va_Y [m/s]
BODY
Velocità angolare
flag SOS
Orientazione
AHRS
Filtro AHRS
pos_es_NED
pos_es_ECEF
pos_NED2ECEF
NED2ECEF
dpp
dps
orient
orient_parete
pos_es
Calcolo distanza pareti
DVL
Filtro DVL
Velocità lineare
Gyro
Velocità Angolare
Filtro Gyro
pos_es
P_es
Velocità lineare
flag_AHRS
flag_DVL
Estended Kalman Filter
P_es
AHRS
DVL
Stop
nav_SOS
SOS_flag_generator
GPS
depth
AHRS
DVL
GP
L_dx
L_sx
GPS_ECEF
GPS_conv_valid
GPS_NED
ECEF2NED
u
Wrap2pi
y
flag_Gyro
Gyro_new
Gyro_check
Vettore POSIZIONE & ORIENTAZIONE
Lat [gradi]
Long [gradi]
flag data_valid
ECEF
GPS
[m]
scalar
profondimetro
R [rad]
P [rad]
Y [rad]
NED
AHRS
Vettore VELOCITA'
vx [m/s]
vy [m/s]
vz [m/s]
BODY
DVL
va_R [rad/s]
va_P [rad/s]
va_Y [rad/s]
BODY
Vel_ang_gyro
Misure dai sonar
[m]
scalar
sonar_prua
[m]
scalar
sonar_dx
[m]
scalar
sonar_sx


FILTRO DI KALMAN
CORREZIONE
MISURA VIRTUALE
PREDIZIONE
Creazione vettore delle misure
EKF_init
calcolo_L
calcolo_jacobiano
DVL_check
AHRS_check
pos_es
P_es
Velocità lineare
Orientazione
GPS
depth
s_pr
s_dx
s_sx
DVL
AHRS

15. SCRIPT MATLAB

Riportiamo ora integralmente di seguito lo script di Matlab dei blocchi descritti nel *Capitolo 6.*

SCRIPT DI INILIZZAZIONE

```
%% Questo script deve essere eseguito solo una volta durante la simulazione
% va eseguito solo una volta che sono stati caricati i dati del bacino nel
% file missionC.m

%% TEMPO DI CAMPIONAMENTO DEL FILTRO DI KALMAN

nav_DT = 0.1;
nav_contatore = [0 0 0];  %[DVL AHRS Gyro]

%% 1. Calcolo coefficenti dei piani delle pareti del bacino

% Definizione terna NED con origine (lat0,long0,h0) coincidente al
% cornerC

nav_lat0 = cornerC(1);
nav_lon0 = cornerC(2);
nav_h0 = 0;

% Conversione ECEF to NED dei corner

[nav_A(1),nav_A(2),nav_A(3)]=
geodetic2ned(cornerA(1),cornerA(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);
[nav_B(1),nav_B(2),nav_B(3)]=
geodetic2ned(cornerB(1),cornerB(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);
[nav_C(1),nav_C(2),nav_C(3)]=
geodetic2ned(cornerC(1),cornerC(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);
[nav_D(1),nav_D(2),nav_D(3)]=
geodetic2ned(cornerD(1),cornerD(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);

%Calcolo coefficienti dei piani delle pareti
nav_plane_AB = coefficient_plane(nav_A,nav_B);
nav_plane_BC = coefficient_plane(nav_B,nav_C);
nav_plane_CD = coefficient_plane(nav_C,nav_D);
nav_plane_DA = coefficient_plane(nav_D,nav_A);

%inizializzazione variabile per la direzione di pattugliamento
if strcmp(inspectionDirection,'clockwise')
    nav_giro = 1;           %orario
else
    nav_giro = 2;           %antiorario
end
%%


% Calcolo coefficienti di un piano verticale passanti per due punti X1 e X2 in ingresso,
in uscita si ha un vettore riga (1x4) con i coefficienti dell'equazione del piano
% cioè y = [a b c d] con equazione ax + by + cz + d = 0


function  y = coefficient_plane(x1,x2)
    y(1) = 1;                               %coefficiente a = 1
    y(2) = (x1(1)-x2(1))/(x2(2)-x1(2));     %coefficiente b
    y(3) = 0;                               %coefficiente c = 0 (perché il piano è
verticale)
    y(4) = -x1(1)-y(2)*x1(2);               %coefficiente d
end
```

Misure vector

```
% Questa funzione impila le informazioni derivanti dai vari sensori in un
% unico vettore mis_vect 7x1.
% Ci sarà un corrispondente vettore flag_vect 7x1 di flag che sarà pari a:

% flag(i) = 0  se la misura è uguale a quella arrivata nell'istante
% predecente, in tal caso la correzione non verrà effettuata
% flag(i) = 1 se la misura è stata aggiornata rispetto a quella precendente

% NB il controllo sulla validità dei dati del GPS è valutata nel blocco
% sucessivo

function [flag_vect,mis_vect] = misure_vector(GPS,depth,s_pr,s_dx,s_sx, mis_vect_prec,...
          J, pos, nav_plane_AB, nav_plane_BC, nav_plane_CD, nav_plane_DA, nav_A, nav_B,
nav_C, nav_D, distanceFromWall, ...
          flag_DVL, flag_AHRS)

    mis_vect = zeros(7,1);
    flag_vect = zeros(7,1);
    mis_vect = [GPS(1) GPS(2) GPS(3) depth s_pr s_dx s_sx]';

    for i=[1:7]
        if mis_vect(i) ~= mis_vect_prec(i)
            flag_vect(i) = 1;
        end
    end
    %si valuta se la misura del profondimetro è valida
    if isnan(depth)
        flag_vect(4) = 0;
    end

    % si controlla che sia disponibile il dato dai sonar
    if isnan(s_pr)
        flag_vect(5) = 0;
    end
    if isnan(s_dx)
        flag_vect(6) = 0;
    end
    if isnan(s_sx)
        flag_vect(7) = 0;
    end

    %se i dati dei sonar qui sono utilizzabili, è necessario assicurarci che il loro
raggio non punti verso uno spigolo, in quel caso potrebbe causare una correzione errata
    if flag_vect(5) == 1                %se i dati del sonar di prua sono nuovi
            versore_son_NED = J*[1 0 0]';          %versore terna    NED sonar prua
            flag_vect(5) = inter(pos, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA, nav_A, nav_B, nav_C, nav_D);
    end

        if flag_vect(6) == 1                %se i dati del sonar destro sono nuovi
                versore_son_NED = J*[0 1 0]';          %versore terna NED sonar dx
                flag_vect(6) = inter(pos, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA, nav_A, nav_B, nav_C, nav_D);
        end
        if flag_vect(7) == 1                %se i dati del sonar sinistro sono nuovi
                versore_son_NED = J*[0 -1 0]';          %versore terna NED sonar sx
                flag_vect(7) = inter(pos, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA, nav_A, nav_B, nav_C, nav_D);
        end
la terza componente di flag_vect non sarà utilizzata. per assicurarci che i dati del GPS
siano validi ci rifaremo sempre su mis_vect
```

```
    if flag_DVL == 0 && flag_AHRS == 0  %i sonar vengono usati sempre nella correzione se
un sensore tra DLV o AHRS non funziona momentaneamente

        soglia_distanza=distanceFromWall*3;
        soglia_variazione=0.8;
        if  mis_vect(5) > soglia_distanza || abs(mis_vect(5)- mis_vect_prec(5)) >
soglia_variazione
            flag_vect(5) = 0;
        end
        if  mis_vect(6) > soglia_distanza || abs(mis_vect(6)- mis_vect_prec(6)) >
soglia_variazione
            flag_vect(6) = 0;
        end
        if  mis_vect(7) > soglia_distanza || abs(mis_vect(7)- mis_vect_prec(7)) >
soglia_variazione
            flag_vect(7) = 0;
        end
    end
end

function [flag] = inter(pos,
versore_son_NED,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA, nav_A, nav_B, nav_C,
nav_D)

%calcolo della distanza dalla posizione ai piani delle pareti seguendo la direzione data
dal versore del sonar

    t_AB = -
(pos(1)+nav_plane_AB(2)*pos(2)+nav_plane_AB(4))/(nav_plane_AB(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_BC = -
(pos(1)+nav_plane_BC(2)*pos(2)+nav_plane_BC(4))/(nav_plane_BC(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_CD = -
(pos(1)+nav_plane_CD(2)*pos(2)+nav_plane_CD(4))/(nav_plane_CD(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_DA = -
(pos(1)+nav_plane_DA(2)*pos(2)+nav_plane_DA(4))/(nav_plane_DA(2)*versore_son_NED(2)+verso
re_son_NED(1));

    if t_AB < 0
        t_AB = inf;
    end
    if t_BC < 0
        t_BC = inf;
    end
    if t_CD < 0
        t_CD = inf;
    end
    if t_DA < 0
        t_DA = inf;
    end

    [dist, index] = min([t_AB t_BC t_CD t_DA]);
    dist = dist - 0.267;       %va tolto il raggio del veicolo rispetto al centro di
massa

    int=zeros(1,3);
    %calcolo l'intersezione
    int(1) = pos(1) + versore_son_NED(1)*dist;
    int(2) = pos(2) + versore_son_NED(2)*dist;
    int(3) = 0;



    soglia_spigolo = 5;          %distanza massima dallo spigolo entro la quale la misura
del sonar non è buona
    flag=1;
    switch index
```

```
        case 1
            if norm(int-nav_A) < soglia_spigolo || norm(int-nav_B) < soglia_spigolo
                flag=0;
            end
        case 2
            if norm(int-nav_B) < soglia_spigolo || norm(int-nav_C) < soglia_spigolo
                flag=0;
            end
        case 3
            if norm(int-nav_C) < soglia_spigolo || norm(int-nav_D) < soglia_spigolo
                flag=0;
            end
        case 4
            if norm(int-nav_D) < soglia_spigolo || norm(int-nav_A) < soglia_spigolo
                flag=0;
            end
        otherwise
            disp('errore nell'intersezione del piano')
    end
end
```

Inizializzazione

```
%blocco di inizializzazione del Filtro
 function [x0, P0, flag] = EKF_init(GPS, depth, flag)

    x0 = zeros(3,1);
    P0 = zeros(3,3);

    % il valore di nav_flag_init è all'inizio della simulazione pari a 0
    % appena c'è il primo dato disponibile del GPS (insieme al profondimetro)
    % si assume quella come punto di inizializzazione del filtro.

    if flag == 0
        if GPS(3)==1
            x0 = [GPS(1), GPS(2), depth]';
            P0 = diag([3 3 0.2]);       %sono le varianze del GPS e del profondimetro
(come nella R)
            flag = 1;                   %se vale 1 bisogna assegnare i primi valori a
x_k_1 e P_k_1
        end

    end

end
```

Calcolo matrice L

```
     %calcolo matrice L
function L = calcolo_L(u_k_1,orient)

    L =  [   u_k_1(2)*((sin(orient(1))*sin(orient(3)))/10 +
(cos(orient(1))*cos(orient(3))*sin(orient(2)))/10) +
u_k_1(3)*((cos(orient(1))*sin(orient(3)))/10 -
(cos(orient(3))*sin(orient(1))*sin(orient(2)))/10),
(u_k_1(3)*cos(orient(1))*cos(orient(3))*cos(orient(2)))/10 -
(u_k_1(1)*cos(orient(3))*sin(orient(2)))/10 +
(u_k_1(2)*cos(orient(3))*cos(orient(2))*sin(orient(1)))/10,
u_k_1(3)*((cos(orient(3))*sin(orient(1)))/10 -
(cos(orient(1))*sin(orient(3))*sin(orient(2)))/10) -
u_k_1(2)*((cos(orient(1))*cos(orient(3)))/10 +
(sin(orient(1))*sin(orient(3))*sin(orient(2)))/10) -
(u_k_1(1)*cos(orient(2))*sin(orient(3)))/10, (cos(orient(3))*cos(orient(2)))/10,
(cos(orient(3))*sin(orient(1))*sin(orient(2)))/10 - (cos(orient(1))*sin(orient(3)))/10,
(sin(orient(1))*sin(orient(3)))/10 +
(cos(orient(1))*cos(orient(3))*sin(orient(2)))/10;...
```

```
          - u_k_1(2)*((cos(orient(3))*sin(orient(1)))/10 -
(cos(orient(1))*sin(orient(3))*sin(orient(2)))/10) -
u_k_1(3)*((cos(orient(1))*cos(orient(3)))/10 +
(sin(orient(1))*sin(orient(3))*sin(orient(2)))/10),
(u_k_1(3)*cos(orient(1))*cos(orient(2))*sin(orient(3)))/10 -
(u_k_1(1)*sin(orient(3))*sin(orient(2)))/10 +
(u_k_1(2)*cos(orient(2))*sin(orient(1))*sin(orient(3)))/10,
u_k_1(3)*((sin(orient(1))*sin(orient(3)))/10 +
(cos(orient(1))*cos(orient(3))*sin(orient(2)))/10) -
u_k_1(2)*((cos(orient(1))*sin(orient(3)))/10 -
(cos(orient(3))*sin(orient(1))*sin(orient(2)))/10) +
(u_k_1(1)*cos(orient(3))*cos(orient(2)))/10, (cos(orient(2))*sin(orient(3)))/10,
(cos(orient(1))*cos(orient(3)))/10 + (sin(orient(1))*sin(orient(3))*sin(orient(2)))/10,
(cos(orient(1))*sin(orient(3))*sin(orient(2)))/10 -
(cos(orient(3))*sin(orient(1)))/10;...

(u_k_1(2)*cos(orient(1))*cos(orient(2)))/10 -
(u_k_1(3)*cos(orient(2))*sin(orient(1)))/10,                                    -
(u_k_1(1)*cos(orient(2)))/10 - (u_k_1(3)*cos(orient(1))*sin(orient(2)))/10 -
(u_k_1(2)*sin(orient(1))*sin(orient(2)))/10,
0,              -sin(orient(2))/10,
(cos(orient(2))*sin(orient(1)))/10,
(cos(orient(1))*cos(orient(2)))/10];
end
```

Calcolo matrice di rotazione J

```
%calcolo matrice J
function J = calcolo_jacobiano(orient)

    Roll  = orient(1);
    Pitch = orient(2);
    Yaw   = orient(3);

    R_x = [ 1              0             0;
            0          cos(Roll)     sin(Roll);
            0          -sin(Roll)    cos(Roll)];

    R_y = [cos(Pitch)       0   -sin(Pitch);
              0             1         0;
           sin(Pitch)       0   cos(Pitch)];

    R_z = [cos(Yaw)     sin(Yaw)      0;
          -sin(Yaw)     cos(Yaw)      0;
              0           0           1];

    J = (R_z'*R_y')*R_x';
end
```

Calcolo matrice R

```
%calcolo matrice R
function R_k = R_cutter(flag_vect, mis_vect,R_)
%valutazione misure GPS
    R=R_;
    i=0;
    if mis_vect(3) == 0                    %se i dati del GPS non sono validi
        R(1,:) = [];
        R(:,1) = [];
        i=i+1;
        R(2-i,:) = [];
        R(:,2-i) = [];
        i=i+1;
        if flag_vect(4) == 0               %se i dati del profondimetro non sono nuovi
            R(3-i,:) = [];
            R(:,3-i) = [];
            i=i+1;
```

```
        end
        if flag_vect(5) == 0                %se i dati del sonar di prua non sono nuovi
            R(4-i,:) = [];
             R(:,4-i) = [];
            i=i+1;
        end
        if flag_vect(6) == 0                %se i dati del sonar sinistro non sono nuovi
            R(5-i,:) = [];
             R(:,5-i) = [];
            i=i+1;
        end
        if flag_vect(7) == 0                %se i dati del sonar destro non sono nuovi
            R(6-i,:) = [];
             R(:,6-i) = [];
        end
    % da qui invece abbiamo il GPS valido
    else
        if flag_vect(1)==0 || flag_vect(2)==0  %se i dati del GPS non sono nuovi
            R(1-i,:) = [];
            R(:,1-i) = [];
            i=i+1;
            R(2-i,:) = [];
            R(:,2-i) = [];
            i=i+1;
        end

        if flag_vect(4) == 0                %se i dati del profondimetro non sono nuovi
            R(3-i,:) = [];
            R(:,3-i) = [];
            i=i+1;
        end

        R(4-i,:) = []; %annulliamo le misure dei sonar
        R(:,4-i) = [];
        i=i+1;
        R(5-i,:) = [];
        R(:,5-i) = [];
        i=i+1;
        R(6-i,:) = [];
        R(:,6-i) = [];

    end
    R_k=R;
end
```

Calcolo matrice Q

```
%calcolo matrice Q
function Q_k = fcn(Q, Q_k_1, flag_AHRS, flag_DVL)

    Q_k = Q;

    K=10;
    if flag_DVL == 1
       Q_k = Q_k_1 + K*5*diag([0,0,0,(0.012)^2,(0.012)^2,(0.012)^2]);
    end

    if flag_AHRS == 1
       Q_k = Q_k_1 + K*50*diag([0.0087, 0.0087, 0.0017,0,0,0]);
    end

end
```

Predizione

```
%% Passo di PREDIZIONE

function [x_k,  P_k, nav_flag_init] = prediction(x0, P0, x_k_1, u_k_1,  P_k_1, J, L,
Q_k_1, nav_DT, nav_flag_init)

    x_k = zeros(3,1);
    P_k = zeros(3,3);

    % il valore di nav_flag_init all'inizio della simulazione é pari a 0
    % appena c'è il primo dato disponibile del GPS (insieme al profondimetro)
    % si assume quella come punto di inizializzazione del filtro.

    if nav_flag_init ~= 0
        if nav_flag_init == 1
            x_k_1 = x0;
            P_k_1 = P0;
            nav_flag_init = 2;      % =2 così l'inizializzazione è fatta una sola volta.
        end

        %passo di predizione
        x_k = x_k_1 + nav_DT*J*u_k_1;
        P_k = P_k_1 + L*Q_k_1*L';               %la matrice F è F = I

    end
end
```

Blocco misure virtuali

```
%calcolo  misure virtuali

function [e, H] = virtual(x_k_k_1, flag_vect, mis_vect, J, nav_plane_AB, nav_plane_BC,
nav_plane_CD, nav_plane_DA)

    e = zeros(6,1);
    h = zeros(6,1);
    z = zeros(6,1);
    H = zeros(6,3);


    %valutazione misure GPS
    if mis_vect(3)==1                                   %se i dati del GPS sono validi
        if flag_vect(1) == 1 && flag_vect(2) == 1       %se i dati del GPS sono nuovi

            z = z + [mis_vect(1) mis_vect(2) 0 0 0 0]';
            h = h + [x_k_k_1(1)   x_k_k_1(2) 0 0 0 0]';

            H(1,:)= [1 0 0];
            H(2,:)= [0 1 0];
        end
    end

    %valutazione misure depth
    if flag_vect(4) == 1                                 %se i dati del profondimetro
sono nuovi

        z = z + [0 0 mis_vect(4) 0 0 0]';
        h = h + [0 0  x_k_k_1(3) 0 0 0]';

        H(3,:)= [0 0 1];
    end

    %valutiamo ora le misure virtuali solo se quelle del GPS non sono
    %valide
    if mis_vect(3)==0                       %se i dati del GPS non sono validi

        if flag_vect(5) == 1                %se i dati del sonar di prua sono nuovi
```

```
                versore_son_NED = J*[1 0 0]';          %versore terna NED sonar prua

                [distanza, plane] = inter(x_k_k_1, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA);
                z = z + [0 0 0 mis_vect(5) 0 0]';
                h = h + [0 0 0 distanza    0 0]';

                H_new=zeros(1,3);
                [H_new(1), H_new(2), H_new(3)] = calc_H_vect(x_k_k_1,
versore_son_NED,plane);
                H(4,:) = H_new;
        end

        if flag_vect(6) == 1              %se i dati del sonar destro sono nuovi

                versore_son_NED = J*[0 1 0]';          %versore terna NED sonar dx

                [distanza, plane] = inter(x_k_k_1, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA);
                z = z + [0 0 0 0 mis_vect(6) 0]';
                h = h + [0 0 0 0 distanza    0]';

                H_new=zeros(1,3);
                [H_new(1), H_new(2), H_new(3)] = calc_H_vect(x_k_k_1,
versore_son_NED,plane);
                H(5,:) = H_new;
        end

        if flag_vect(7) == 1              %se i dati del sonar sinistro sono nuovi

                versore_son_NED = J*[0 -1 0]';          %versore terna NED sonar sx

                [distanza, plane] = inter(x_k_k_1, versore_son_NED
,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA);
                z = z + [0 0 0 0 0 mis_vect(7)]';
                h = h + [0 0 0 0 0 distanza   ]';

                H_new=zeros(1,3);
                [H_new(1), H_new(2), H_new(3)] = calc_H_vect(x_k_k_1,
versore_son_NED,plane);
                H(6,:) = H_new;
        end
    end

    e = z - h;

    %%si riducono le dimensioni della matrice H e del vettore e

    i=0;
    if mis_vect(3) == 0                   %se i dati del GPS non sono validi
        H(1-i,:) = [];
        e(1-i) = [];
        i=i+1;
        H(2-i,:) = [];
        e(2-i) = [];
        i=i+1;
        if flag_vect(4) == 0              %se i dati del profondimetro non sono nuovi
            H(3-i,:) = [];
            e(3-i) = [];
            i=i+1;
        end
        if flag_vect(5) == 0              %se i dati del sonar di prua non sono nuovi
            H(4-i,:) = [];
            e(4-i) = [];
            i=i+1;
        end
        if flag_vect(6) == 0              %se i dati del sonar sinistro non sono nuovi
            H(5-i,:) = [];
```

```
            e(5-i) = [];
            i=i+1;
        end
        if flag_vect(7) == 0               %se i dati del sonar destro non sono nuovi
            H(6-i,:) = [];
            e(6-i) = [];
        end
    % da qui invece si ha il GPS valido
    else
        if flag_vect(1)==0 || flag_vect(2)==0  %se i dati del GPS non sono nuovi
            H(1-i,:) = [];
            e(1-i) = [];
            i=i+1;
            H(2-i,:) = [];
            e(2-i) = [];
            i=i+1;
        end

        if flag_vect(4) == 0               %se i dati del profondimetro non sono nuovi
            H(3-i,:) = [];
            e(3-i) = [];
            i=i+1;
        end

        H(4-i,:) = [];                      %annulliamo le misure dei sonar
        e(4-i) = [];
        i=i+1;
        H(5-i,:) = [];
        e(5-i) = [];
        i=i+1;
        H(6-i,:) = [];
        e(6-i) = [];
    end
    i=0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [dist, plane] = inter(pos,
versore_son_NED,nav_plane_AB,nav_plane_BC,nav_plane_CD,nav_plane_DA)

    %calcolo della distanza dalla posizione ai piani delle pareti seguendo
    %la direzione data dal versore del sonar

    t_AB = -
(pos(1)+nav_plane_AB(2)*pos(2)+nav_plane_AB(4))/(nav_plane_AB(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_BC = -
(pos(1)+nav_plane_BC(2)*pos(2)+nav_plane_BC(4))/(nav_plane_BC(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_CD = -
(pos(1)+nav_plane_CD(2)*pos(2)+nav_plane_CD(4))/(nav_plane_CD(2)*versore_son_NED(2)+verso
re_son_NED(1));
    t_DA = -
(pos(1)+nav_plane_DA(2)*pos(2)+nav_plane_DA(4))/(nav_plane_DA(2)*versore_son_NED(2)+verso
re_son_NED(1));

    if t_AB < 0
        t_AB = inf;
    end
    if t_BC < 0
        t_BC = inf;
    end
    if t_CD < 0
        t_CD = inf;
    end
    if t_DA < 0
        t_DA = inf;
    end
```

```
    [dist, index] = min([t_AB t_BC t_CD t_DA]);
    dist = dist - 0.267;        %va tolto il raggio del robot rispetto al centro di massa

    switch index
        case 1
            plane = nav_plane_AB;
        case 2
            plane = nav_plane_BC;
        case 3
            plane = nav_plane_CD;
        case 4
            plane = nav_plane_DA;
        otherwise
            plane=[0 0 0 0];
            disp('errore nella intersezione del piano')
    end

end

function [H1, H2, H3]= calc_H_vect(pos, sonar, plane)

    H1 = sign(pos(1) + plane(4) + pos(2)*plane(2))/abs(sonar(1) + plane(2)*sonar(2));
    H2 = (plane(2)*sign(pos(1) + plane(4) + pos(2)*plane(2)))/abs(sonar(1) +
plane(2)*sonar(2));
    H3 = 0;
end
```

Correzione

```
%%Passo di Correzione

function [x_k, P_k] = correction(x_k_k_1,P_k_k_1, e, H, R_k,nav_flag_init)

     x_k = zeros(3,1);
     P_k = zeros(3,3);


    if nav_flag_init ~= 0
        %matrice ottenuta da derivata di funzione h (6x1) per i rumori sei vari
        %sensori, che sono 6. (ha dimensione 6x6)
        %andrebbero anche considerati i rumori legati all'AHRS che al momento
        %sono trascurati
        %M = eye(6,6);              %M è l'dientità
        S_k = H*P_k_k_1*H'+ R_k;
        K_k = P_k_k_1*H'*(inv(S_k));

        P_k = (eye(3)-(K_k*H))*P_k_k_1;
        x_k = x_k_k_1 + K_k * e;
    end

end
```

Conversione ECEF2NED

```
%Componente di ingresso in terna ECEF
%Lat  [gradi]
%Long  [gradi]
%flag data_valid
function GPS_NED = GPS_conv_valid(GPS_ECEF, nav_lat0, nav_lon0, nav_h0)

    coder.extrinsic('evalin', 'wgs84Ellipsoid', 'geodetic2ned');

    if GPS_ECEF(3)==1
        GPS_NED = zeros(3,1);
        [GPS_NED(1),GPS_NED(2),GPS_NED(3)]=
geodetic2ned(GPS_ECEF(1),GPS_ECEF(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);
```

```
        GPS_NED(3) = GPS_ECEF(3);
    else                            %qui non si effettua la conversione se il dato non è
valido
        GPS_NED = GPS_ECEF;
    end
end
```

Conversione NED2ECEF

```
%%Blocco di conversione NED2ECEF
function pos_es_ECEF = pos_NED2ECEF(pos_es_NED,nav_lat0, nav_lon0, nav_h0)

    coder.extrinsic('evalin', 'wgs84Ellipsoid', 'ned2geodetic');
    pos_es_ECEF = zeros(3,1);
    depth = pos_es_NED(3);

    [pos_es_ECEF(1),pos_es_ECEF(2),pos_es_ECEF(3)]=
ned2geodetic(pos_es_NED(1),pos_es_NED(2),0,nav_lat0,nav_lon0,nav_h0,wgs84Ellipsoid);
    pos_es_ECEF(3) = depth;
end
```

Blocco di calcolo di distanza dalle pareti

```
% Applicando la formula della distanza di un punto da un piano si ottiene la
% distanza della posizione dell'AUV dalla parete d'interesse.

function [dpp,dps] = dpp_dps_calculator(pos_es, parete_attuale, nav_plane_AB,
nav_plane_BC, nav_plane_CD, nav_plane_DA, nav_giro)

%selezione della parete di pattugliamento e successiva
    switch parete_attuale

        case 1
           coef = nav_plane_AB;            %parete attuale AB
           if nav_giro == 1
               coef_succ = nav_plane_BC;   %parete succ BC
           else
               coef_succ = nav_plane_DA;   %parete succ DA
           end

        case 2
           coef = nav_plane_BC;            %parete attuale BC
           if nav_giro == 1
               coef_succ = nav_plane_CD;   %parete succ CD
           else
               coef_succ = nav_plane_AB;   %parete succ AB
           end

        case 3
           coef = nav_plane_CD;            %parete attuale CD
           if nav_giro == 1
               coef_succ = nav_plane_DA;   %parete succ DA
           else
               coef_succ = nav_plane_BC;   %parete succ BC
           end
        case 4
           coef = nav_plane_DA;            %parete attuale DA
           if nav_giro == 1
               coef_succ = nav_plane_AB;   %parete succ AB
           else
               coef_succ = nav_plane_CD;   %parete succ CD
           end
        otherwise
            coef = [0 0 0 0];
            coef_succ = [0 0 0 0];
    end
```

```
    dpp = abs(coef(1)*pos_es(1)+coef(2)*pos_es(2)-
coef(3)*pos_es(3)+coef(4))/norm([coef(1) coef(2) coef(3)]);
    dps = abs(coef_succ(1)*pos_es(1)+coef_succ(2)*pos_es(2)-
coef_succ(3)*pos_es(3)+coef_succ(4))/norm([coef_succ(1) coef_succ(2) coef_succ(3)]);

end
```

Blocco di calcolo di orientazione relativa alla parate

```
function orient_parete  = orient_parete_calculator(pos_es, parete_attuale, orient,
nav_plane_AB, nav_plane_BC, nav_plane_CD, nav_plane_DA)

    %consideriamo angoli di pitch e roll nulli, cioè si calcola
    %l'orientazione rispetto alla parete solo considerando rotazioni di yaw
    %(piano xy)

    Yaw   = orient(3);
    R_z = [cos(Yaw)      sin(Yaw)       0     ;
           -sin(Yaw)     cos(Yaw)       0     ;
              0             0           1    ];
    v_pos= R_z'*[1 0 0]';  %versore della retta lungo l'asse x (in body) in NED,
                        %cioè la direzione di riferimento di prua del veicolo

    %selezione della parete di pattugliamento
switch parete_attuale
    case 1
        %% Piano di pattugliamento AB
        versore_plane_NED = [nav_plane_AB(1) nav_plane_AB(2) 0]';    %versore normale al
piano
        versore_plane_NED = versore_plane_NED/norm(versore_plane_NED);

        %calcolo distanza e intersezione lungo la normale al piano
        t_ = -
(pos_es(1)+nav_plane_AB(2)*pos_es(2)+nav_plane_AB(4))/(nav_plane_AB(2)*versore_plane_NED(
2)+versore_plane_NED(1));
        inter_perp=zeros(3,1);
        inter_perp(1) = pos_es(1) + versore_plane_NED(1)*t_;
        inter_perp(2) = pos_es(2) + versore_plane_NED(2)*t_;
        inter_perp(3) = pos_es(3) + versore_plane_NED(3)*t_;

        %calcolo distanza e intersezione lungo la direzione di prua
        t_ = -
(pos_es(1)+nav_plane_AB(2)*pos_es(2)+nav_plane_AB(4))/(nav_plane_AB(2)*v_pos(2)+v_pos(1))
;
        inter_prua=zeros(3,1);
        inter_prua(1) = pos_es(1) + v_pos(1)*t_;
        inter_prua(2) = pos_es(2) + v_pos(2)*t_;
        inter_prua(3) = pos_es(3) + v_pos(3)*t_;

        coef = nav_plane_AB;
        coef(4)=[];
        orient_parete = rad2deg(asin(abs(coef*v_pos)/(norm(coef)*norm(v_pos))));
        %se orient_parete è positiva, allora ci si trova tra - 90 e + 90 rispetto alla
normale della parete
        if t_ > 0
            if inter_prua(1) > inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = orient_parete - 90;
            else
                orient_parete = -(orient_parete - 90);
            end
        else
            if inter_prua(1) > inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = (orient_parete + 90);
            else
                orient_parete = -(orient_parete + 90);
            end
        end
    case 2
```

```
        %% Piano di pattugliamento BC
        versore_plane_NED = [nav_plane_BC(1) nav_plane_BC(2) 0]';    %versore normale al
piano
        versore_plane_NED = versore_plane_NED/norm(versore_plane_NED);

        %calcolo distanza e intersezione lungo la normale al piano
        t_ = -
(pos_es(1)+nav_plane_BC(2)*pos_es(2)+nav_plane_BC(4))/(nav_plane_BC(2)*versore_plane_NED(
2)+versore_plane_NED(1));
        inter_perp=zeros(3,1);
        inter_perp(1) = pos_es(1) + versore_plane_NED(1)*t_;
        inter_perp(2) = pos_es(2) + versore_plane_NED(2)*t_;
        inter_perp(3) = pos_es(3) + versore_plane_NED(3)*t_;

        %calcolo distanza e intersezione lungo la direzione di prua
        t_ = -
(pos_es(1)+nav_plane_BC(2)*pos_es(2)+nav_plane_BC(4))/(nav_plane_BC(2)*v_pos(2)+v_pos(1))
;
        inter_prua=zeros(3,1);
        inter_prua(1) = pos_es(1) + v_pos(1)*t_;
        inter_prua(2) = pos_es(2) + v_pos(2)*t_;
        inter_prua(3) = pos_es(3) + v_pos(3)*t_;

        coef = nav_plane_BC;
        coef(4)=[];
        v_pos=-v_pos;
        orient_parete = rad2deg(asin(abs(coef*v_pos)/(norm(coef)*norm(v_pos))));
        %se orient_parete è positiva, allora ci si trova tra - 90 e + 90 rispetto alla
normale della parete
        if t_ > 0
            if inter_prua(1) > inter_perp(1) && inter_prua(2) > inter_perp(2)
                orient_parete = orient_parete - 90;
            else
                orient_parete = -(orient_parete - 90);
            end
        else
            if inter_prua(1) > inter_perp(1) && inter_prua(2) > inter_perp(2)
                orient_parete = orient_parete + 90;
            else
                orient_parete = -(orient_parete + 90);
            end
        end

    case 3
        %% Piano di pattugliamento CD
        versore_plane_NED = [nav_plane_CD(1) nav_plane_CD(2) 0]';    %versore normale al
piano
        versore_plane_NED = versore_plane_NED/norm(versore_plane_NED);

        %calcolo distanza e intersezione lungo la normale al piano
        t_ = -
(pos_es(1)+nav_plane_CD(2)*pos_es(2)+nav_plane_CD(4))/(nav_plane_CD(2)*versore_plane_NED(
2)+versore_plane_NED(1));
        inter_perp=zeros(3,1);
        inter_perp(1) = pos_es(1) + versore_plane_NED(1)*t_;
        inter_perp(2) = pos_es(2) + versore_plane_NED(2)*t_;
        inter_perp(3) = pos_es(3) + versore_plane_NED(3)*t_;

        %calcolo distanza e intersezione lungo la direzione di prua
        t_ = -
(pos_es(1)+nav_plane_CD(2)*pos_es(2)+nav_plane_CD(4))/(nav_plane_CD(2)*v_pos(2)+v_pos(1))
;
        inter_prua=zeros(3,1);
        inter_prua(1) = pos_es(1) + v_pos(1)*t_;
        inter_prua(2) = pos_es(2) + v_pos(2)*t_;
        inter_prua(3) = pos_es(3) + v_pos(3)*t_;

        coef = nav_plane_CD;
        coef(4)=[];
```

```
        v_pos=-v_pos;
        orient_parete = rad2deg(asin(abs(coef*v_pos)/(norm(coef)*norm(v_pos))));
        %se orient_parete è positiva, allora ci si trova tra - 90 e + 90 rispetto alla
normale della parete
        if t_ > 0
            if inter_prua(1) > inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = -orient_parete + 90;
            else
                orient_parete = orient_parete - 90;
            end
        else
             if inter_prua(1) > inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = -orient_parete - 90;
            else
                orient_parete = orient_parete + 90;
            end
        end

    case 4
        %% Piano di pattugliamento DA
        versore_plane_NED = [nav_plane_DA(1) nav_plane_DA(2) 0]';    %versore normale al
piano
        versore_plane_NED = versore_plane_NED/norm(versore_plane_NED);

        %calcolo distanza e intersezione lungo la normale al piano
        t_ = -
(pos_es(1)+nav_plane_DA(2)*pos_es(2)+nav_plane_DA(4))/(nav_plane_DA(2)*versore_plane_NED(
2)+versore_plane_NED(1));
        inter_perp=zeros(3,1);
        inter_perp(1) = pos_es(1) + versore_plane_NED(1)*t_;
        inter_perp(2) = pos_es(2) + versore_plane_NED(2)*t_;
        inter_perp(3) = pos_es(3) + versore_plane_NED(3)*t_;

        %calcolo distanza e intersezione lungo la direzione di prua
        t_ = -
(pos_es(1)+nav_plane_DA(2)*pos_es(2)+nav_plane_DA(4))/(nav_plane_DA(2)*v_pos(2)+v_pos(1))
;
        inter_prua=zeros(3,1);
        inter_prua(1) = pos_es(1) + v_pos(1)*t_;
        inter_prua(2) = pos_es(2) + v_pos(2)*t_;
        inter_prua(3) = pos_es(3) + v_pos(3)*t_;

        coef = nav_plane_DA;
        coef(4)=[];
        orient_parete = rad2deg(asin(abs(coef*v_pos)/(norm(coef)*norm(v_pos))));
        %se orient_parete è positiva, allora ci si trova tra - 90 e + 90 rispetto alla
normale della parete
        if t_ > 0
            if inter_prua(1) < inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = orient_parete - 90;
            else
                orient_parete = -orient_parete + 90;
            end
        else
            if inter_prua(1) < inter_perp(1) && inter_prua(2) < inter_perp(2)
                orient_parete = orient_parete + 90;
            else
                orient_parete = -orient_parete - 90;
            end
        end

    otherwise
        %%
       coef=[0 0 0 0];
       orient_parete = NaN;
end


end
```

Blocco di gestione delle misure errate di giroscopio, DVL, AHRS

```
% gestione di misura errata del giroscopio
function [flag_gyro, gyro_now] = fcn(gyro_new, gyro_old)

    if isnan(gyro_new(1,1)) || isnan(gyro_new(2,1)) || isnan(gyro_new(3,1))
        gyro_now = gyro_old;
        flag_gyro = 1;
        disp('Misura gyro non valida')
    else
        gyro_now = gyro_new;
        flag_gyro = 0;
    end
end

%gestione di misura errata del DVL
function [flag_DVL, DVL_now] = fcn(DVL_new, DVL_old)

    if isnan(DVL_new(1,1)) || isnan(DVL_new(2,1)) || isnan(DVL_new(3,1))
        DVL_now = DVL_old;
        flag_DVL = 1;
        disp('Misura DVL non valida')
    else
        DVL_now = DVL_new;
        flag_DVL = 0;
    end
end

%gestione di misura errata di AHRS
function [flag_AHRS, AHRS_now] = fcn(AHRS_new, AHRS_old)

    if isnan(AHRS_new(1,1)) || isnan(AHRS_new(2,1)) || isnan(AHRS_new(3,1))
        AHRS_now = AHRS_old;
        flag_AHRS = 1;
        disp('Misura AHRS non valida')
    else
        AHRS_now = AHRS_new;
        flag_AHRS = 0;
    end
end
```

Blocco di SOS

```
%% Generazione flag di emergenza
% 1. Assenza di segnali dai sensori per un determinato tempo

function [SOS, contatore] = SOS_flag(P_es, DVL, AHRS, gyro, contatore)
    SOS=0;

    soglia_P = 10; %[m^2]
    if P_es(1,1)+P_es(2,2) > soglia_P || P_es(3,3) > soglia_P
        SOS=1;
    end

    if DVL == 1
        contatore(1) = contatore(1) + 1;
    else
        contatore(1) = 0;
    end

    if AHRS == 1
        contatore(2) = contatore(2) + 1;
    else
        contatore(2) = 0;
    end
```

```
    if gyro == 1
        contatore(3) = contatore(3) + 1;
    else
        contatore(3) = 0;
    end

    if max(contatore) > 50 %50 equivale a 5 secondi
        SOS=1;
    end

end
```

PLOT 2D CON COVARIANZE

```
%GRUPPO Navigation Systems Team C

%Eseguendo questo file viene creato uno schema 2D della posizione reale
%eseguita dall'AUV durante la missione e della sua posizione stimata.
%Si possono anche disegnare le ellissi d'incertezza con semiassi pari a tre
%volte le deviazioni standard.

close
hold on

%creazione del bacino
plot(nav_A(2),nav_A(1),'r*')
text(nav_A(2),nav_A(1),' A')
plot(nav_B(2),nav_B(1),'g*')
text(nav_B(2),nav_B(1),' B')
plot(nav_C(2),nav_C(1),'r*')
text(nav_C(2),nav_C(1),' C')
plot(nav_D(2),nav_D(1),'r*')
text(nav_D(2),nav_D(1),' D')
plot([nav_A(2) nav_B(2)],[nav_A(1) nav_B(1)],'g')
plot([nav_B(2) nav_C(2)],[nav_B(1) nav_C(1)],'g')
plot([nav_C(2) nav_D(2)],[nav_C(1) nav_D(1)],'g')
plot([nav_D(2) nav_A(2)],[nav_D(1) nav_A(1)],'g')
xlabel('yEast [m]')
ylabel('xNorth [m]')

%disegno la traiettoria della posizione reale NED dal Vehicle Model
plot(out.Lon_ts.data,out.Lat_ts.data,'black-');

%disegno la traiettoria della posizione stimata NED dal Navigation System
plot(out.Lon_es.data,out.Lat_es.data,'r.');

%% Ciclo per disegnare gli ellissi

%Commentare da qui in poi per avere solo il plot senza ellissi

for i=1:1:2000
    xell = 3*out.devx.data(i*10);
    yell = 3*out.devy.data(i*10);
    [x,y] = getEllipse(yell,xell,[out.Lon_es.data(i) out.Lat_es.data(i)]);
    plot(x,y);
end

function [x,y] = getEllipse(r1,r2,C)
    beta = linspace(0,2*pi,100);
    x = r1*cos(beta) - r2*sin(beta);
    y = r1*cos(beta) + r2*sin(beta);
    x = x + C(1,1);
    y = y + C(1,2);
end
```