Politécnica de Madrid

Robot Móviles de Trabajo individuales

PROFESOR:

Fernando Matia

ALUMNOS:

Marco Montagni

Curso 2011/2012

DIVISIÓN DE INGENIERÍA DE
SISTEMAS Y AUTOMÁTICA
ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

Índice

Introducción

Existen varios métodos para estimar la posición de un vehículo.

En mi caso para una mejor evaluación de la prueba escrita (en el recuento de Italia del examen de evaluación para mucho), me decidí a evaluar la posición de un vehículo bajo el agua (en un espacio 3D) desde el final de mi carrera he trabajado en el proyecto con un robot de este tipo. [1]

Por lo tanto, asume la posición y velocidad determinada por los sensores [2] con un EKF y evaluado, además de gráficamente, incluso con pruebas de Montecarlo la bondad del filtro. También usé un filtro en el asesoramiento de la de Gauss-Newton [5] para obtener medidas convergentes de los sensores.

También usé Simulink para la estructura del programa que creo que es la línea de comandos más intuitivo para obtener una forma gráfica más inmediato. R2009b de 32 bits

.

La teoría

En este capítulo me he decidido a incluir una descripción de los filtros utilizados en la formulación matemática estandarizada. [3] [4] [5] [6] [7]

1.1 - El filtrado recursivo

El problema de filtrado es estimar recursivamente el estado de un sistema dinámico utilizando un conjunto de mediciones ruidosas [5]. Un sistema dinámico de tiempo discreto es modelada por una ecuación de la ecuación de estado y de salida:

$$x(t+1) = f(t, x(t), w(t))$$
$$y(t) = h(t, x(t), v(t))$$

¿Dónde está el vector de estado, es el ruido del proceso, es el vector de las mediciones y el ruido de la medición. Si el sistema es lineal y es aditiva ruido Gaussiano la solución óptima está dada por el filtro de Kalman. En el caso de la no-lineal del sistema y / o no-gaussiana ruido o no aditivos no siempre es conocida una solución en forma cerrada.$x(t)w(t)y(t)v(t)$

1.2 - El filtro de Kalman extendido

[8] En el caso de un sistema lineal de tiempo discreto con aditivo ruido Gaussiano tenemos:

$$x(t+1) = Ax(t) + w(t)$$
$$y(t) = Cx(t) + v(t)$$

donde (proceso de ruido) es un ruido blanco con cero matriz de covarianza media:$w(t)$

$$Q(t) = E[w(t)w'(t)]$$

Mientras que v (t) (medición del ruido) es un ruido blanco con cero matriz de covarianza media:

$$R(t) = E[v(t)v'(t)]$$

También el estado inicial se modela con una variable aleatoria gaussiana caracteriza por una cierta incertidumbre y un promedio determinado, también se supone perturbaciones del proceso y la medición son independientes el uno del otro e independiente del estado inicial.

$$L(t) = P(t \mid t-1)C'(CP(t \mid t-1)C' + R)^{-1}$$

e

$$x(t \mid t) = x(t \mid t-1) + L(t)(y(t) - Cx(t \mid t-1))$$ Medición de actualización

e

$$P(t \mid t) = P(t \mid t-1) - L(t)P(t \mid t)L(t)'$$

$$x(t+1 \mid t) = Ax(t \mid t)$$

e

$$P(t+1 \mid t) = AP(t \mid t)A' + Q$$ Tiempo de actualización

Como ya se ha mencionado anteriormente en el caso no lineal no es una solución óptima en forma cerrada, pero es posible autorizar el sistema de línea en el barrio de la estimación actual, y aplicar el filtro de Kalman para el sistema linealizado, dicho filtro se denomina filtro de Kalman extendido (= filtro de Kalman extendido EKF). Considere entonces el sistema:

$$x(t+1) = f(t, x_t) + w(t)$$
$$y(t) = h(t, x(t)) + v(t)$$

¿Dónde está el Estado, procesos de ruido, medición del ruido y el vector de las mediciones. Como ya se mencionó, ee se aproximan a los de primer orden de Taylor, tenemos:$x(t)w(t)v(t)y(t)f()h()$

$$A(t) = \left[\frac{\partial f(\cdot)}{\partial x}\right]_{x=x(t|t)}$$

$$C(t) = \left[\frac{\partial h(\cdot)}{\partial x}\right]_{x=x(t|t-1)}$$

Y luego aproximar las funciones anteriores en la forma siguiente:

$$f(t, x(t)) \cong f(t, \hat{x}(t \mid t)) + A(t)[x(t) - \hat{x}(t \mid t)]$$

$$h(t, x(t)) \cong h(t, \hat{x}(t \mid t-1)) + C(t)[x(t) - \hat{x}(t \mid t-1)]$$

Habiendo hecho esta aproximación el sistema linealizado es y se puede aplicar el filtro de Kalman:

$$L(t) = P(t \mid t-1)C(t)'(C(t)P(t \mid t-1)C(t)' + R(t))^{-1}$$
e
$$x(t \mid t) = x(t \mid t-1) + L(t)(y(t) - C(t)x(t \mid t-1))$$
e
$$P(t \mid t) = P(t \mid t-1) - L(t)P(t \mid t)L(t)'$$

$$x(t+1 \mid t) = A(t)x(t \mid t)$$
$$e$$
$$P(t+1 \mid t) = A(t)P(t \mid t)A(t)' + Q(t)$$

Se observa que en el filtro de Kalman extendido, además de los errores en el proceso y los errores de medición están presentes, la linealización, que no se puede cuantificar y que no se toma en cuenta al evaluar la covarianza del error de estimación.

1.3 - Regresión

1.3.1 - La identificación paramétrica

[5] La teoría de la identificación tiene el objetivo de determinar, a partir de un conjunto de datos, un modelo del sistema dinámico, que es una función del tipo:

$$\hat{y} = g(z^{t-1}, \theta)$$

donde con indica el conjunto de las mediciones hasta que el tiempo y con un vector de parámetros que deben ser tales como para minimizar el error de predicción. El problema de encontrar el vector de parámetros se pueden formular como sigue:$z^{t-1}t - 1\theta\theta$

$$\hat{\theta_N} = argmin_\theta V_N(\theta, z^N)$$

donde el índice de rendimiento para ser minimizada se define como:

$$V_N(\theta, z^n) \triangleq \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon_{t,\theta}^F)$$

con:

$$\varepsilon_{t,\theta}^F \triangleq L(z)(y_t - \widehat{y_{t,\theta}})$$

$L(z)$BIBO estable filtro digital

$l(.)$función arbitraria positiva que es tal que para cada $l(\varepsilon) > 0\varepsilon \neq 0$

En el caso en el que el predictor es lineal y el vector de mediciones es suficientemente informativa, la solución se conoce en forma cerrada.$\hat{y} = g(z^{t-1}, \theta)z^n$

En el caso tratado en este trabajo sin embargo, el predictor no es lineal y por tanto debe recurrir a una solución numérica del problema de optimización.

1.3.2 - La regresión lineal

[4] [5] [7] El objetivo es encontrar una solución al problema de optimización:

$$minV_N(\theta, z^N)$$

Por lo general, utilizado para los algoritmos de búsqueda que operan en el tipo:

$$\theta^{(\hat{\imath}+1)}{}_N = \hat{\theta}_N^{(i)} + \alpha^{(i)} f^{(i)}$$

en la que representa el paso de actualización (escalar) con el fin de disminuir la función objetivo e indica la dirección de búsqueda; la dirección de búsqueda debe ser tal como para causar una disminución en la función de destino.αf

Uno de los métodos más comunes es el método de Newton:

$$f^{(i)} = -[V_N{}''(\hat{\theta}_N^{(i)}), z^N]^{-1} V_n'(\theta^{\hat{(\imath)}}{}_N, z^N)$$

1.3.3 - Cálculo del Jacobiano y Hessiano

[7] Supongamos que podemos expresar el factor de predicción en forma regresor lineal de:

$$\widehat{y_{t,\theta}} = \phi'{}_{t,\theta}\theta$$

donde se indica con el regresor seudo-que es dependiente y theta% indica el vector de parámetros. Por otra parte, se supone:$\phi_{t,\theta}\theta l(\varepsilon) = \frac{1}{2}\varepsilon^2$

$$V_N(\theta, z^n) = \frac{1}{2\mathrm{N}} \sum_{t=1}^{N} \varepsilon_{t,\theta}^2 \varepsilon_{t,\theta} \triangleq y_t - \phi'_{t,\theta}\theta$$

Para el cálculo del gradiente en lugar tenemos:

$$V'_N(\theta, z^N) \triangleq \frac{\partial}{\partial\theta} V_N(\theta, z^N) = -\frac{1}{N} \sum_{t=1}^{N} \partial \frac{\hat{y_{t,\theta}}}{\partial\theta} \varepsilon_{t,\theta}$$

Se define:

$$\Psi_{t,\theta} \triangleq \frac{\partial \hat{y}_t, \theta}{\partial\theta} = -\frac{\partial \varepsilon_{t,\theta}}{\partial\theta}$$

y luego tenemos el gradiente de la función objetivo:

$$V'_N(\theta, z^N) = -\frac{1}{N} \sum_{t=1}^{N} \psi_{t,\theta} \varepsilon_{t,\theta}$$

a partir de los cuales:

$$V''_N(\theta, z^N) \triangleq \frac{\partial}{\partial\theta} \left[\frac{\partial}{\partial\theta} V_N(\theta, z^N)\right]' = -\frac{1}{N} \sum_{t=1}^{N} \psi_{t,\theta}\, \partial \frac{\varepsilon_{t,\theta}}{\partial\theta} - \frac{1}{N} \sum_{t=1}^{N} \frac{\partial \psi_{t,\theta}}{\partial\theta} \varepsilon_{t,\theta}$$

El segundo término de la expresión es insignificante cerca del mínimo doveè pequeña, por lo que la expresión de Hesse se aproxima:ε

$$V''_N(\theta, z^N) \cong \frac{1}{N} \sum_{t=1}^{N} \psi_{t,\theta} \psi'_{t,\theta}$$

La estimación de la posición

Para la estimación de los sensores de posición se utilizan TOA [3] (tiempo de llegada), es generalmente sensores ultrasónicos que miden el tiempo de vuelo para volver a la distancia del vehículo objeto, a continuación, las medidas se incluirá en un algoritmo de regresión pseudolineare .

1.4 - Gauss-Newton: estimación de la posición

El objetivo es desarrollar un algoritmo recursivo que a través del método del gradiente permite actualizar la estimación de la disminución en cada iteración un índice de rendimiento especial [5]. En el caso específico, es como si calcolasse la intersección de las distancias en un espacio 3D y luego calcular la intersección de las esferas, que las medidas sería detectado por los sensores, con una incertidumbre mayor que la del sensor se utiliza un solo y por esta apropiada una EKF filtro.

Se define:

- $D(P) = [D^{1,}D^2 \cdots D^n] = [\| P - P_1 \|, \| P - P_2 \|, \ldots\ldots \| P - P_n \|]$medidas
- $D_{pr}(\hat{P}) = [D_{pr}^1, D_{pr}^2, \cdots, D_{pr}^n] = [\| \hat{P} - P_1 \|, \| \hat{P} - P_2 \|, \ldots\ldots \| \hat{P} - P_n \|]$ las medidas necesarias

- $\varepsilon(P, \hat{P}) = D(P) - D_{pr}(\hat{P})$ estimación del error de las medidas
- $\hat{P}_k, P_k$posición de estima y momento exacto t_k

Por otra parte se define el índice de rendimiento:

$V(P, \hat{P}) = \sum_{i=1}^{N} \varepsilon^2 = \sum_{t=1}^{N} (D(P) - D_{pr}(\hat{P}))^2$

Para desarrollar el algoritmo es necesario calcular el Jacobiano de la función:$\frac{\partial \varepsilon(P, \hat{P})}{\partial \hat{P}}$

$$\frac{\partial \epsilon(P,\hat{P})}{\partial \hat{P}} = -\frac{\partial D_{pr}(\hat{P})}{\partial \hat{P}} = -\begin{bmatrix} \frac{\partial D_{pr}^1}{\partial x} & \frac{\partial D_{pr}^1}{\partial y} & \frac{\partial D_{pr}^1}{\partial z} \\ \frac{\partial D_{pr}^2}{\partial x} & \frac{\partial D_{pr}^2}{\partial y} & \frac{\partial D_{pr}^2}{\partial z} \\ \vdots & & \vdots \\ \frac{\partial D_{pr}^n}{\partial x} & \frac{\partial D_{pr}^n}{\partial y} & \frac{\partial D_{pr}^n}{\partial z} \end{bmatrix}$$

Desde

$D_{pr}^1 = \sqrt{\left(\left(\widehat{P_x} - P_x^1\right)^2 + \left(\widehat{P_y} - P_y^1\right)^2 + \left(\widehat{P_z^1} - P_z^1\right)^2\right)}$

tenemos:

$$J_{1,1}(\hat{P}) = \frac{\partial D_{pr}^1}{\partial X} = \frac{\widehat{P_x}}{\sqrt{((\widehat{P_x} - P_x^1)^2 + (\widehat{P_y} - P_y^1)^2 + (\widehat{P_z^1} - P_z^1)^2)}}$$

y, a continuación:

$$J(\hat{P}) = \begin{bmatrix} \frac{\widehat{P_x}}{\sqrt{((\widehat{P_x} - P_x^1)^2 + (\widehat{P_y} - P_y^1)^2 + (\widehat{P_z} - P_z^1)^2)}} & \cdots & \frac{\widehat{P_z}}{\sqrt{((\widehat{P_x} - P_x^1)^2 + (\widehat{P_y} - P_y^1)^2 + (\widehat{P_z} - P_z^1)^2)}} \\ \vdots & & \vdots \\ \frac{\widehat{P_x}}{\sqrt{((\widehat{P_x} - P_x^n)^2 + (\widehat{P_y} - P_y^n)^2 + (\widehat{P_z} - P_z^n)^2)}} & \cdots & \frac{\widehat{P_z}}{\sqrt{((\widehat{P_x} - P_x^n)^2 + (\widehat{P_y} - P_y^n)^2 + (\widehat{P_z} - P_z^n)^2)}} \end{bmatrix}$$

Para la estimación actualizada a utilizar:

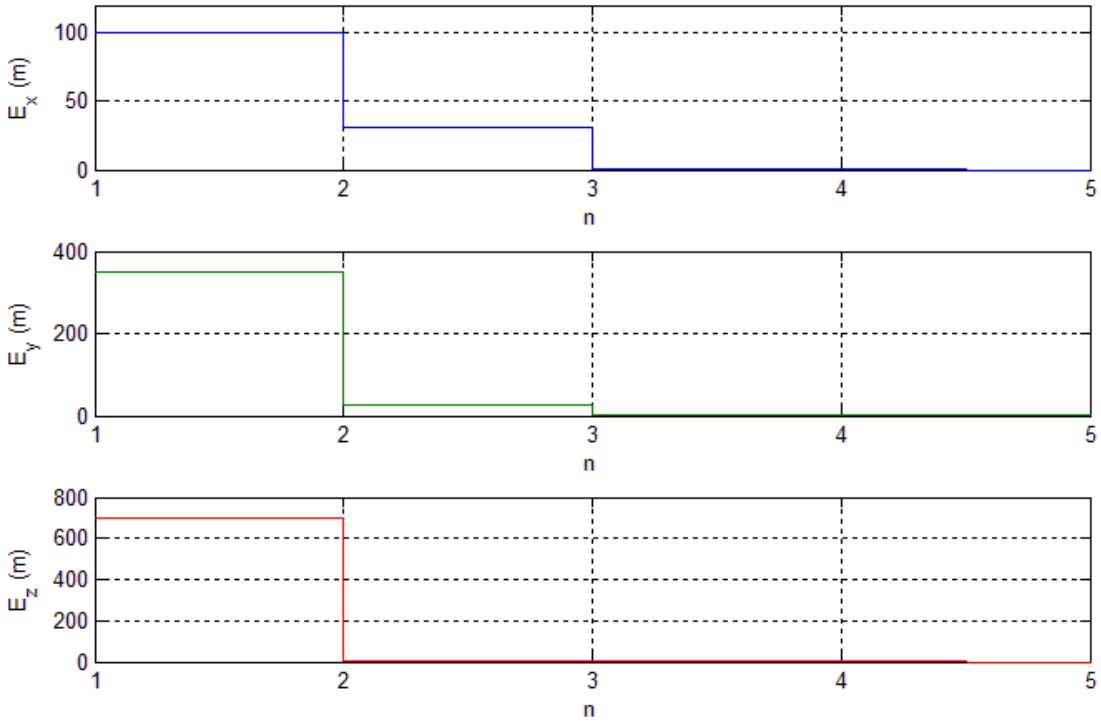
$$P_{(k+1)} = P_k + (J(\hat{P}_k)^T J(\hat{P}_k))^{-1} J(\hat{P}_k) \epsilon_k(\hat{P}_k, P_k)$$

1.5 - Estudio de la convergencia

estimación inicial (el vehículo permanece inmóvil) → [0,0,0]

posición exacta → [100350700]

Iteración n °	Error en la x	Error en y	Error en la z
0	100.0000	350.0000	700.0000
1	30.8862	25.8721	7,1209
2	0,0490	0,0610	0,0906
3	00.0000	0,0000	0,0000



1.6 - Filtro Kalman para estimar la posición

Para hacer más eficiente la estimación de las medidas de posición en la salida por el algoritmo de Gauss Newton se filtra con un filtro de Kalman extendido. El filtro de Kalman se pierde sólo si la ubicación no es cierto [8]. Considere el sistema de tiempo continuo:

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \ddot{x}\end{bmatrix}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix}$$

En el sistema se han incluido en el estado sólo la posición y velocidad, esto corresponde a una hipótesis de aceleración casi constante.

$$A^c = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \rightarrow A^k = e^A = \begin{bmatrix}1 & \Delta t\\ 0 & 1\end{bmatrix}$$

y el modelo completo es:

$$\begin{bmatrix}x(t+1)\\ y(t+1)\\ z(t+1)\\ \dot{x}(t+1)\\ \dot{y}(t+1)\\ \dot{z}(t+1)\end{bmatrix} = \begin{bmatrix}I_{3\times3} & I_{3\times3}\Delta t\\ 0_{3\times3} & I_{3\times3}\end{bmatrix}\begin{bmatrix}x(t)\\ y(t)\\ z(t)\\ \dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{bmatrix}$$

$$\begin{bmatrix}Y_x(t)\\ Y_y(t)\\ Y_z(t)\end{bmatrix} = \begin{bmatrix}I_{r\times3} & 0_{3\times3}\end{bmatrix}\begin{bmatrix}x(t)\\ y(t)\\ z(t)\\ \dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{bmatrix} + \begin{bmatrix}v_x(t)\\ v_y(t)\\ v_z(t)\end{bmatrix}$$

donde:

$[x(t), y(t), z(t), \dot{x}(t), \dot{y}(t), \dot{z}(t)]$ indica el vector de estados en el tiempo t

indica el vector de mediciones de posición en el instante t$[Y_x(t), Y_y(t), Y_z(t)]$

Por el ruido del proceso y la medición tendremos:

$$v_i(t) = N(0, \sigma_{v,i}^2) i = x, y, z v_i(t) \perp v_j(t) \forall i \neq j$$

$$w_i(t) \perp v_j(t) \perp x_j(t) \forall i \neq j \neq k$$

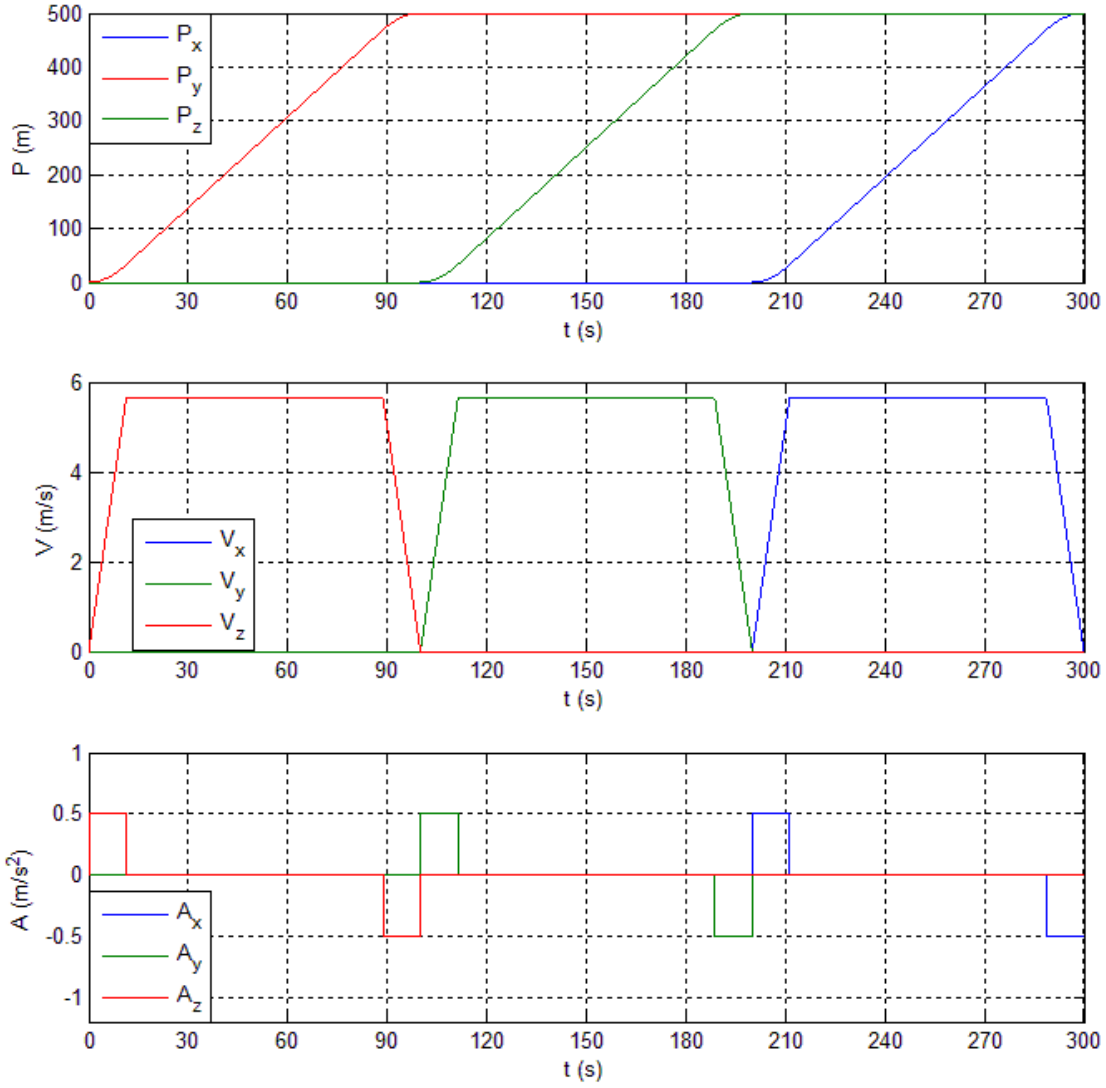
$$w_i(t) = N(0, \sigma_{w,i}^2) i = x, y, z w_i(t) \perp w_j(t) \forall i \neq j$$

Las hipótesis anteriores suponen que los disturbios son gaussiana con media cero y no correlacionados, y también que los estados son independientes de los disturbios.

Las incertidumbres de la medición del ruido y el ruido del proceso, así que he evaluado empíricamente y luego son las matrices de covarianza de ruido y de proceso:

$$R = \begin{bmatrix} 1.1 & 0 & 0 \\ 0 & 1.1 & 0 \\ 0 & 0 & 1.1 \end{bmatrix} Q = \begin{bmatrix} 0.02 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

1.7 - Monte Carlo pruebas

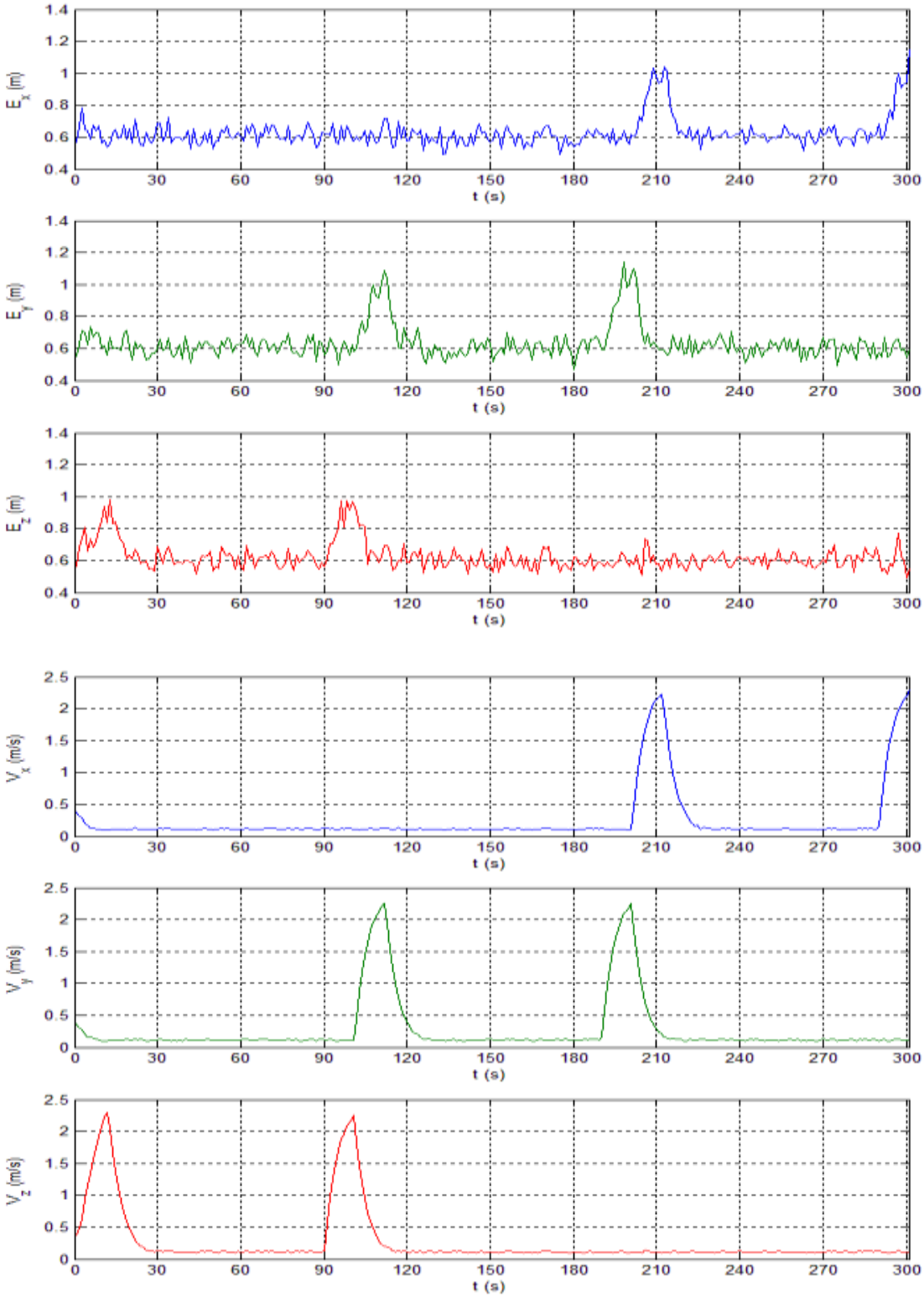


La trayectoria exacta

Estas son las trayectorias implementadas en el programa. [1] me decidí a evaluar los errores de n ensayos, al gusto personal como lo había hecho un trabajo con las pruebas de Monte Carlo, para mi proyecto de final de la carrera y ya que el error varía de un ensayo a otro. Omita una descripción detallada ya que se requiere explícitamente sólo la comprensión de la EKF.

Monte Carlo resultados de las pruebas:

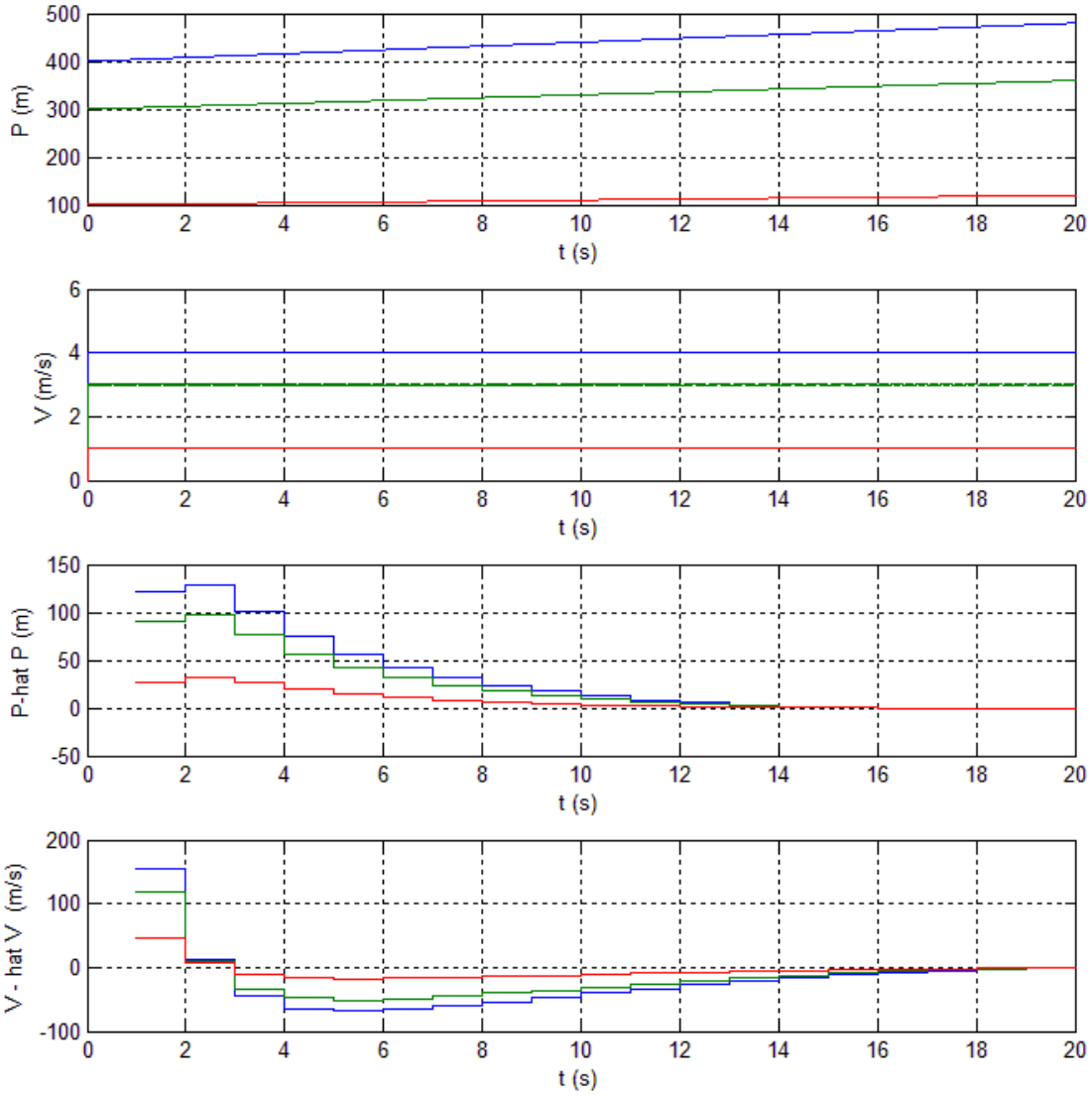
Evaluación del desempeño con la estimación inicial exacto del estado.



Evaluación del desempeño de los transitorios

Mal estado inicial:

$$x(0 \mid -1) = \begin{bmatrix} x(0) \\ y(0) \\ z(0) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} x(0) = \begin{bmatrix} 400 \\ 300 \\ 100 \\ 4 \\ 3 \\ 1 \end{bmatrix}$$



Los resultados de la pruebas de Monte Carlo

El programa

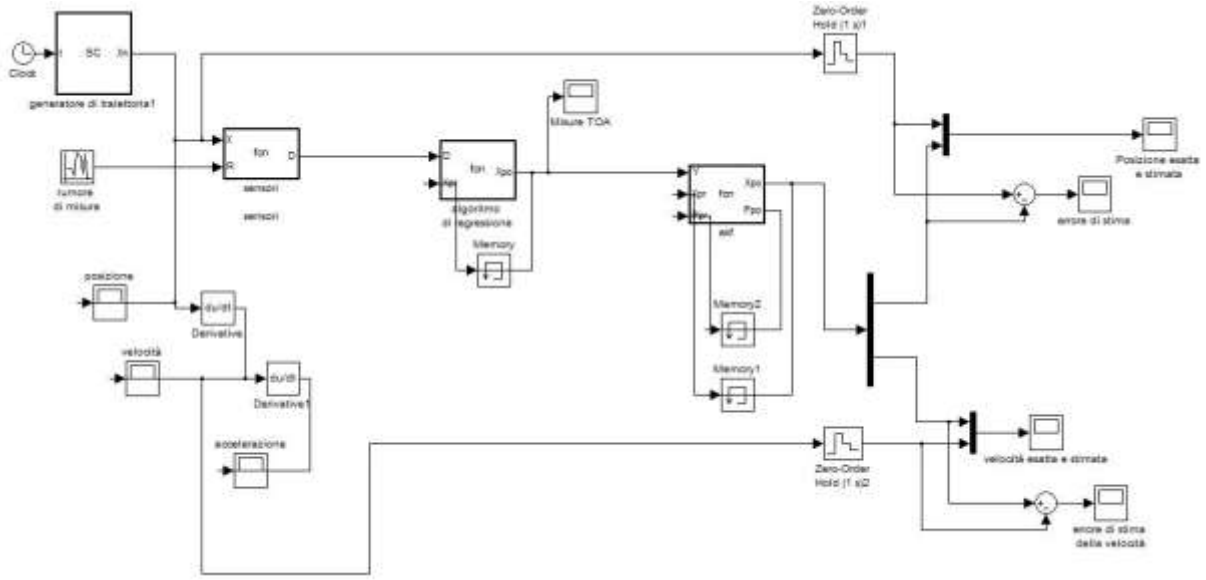


Diagrama de Simulink

El funcionamiento general del programa es como sigue:

El bloque de "generador de trayectorias" genera una trayectoria inicializado con la función "fcline", que implementan un perfil trapezoidal, que es el más utilizado en robótica para la generación de trayectorias, incluso en robots industriales, los sensores simples medirá la adición de ruido. Todo lo que entra en regresión un'algoritno estudiado previamente estima que una especie de intersección en el espacio de posibles soluciones de los 3 sensores, los bloques de memoria se almacenan con las variables que se pueden ver. La señal generada luego entra en el bloque "EKF" algoritmo en cuestión y me decidí a evaluar los errores de estimación de los gráficos.

1.8 - Código generador de trayectorias

El código para la generación de la trayectoria se compone de una función que genera la "fcline" mencionado por el bloque que genera la secuencia apropiada de la posición. El código permite que tanto para generar una trayectoria con perfil trapezoidal visto anteriormente antes de Monte Carlo pruebas, y generar un movimiento del robot a x e y luego

finalmente a z.con posición conocida inicial es malo en el estado inicial, cambiando el parámetro de Xn . El código es el siguiente:

```
function Xn       = SC(t)

eml.extrinsic('fcline');
Quat = [0;0;0;1];
om = [0;0;0];
Xn = [0;0;0];
Xnf = [Xn;0];
f = 0;
a = 0.5;
A = 500;
B = 500;
C = 500;
T = 100;

%Xn = [100;350;700];

if t>0

if t>=0 && t < T%Z
Xnf = fcline(t,[0;0;0],[0;0;A],T,a);
end
if t>=T && t< 2*T%Y
Xnf = fcline(t-T,[0;0;A],[0;B;A],T,a);
end
if t>=2*T && t < 3*T%X
Xnf = fcline(t-2*T,[0;B;A],[C;B;A],T,a);
end
if t>=3*T && t< 4*T%-Y
Xnf = fcline(t-3*T,[C;B;A],[C;0;A],T,a);
end
if t>=4*T && t< 5*T%X
Xnf = fcline(t-4*T,[C;0;A],[0;0;A],T,a);
end
if t>=5*T && t< 6*T%X
Xnf = fcline(t-5*T,[0;0;A],[0;0;0],T,a);
end

if t>=6*T
Xn = [0;0;0];
end

Xn = Xnf(1:3);
f = Xnf(4);

function [Xn] = fcline(t,Xn1,Xn2,T,acc)
r = [0;0;0];
tc = 0;
d = 0;
Xn = [0;0;0;0];
Xn(1:3) = Xn1;
a = acc;
d = norm(Xn2-Xn1);
r = (Xn2-Xn1)/d;
```

```
tc = T/2-0.5*sqrt(1/a*(T^2*a-4*d));

if t<=tc && t>0
    Xn(1:3) =Xn1+ r*0.5*a*t*t;
end
if t>tc && t<=T-tc
    Xn(1:3) =Xn1+ r*(0.5*a*tc*tc+ a*tc*(t-tc));
end
if t>(T-tc) && t<=T
    Xn(4) = 1;
    Xn(1:3) = Xn2 -r*(0.5*a*(T-t)*(T-t));
end
if t > T
    Xn(4) = 0;
    Xn(1:3) = Xn2;
end
```

1.9 - Los sensores

He optado por utilizar los sensores "El tiempo de llegada", también conocido como TOA [3], fácil de implementar, que mediante el cálculo del tiempo de vuelo de la señal y conociendo la velocidad de propagación de los medios de calcular la distancia mínima y son puntiformes y por lo tanto con el " intersección de 3 mediciones, siendo posicionadas en puntos diferentes Xp, la posición del vehículo.

```
function D  = fcn(X,R)
Xp = zeros(3,6);
Xp(1:3,1) = [10000;0;0];
Xp(1:3,2) = [0;10000;0];
Xp(1:3,3) = [0;0;10000];



D = zeros(3,1);
for i = 1:length(D)
D(i) = norm(X-Xp(1:3,i));
end
D = D + R;
```

1.10 - Gauss-Newton

El algoritmo es fácil de implementar después de leer el halp de MATLAB y [4] [5]. El algoritmo se encarga de 3 tamaños de los tres sensores e interpola ¿cómo 'para obtener la intersección de las esferas en 3D y luego obtener la posición claramente por encima del ruido

en todo lo que será "más" de ruido en las mediciones individuales . En este caso, el algoritmo del código es como sigue:

```
function Xpo    = fcn(D,Xpr)
Xp = zeros(3,3);
Xp(1:3,1) = [10000;0;0];
Xp(1:3,2) = [0;10000;0];
Xp(1:3,3) = [0;0;10000];


%Algoritmo di Gauss-Newton------------------
Dpr = zeros(length(D),1);
for i = 1:length(D);
Dpr(i) = norm(Xpr(1:3)-Xp(1:3,i));
end
e = D-Dpr;
g = -[  (Xp(1:3,1)-Xpr(1:3))'/norm(Xpr(1:3)-Xp(1:3,1));...
        (Xp(1:3,2)-Xpr(1:3))'/norm(Xpr(1:3)-Xp(1:3,2));...
        (Xp(1:3,3)-Xpr(1:3))'/norm(Xpr(1:3)-Xp(1:3,3));...
     ]';


%Xpo = Xpr(1:3) + g*e;
Xpo = Xpr(1:3) +inv(g*g')*g*e;
%Fine algoritmo di Gaus-Newton------------------
```

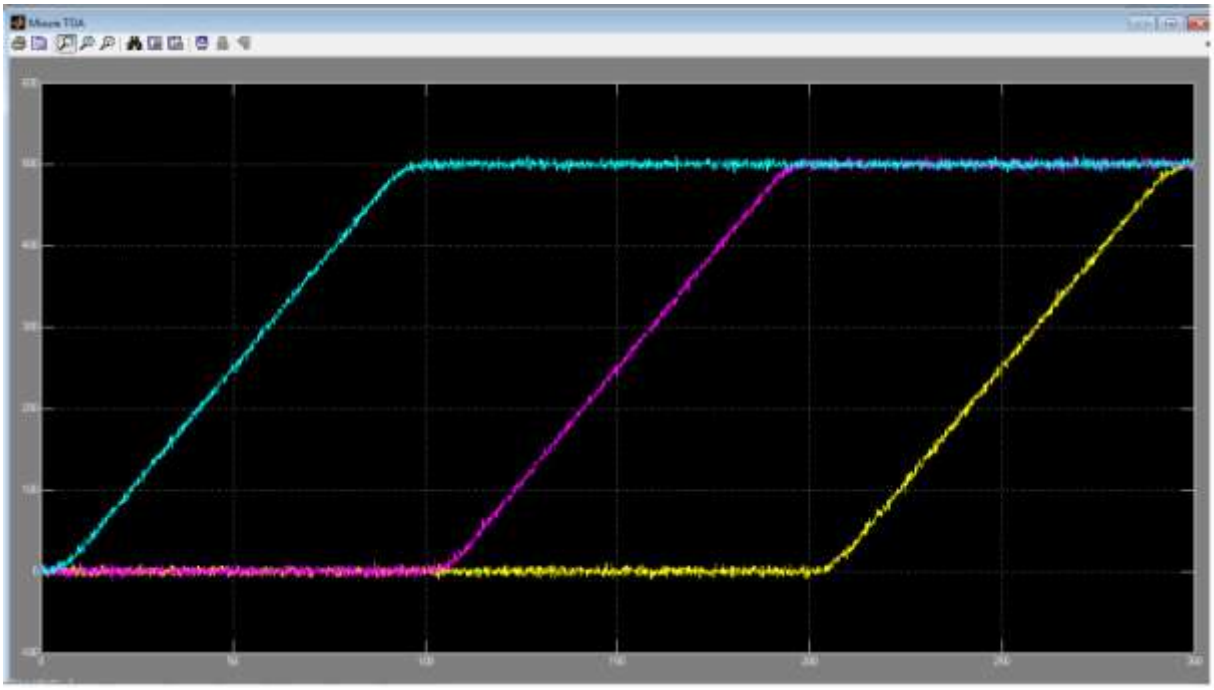
1.11 - EKF

En el lanzamiento del programa se puede ver hacer varias comparaciones y evaluar la bondad del filtro.

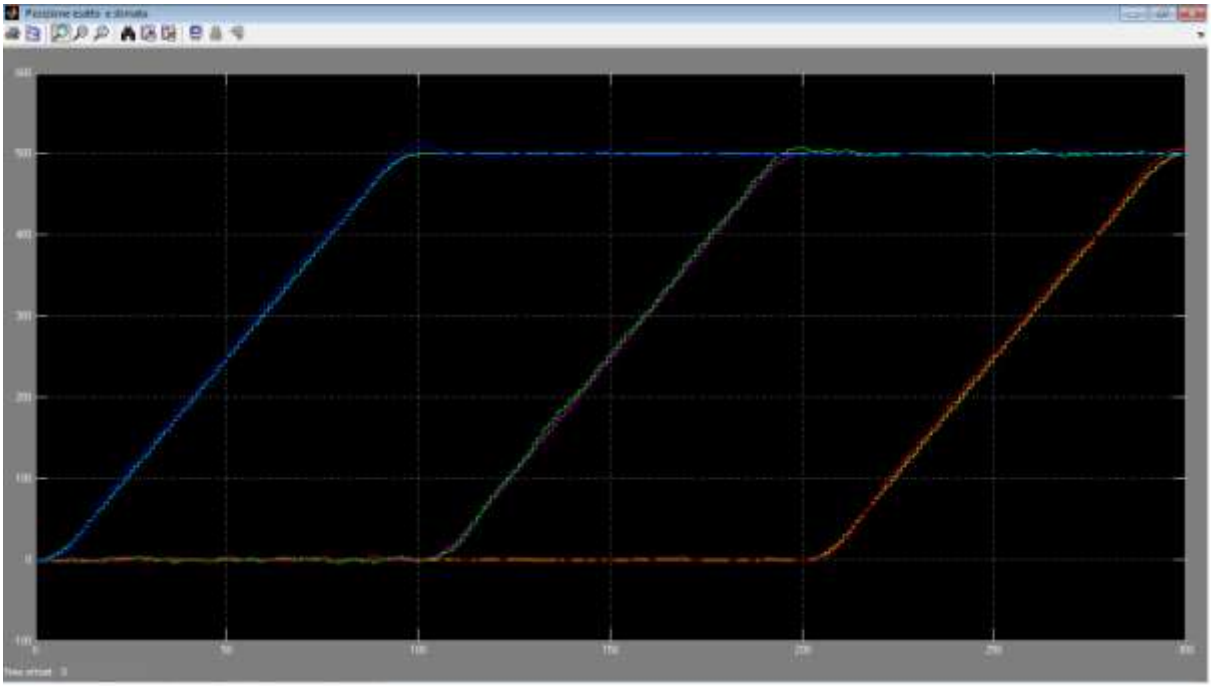
```
function [Xpo,Ppo]   = fcn(Y,Xpr,Ppr)
Xpo = zeros(6,1);
Xsucc = zeros(6,1);
Psucc = zeros(6);
dt = 1;
K = zeros(6,3);
A = [eye(3),eye(3)*dt;zeros(3),eye(3)];
C = [eye(3),zeros(3)];
R = diag([[1;1;1]*1.1])*10;
Q = diag([[1;1;1]*0.01;[1;1;1]*0.05]);
K = Ppr*C'*inv(C*Ppr*C'+R);
Xpo = Xpr+K*(Y-C*Xpr);
Ppo = Ppr-K*(C*Ppr*C'+R)*K';

Xpo = A*Xpo;
Ppo = A*Ppo*A'+Q;
```

Una de las comparaciones que pone en pie a cabo la bondad del filtro es el equilibrio entre la entrada y la salida donde se ve que la posición de x, y, z obtiene una señal

extraordinariamente limpio. Usted puede obtener una evaluación más a fondo la ampliación de los mismos puntos en el programa gráfico.



Por encima de las mediciones de los sensores. Bajo la salida del filtro Kalman.



Se puede ver claramente una disminución significativa en el ruido.

Referencias

[1] Marco montagni "Diseño e implementación de un motor para submarinos", Julio 2011

[2] Fecha imaginex

[3] Yee-Cheong Jin Jong-Hwan Kim, "ya que en fragancia Filtrado de un cuaternión Espacio párrafo de la Unidad de la nave estimación Espacial Actitud" 1-4244-0755-9/07 / '2007 IEEE

[4] "Modelado y control de sistemas mecánicos y eléctricos" notas del curso, Florencia 2011

[5] "Notas de Identificación de las estimación y" Notas del Curso, Florencia 2011

[6] B. Siciliano, L. Sciavicco, L. Y G. Villani Oriolo, "Modelado, programación y control, McGraw-Hill, 2008 Milán.

[7] "Robótica Industrial" Notas del curso, Florencia 2010

[8] "Robot Móviles" Notas del curso, 2012 Madird