INTRODUCCIÓN A LA ROBÓTICA

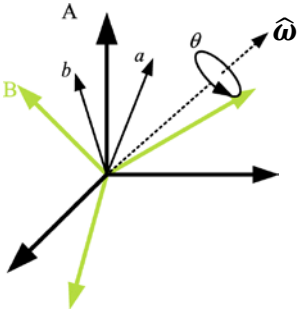
TIEMPO Y MOVIMIENTO

Indice:

Indice:

Matriz exponencial.

- Considerando una rotación de ángulo θ alrededor de un eje definido por un vector unitario $\hat{\boldsymbol{\omega}}$, podemos describir esta rotación utilizando la notación de matriz exponencial y matrices sesgadas simétricas:



Otra forma de expresar rotaciones!

$$\boldsymbol{R_x(\theta)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} = e^{[\hat{\omega}]_\times \theta} \in \boldsymbol{SO(3)}$$

- Donde la notación $[\cdot]_\times : \mathbb{R}^3 \longmapsto \mathbb{R}^{3\times 3}$ indica un mapeo de un vector a una matriz sesgada simétrica
- Llamábamos matriz sesgada simétrica a aquella matriz cuyos elementos cumplen: $a_{ij} = -a_{ji}$

$$[\hat{\omega}]_x = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

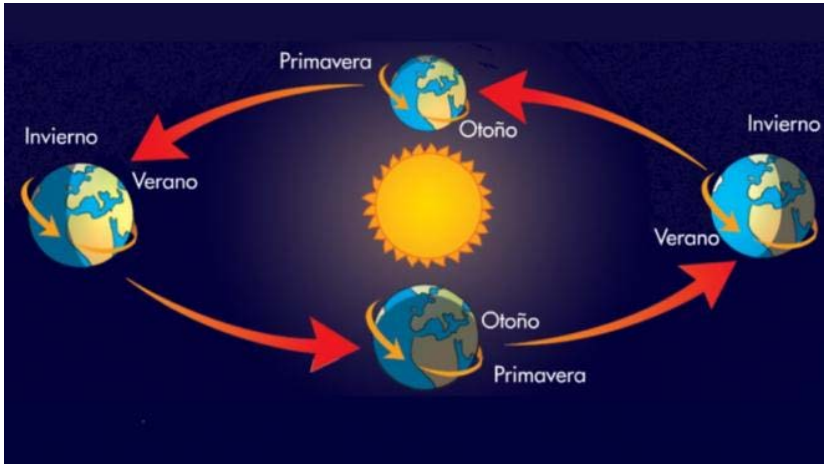
Notar que con esta matriz puedo convertir el producto vectorial en producto matricial:

$$\boldsymbol{v_1 \times v_2 = [v_1]_\times \cdot v_2}$$

Indice:

IR: 2 – Derivada de una pose

- Una pose se compone de la posición y la rotación relativa a un marco de coordenadas, por tanto su derivada contendrá una parte debida a una velocidad traslacional y otra rotacional.
  - La **velocidad traslacional** es directamente la derivada del cambio de posición (vector).
  - La **velocidad rotacional** requerirá la derivada de las diversas formas para su representación.



Velocidad rotacional.

- Tomando la forma exponencial para una rotación y calculando su **derivada** podemos obtener (supondremos por simplicidad que el eje de rotación instantáneamente no cambia de dirección):

$$^{A}\mathbf{R}_{B}(t) = e^{[^{A}\hat{\omega}(\mathbf{t})]_{\times}\theta(t)} \in \mathbf{SO(3)}$$

$$^{A}\dot{\mathbf{R}}_{B}(t) = [^{A}\hat{\omega}(\mathbf{t})]_{\times}\dot{\theta}e^{[^{A}\hat{\omega}(\mathbf{t})]_{\times}\theta(t)} = [^{A}\hat{\omega}(\mathbf{t})]_{\times}\dot{\theta}\,^{A}\mathbf{R}_{B}(t) \in \mathbb{R}^{3\times 3}$$

- Que puede ser escrito como: $^{A}\dot{\mathbf{R}}_{B}(t) = [^{A}\omega]_{\times}\,^{A}\mathbf{R}_{B}(t) \in \mathbb{R}^{3\times 3}$

Expresa como cambia la orientación entre los marcos de referencia en el tiempo!

- Donde $^{A}\omega = ^{A}\hat{\omega}\dot{\theta}$ es la velocidad angular en el marco $\{A\}$ (**ver que se trata de un vector**) y que $\left\|^{A}\omega\right\|$ es la magnitud de la rotación alrededor del eje descrito por el vector unitario.
- Si deseamos utilizar la velocidad rotacional expresada en el marco $\{B\}$ para expresar el cambio podemos relacionar ambos marcos con $^{A}\omega = {}^{A}\mathbf{R}_{B}\,{}^{B}\omega$ y luego con la propiedad $[Av]_{\times} = A[v]_{\times}A^{T}$ llegar a: $^{A}\dot{\mathbf{R}}_{B}(t) = {}^{A}\mathbf{R}_{B}[^{B}\omega]_{\times} \in \mathbb{R}^{3\times 3}$
- En forma equivalente se puede llegar a una expresión de la **derivada del cuaternión unitario**:

$$^{A}\dot{\mathbf{q}}_{B} = \frac{1}{2}\,^{A}\mathring{\omega} \circ {}^{A}\mathring{\mathbf{q}}_{B} = \frac{1}{2}\,^{A}\mathring{\mathbf{q}}_{B} \circ {}^{B}\mathring{\omega} \in \mathbb{H}$$

$\mathring{\omega}$ Es un cuaternión puro formado por el vector de velocidad angular

Derivada de una pose. Matriz de transformación homogénea

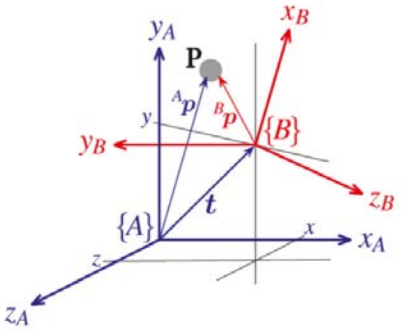
- Considerando la expresión de una pose mediante una matriz de transformación homogénea, y realizando la derivada con respecto al tiempo podemos obtener:

$$\xi \sim {}^{A}\mathbf{T}_{B} = \begin{pmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\mathbf{t}_{B} \\ 0_{1\times 3} & 1 \end{pmatrix}$$

$$\dot{\xi} \sim {}^{A}\dot{\mathbf{T}}_{B} = \begin{pmatrix} {}^{A}\dot{\mathbf{R}}_{B} & {}^{A}\dot{\mathbf{t}}_{B} \\ 0_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} [{}^{A}\omega]_{\times}\,{}^{A}\mathbf{R}_{B} & {}^{A}\dot{\mathbf{t}}_{B} \\ 0_{1\times 3} & 0 \end{pmatrix}$$

- La tasa de cambio queda descripta por la orientación ${}^{A}\mathbf{R}_{B}$ y dos velocidades:
  - La velocidad linear o traslacional $v = {}^{A}\dot{\mathbf{t}}_{B}$ velocidad del origen de $\{B\}$ respecto de $\{A\}$
  - La velocidad angular ${}^{A}\omega_{B}$
- Combinando ambas velocidades se puede formar el vector de velocidad espacial, que describe la velocidad instantánea del marco $\{B\}$ con respecto al marco $\{A\}$:

$${}^{A}\nu_{B} = ({}^{A}v_{B}, {}^{A}\omega_{B}) \in \mathbb{R}^{6}$$



Movimiento incremental (discreto).

- Considerando la aproximación de primer orden para la derivada de una rotación:

$$\dot{\mathbf{R}} = \frac{\mathbf{R}\langle t+\delta_t\rangle - \mathbf{R}\langle t\rangle}{\delta_t} \in \mathbb{R}^{3\times 3}$$

- Y considerando el movimiento del marco $\{B\}$ en dos pasos de tiempo consecutivos puede relacionarse por una pequeña rotación: ${}^{\mathbf{B}}\mathbf{R}_{\boldsymbol{\Delta}}$

$$\mathbf{R}_{\mathbf{B}}\langle t+\delta_t\rangle = \mathbf{R}_{\mathbf{B}}\langle t\rangle {}^{\mathbf{B}}\mathbf{R}_{\boldsymbol{\Delta}}$$

- Tomando la formula ${}^{A}\dot{\mathbf{R}}_B(t) = {}^{A}\mathbf{R}_B[{}^{B}\omega]_\times \in \mathbb{R}^{3\times 3}$ puede despejarse:

$${}^{\mathbf{B}}\mathbf{R}_{\boldsymbol{\Delta}} \approx \delta_t[{}^{\mathbf{B}}\omega]_\times + \mathbf{I}_{3\times 3}$$

- **Esta última ecuación nos indica que una pequeña rotación puede aproximarse por la suma de una matriz sesgada simétrica y una matriz identidad.**

Movimiento incremental (discreto).

- Considerando lo anterior podemos aproximar el vector de velocidad angular:

$$\omega \approx \frac{1}{\delta_t} V_\times(\mathbf{R_B}\langle t\rangle^T \mathbf{R_B}\langle t+\delta_t\rangle - \mathbf{I}_{3\times 3})$$

- Donde $V_\times(\cdot)$ es la inversa del operador de matriz sesgada simétrica $\mathbf{S} = [v]_\times \Longrightarrow v = V_\times(\mathbf{S})$
- Alternativamente si la velocidad angular en el marco $\{B\}$ es conocida, podemos aproximar la actualización de la matriz de rotación como:

Esto sigue siendo ortonormal? => Necesito normalización!

$$\mathbf{R_B}\langle t+\delta_t\rangle \approx \mathbf{R_B}\langle t\rangle + \delta_t \mathbf{R_B}\langle t\rangle [\omega]_\times$$

Base para la navegación inercial!

- Un resultado análogo puede alcanzarse con el cuaternión unitario

$$\mathring{\mathbf{q}}\langle k+1\rangle \approx \mathring{\mathbf{q}}\langle k\rangle + \frac{\delta_t}{2}\mathring{\mathbf{w}} \circ \mathring{\mathbf{q}}\langle k\rangle$$

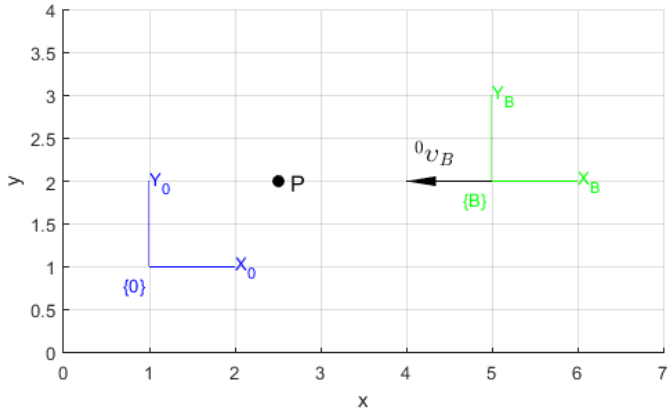
- Finalmente el movimiento incremental de un cuerpo rígido puede aproximarse por

$$\xi_\Delta \sim \mathbf{T}_\Delta = \begin{pmatrix} \mathbf{R}_\Delta & \mathbf{t}_\Delta \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} [\mathbf{\Delta}_R]_\times + \mathbf{I}_{3\times 3} & \mathbf{\Delta}_t \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}$$

Indice:

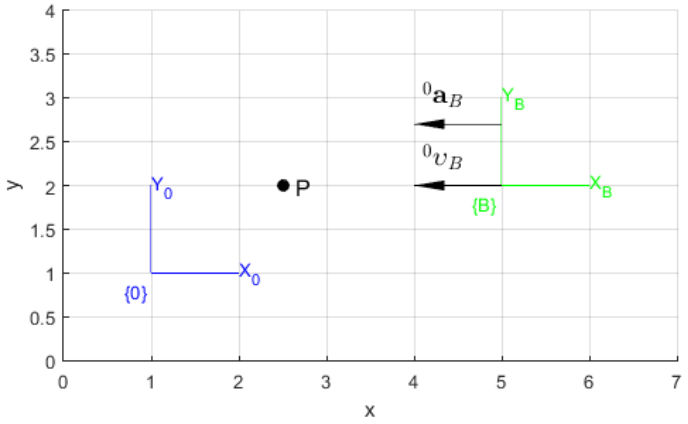
Marco de referencia inercial.

- Se trata de "**un marco que no esta acelerando o rotando**".
- Si consideramos una partícula P, un marco estacionario $\{0\}$ y otro en movimiento $\{B\}$



Si consideramos solamente una velocidad constante -> ambos marcos son inerciales.

Si esta presente una aceleración {B} es un marco no inercial.



Dinámica de cuerpos en movimiento.

- La segunda Ley de Newton, **en un marco inercial**, describe la aceleración de una partícula con posición x y masa m debida a una fuerza f como:

$$m\,{}^{0}\ddot{\mathbf{x}} = {}^{0}\mathbf{f}$$

- De la misma forma para el **movimiento rotacional en $SO(3)$**, la ecuación de Euler de rotación relaciona la aceleración angular de un cuerpo con el torque aplicado y una matriz de inercias rotacionales J

$${}^{B}\mathbf{J}\,{}^{B}\dot{\omega} + {}^{B}\omega \times ({}^{B}\mathbf{J}\,{}^{B}\omega) = {}^{B}\tau$$

En el caso 3D, aun en ausencia de torque la velocidad es variante en el tiempo!

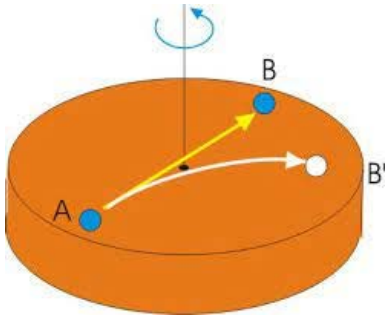
- Ver que la matriz de inercias rotacionales J es simétrica. Los elementos diagonales son los momentos positivos de inercia, mientras que los que se encuentran fuera de la diagonal son los llamados productos de inercia. Ver que solo 6 elementos son únicos. En el caso que la distribución de masa del cuerpo rotante respecto del eje de giro sea simetría los productos inerciales son cero.

$$\mathbf{J} = \begin{pmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{pmatrix}$$

Notar que J tiene un marco de referencia asociado

Marco de referencia inercial.

- Para el caso de un marco de referencia rotando las cosas son un poco más complejas:



- Para el caso que el marco de referencia $\{B\}$ este rotando con velocidad angular ω la segunda ley de Newton es:

$$m\left({}^{B}\dot{v} + \underbrace{\omega \times (\omega \times {}^{B}\mathbf{p})}_{\text{Centrífuga}} + \underbrace{2\omega \times {}^{B}v}_{\text{Coriolis}} + \underbrace{\frac{\partial \omega}{\partial t} \times {}^{B}\mathbf{p}}_{\text{Euler}}\right) = {}^{0}\mathbf{f}$$

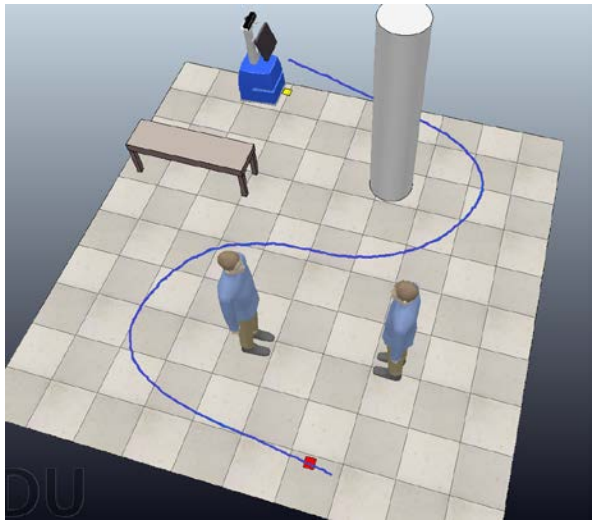
- En robótica el termino **marco inercial (inertial frame)**)y **marco mundo (world frame)** se utiliza en forma indistinta para indicar un marco de referencia fijo en algún punto de la Tierra. Como contraparte se hacer referencia al **marco "cuerpo" (body-frame)** al marco situado en el cuerpo del robot.

Camino vs. trayectoria.

- **Camino:** secuencia de puntos en el espacio a recorrer en un determinado orden que estable el sentido del mismo.

- **Trayectoria:** secuencia de puntos en el espacio a recorrer donde cada punto tiene asociada una marca temporal. (Además de estar en un punto del espacio debemos estar en un tiempo determinado)

Una característica importante de una trayectoria es que es "suave". Posición y orientación varían suavemente en el tiempo. Suave significa que las primeras derivadas son continuas: velocidad, aceleración y opcionalemnte jerk (tirón)



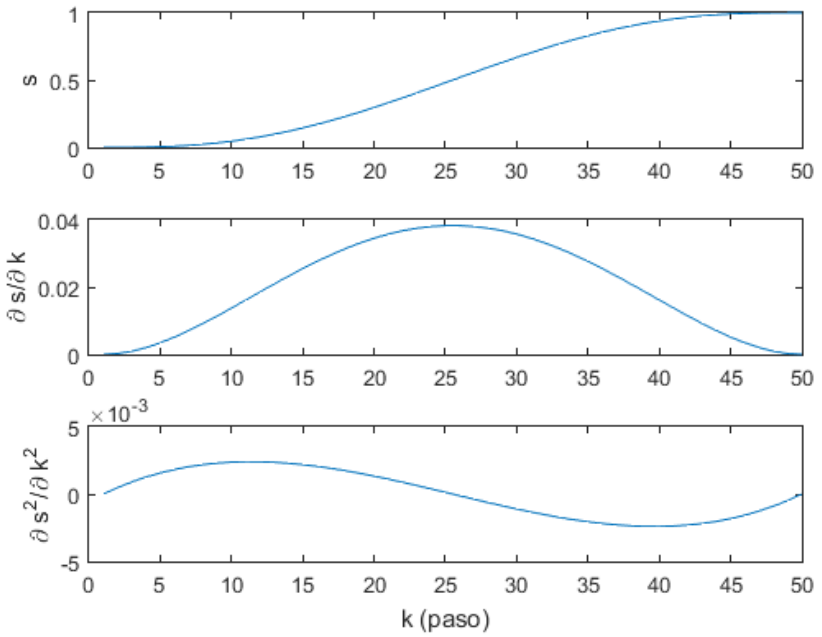
Trayectorias suaves 1D. Polinomios de 5to orden.

- **Objetivo:** función escalar con punto inicial y final definidos, y primeras derivadas continuas. -> **polinomios en función del tiempo** $t \in [0, T]$

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

- Por ejemplo, si generamos un polinomio para una trayectoria entre 0 y 1 con derivadas iniciales y finales nulas obtenemos:

```
[s,sd,sdd]=tpoly(0,1,50);
subplot(3,1,1)
plot(s)
ylabel('s')
subplot(3,1,2)
plot(sd)
ylabel('\partial s/\partial k')
subplot(3,1,3)
plot(sdd)
ylabel('\partial s^2/\partial k^2')
xlabel('k (paso)')
```



Código MATLAB – con equivalente similar en python!

Trayectorias suaves 1D. Polinomios de 5to orden.

- **Objetivo:** función escalar con punto inicial y final definidos, y primeras derivadas continuas. -> **polinomios en función del tiempo** $t \in [0, T]$

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

- Por ejemplo si generamos un polinomio para una trayectoria entre 0 y 1 con derivadas iniciales y finales nulas obtenemos:

```
[s,sd,sdd]=tpoly(0,1,50);
subplot(3,1,1)
plot(s)
ylabel('s')
subplot(3,1,2)
plot(sd)
ylabel('\partial s/\partial k')
subplot(3,1,3)
plot(sdd)
ylabel('\partial s^2/\partial k^2')
xlabel('k (paso)')
```

Código MATLAB – con equivalente similar en python!

No aprovecho velocidad máxima de actuador en todo el recorrido!

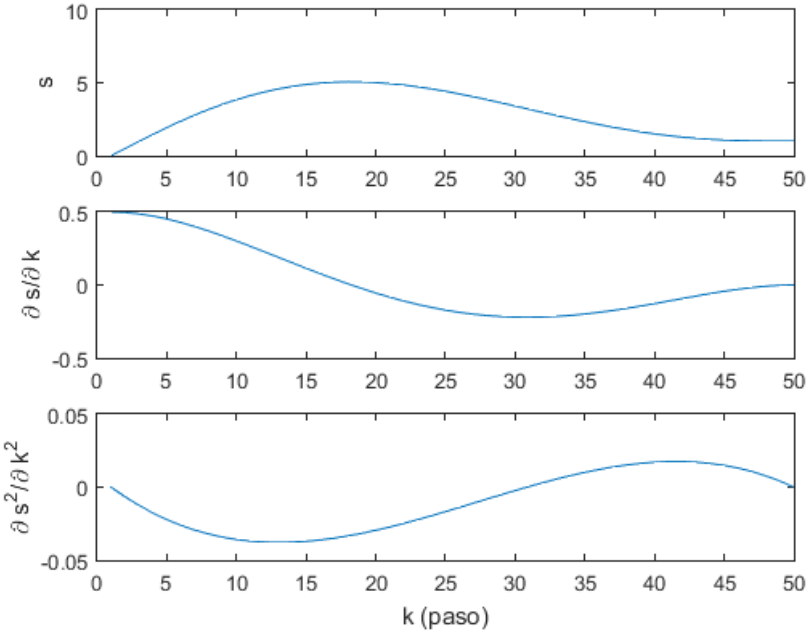
Trayectorias suaves 1D. Polinomios de 5to orden.

- **Objetivo:** función escalar con punto inicial y final definidos, y primeras derivadas continuas. -> **polinomios en función del tiempo** $t \in [0, T]$

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

- Por ejemplo, si generamos un polinomio para una trayectoria entre 0 y 1 con velocidad inicial 0.5 y final 0 obtenemos:

```
[s,sd,sdd]=tpoly(0,1,50,0.5,0);
subplot(3,1,1)
plot(s)
ylabel('s')
subplot(3,1,2)
plot(sd)
ylabel('\partial s/\partial k')
subplot(3,1,3)
plot(sdd)
ylabel('\partial s^2/\partial k^2')
xlabel('k (paso)')
```



Código MATLAB – con equivalente similar en python!

Trayectorias suaves 1D. Polinomios de 5to orden.

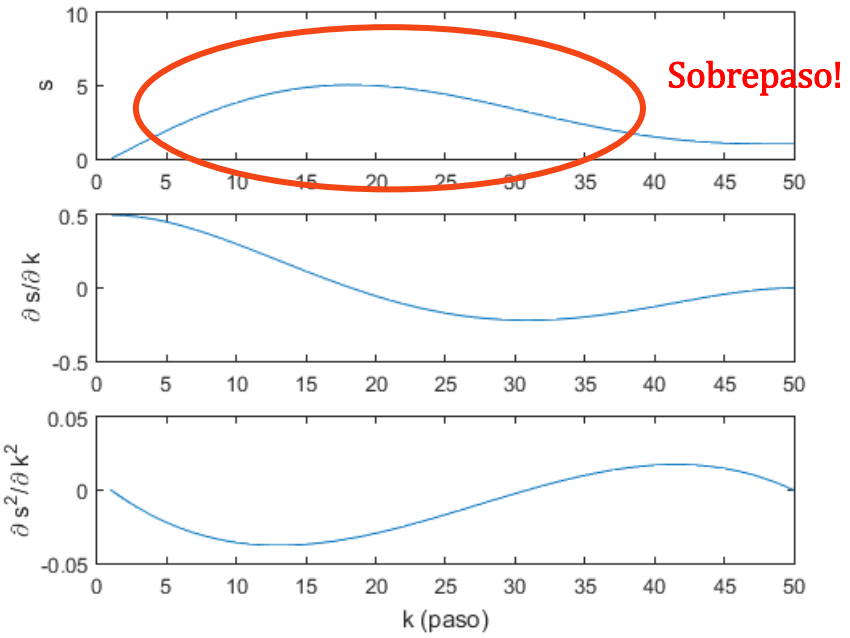
- **Objetivo:** función escalar con punto inicial y final definidos, y primeras derivadas continuas. -> **polinomios en función del tiempo** $t \in [0, T]$

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

- Por ejemplo, si generamos un polinomio para una trayectoria entre 0 y 1 con velocidad inicial 0.5 y final 0 obtenemos:

```
[s,sd,sdd]=tpoly(0,1,50,0.5,0);
subplot(3,1,1)
plot(s)
ylabel('s')
subplot(3,1,2)
plot(sd)
ylabel('\partial s/\partial k')
subplot(3,1,3)
plot(sdd)
ylabel('\partial s^2/\partial k^2')
xlabel('k (paso)')
```

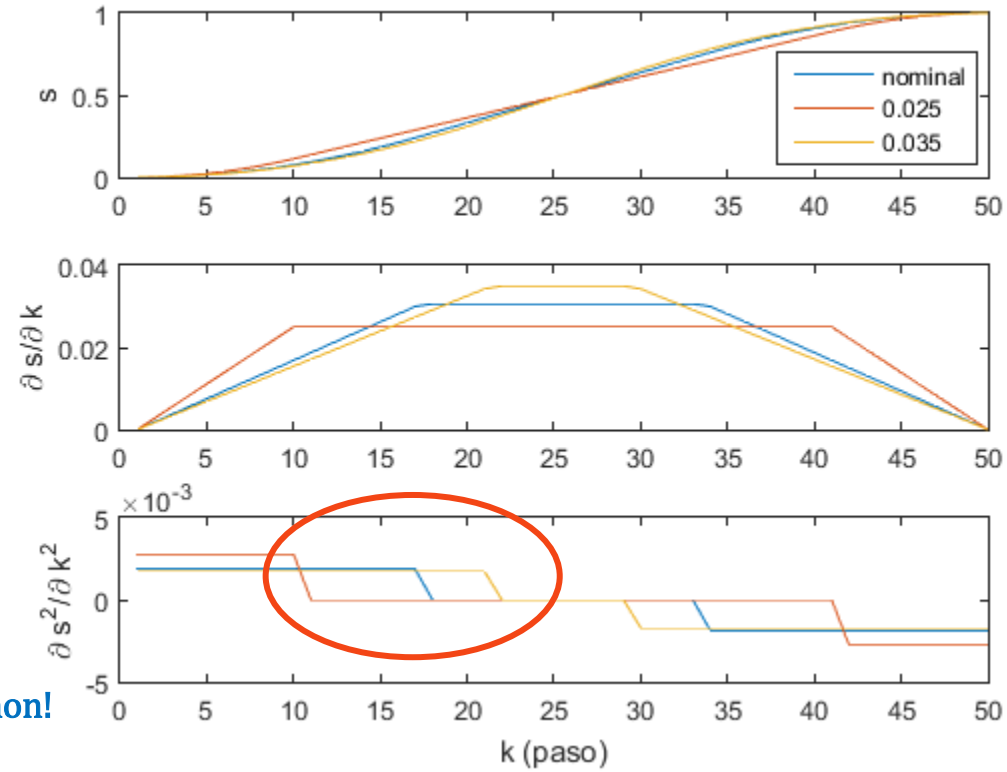
Código MATLAB – con equivalente similar en python!



Trayectorias suaves 1D. Trayectorias hibridas (trapezoidales).

- Funciones que mezclan tramos lineales (a velocidad constante) y uniones de tipo parabólicas (para aceleraciones y desaceleraciones)
- Por ejemplo, si generamos un polinomio para una trayectoria entre 0 y 1 :

```
[s,sd,sdd]=lspb(0,1,50);
%[s,sd,sdd]=lspb(0,1,50,0,025);
%[s,sd,sdd]=lspb(0,1,50,0,035);
subplot(3,1,1)
plot(s)
ylabel('s')
subplot(3,1,2)
plot(sd)
ylabel('\partial s/\partial k')
subplot(3,1,3)
plot(sdd)
ylabel('\partial s^2/\partial k^2')
xlabel('k (paso)')
```



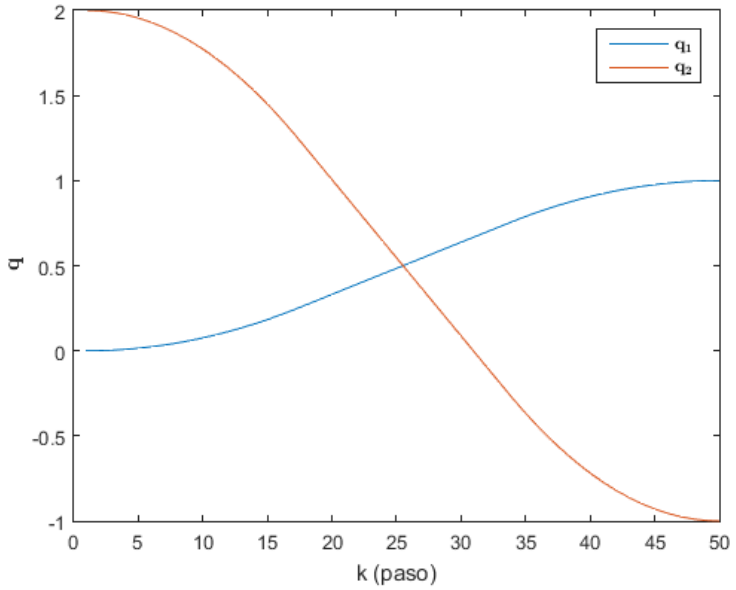
Código MATLAB – con equivalente similar en python!

Trayectorias suaves Multi-Dimensión.

- **Objetivo**: extender lo visto en escalar a vectorial.
- En el caso multi dimensional, representaremos los movimientos en el **espacio de configuración (articular) del robot**. Este será representado por un vector $\mathbf{q} \in \mathbb{R}^N$ donde N son los grados de libertad.
- Para un robot móvil con ruedas podemos expresarlo como: $\mathbf{q} = (x, y) \quad o \quad \mathbf{q} = (x, y, \theta)$
- Para el caso de la orientación de un cuerpo en 3d $\mathbf{q} = (\theta_r, \theta_p, \theta_y)$
- Para una pose en $SE(3)$: $\mathbf{q} = (x, y, z, \theta_r, \theta_p, \theta_y)$

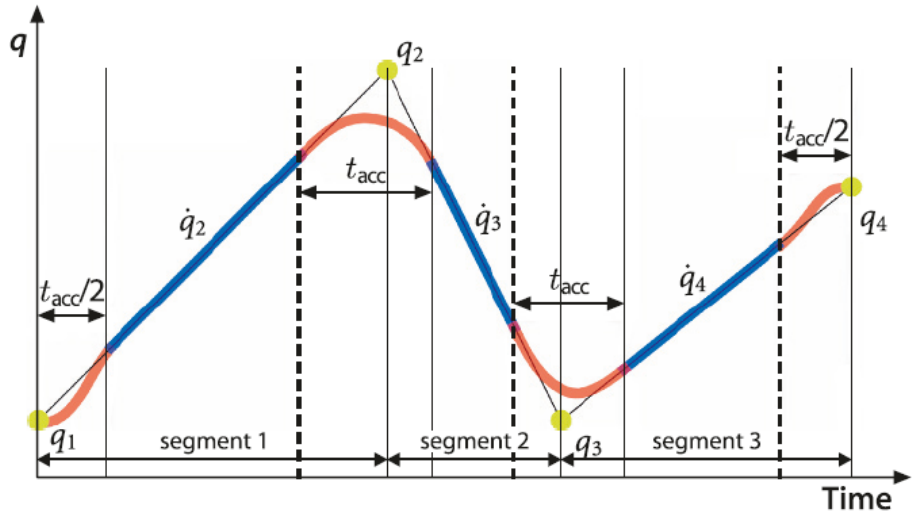
```
q=mtraj(@lspb,[0 2],[1 -1], 50);
plot(q)
ylabel('$\mathbf{q}$','Interpreter',
'latex')
xlabel('k (paso)')
legend({'$\mathbf{q_1}$','$\mathbf{q_2}$'},
'Interpreter', 'latex')
```

Código MATLAB – con equivalente similar en python!



Trayectorias suaves Multi-segmento.

- **Problema**: trayectoria definida por Mconfiguraciones q_k, $k \in [1, M]$, donde existen M−1 segmentos de movimiento.
- El robot se desplaza del punto inicial al final, pasando por las posiciones intermedias (o en forma cercana).



Como parámetros de configuración son usuales:

- Velocidad de cada segmento
- El tiempo en cada punto t_{acc}
- Conociendo la máxima aceleración del eje, el tiempo t_{acc} puede despejarse a partir de:

$$\ddot{q} = \frac{\dot{q}_{k+1} - \dot{q}_k}{t_{acc}}$$

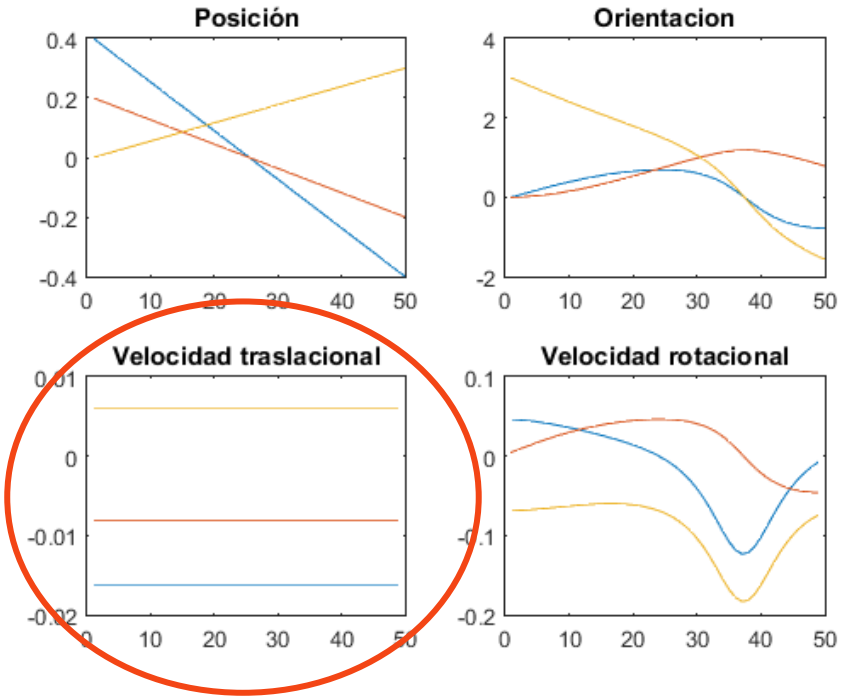
Trayectorias suaves. Movimiento cartesiano

- **Problema**: generación de caminos entre poses en $\boldsymbol{SE}(3)$.
- Opciones:
- Interpolar transformaciones homogéneas -> problemas de condiciones iniciales
- Combinar enfoque polinomial, o hibrido junto con interpolación de transformaciones homogéneas.

Mediante interpolación de transformaciones homogéneas.

```
T0=SE3([0.4 .2 0])*SE3.rpy(0,0,3);
T1=SE3([-0.4 -.2 0.3])*SE3.rpy(-
pi/4,pi/4,-pi/2);
TS=interp(T0,T1,50);
TS.animate()
```

Código MATLAB – con equivalente similar en python!



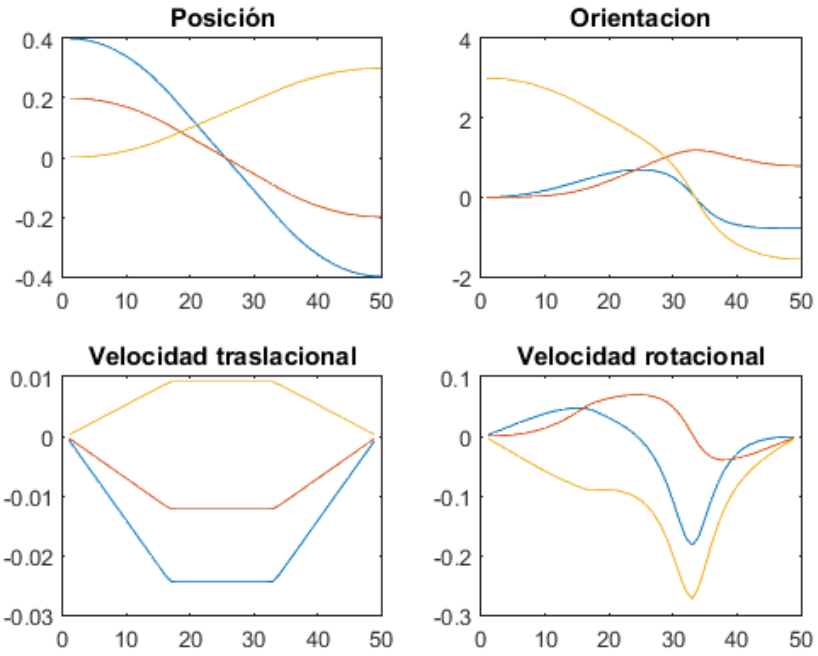
Trayectorias suaves. Movimiento cartesiano

- **Problema**: generación de caminos entre poses en $\boldsymbol{SE}(3)$.
- Opciones:
- Interpolar transformaciones homogéneas -> problemas de condiciones iniciales
- Combinar enfoque polinomial, o hibrido junto con interpolación de transformaciones homogéneas.

Mediante interpolación de transformaciones homogéneas + Polinomios de 5to orden.

```
T0=SE3([0.4 .2 0])*SE3.rpy(0,0,3);
T1=SE3([-0.4 -.2 0.3])*SE3.rpy(-
pi/4,pi/4,-pi/2);
TS=T0.interp(T1,lspb(0,1,50));
TS.animate()
```

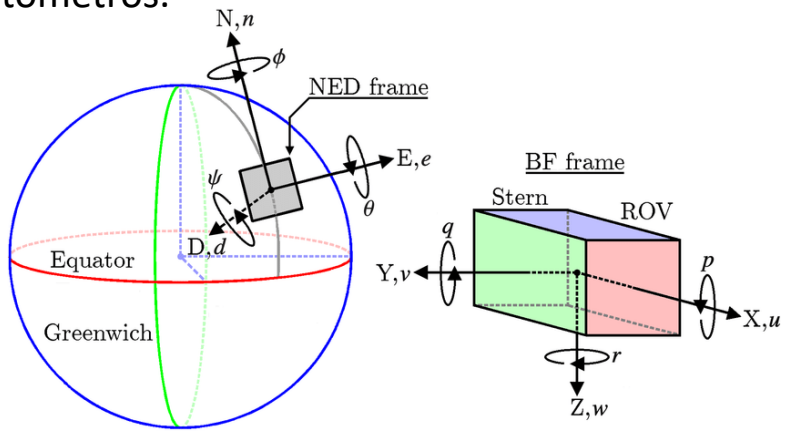
Código MATLAB – con equivalente similar en python!



Indice:

Navegación inercial.

- Consiste en estimar la posición respecto de un marco inercial, usualmente fijo en algún punto de la tierra (marco mundo). Convenciones usuales north-east-down (**NED**) y east-nort-up (**ENU**).
- No utiliza referencias externas (satélite, GPS, balizas, etc.).
- Compuestos por: acelerómetros, giroscopios y magnetómetros.



Giroscopio.

- Basan su funcionamiento en la conservación del momento angular.
- De la ecuación de rotación en 3D, si nos encontramos en un marco inercial obtenemos:

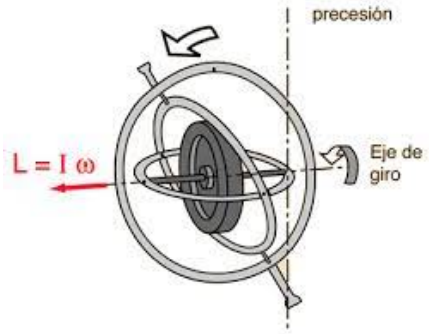
$$\tau = \omega \times \mathbf{h}$$

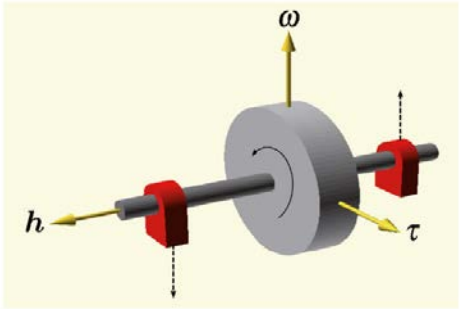
- Donde $\mathbf{h}$ es el momento angular del giroscopio, un vector paralelo al eje de rotación con magnitud

$$\|\mathbf{h}\| = \mathbf{J}\overline{\omega}$$

con $\mathbf{J}$ la inercia del rotor y $\overline{\omega}$ la velocidad rotacional (del disco rotante)

- Aquí ω representa la velocidad angular de todo el giroscopio (no del disco rotante)
- Pueden utilizarse para determinar el movimiento respecto de una dirección fija en el espacio, o fijo a un vehículo (configuración **strapdown**) para detectar variaciones de velocidad.
- Distintos tipos: Mecánicos, Laser (RLG), Fibra óptica (FOG), MEMS (micro-electro-mechanical systems).





Giroscopio. Estimando orientación

- Asumiendo que la velocidad en el marco del robot ${}^{B}\omega$ es contante durante un intervalo ∂_t el equivalente de rotación en un paso de tiempo es:

$$ {}^{B}\xi_{\Delta} \sim e_{\times}^{[{}^{B}\omega^{\#}]}\partial_t $$

- Si la orientación inicial del sensor es ξ_B la evolución estimada de la pose se puede escribir en forma discreta como

$$ \hat{\xi}_B\langle k+1\rangle = \hat{\xi}_B\langle k\rangle \oplus {}^{B}\xi_{\Delta}\langle k\rangle $$

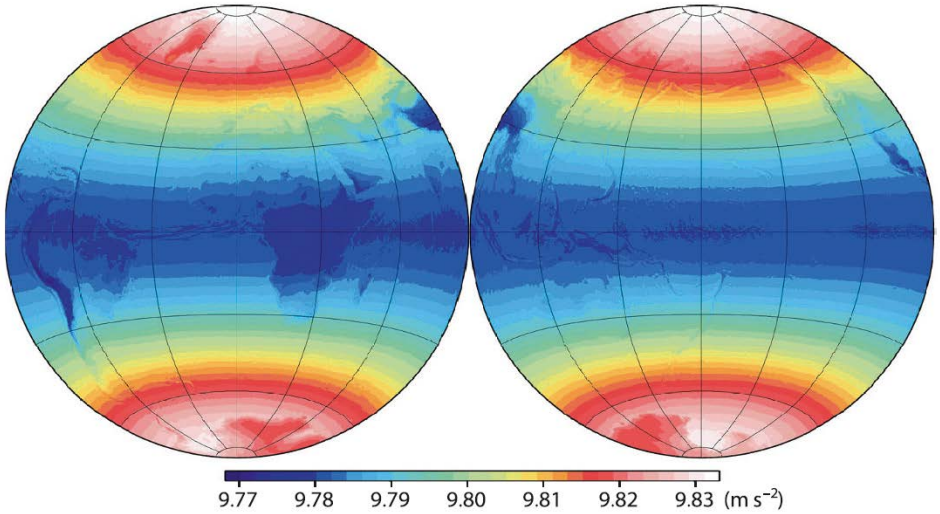
Código MATLAB – con equivalente similar en python!

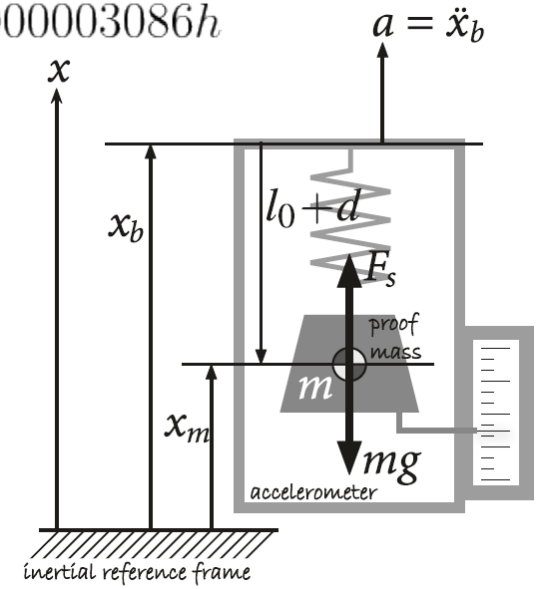
```
%% estimacion de orientacion
close all
clear all
ex_tumble
attitude(1)=UnitQuaternion();
for k=1:numcols(w)-1
    attitude(k+1)=attitude(k).*UnitQuaternion.omega(w(:,k)*dt);
end
attitude.animate('time',t)
figure();
mplot(t,attitude.torpy())
```

Acelerómetro.

- Miden aceleración y su funcionamiento más simple se basa en una masa conocida un resorte y la consideración de un marco de referencia inercial.
- Aun en reposo los acelerómetros miden la aceleración debida a la gravedad lo que permite detectar la dirección “hacia abajo”.
- Como la tierra no es una esfera la gravedad no es uniforme y puede aproximarse por la siguiente ecuación donde θ es el ángulo de latitud y h es la altura respecto del nivel del mar.

$$g \approx 9.780327(1 + 0.0053024 \sin^2(2\theta)) - 0.000003086h$$





Acelerómetro. Estimación de orientación

- Si tomamos la aceleración en el marco $\{0\}$ con el eje z vertical hacia arriba, podemos escribir el vector de aceleración gravitacional como: $${}^0\mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$
- En el marco del robot $\{B\}$ una orientación arbitraria en termino de los ángulos de roll-pich-yaw (ZYZ) puede escribirse como: $${}^0\xi_B = R_z(\theta_y) \oplus R_y(\theta_p) \oplus R_x(\theta_r)$$
- En el marco $\{B\}$ el vector aceleración gravitacional se puede escribir como: $${}^B\mathbf{a} = (\ominus {}^0\xi_B) \cdot {}^0\mathbf{a} = \begin{pmatrix} -g\sin(\theta_p)) \\ g\cos(\theta_p)\sin(\theta_R) \\ g\cos(\theta_p)\cos(\theta_r) \end{pmatrix}$$
- Considerando la medida de nuestro acelerómetro: $${}^B\mathbf{a}^{\#} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$
- Trabajando un poco con las ecuaciones podemos llegar a: $$\sin(\hat{\theta_p}) = \frac{-a_x}{g} \qquad \tan(\hat{\theta_r}) = \frac{a_y}{a_z}, \theta_p \neq \pm\frac{\pi}{2}$$

Aquí se hace el supuesto fuerte que la aceleración medida se debe solo la gravedad. En un robot al desplazarse se generarán aceleraciones adicionales que introducirán error en la estimación.

El símbolo # denota valor medido

El símbolo ^ denota estimación del valor real

Acelerómetro. Estimación de posición

- Considerando la aceleración total medida en el marco $\{0\}$, esta se debe a la gravedad y el movimiento:

$$^0\mathbf{a}^{\#} = {}^0\mathbf{g} - {}^0\mathbf{a}_v$$

- Suponiendo conocer una estimación de la orientación del robot y la aceleración de la gravedad podemos tener una estimación de la aceleración en el marco inercial:

$$^0\hat{\mathbf{a}}_v = {}^0\hat{\mathbf{R}}_B {}^B\mathbf{a}^{\#} - {}^0\mathbf{g}$$

Estimar la orientación con exactitud en la práctica no es simple.

- Integrado con respecto al tiempo podemos obtener la velocidad del robot:

$$^0\hat{v}_v(t) = \int {}^0\hat{\mathbf{a}}_v(t)\partial t$$

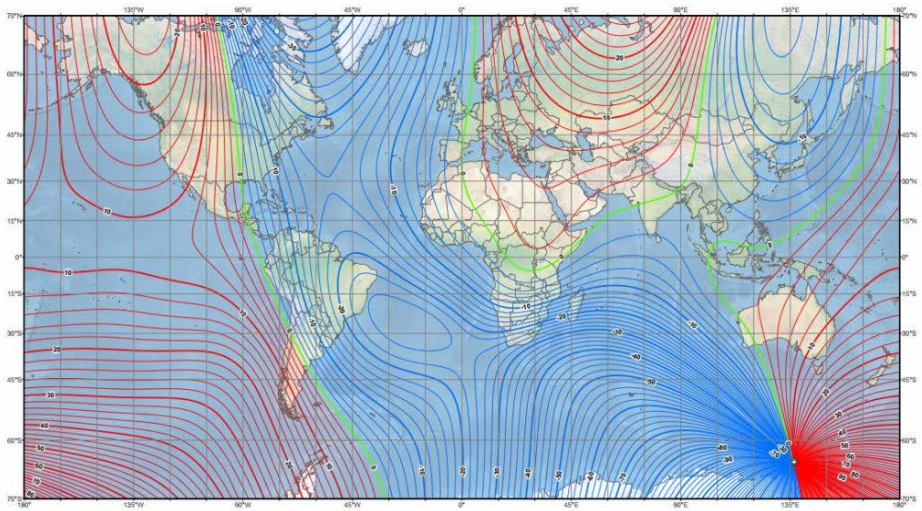
- Integrando nuevamente podemos obtener la posición del robot:

$$^0\hat{\mathbf{p}}_v(t) = \int {}^0\hat{v}_v(t)\partial t$$

- **Notar que podemos asumir que la aceleración del vehículo es cero y estimar la orientación, o asumir la orientación y estimar la aceleración del robot. No podemos estimar ambas ya que tendríamos mayor número de incógnitas que mediciones.**

Magnetómetro.

- Miden la intensidad y dirección del campo magnético terrestre. Se basan en el efecto Hall.
- El norte magnético no coincide con el norte geográfico, de hecho esta en continuo movimiento.
- Las líneas de campo magnético pueden considerarse como un vector m con una dirección y magnitud.
- La dirección se describe en termino de dos ángulos: declinación e inclinación.
  - Una proyección horizontal de m apunta al norte magnético, el **ángulo de declinación** D es la medida en sentido horario del ángulo entre esta proyección y el norte geográfico.
  - El **ángulo de inclinación** Ies el ángulo del campo magnético con la horizontal en el punto de consideración.
  - La longitud del vector, es la **intensidad de campo** y se mide en Tesla (aprox 25 – 65 μT)



Magnetómetro. Estimación de rumbo (heading)

- Considerando el marco inercial $\{0\}$ con el eje z apuntando verticalmente hacia arriba y el eje x apuntando al norte magnético. El vector de campo magnético se encuentra en el plano xz, con B la intensidad de campo e I la inclinación magnética.

$$ {}^{0}\mathbf{m} = B \begin{pmatrix} \cos(I) \\ 0 \\ \sin(I) \end{pmatrix} $$

- En el marco del robot $\{B\}$ una rotación arbitraria en términos de los ángulos roll-pitch-yaw puede escribirse como:

$$ {}^{0}\xi_B = R_z(\theta_y) \oplus R_y(\theta_p) \oplus R_x(\theta_r) $$

- El campo magnético en el marco $\{B\}$ puede expresarse como:

$$ {}^{B}\mathbf{m} = \left(\ominus {}^{0}\xi_B\right) \cdot \mathbf{m} $$

- La medida del campo magnético producida por el sensor en el marco $\{B\}$ es: $ {}^{B}\mathbf{m}^{\#} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} $
- Luego trabajando un poco con las ecuaciones podemos obtener:

$$ \hat{\theta}_y = \tan^{-1} \frac{\cos(\theta_p)(m_z \sin(\theta_r) - m_y \cos(\theta_r))}{m_x + B \sin(I) \sin(\theta_p)} $$

Aquí se suponen θ_r ángulo roll y θ_p ángulo pitch conocidos (por ejemplo estimados mediante una medida de aceleración).

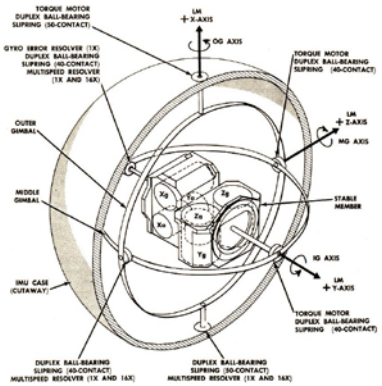
- Luego si deseamos obtener el rumbo con respecto al norte geográfico (true-north) debemos restar a la estimación la declinación local D:

$$ {}^{tn}\hat{\theta}_y = \hat{\theta}_y - D $$

Fusión de sensores.

- Un sistema de **navegación inercial** utiliza los sensores analizados para determinar la pose de un vehículo (posición y orientación).
- Diseños originales basados en cardans (gymbals), nuevos diseños tipo strapdown llamados **IMU** (Inertial Measured Unit) (6-DOF Y 9DOF)
- Los sensores, especialmente los de bajo costo, no son perfectos. Su medida general puede expresarse como:

$$x^{\#} = sx + b + \varepsilon$$



- Donde x es el valor real, s un **factor de escala** usualmente brindado por el fabricante con alguna tolerancia, b el offset o **bias** y ε **ruido aleatorio**.
- El mayor problema es el bias, que es variante en el tiempo y con la temperatura.
  - Solución 1: **estimar el bias medio** de todos los sensores dejando la IMU estacionaria durante unos segundos, en realidad esto solo será validado durante un periodo corto de tiempo ya que el bias no es constante.
  - Solución 2: **estimar el bias online**, pero para esto necesitamos combinar la información de los diversos sensores, a esto se lo llama fusión de sensores.

Fusión de sensores. Filtro complementario.

- La forma más común es utilizar un **filtro de Kalman extendido**, que da una estimación optima.
- Una forma alternativa, más simple pero también efectiva es utilizar un **filtro complementario**.
- La actualización de la rotación es similar a lo ya utilizado pero con algunos agregados:

$$^{B}\xi_{\Delta}\left\langle k\right\rangle = e^{[^{B}\omega^{\#}\left\langle k\right\rangle - \hat{\mathbf{b}}\left\langle k\right\rangle + k_{p}\sigma_{r}\left\langle k\right\rangle]_{\times}\partial t}$$

- Las claves aquí son la estimación del bias $\hat{\boldsymbol{b}}$ que es restada de la medida del sensor y el termino de error en orientación σ_r
- La estimación del bias cambia en el tiempo como: $\hat{\mathbf{b}}\left\langle k+1\right\rangle \leftarrow \hat{\mathbf{b}}\left\langle k\right\rangle - k_{1}\sigma_{r}\left\langle k\right\rangle$

  (con $\sigma_r \cdot k_p > 0$ y $k_i > 0$ constantes elegidas adecuadamente)
- El error de orientación se deriva de N medidas de sensor $^{0}v_{\mathbf{i}}{}^{\#}$, donde $^{0}v_{\mathbf{i}}$ es el valor conocido de un vector en el marco inercial (por ejemplo la aceleración gravitacional)

  Ver que solo peso la medida de algo conocido!

$$\sigma_{r}\left\langle k+1\right\rangle = \sum_{i=1}^{N} k_{i}^{0}v_{\mathbf{i}} \times {}^{0}v_{\mathbf{i}}{}^{\#}\left\langle k\right\rangle$$

- Con $^{0}v_{\mathbf{i}}{}^{\#}$ el valor medido en el marco del robot y rotado al marco inercial por la estimación de la orientación:

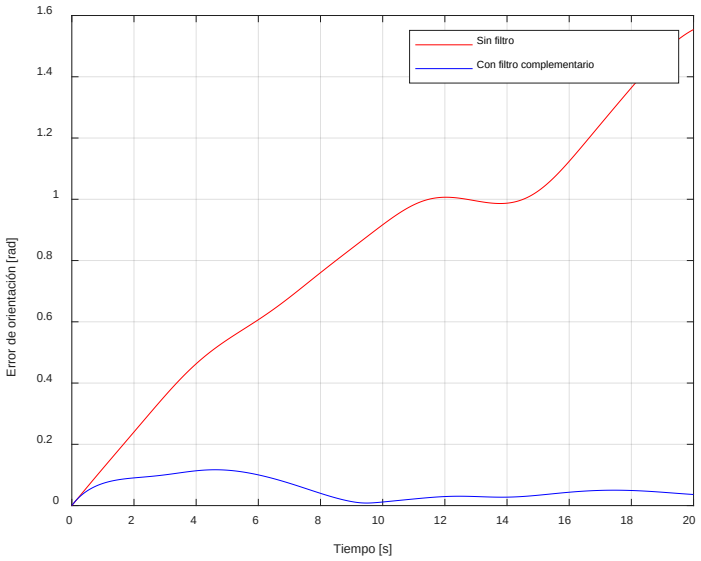
$$^{0}v_{\mathbf{i}}{}^{\#}\left\langle k\right\rangle = {}^{0}\hat{\xi}\left\langle k\right\rangle \cdot {}^{B}v_{\mathbf{i}}{}^{\#}\left\langle k\right\rangle$$

- Este filtro permite agregar como fuente de información tantas variables vectoriales $^{0}v_{\mathbf{i}}$ como se desee, por ejemplo medidas de campo magnético, altitud y azimut de marcas visuales, estrellas o planetas.

Fusión de sensores. Filtro complementario.

```
%% ejemplo filtro complementario
close all;
clear all;
ex_tumble;
attitude(1)=UnitQuaternion();
for k=1:numcols(wm)-1
attitude(k+1)=attitude(k).*UnitQuaternion.omega(wm(:,k)*dt);
end
plot(t,angle(attitude, truth),'r');
hold on;
ki=0.2;kp=1;b=zeros(3,numcols(w));
attitude_ecf(1)=UnitQuaternion();
b=[0 0 0]';
for k=1:numcols(wm)-1
       invq=inv(attitude_ecf(k));
       sigmaR=cross(gm(:,k), invq*g0)+cross(mm(:,k), invq*m0);
       wp=wm(:,k)-b(:,k)+kp*sigmaR;

attitude_ecf(k+1)=attitude_ecf(k).*UnitQuaternion.omega(wp*dt);
       b(:,k+1)=b(:,k)-ki*sigmaR*dt;
end
plot(t,angle(attitude_ecf, truth),'b');
grid on;
xlabel('Tiempo [s]');
ylabel('Error de orientación [rad]')
legend('Sin filtro','Con filtro complementario')
```



Código MATLAB – con equivalente similar en python!

Lo visto…

- Derivada de pose –translación y orientación.
- Camino y trayectorias.
- Inicios de navegación inercial.

Próxima clase:

- ROS

Bibliografía:

- Robotics, Vision and Control Fundamental Algorithms in MATLAB. Peter Corke. 2da edición Springers.