DINÁMICA DE VEHÍCULOS ESPACIALES

Práctica 1: Representación de la Actitud. Visualización de la actitud mediante MATLAB/STK. Determinación (estática) de la Actitud.

En esta práctica se emplearán los conceptos de representación y determinación estática de actitud aprendidos en la asignatura y se utilizarán Matlab y STK para visualizar y demostrar dichos conceptos.

Arrancar STK, Matlab, y bajar de enseñanza virtual y descomprimir el archivo practica1.zip en un directorio elegido para trabajar.

1. Representación de la actitud con STK

A continuación vamos a aprender las opciones básicas de representación de actitud de STK.
En primer lugar creamos un escenario nuevo con las opciones por defecto y en él creamos un satélite (insert default), al cual llamamos "Sat", modificamos su órbita de forma que tenga de semieje mayor 8800 km y excentricidad 0.2, y el resto de sus elementos orbitales (que no son muy relevantes a efectos de esta demostración) a elección del alumno. Pinchamos en el icono "View From/To" de la ventana 3D, y pinchamos nuestro satélite en "View from" y en "View to". Cambiar la vista para ver desde el sistema de referencia inercial, pinchando en "Reference frame" la opción "Earth Inertial Axes". Pulsamos OK. Hacer zoom hacia el satélite y encuadrarlo en la pantalla. Animando la simulación, observemos que por defecto tiene una actitud que permanece prácticamente constante relativa a la órbita (girando con ella) y sus paneles solares siempre apuntan al sol. Esto se debe al perfil de actitud por defecto, que estudiamos a continuación. Cambiar el modelo del satélite (3D graphics->Model) al de Hubble para ver con más precisión lo que sigue.

Para estudiar adecuadamente la actitud, primero añadimos una representación de los ejes cuerpo (Body Axes), ejes de la órbita (VVLH, Vehicle Velocity Local Horizontal), ejes inerciales (J2000) y del vector del Sol: para ello hacemos doble click en el satélite para obtener sus propiedades y pinchamos en "3D->Vector". Pinchamos "Show" en Body Axes , en VVLH y en "Sun Vector". Pinchamos OK y simulamos.

Animar la simulación. Observar en primer lugar el comportamiento de los ejes VVLH: el eje Z apunta a la tierrra, el eje X está contenido en el plano de la órbita (coincidirá con ella en una órbita circular, pero no en una órbita elíptica) y el eje Y es normal a la órbita. Observemos que los ejes cuerpos permanecen con actitud casi constante respecto a estos ejes (sería totalmente constante si no hubiera excentricidad). Observamos también que los paneles solares siguen constantemente al vector del Sol. Los ejes inerciales se mueven de forma aparente debido a que estamos manteniendo una visión fija con respecto a la Tierra. En realidad, son ellos los que se mantienen fijos en el espacio.

Para cambiar el comportamiento por defecto, pinchamos "Attitude" en las propiedades del satélite. En la versión básica sólo se puede fijar la actitud de dos formas:

- Basic: Perfil predefinido por STK (no obedece necesariamente las ecuaciones de la dinámica de

actitud). El tipo (type) que viene por defecto, Nadir alignment with ECF velocity constraint, alinea el satélite con la órbita, manteniendo el eje Z cuerpo apuntando hacia el nadir (eje Z de VVLH). De ahí viene "... nadir constraint". La otra dirección la determina la velocidad, que en una órbita excéntrica no es constante respecto a la órbita, de ahí la lenta variación del sistema de referencia. Hay otros tipos pero no son demasiado interesantes.

- Precomputado: Proporcionamos la información de actitud en el tiempo a través de un archivo de actitud con extensión .a. Este archivo contiene un vector de tiempos y una matriz de cuaterniones que da la actitud para cada instante de tiempo del vector de tiempos. Esta será la opción que usaremos en la práctica.

El hecho de que los paneles solares apunten al Sol viene dado por la propiedad "Model pointing" en 3D Graphics. Otros modelos permiten apuntar el cuerpo del satélite, antenas, etc.. a diferentes blancos.

2. Representación de una actitud generada por Matlab

Ahora pasamos a ver como podemos representar actitudes generadas por Matlab. Para simplificar la visualización, eliminar el vector Sol y el sistema de referencia VVLH. Para poder vigilar los cuaterniones y otros parámetros en tiempo real, en 3D graphics/Data Display se puede seleccionar "Attitude Quaternions" (también Euler Angles).

IMPORTANTE: Se sigue la convención de STK de forma que la parte escalar del cuaternión es la cuarta componente y la parte vectorial las primeras tres componentes.

Las siguientes funciones de Matlab están disponibles para cálculos de actitud:

- *C=euler3132DCM(euler313)* obtiene la DCM a partir de un vector de ángulos de Euler (en radianes) en la secuencia 3-1-3.
- *euler313=DCM2euler313(C)* obtiene un vector con los ángulos de Euler en la secuencia 3-1-3 a partir de la DCM. Se ha fijado que el segundo ángulo se encuentre entre -90 y +90.
- *q=DCM2quat(C)* a partir de la DCM genera el cuaternión correspondiente.
- *C=quat2DCM(q)* a partir del cuaternión genera la DCM correspondiente.
- *qp=qprod(q1,q2)* da el producto de dos cuaterniones q1 y q2.
- *qc=qconj(q)* da el conjugado de un cuaternión.
- *[eje,angulo]=q2euler(q)* permite obtener el eje y ángulo de Euler a partir del cuaternión.
- *q=euler2q(eje,angulo)* permite obtener el cuaternión a partir del eje y ángulo de Euler.

Además se dispone de la siguiente función:

- generar_archivo_actitud (nombre,T,Q), genera un archivo de actitud (.a) para usar con STK, recibe el nombre del archivo (string) sin extensión, un vector columna T con todos los tiempos en los que se tiene un valor de actitud (supuesto ordenado, debe empezar por 0), y una matriz Q cada una de cuyas filas es el cuaternión de actitud correspondiente al tiempo T en dicha fila.

Ejemplo 1. En primer lugar, vamos a generar un perfil de actitud constante en el tiempo, respecto al sistema inercial J2000.
Para ello, seguir los siguientes pasos:

1. Generar una actitud inicial. Por ejemplo eligiendo unos ángulos de Euler en el conjunto 3-1-3, digamos 30º,45º,0º. Calculamos con los scripts la matriz correspondiente a este ángulo, la pasamos a cuaterniones y así obtenemos q0.

2. Generamos un vector de tiempos entre 0 y 3600, con paso 1 segundo, es decir, *t=[0:3600]*.
3. Generamos la matriz que va a contener los valores de los cuaterniones. Para ello, simplemente *Q=zeros(3601,4)*.
4. Rellenamos la matriz con el valor del cuaternión: *for i=1:3601, Q(i,:)=q0'; end*
5. Generar el archivo .a: generar_archivo_actitud *('ejemplo1',T,Q)*.
6. Ver el archivo con un editor de texto y entender todas sus lineas.
7. En Sat1->propiedas->Basic,attitude->marcar precomputed, elegir el archivo en file y apply.

Pulsar reset y play en STK. Visualizar el resultado. Ojo: Verificar que la animación se visualiza en el periodo de tiempo correcto (desde el inicio de la simulación hasta una hora).

Ahora se van a repetir otros ejemplos. Para los ejemplos, se repiten los mismos pasos variando el procedimiento de generación de Q.

Ejemplo 2. Creamos ahora un perfil de actitud de maniobra.
La maniobra partirá de unas actitudes inicial y final dadas, en un tiempo de maniobra dado.

Para ello, usamos la siguiente propiedad: si partimos de un cuaternión inicial q_0 a otro final, q_F , existe un cuaternión que representa la rotación de velocidad angular mínima q_{rot} que cumple:

$$q_F = q_0 * q_{rot}$$

Por tanto:

$$q_{rot} = \frac{1}{q_0} * q_F = \frac{q_0^*}{\|q_0\|^2} * q_F = q_0^* * q_F$$

Estas multiplicaciones de cuaterniones pueden hacerse fácilmente en Matlab con los scripts proporcionados.

Una vez obtenemos el cuaternión de rotación, extraemos sus eje y ángulo de Euler, y construimos el cuaternión de rotación instantánea:

$$q_{rot}(t) = \begin{bmatrix} \vec{e}\sin\left(\frac{\theta t}{2T}\right) \\ \cos\left(\frac{\theta t}{2T}\right) \end{bmatrix}$$

siendo θ el ángulo de Euler y T el tiempo de maniobra dado.

De esta manera, la actitud en función del tiempo es:

$$q(t) = q_0 * q_{rot}(t)$$

En este ejemplo se pide: partiendo de la misma actitud inicial que antes, generar una velocidad angular que permita llegar a la actitud (dada en ángulos de Euler en el conjunto 3-1-3) 60º,15º,180º a lo largo de una hora. Posteriormente "quedarse" en dicha actitud durante media hora.

Ejemplo 3: Generar un giro de forma que un vector en ejes cuerpo coincida con un vector en ejes inerciales.

Supongamos ahora que partimos de la misma actitud inicial y queremos que el eje z coincida con el vector Sol (aproximadamente constante en ejes inerciales). En este caso también usamos los

conceptos de ángulo y eje de Euler, pero calculados usando el vector z (es decir [0 0 1], que llamamos v0) y el vector Sol en ejes cuerpo (v1). Para calcular el vector Sol en ejes cuerpo, tomamos su valor en ejes J2000 (usar Report&Graph manager), y con C0 lo pasamos a ejes cuerpo y lo normalizamos. Ahora, el eje de Euler será el producto vectorial normalizado de v0 por v1 (en dicho orden), y el ángulo de Euler será el ángulo entre ambos vectores (arcocoseno del producto escalar).

3. Métodos TRIAD y q

Ahora revisaremos el funcionamiento de los métodos TRIAD y q. Recordemos que estos métodos permiten encontrar una estimación de la actitud de un vehículo con respecto a un cierto sistema de referencia base (típicamente el inercial) si se conocen al menos dos direcciones (vectores) en el sistema de referencia base, y sus medidas en ejes cuerpo obtenidas mediante sensores del vehículo. El método TRIAD funciona con dos referencias, mientras que el q acepta cualquier número de referencias igual o mayor que dos, a las que hay que asignarles pesos.

Vamos a generar en primer lugar dos referencias para comprender y comparar los métodos TRIAD y . qLas referencias podrían ser inventadas, pero vamos a aprender a extraerlas de STK. Para ello, siguiendo los pasos dados en la práctica 1, arrancar STK, generar un escenario por defecto, insertar un satélite, y dotar al satélite de un perfil constante en el tiempo con ángulos de Euler, en el conjunto 3-1-3, de magnitud (30º, 45º, 0º), que se pasará a un cuaternión inicial q0, tal como se hizo en la primera práctica.

Una vez hecho eso, pinchando con el botón derecho sobre el satélite y yendo al "report&graph manager", generar cuatro nuevos estilos de informe, que serán:

- Vector Sol (J2000)
- Vector Sol (Ejes cuerpo)
- Campo Magnético (J2000)
- Campo Magnético (Ejes cuerpo)

Los estilos se crean usando el tercer icono (``Create new report style''), nombrando el nuevo informe creado, haciendo doble click sobre él, y seleccionando en Vectors (Body o J2000, según sea el sistema de referencia deseado), bien Sun, bien MagField (IGRF), los campos ``Time'', ``x/Magnitude'', ``y/Magnitude'', y ``z/Magnitude'' (de esa forma los vectores son unitarios). Los resultados estarán limitados por la precisión numérica. Observemos que al generar los estilos se podría aumentar el número de decimales, pero en cualquier caso tampoco podríamos esperar que los sensores reales obtuvieran resultados totalmente exactos.

Una vez creados los estilos, generar los informes con ``Generate''. Mediante Copiar y Pegar, extraer los cuatro vectores, llamando v1 y v2 a los vectores en ejes J2000, respectivamente el vector Sol y el campo magnético, e igualmente w1 y w2 los mismos vectores en ejes cuerpo. Todos deberán ser extraídos en el mismo instante de tiempo por consistencia, y deberán ser vectores columna.

Probemos en primer lugar el método TRIAD. La forma de utilizarlo es la siguiente: *C=TRIAD(v1,v2,w1,w2)*. Nos devuelve la actitud en forma de DCM, y podemos obtener el cuaternión con *q_triad=DCM2quat(C)*. Comparándolo con el cuaternión inicial q0 parecerá en principio idéntico, pero la forma de compararlo es obtener el cuaternión de error entre ambos multiplicando uno por el conjugado del otro: *q_error_triad=qprod(q0,qconj(q_triad))*.

Para obtener un único número que cuantifique el error, se puede extraer el ángulo de Euler, en grados, de dicho cuaternión de error *error_triad=180/pi*2*acos(q_error_triad(4))*. Multiplicando dicho número por 3600 obtenemos el error en segundos de arco. Observamos que el error no es cero a pesar de que los datos son supuestamente exactos, se debe a la precisión numérica elegida (6 decimales de las medidas), en cualquier caso es muy pequeño.

Para probar el método q, unimos las referencias en una matriz: *v=[v1 v2]* e igualmente las medidas: *w=[w1 w2]*, y asignamos unos pesos: *p=[1/2 1/2]'*, que inicialmente serán los mismos. Estudiamos los resultados de la misma forma que el método TRIAD y comparemos los resultados, cuyo error también será muy pequeño.

Una vez se ha comprendido como usar los métodos, se van a ``corromper'' con error las medidas, de forma que se perderá la exactitud. Esto nos permitirá cotejar los métodos en presencia de error (que siempre existirá en la práctica). Vamos a añadir al sensor solar un error de 0.5 grados, mientras que al magnetómetro le vamos a dar un error de 2 grados. Ignoraremos el hecho de que el modelo de campo magnético tampoco se conoce con total precisión, por lo que también habría errores en la referencia, en este caso.

Generamos cuaterniones en base a un eje de Euler cualquiera (se tomará la dirección x para el sensor solar y la dirección y para el magnetómetro) y con el ángulo de Euler elegido:

```
q_error1=[sin(0.5*pi/360)*[1 0 0]' ; cos(0.5*pi/360)];
q_error2=[sin(2*pi/360)*[0 1 0]' ; cos(2*pi/360)];
```

Podríamos generarlo aleatoriamente pero lo hacemos todos igual para obtener todos el mismo resultado. Ahora giramos los vectores de medida usando estos cuaterniones:

```
w_error1=getfield(qprod(qprod(qconj(q_error1),[w1;0]),q_error1),{1:3});
w_error2=getfield(qprod(qprod(qconj(q_error2),[w2;0]),q_error2),{1:3});
```

Probemos ahora el método TRIAD (comprobar el efecto del orden de los vectores) y el método q con distintos pesos.

Una de las ventajas del método q es que permite añadir más medidas. Por ejemplo, podemos añadir un sensor de horizonte terrestre. Estos sensores esencialmente miden la dirección del vector nadir. Para ello, añadimos este vector en STK (tanto en ejes J2000 como en ejes cuerpo). Elegir centric. Llamar v3 a la referencia y w3 a la medida y corromperlo con un error de 1 grado girado en la dirección z. Probar otra vez el método q y comparar los resultados con los anteriores.