RESOLUCION ECUACIONES POSICIONAMIENTO en GPS

Ejercicio 1a: Escribir una función get_data_sats() con el siguiente template:

```
function [XYZ_sat, cdT]=get_data_sats(sp,t,prn,N)
if nargin==3, N=8; end  % Si no se especifica, usamos N = 8 nodos
 nsat = length(prn); XYZ_sat=zeros(3,nsat); cdT=zeros(nsat,1);
 if length(t)==1, t = t*ones(1,nsat); end  % Si unico t, replica
   ....
return
```

Sus argumentos son los mismos que los de interp_sat() pero ahora prn es un vector 1 x nsat con una lista de satélites para calcular sus datos. El tiempo t puede ser un valor único o un vector con diferentes tiempos. Si es un único dato se crea un vector con valores idénticos, tantos como satélites. En vuestro código asumiréis que tanto t como prn son vectores del mismo tamaño (nsat).

Se trata de hacer un bucle llamando a interp_sat() con el correspondiente tiempo y satélite. Las posiciones obtenidas se guardan en las columnas de una matriz XYZ_sat (3 x nsat). En el **vector columna** cdT (nsat x 1) se devuelven los errores de reloj para todos los satélites.

Probad la función con los datos del fichero **'IGS13230.sp3'** dado, calculando las posiciones de los satélites con **PRNs = [1 2 5 6 14]** para **t=10000** seg. usando **N=8** nodos en la interpolación.

Las posiciones XYZ de los 5 satélites deben salir (en mt):

```
 15673977.06   -10275418.40    12013639.67    13388074.25    19169461.57
-21411706.08    13567917.10    23678940.35     8523947.99   -12429984.86
  -551126.02    20062283.21    -2476393.99    21493936.38   -13476675.41
```

y los errores de reloj (mt): **120912.49 -7719.52 29645.68 152168.03 -8953.75**

Usad `fprintf('%12.2f ',XYZ_sat(1,:)); fprintf('\n');` para volcar las sucesivas líneas de XYZ_sat y comparadlas con los resultados dados. Haced lo mismo con `fprintf('%12.2f ',cdT);` para volcar los errores de reloj (cdT).

Volcad las posiciones y errores de reloj para esos mismos satélites (y mismo tiempo, t=10000) pero usando N=4 nodos. ¿De qué orden son las diferencias en las posiciones entre ambas interpolaciones? ¿Y en los errores de reloj?

Adjuntar vuestra función completada.

Ejercicio 1b: Lo siguiente que necesitamos es una función con el siguiente template:

```
function [H,R]=get_HR(XYZ,pos)
```

La función recibe una matriz XYZ con las posiciones de los satélites (3 x Nsat) y la posición de un observador (pos, vector 3x1) y construye la matriz H del sistema (Nsat x 4). Como segundo argumento la función devuelve un **vector columna** R (Nsat x 1) con las distancias entre la posición del observador y todos los satélites. Probad la función con los siguientes argumentos:

- Para la posición de los satélites XYZ, usad la matriz obtenida en el apartado anterior (satélites 1 2 5 6 y 14, tiempo t=10000, N=8 nodos).

- Para la posición del observador, usad la posición aproximada de Madrid:

```
pos = [4855000;  -325000;  4115000];
```

Los resultados deben salir:

```
     -0.44789   0.87297   0.19317   1.00          24155168.594
      0.58183  -0.53424  -0.61324   1.00          26004971.607
H =  -0.27638  -0.92674   0.25448   1.00      R =  25901385.081
     -0.40085  -0.41569  -0.81640   1.00          21287194.888
     -0.55681   0.47086   0.68429   1.00          25708004.828
```

Adjuntar código de vuestra función get_HR.

Calcular la matriz H y vector R para los mismos sats, en el tiempo t = 12000 usando 8 nodos. Volcar vuestros resultados en un formato similar al anterior usando: `fprintf('%9.5f %9.5f %9.5f %6.2f %12.3f\n',[H R]');`

Ejercicio 2a: Completad la función get_pos() suministrada:

```
function X=get_pos(sp,Tr,sats,obs,X,N)
if nargin<6, N=8; end  % Si no se especifica, usar nodos N=8
if nargin<5, X=[0; 0; 0; 0]; end % Por defecto X inicial =[0; 0; 0; 0];
```

Argumentos de ENTRADA:

- Estructura sp3 con los datos de los satélites durante el día (sp).
- Los datos suministrados por el GPS:
  - El tiempo n que se tomo la medida según receptor (Tr),
  - Los PRNs de los satélites observados (sats)
  - Medidas tomadas a dichos satélites (vector **COLUMNA** obs).
- X (opcional): vector 4x1 con la hipótesis inicial (posición inicial XYZ0 + error inicial cdt0) en metros. Si no se especifica se usa X0=[0;0;0;0].
- N (opcional): número de nodos a usar en la interpolación. Por defecto usamos N=8.

Argumentos de SALIDA:

- Vector X (tamaño 4x1) con los resultados actualizados, agrupando la posición pos (3x1) y el error de reloj cdt del receptor.

La rutina debe implementar la iteración descrita en las transparencias (por ahora en su versión más básica sin usar tiempos de vuelo variables, la rotación de la Tierra ni las correcciones relativistas).

Se trata de entrar en una iteración donde:

1. Extraemos del vector X la posición aproximada del receptor, pos=X(1:3), y su error de reloj, cdt = X(4).

2. Corregimos Tr por el error de reloj del receptor (recordad pasarlo antes a segundos) para obtener Tr_gps (tiempo de recepción GPS).

3. Estimar el tiempo de transmisión Tx_gps. En esta primera aproximación usamos el mismo tiempo de vuelo para todos los satélites (un valor medio ~70 msec), por lo que: Tx_gps = Tr_gps - 0.070

4. Usar la rutina get_data_sats() con el tiempo de transmisión Tx_gps para obtener los datos de los satélites (matriz de posiciones + vector cdT con los errores de reloj).

5. A partir de la matriz de posiciones de los satélites y la estimación actual de la posición del receptor (pos) usad get_HR() para obtener la matriz H y las distancias R a los satélites.

6. Calcular los pseudo-rangos predichos por nuestro modelo:

$$pred = \sqrt{(X_S - x)^2 + (Y_S - y)^2 + (Z_S - z)^2} + cdt - cdT = R + cdt - cdT$$

   Comprobad que pred es vector columna (R y cdT deben serlo también).

7. Restar los valores predichos (pred) de las medidas reales del GPS (obs) para obtener el vector de discrepancias observado-esperado ($\Delta\rho$). El resultado $\Delta\rho$ deber ser igualmente **un vector columna (Nsat x 1).**

8. Resolver el sistema $H \cdot \delta x = \Delta\rho$ y determinar la corrección δx a aplicar (δx debe ser un vector 4x1). Actualizar nuestra solución X = X + δx

9. Calcular la norma del desplazamiento, norm(δx), y si es suficientemente pequeña (por ejemplo menor de 1cm = 0.01 m) terminar la iteración. En caso contrario repetir los pasos 1 a 8 con los datos de la nueva solución X actualizada.

Probad vuestra rutina con los siguientes datos:

- **Estructura sp --> leida del fichero IGS13230.SP3.**
- **Tiempo de recepción: Tr = 7200.00;**
- **Satélites observados = [1 2 5 6];**
- **obs=[23096478.313; 24501265.177; 24401880.133; 20555740.449];**

No hace falta pasarle valores para N (número de nodos) y X (hipótesis inicial). El programa usará los valores por defecto, N=8 y X0=[0;0;0;0].

Con estos datos debéis obtener los resultados del fichero **traza0_4sat.txt.** En esta traza están los resultados de todas las iteraciones paso a paso. De esta forma, si observáis discrepancias con vuestra solución podéis detectar en qué paso os estáis equivocando. Adjuntar código de get_pos(), una vez verificada.

Los datos siguientes corresponden a medidas tomadas por el GPS a otros 4 satélites diferentes en el mismo instante de tiempo:

```
prn = [14 16 21 25];
obs = [ 24258653.281; 24581854.516;  23456029.925; 21447276.383];
```

Calcular la posición del GPS en base a estas nuevas observaciones Volcar las posiciones obtenidas en cada iteración con 2 decimales. ¿Es el resultado final el mismo que antes?

Ejercicio 2b: En realidad, en ese instante de tiempo (Tr=7200) nuestro GPS estaba observando un total de 9 satélites:

PRN	OBS
1	23096478.313
2	24501265.177
5	24401880.133
6	20555740.449
14	24258653.281
16	24581854.516
21	23456029.925
25	21447276.383
30	21280828.168

Volver a correr vuestra función get_pos usando todas las medidas. Si habéis hecho las cosas bien (no suponiendo que siempre tendríamos 4 sats) debería funcionar sin cambios. El correspondiente volcado está en **traza0_9sat.txt**.

¿Qué distancia (en mt) hay entre esta solución y las del ejercicio anterior?
Dad los valores del vector de discrepancias $\Delta\rho$ final. ¿Se hace dicho vector cero como sucedía antes? ¿Y la corrección δX? ¿Os fiaríais más de las soluciones del ejercicio anterior (usando 4 sats cada una) o de las de éste (usando 9 sats)?

Volver a resolver pero ahora sumando 5000 mt. a todas las observaciones. ¿Cambia la posición obtenida? ¿Y el error de reloj? ¿Por cuántos metros?

Ejercicio 3: (elevaciones de los satélites sobre el horizonte). La función suministrada elaz.m recibe unos argumentos similares a los de get_HR:

1) Una matriz 3xN con las posiciones de N satélites en un instante dado.
2) Una posición del observador (vector 3x1)

La función calcula los ángulos (en grados) de elevación (sobre el horizonte) y azimuth (respecto al Norte) de los N satélites vistos desde la posición del observador y los devuelve en dos arrays 1xN. Usar help elaz para ver su uso.

Con los datos de 'IGS13230.sp3', calcular la elevación de los satélites desde el punto de vista de un observador en Madrid **([4855000;-325000;4115000])** a las 14:30h. Volcar las elevaciones de los satélites.

El campo sp.nsat indica que ese día había 29 satélites activos, ¿cuántos de esos 29 satélites eran visibles desde Madrid a esa hora?
Si definimos que para ser usable un satélite debe estar a más de 10º sobre el horizonte, ¿cuál sería la lista de satélites operativos en ese caso?
Hacer una gráfica con el número de satélites visibles (con el criterio de 10º) desde las 02:00 hasta las 22:00 de ese día a intervalos de 5 minutos.

Ejercicio 4: La función get_pos() del ejercicio 2) es la versión mínima pedida, pero sus resultados no son todo lo precisos que podrían ser porque nos falta incorporar algunas correcciones adicionales en nuestro modelo. En particular podríamos considerar:

1) Tiempos de vuelo diferentes para cada satélite. Al principio usaremos como antes un valor inicial medio $\tau = 70$ msec, y al calcular el tiempo de transmisión haremos Tx_gps = Tr_gps - τ. La diferencia es que ahora, tras completar cada iteración, recalcularemos los tiempos de vuelo dividiendo la distancia R a los satélites por la velocidad de la luz c. Notad que ahora tendremos un τ diferente para cada satélite.

2) Corrección por la rotación de la Tierra durante el tiempo de vuelo.

3) Corrección relativista que modifica ligeramente el error de reloj (cdT) de cada satélite debido a su velocidad. Si queréis usar esta corrección vuestra función interp_sat() necesita calcular la velocidad de los satélites (ejercicio 5 de la primera hoja).

Si se incorporan estas correcciones hace falta hacerlo en el orden indicado y de forma acumulativa. Esto es, primero modificaríamos get_pos para incorporar la influencia del tiempo de vuelo (llamando p.e. get_pos1 a la nueva función). Sobre get_pos1 añadiríamos lo necesario para tener en cuenta la rotación de la Tierra y obtendríamos get_pos2, etc. La idea es que no tiene sentido implementar la 3ª corrección (del orden de 1-2 metros) si no se han aplicado previamente las anteriores (que pueden alcanzar decenas de metros).

Para depurar, podéis usar los siguientes ficheros con volcados de resultados:

- **traza1.txt** -> incorpora tiempos de vuelo τ's distintos.
- **traza2.txt** -> τ's + rotación Tierra.
- **traza3.txt** -> τ's + rotación + corrección relativista de dT.