

Satélites Artificiales y Geomática

PRÁCTICA 1

Movimiento del Polo Actualizado

ADRIÁN FERNÁNDEZ TEJADA

ADRIÁN SÁNCHEZ LOUREIRO

2023

Tratamiento de Datos Imprecisos Actualizados.

Trataremos el archivo BULLET_A.ERP extraído del International Earth Rotation and Reference Systems Service.

Tablas de datos

En este proyecto, emplearemos un tipo de variable que almacena los datos como una tabla.

Definamos la tabla tablaI que almacenará los datos:

```
fecha, X-polo, Y-polo, UT1-UTC, RMS-xp, RMS-yp, RMS-dt.
```

Tomando los datos del fichero BULLET.ERP, que almacena datos desde 2000-01-02 a 2018-09-22.

Primero, definimos el tipo de dato que almacenará cada columna y su nombre.

```
varTipos = [ "double", "double", "double", "double", "double", "double", "double" ];  
varNames = [ "Fecha", "X", "Y", "UT", "RMS-X", "RMS-Y", "RMS-T" ];
```

Segundo, una variable auxiliar en la que almacenaremos el número de columnas.

```
sz = [1 length(varNames)];
```

Finalmente, definimos la tabla con lo citado anteriormente.

```
tablaI = table('Size',sz,'VariableTypes',varTipos,'VariableNames',varNames);
```

A continuación, usamos la variable auxiliar fila para almacenar el nº de filas que tiene la tabla.

```
fila = 0;
```

Código Realizado

Definimos en primer lugar la ruta donde se encuentra nuestro archivo de datos en la variable *directorio_raiz*. También una variable, *directorio_datos*. Finalmente, la ruta completa se almacena en *pathe_entrada*.

```
directorio_raizs = '/home/desk-chaguarma/Documents/UNI/SAG/PRACTICAS/PR1/';  
directorio_dats = 'IMPRECISOS_ACTUALIZADO/';  
directorio_sal= 'SALIDA/';  
directorio_diag= 'DIAGRAMAS/';  
  
pathe_entrada=[directorio_raizs directorio_dats];  
pathe_diag=[directorio_raizs directorio_diag];
```

El nombre del documento a trabajar lo almacenamos en *doc*.

```
doc = "BULLET_A.ERP";
```

Abrimos el documento deseado con la función `fopen` habiendo concatenado la ruta y el nombre del archivo.

```
fidin = fopen(strcat(path_entrada,doc), 'r')  
  
fidin = 7
```

Definimos una variable para quitar las primeras 6 líneas de documentación del archivo. Saltamos el nº de líneas deseado con un bucle.

```
elim = 6;  
for i = 1:elim  
    fgetl(fidin);  
end
```

A partir del patrón del archivo, obtenemos datos de cada fila, con un bucle `while` con la condición que lea el archivo hasta que llegue al final de él.

```
while ~feof(fidin)
```

Leemos la línea con la función `fgetl` y tomamos los tres primeros caracteres para controlar si hemos llegado al final del archivo. A partir de una función `if` controlamos si ha llegado al final.

```
registro = fgetl(fidin);  
control_fin=registro(1:3);  
if control_fin=='EOF' %Puedes introducir este texto al final del doc como  
parada  
    break  
else
```

Una vez comprobado que no es el final del documento, con la función `split`, dividimos la fila de datos que están separados por espacios y los almacenamos como un vector de caracteres en `registro`. Además, al añadir una nueva fila, aumentamos el contador `fila`,

```
fila = fila + 1;  
registro = split(registro);  
registro = registro (1:13);
```

Para almacenar la fecha, primero obtenemos la cadena de la fecha (yyyy-MM-dd) siguiendo la disposición de los datos en el archivo. Tras ello, a partir de la función `date2doy` obtenida de [aquí](#), transformamos la fecha a año con decimal.

```
fecha = strcat(registro(1),"-",registro(2),"-", registro(3));  
  
[doy,frac] = date2doy(datenum(fecha));  
  
frac = extractAfter(string(frac),1);  
fecha = strcat(registro(1),frac);
```

Finalmente, almacenamos los datos deseados en la matriz definida anteriormente. Almacenamos los datos en formato double.

```
tablaI(fila,:) = {double(string(fecha)), double(string(registro(6))),  
double(string(registro(7))), double(string(registro(8))),  
double(string(registro(11))), double(string(registro(12))),  
double(string(registro(13)))};  
end  
end  
  
fclose(fidin);
```

Aquí tenemos el numero de datos almacenados y la tabla obtenida.

```
fila
```

```
fila = 11320
```

```
tablaI(1:8,:)
```

```
ans = 8x7 table
```

	Fecha	X	Y	UT	RMS-X	RMS-Y	RMS-T
1	1.9923e+03	-0.1331	0.2301	-0.4388	3.2000e-04	3.0000e-04	1.3000e-05
2	1.9923e+03	-0.1350	0.2323	-0.4413	3.0000e-04	3.8000e-04	1.2000e-05
3	1.9923e+03	-0.1371	0.2345	-0.4436	2.9000e-04	2.6000e-04	1.2000e-05
4	1.9923e+03	-0.1392	0.2366	-0.4458	2.9000e-04	2.6000e-04	1.0000e-05
5	1.9923e+03	-0.1412	0.2387	-0.4478	2.8000e-04	3.0000e-04	1.2000e-05
6	1.9923e+03	-0.1428	0.2408	-0.4498	2.8000e-04	3.0000e-04	1.2000e-05
7	1.9923e+03	-0.1441	0.2432	-0.4519	2.8000e-04	3.0000e-04	2.1000e-05
8	1.9923e+03	-0.1451	0.2457	-0.4541	2.8000e-04	2.6000e-04	2.6000e-05

Guardamos la tabla tablaI de los datos imprecisos para compararla en otro archivo con los datos precisos.

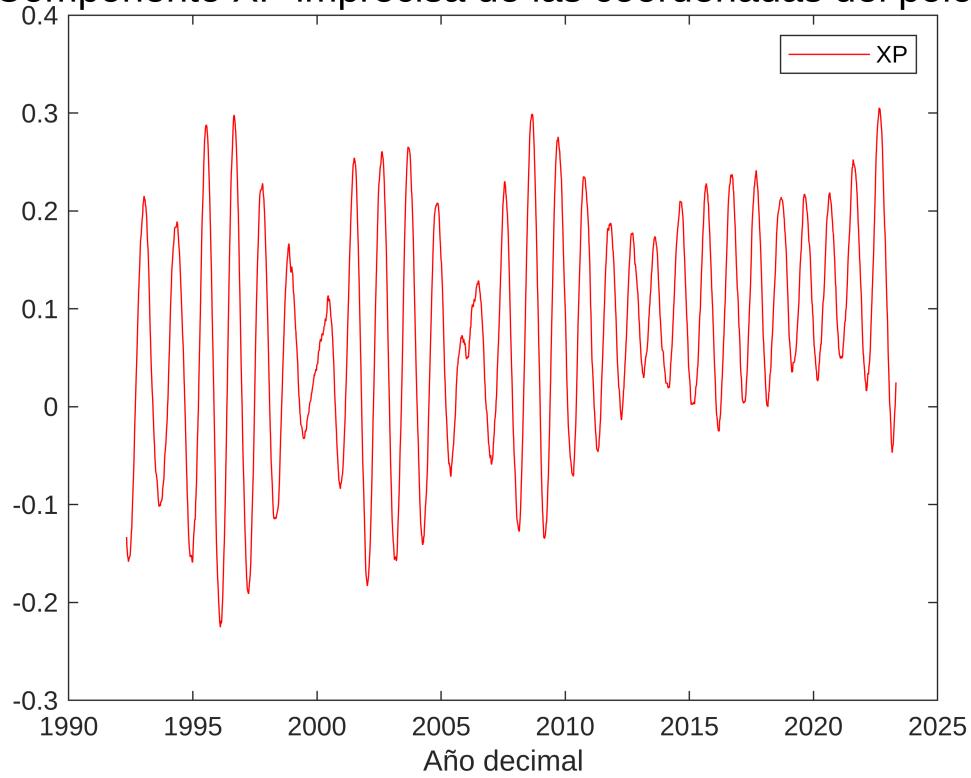
```
save([directorio_raizs directorio_sal,  
'tablaImprecisosActualizado.mat','tablaI'])
```

Graficando Datos

En esta gráfica representamos la variación de posición en X del polo durante el tiempo.

```
figure;  
plot(tablaI.Fecha,tablaI.X,"red")  
legend("XP")  
xlabel('Año decimal')  
sgtitle('Componente XP imprecisa de las coordenadas del polo')
```

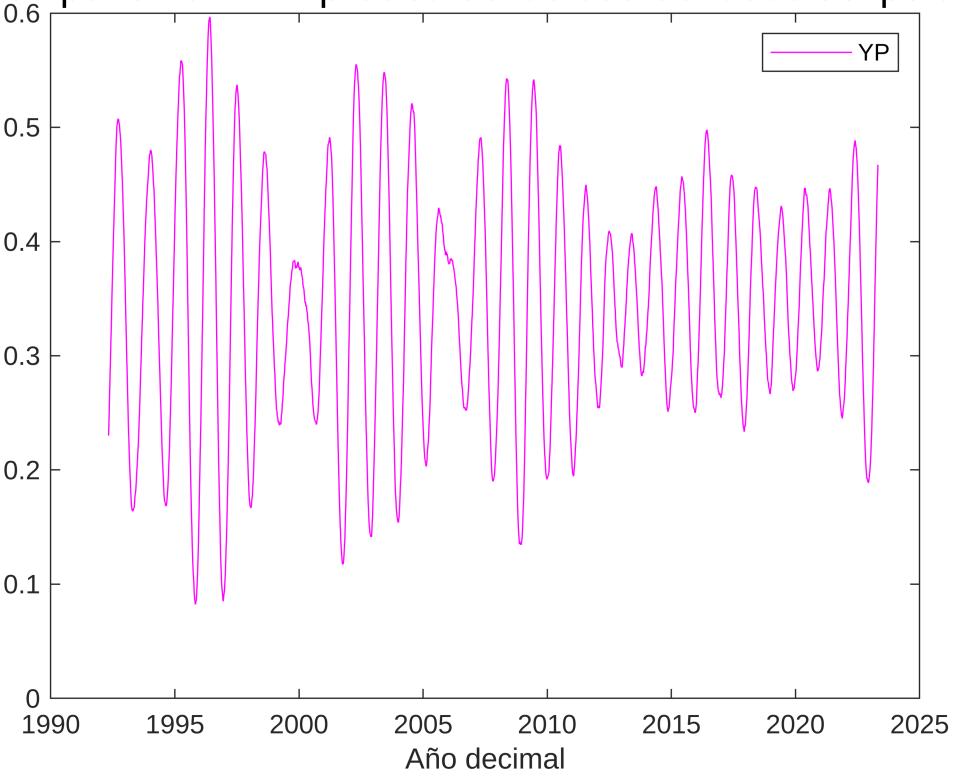
Componente XP imprecisa de las coordenadas del polo



En esta gráfica representamos la variación de posición en Y del polo durante el tiempo.

```
figure;
plot(tablaI.Fecha,tablaI.Y, "magenta")
legend( "YP" )
xlabel( 'Año decimal' )
sgtitle( 'Componente YP imprecisa de las coordenadas del polo' )
```

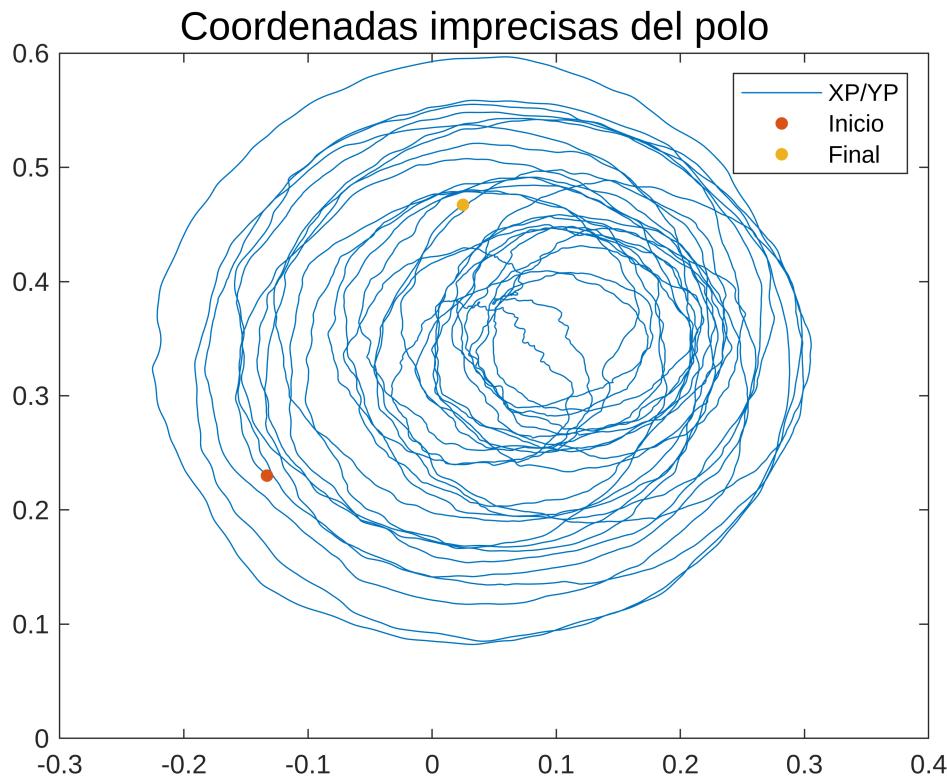
Componente YP imprecisa de las coordenadas del polo



Se puede ver que ambas componentes varían de forma parecida, pero la componente Y toma valores mayores.

Ahora vemos la relación de posición de X e Y, y vemos que la posición del polo va variando describiendo circunferencias.

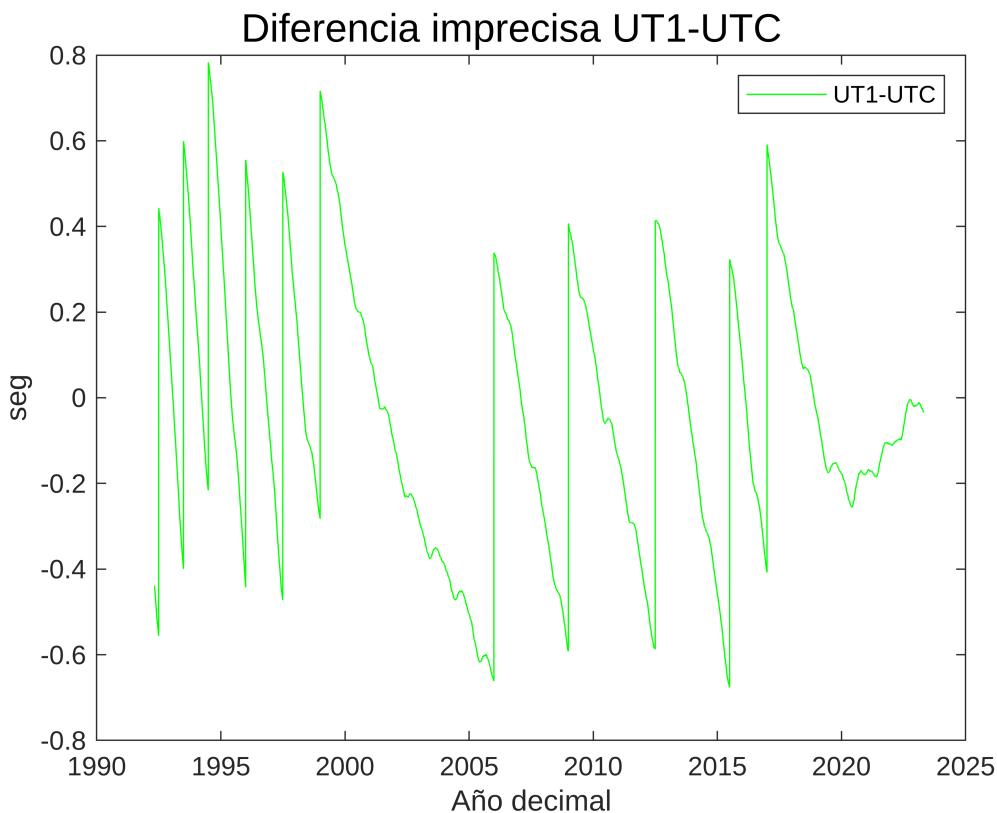
```
figure;
plot(tablaI.X,tablaI.Y)
hold on
plot(tablaI.X(1),tablaI.Y(1),".",'MarkerSize',16)
plot(tablaI.X(fila),tablaI.Y(fila),".",'MarkerSize',16)
legend("XP/YP","Inicio","Final")
sgtitle('Coordenadas imprecisas del polo')
hold off
```



Vemos que el polo describe elipses irregulares, superponiéndose.

Por último, graficamos las diferencias de UT1, que es la corrección de UT0 por el efecto del movimiento polar en la longitud del sitio de observación, y UTC, que es el tiempo universal coordinado.

```
figure;
plot(tablaI.Fecha,tablaI.UT, "green")
legend( "UT1-UTC" )
xlabel( 'Año decimal' )
ylabel( 'seg' )
sgtitle( 'Diferencia imprecisa UT1-UTC' )
```



Finalmente, para el tiempo observamos que su corrección hace una gráfica escalonada.

Adrián Fernández Tejada - Adrián Sánchez Loureiro

Funciones definidas

Aquí tenemos la función citada anteriormente, que nos transforma la fecha a un número decimal.

```
function [doy,fraction] = date2doy(inputDate)
%DATE2DOY Converts the date to decimal day of the year.
% [doy,fraction] = date2doy(inputDate)
%
% Descriptions of Input Variables:
% inputDate: Input date as a MATLAB serial datenumber
%
% Descriptions of Output Variables:
% doy: Decimal day of the year. For instance, an output of 1.5 would
%       indicate that the time is noon on January 1.
% fraction: Outputs the fraction of a year that has elapsed by the input
%           date.
%
% Example(s):
% >> [doy,frac] = date2doy(datenum('1/25/2004'));
%
```

```

% See also:

% Author: Anthony Kendall
% Contact: anthony [dot] kendall [at] gmail [dot] com
% Created: 2008-03-11
% Copyright 2008 Michigan State University.

%Want inputs in rowwise format
[doy,fraction] = deal(zeros(size(inputDate)));
inputDate = inputDate(:);

%Parse the inputDate
[dateVector] = datevec(inputDate);

%Set everything in the date vector to 0 except for the year
dateVector(:,2:end) = 0;
dateYearBegin = datenum(dateVector);

%Calculate the day of the year
doyRow = inputDate - dateYearBegin;

%Optionally, calculate the fraction of the year that has elapsed
flagFrac = (nargout==2);
if flagFrac
    %Set the date as the end of the year
    dateVector(:,1) = dateVector(:,1) + 1;
    dateYearEnd = datenum(dateVector);
    fracRow = (doyRow - 1) ./ (dateYearEnd - dateYearBegin);
end

%Fill appropriately-sized output array
doy(:) = doyRow;
if flagFrac
    fraction(:) = fracRow;
end
end

```

Tratamiento de Datos Precisos Actualizados

Trataremos el archivo `igs95p02.erp` extraído de la [NASA's Archive of Space Geodesy Data](#). Contiene los archivos IGS erp finales acumulados comenzando en la semana GPS 0860 o MJD 50264.5.

Tablas de datos

Definamos la tabla `tablaP` que almacenará los siguientes datos que extraeremos de los archivos en la carpeta `/PRECISOS`:

```
fecha, X-polo, Y-polo, UT1-UTC, Xsig, Ysig, UTsig.
```

Tomaremos datos desde 51545.50 a 58383.50 en fecha juliana modificada.

Primero, definimos el tipo de dato que almacenará cada columna y su nombre.

```
varTipos = [ "double", "double", "double", "double", "double", "double", "double" ];
varNames = [ "Fecha", "X", "Y", "UT", "X-SIG", "Y-SIG", "UT-SIG" ];
```

Segundo, una variable auxiliar en el que almacenaremos el número de columnas.

```
sz = [ 1 length(varNames) ];
```

Finalmente, definimos la tabla con lo citado anteriormente.

```
tablaP = table('Size',sz,'VariableTypes',varTipos,'VariableNames',varNames);
```

A continuación, la variable auxiliar `fila` para almacenar el nº de filas que tiene la tabla.

```
fila = 0;
```

Código Realizado

Definimos en primer lugar la ruta donde se encuentra nuestro archivo de datos en la variable `directorio_raiz`. También una variable, `directorio_datos`, con el nombre de la carpeta donde están los datos. Finalmente, la ruta completa se almacena en `pathe_entrada`.

```
directorio_raiz = '/home/desk-chaguarma/Documents/UNI/SAG/PRACTICAS/PR1/';
directorio_datos = 'PRECISOS_ACTUALIZADO/';
directorio_sal= 'SALIDA/';
directorio_diag= 'DIAGRAMAS/';

pathe_entrada=[directorio_raiz directorio_datos];
pathe_diag=[directorio_raiz directorio_diag];
```

Primero, la función `fullfile` crea una especificación de archivo completa a partir de la carpeta y nombres de archivo especificados. Finalmente, almacenamos en `archivos` la enumeración dada por la función `dir`. Esta enumera los archivos y carpetas que coinciden con el nombre dado.

```
archivos = dir(fullfile(path_entrada, "ig*"));
```

Tras lo anterior, almacenamos el número de archivos reconocidos en `numero`.

```
numero=size(archivos)
```

```
numero = 1x2  
1 1
```

Para obtener los datos de los archivos .erp, leeremos línea por línea estos documentos. Para llegar a las líneas de datos, saltaremos un número determinado de líneas de cabecera.

Hay dos formatos de cabecera en los archivos .erp, con su número específico de líneas a omitir. Hemos almacenado estos números en `elim` y `elimEOP`.

```
elim = 1;  
elimEOP = 2;
```

Para leer cada archivo, realizaremos un bucle `for` en el que leeremos del primer al último archivo reconocido. En cada uno, definiremos la variable `cabe` para controlar si hemos leído la cabecera y ha encontrado los datos o si ha leído el documento y no ha encontrado datos.

```
for i=1:numero(1)  
    % Tomamos el nombre del archivo i  
    fichero_entrada=[archivos(i).name];  
    % Abrimos el archivo i empleando fopen.  
    fidin = fopen([path_entrada fichero_entrada], 'r');  
  
    cabe=0;
```

Distinguiremos el formato del documento abierto en `fidin`, Tras ello, con un bucle `for`, saltamos el número de líneas necesarias. En caso de no seguir un formato (`cabe == 0`), se devuelve el nombre del archivo y paramos el bucle.

```
while ~feof(fidin)  
    registro = fgetl(fidin);  
  
    if length(registro)>=3  
        if registro(1:3) == 'EOP'  
            % Nos saltamos elimEOP filas  
            for j = 1:elimEOP  
                fgetl(fidin);  
            end  
        cabe = 1;
```

```

        break
elseif (length(registro)>=5) & (registro(1:5) == ' MJD ')
    % Nos saltamos elim filas
    for j = 1:elim
        fgetl(fidin);
    end
    cabe = 1;
    break
end
end

end

```

Una vez omitidas las líneas de cabecera, identificado el patrón del archivo, obtenemos los datos de cada fila.

Para ello, emplearemos un bucle `while`, que lea el archivo hasta su final.

Leemos la línea con la función `fgetl` y tomamos los tres primeros caracteres para controlar si hemos llegado al final del archivo. A partir de una función `if` controlamos si ha llegado al final.

```

if cabe == 0
    break
end

while ~feof(fidin)
    registro = fgetl(fidin);

    if registro(1:3) == 'EOF'
        break
    end

```

Una vez comprobado que no es el final del documento, con la función `split`, dividimos la fila de datos que estan separados por espacios y los almacenamos como un vector de caracteres en `registro`. Además, al añadir una nueva fila, aumentamos el contador `fila`,

```

else
    fila = fila + 1;
    registro = split(registro);
    registro = registro(1:8);

```

Para almacenar la fecha, primero transformamos el dato de fecha juliana modificada a formato año-mes-día (yyyy-MM-dd). Tras ello, a partir de la función `date2doy` obtenida [aquí](#), transformamos la fecha a año con decimal.

```

fecha = datetime(str2num(cell2mat(registro(1))),
'ConvertFrom','mjd', 'Format','yyyy-MM-dd');

[doy,frac] = date2doy(datenum(fecha));

frac = extractAfter(string(frac),1);
fecha = strcat(string(year(fecha)),frac);

```

Finalmente, almacenamos los datos deseados en la tabla definida anteriormente. Almacenamos los datos en formato double.

```
tablaP(fila,:) = {double(string(fecha)),
double(string(registro(2)))*10^(-6), double(string(registro(3)))*10^(-6),
double(string(registro(4)))*10^(-7), double(string(registro(6))), 
double(string(registro(7))), double(string(registro(8)))};
end
fclose(fidin);
end
%fclose(fidsal);
```

Tabla Obtenida

Aquí tenemos la tabla obtenida y el número de datos almacenados.

```
fila
```

```
fila = 9709
```

```
tablaP(1:8,:)
```

```
ans = 8×7 table
```

	Fecha	X	Y	UT	X-SIG	Y-SIG	UT-SIG
1	1.9965e+03	0.1774	0.5491	0.1877	70	200	0
2	1.9965e+03	0.1807	0.5465	0.1865	50	200	0
3	1.9965e+03	0.1838	0.5437	0.1853	100	170	0
4	1.9965e+03	0.1870	0.5409	0.1840	100	170	0
5	1.9965e+03	0.1901	0.5380	0.1826	100	150	0
6	1.9965e+03	0.1938	0.5350	0.1812	80	170	0
7	1.9965e+03	0.1978	0.5326	0.1798	60	170	0
8	1.9965e+03	0.2018	0.5303	0.1784	70	210	0

Guardamos la tabla tablaP de los datos precisos.

```
save([directorio_raiz directorio_sal,
'tablaPrecisosActualizado.mat'], "tablaP")
```

Graficando Datos

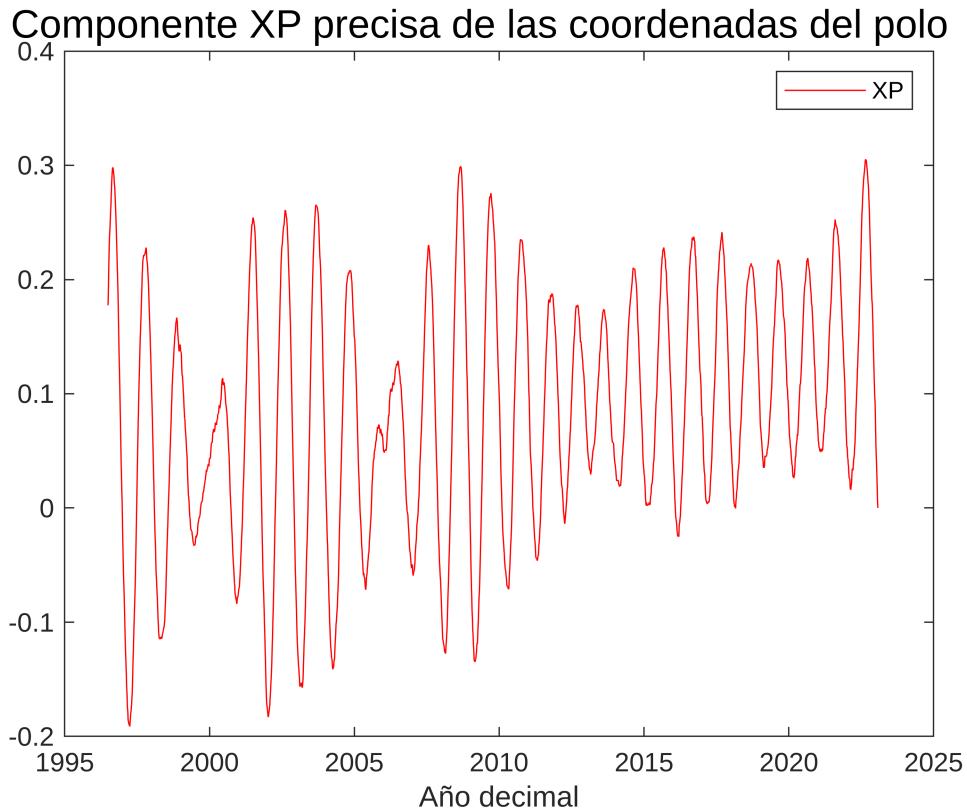
En esta gráfica representamos la variación de posición en X del polo durante el tiempo.

```
figure;
```

```

plot(tablaP.Fecha,tablaP.X, "red")
legend( "XP" )
xlabel( 'Año decimal' )
sgtitle( 'Componente XP precisa de las coordenadas del polo' )

```



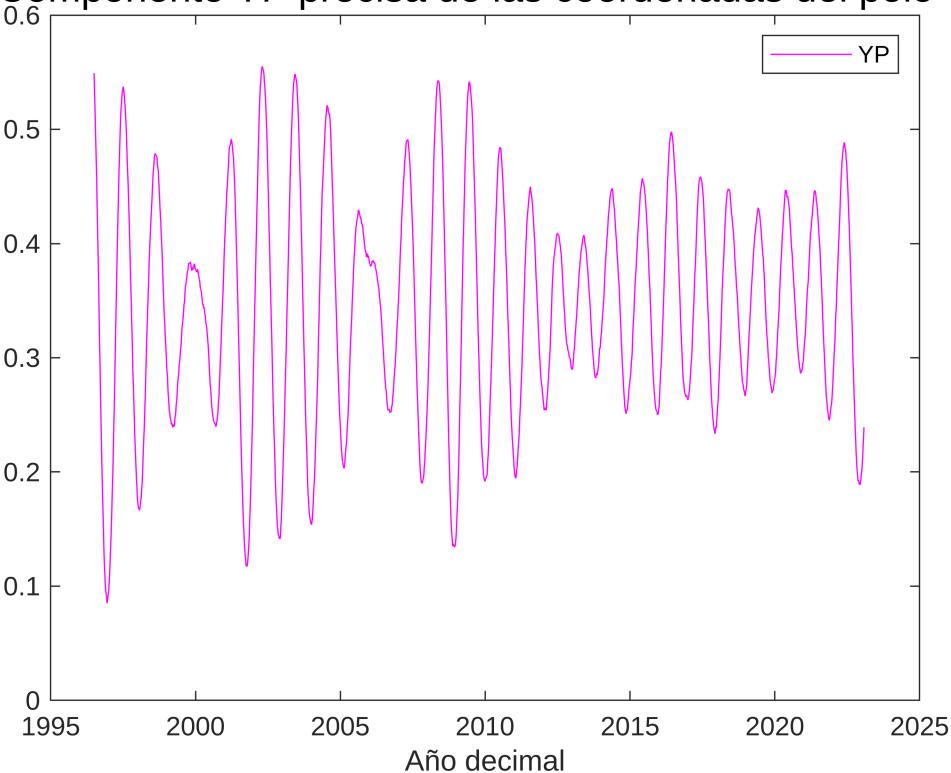
En esta gráfica representamos la variación de posición en Y del polo durante el tiempo.

```

figure;
plot(tablaP.Fecha,tablaP.Y, "magenta")
legend( "YP" )
xlabel( 'Año decimal' )
sgtitle( 'Componente YP precisa de las coordenadas del polo' )

```

Componente YP precisa de las coordenadas del polo

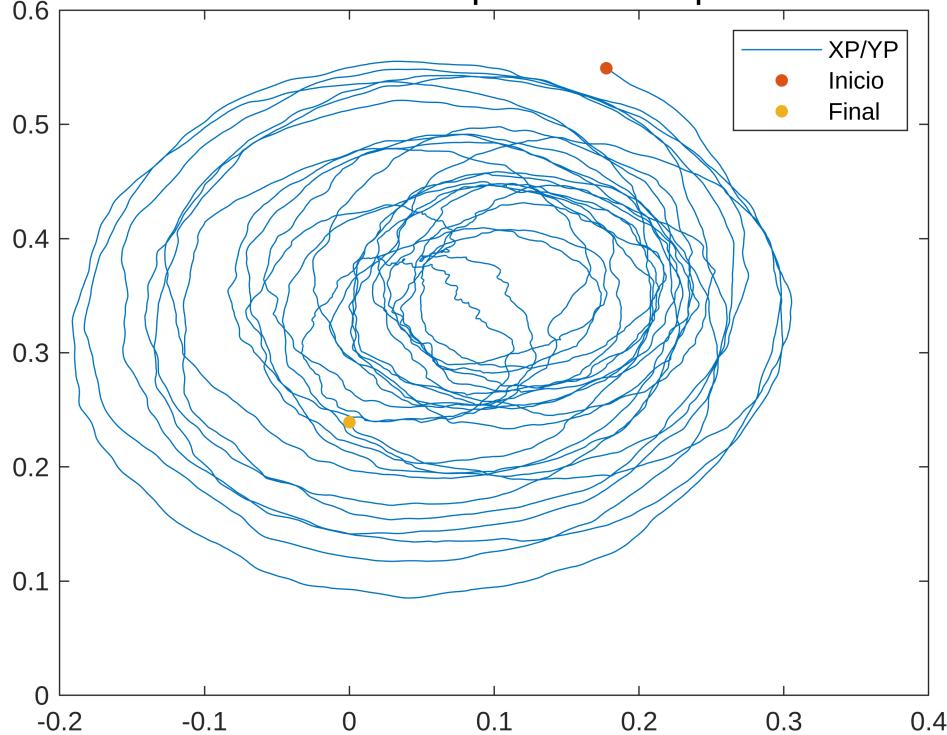


Se puede ver que ambas componentes varían de forma parecida, pero la componente Y toma valores mayores.

Ahora vemos la relación de posición de X e Y, y vemos que la posición del polo va variando describiendo circunferencias.

```
figure;
plot(tablaP.X,tablaP.Y)
hold on
plot(tablaP.X(1),tablaP.Y(1),".",'MarkerSize',16)
plot(tablaP.X(fila),tablaP.Y(fila),".",'MarkerSize',16)
legend("XP/YP","Inicio","Final")
sgtitle('Coordenadas imprecisas del polo')
hold off
```

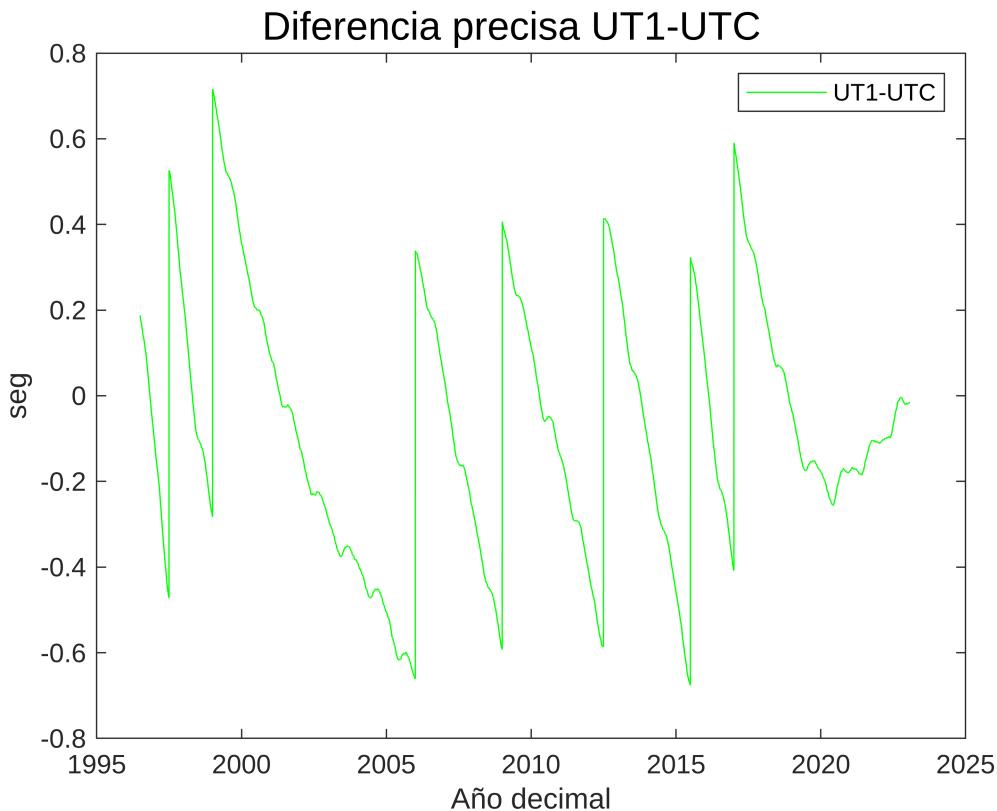
Coordenadas imprecisas del polo



Vemos que el polo describe elipses irregulares, superponiéndose.

Por último, graficamos las diferencias de UT1, que es la corrección de UT0 por el efecto del movimiento polar en la longitud del sitio de observación, y UTC, que es el tiempo universal coordinado.

```
figure;
plot(tablaP.Fecha,tablaP.UT, "green")
legend( "UT1-UTC" )
xlabel( 'Año decimal' )
ylabel( 'seg' )
sgtitle( 'Diferencia precisa UT1-UTC' )
```



Finalmente, para el tiempo observamos que su corrección hace una gráfica escalonada.

Adrián Fernández Tejada - Adrián Sánchez Loureiro

Funciones definidas

Aquí tenemos la función citada anteriormente, que nos transforma la fecha a un número decimal.

```
function [doy,fraction] = date2doy(inputDate)
%DATE2DOY Converts the date to decimal day of the year.
% [doy,fraction] = date2doy(inputDate)
%
% Descriptions of Input Variables:
% inputDate: Input date as a MATLAB serial datenumber
%
% Descriptions of Output Variables:
% doy: Decimal day of the year. For instance, an output of 1.5 would
%       indicate that the time is noon on January 1.
% fraction: Outputs tratas_MAM_2_3he fraction of a year that has elapsed
by the input
%       date.
%
% Example(s):
% >> [doy,frac] = date2doy(datenum('1/25/2004'));
%
```

```

% See also:

% Author: Anthony Kendall
% Contact: anthony [dot] kendall [at] gmail [dot] com
% Created: 2008-03-11
% Copyright 2008 Michigan State University.

%Want inputs in rowwise format
[doy,fraction] = deal(zeros(size(inputDate)));
inputDate = inputDate(:);

%Parse the inputDate
[dateVector] = datevec(inputDate);

%Set everything in the date vector to 0 except for the year
dateVector(:,2:end) = 0;
dateYearBegin = datenum(dateVector);

%Calculate the day of the year
doyRow = inputDate - dateYearBegin;

%Optionally, calculate the fraction of the year that has elapsed
flagFrac = (nargout==2);
if flagFrac
    %Set the date as the end of the year
    dateVector(:,1) = dateVector(:,1) + 1;
    dateYearEnd = datenum(dateVector);
    fracRow = (doyRow - 1) ./ (dateYearEnd - dateYearBegin);
end

%Fill appropriately-sized output array
doy(:) = doyRow;
if flagFrac
    fraction(:) = fracRow;
end
end

```

Comparando los Datos

Cargando Datos

En primer lugar cargamos los dos archivos donde guardamos las tablas de datos anteriormente creados.

```
directorio_raiz = '/home/desk-chaguarma/Documents/UNI/SAG/PRACTICAS/PR1/' ;
directorio_datos = 'SALIDA/' ;
directorio_diag= 'DIAGRAMAS/' ;

pathe_diag=[directorio_raiz directorio_diag]
```

```
pathe_diag =
'/home/desk-chaguarma/Documents/UNI/SAG/PRACTICAS/PR1/DIAGRAMAS/'
```

```
load([directorio_raiz directorio_datos 'tablaImprecisosActualizado.mat']) 
load([directorio_raiz directorio_datos 'tablaPrecisosActualizado.mat'])
```

```
size(tablaI)
```

```
ans = 1x2
    11320      7
```

```
size(tablaP)
```

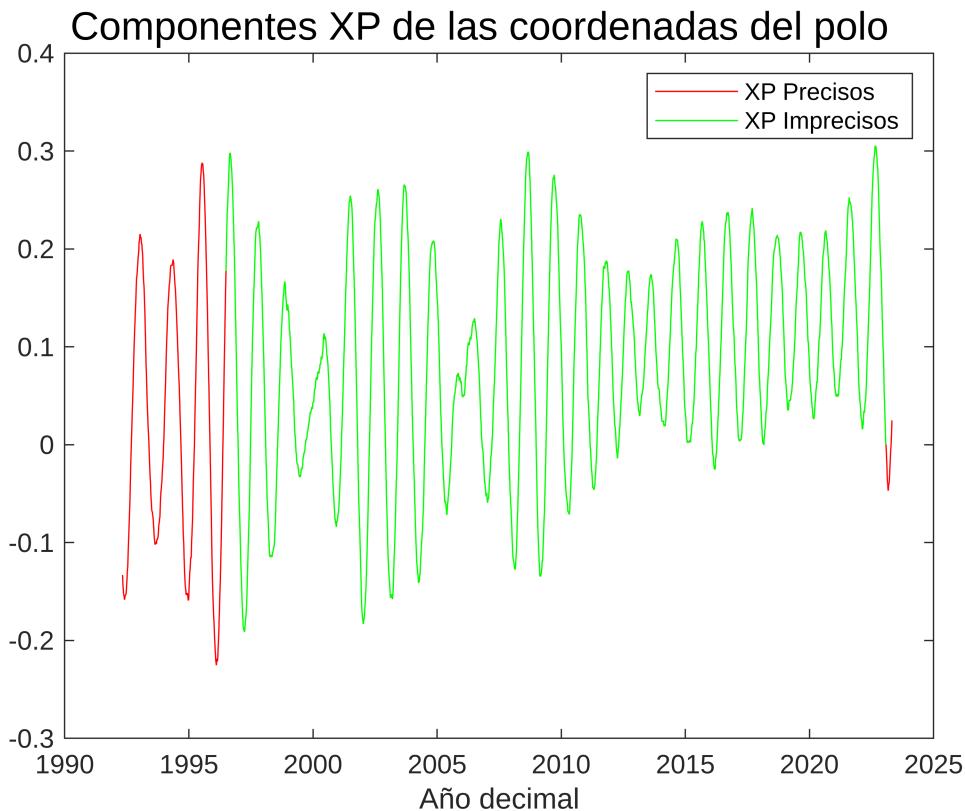
```
ans = 1x2
    9709      7
```

Al tener más datos imprecisos que precisos, no haremos la diferencia de los datos. Simplemente, mostraremos superpuestas ambas gráficas.

Polo X respecto al tiempo

En primer lugar comparamos las gráficas de la posición X del polo, y no apreciamos ninguna variación.

```
figure;
plot(tablaI.Fecha,tablaI.X, "red")
hold on
plot(tablaP.Fecha,tablaP.X, "green")
legend("XP Precisos", "XP Imprecisos")
xlabel('Año decimal')
sgtitle('Componentes XP de las coordenadas del polo')
hold off
```

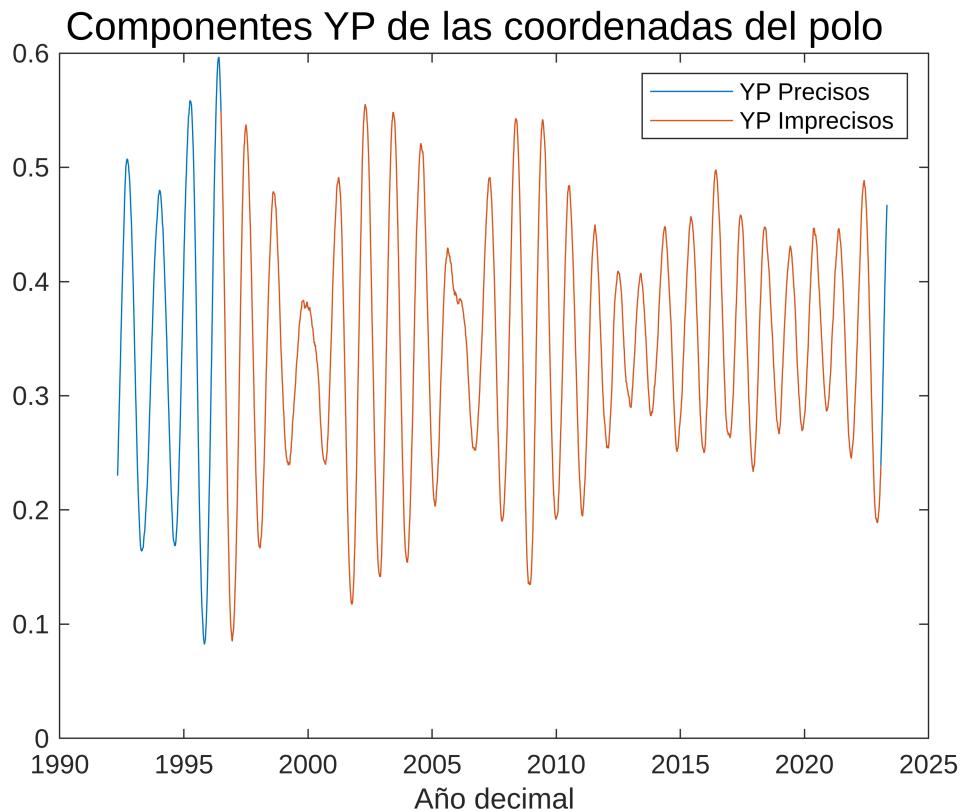


Sin embargo, si en la gráfica anterior nos acercamos lo suficiente, vemos pequeñas diferencias entre los datos.

Polo Y respecto al tiempo

Lo siguiente es comparar la posición Y del polo, y como antes no observamos ninguna diferencia.

```
figure;
plot(tablaI.Fecha,tablaI.Y)
hold on
plot(tablaP.Fecha,tablaP.Y)
legend("YP Precisos","YP Imprecisos")
xlabel('Año decimal')
sgtitle('Componentes YP de las coordenadas del polo')
hold off
```

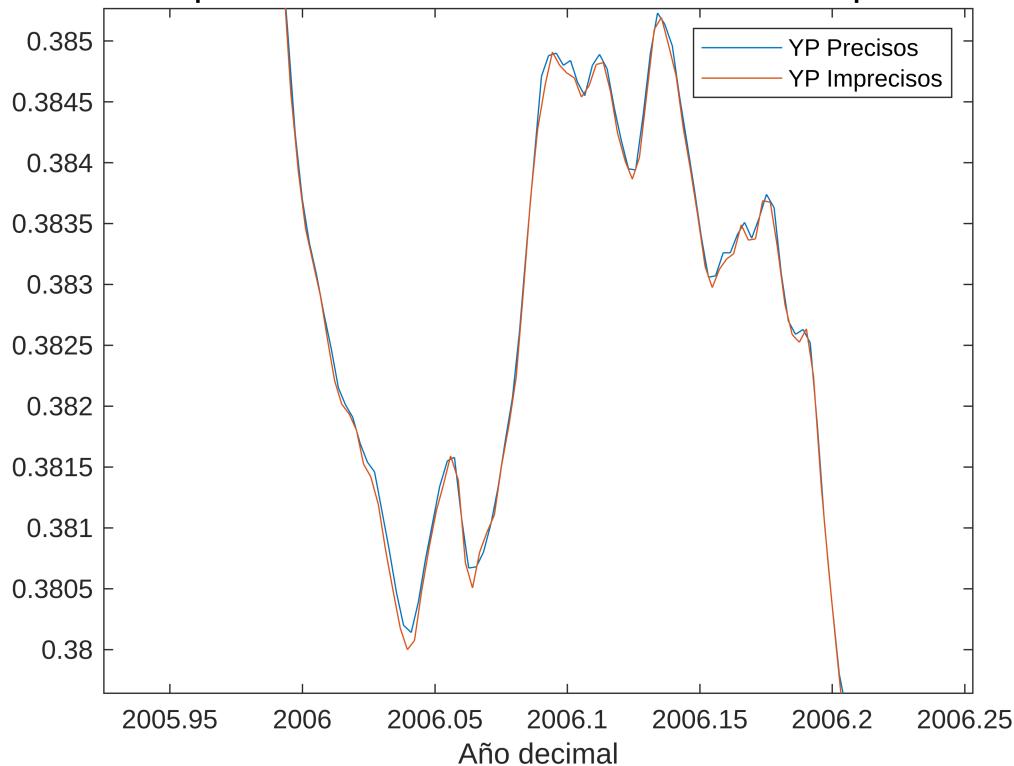


```

figure;
plot(tablaI.Fecha,tablaI.Y)
hold on
plot(tablaP.Fecha,tablaP.Y)
legend( "YP Precisos" , "YP Imprecisos" )
xlabel( 'Año decimal' )
sgtitle( 'Componentes YP de las coordenadas del polo' )
hold off
xlim([2005.925 2006.253])
ylim([0.37964 0.38527])

```

Componentes YP de las coordenadas del polo

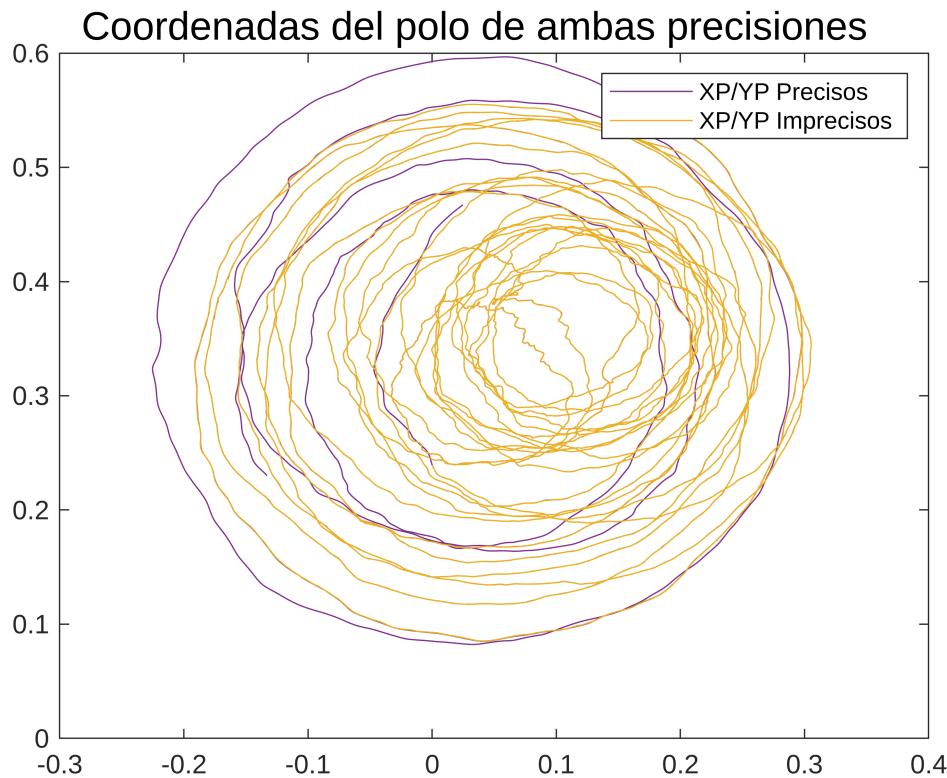


Como pasó anteriormente, si nos acercamos lo suficiente vemos pequeñas diferencias.

Posiciones del polo X e Y

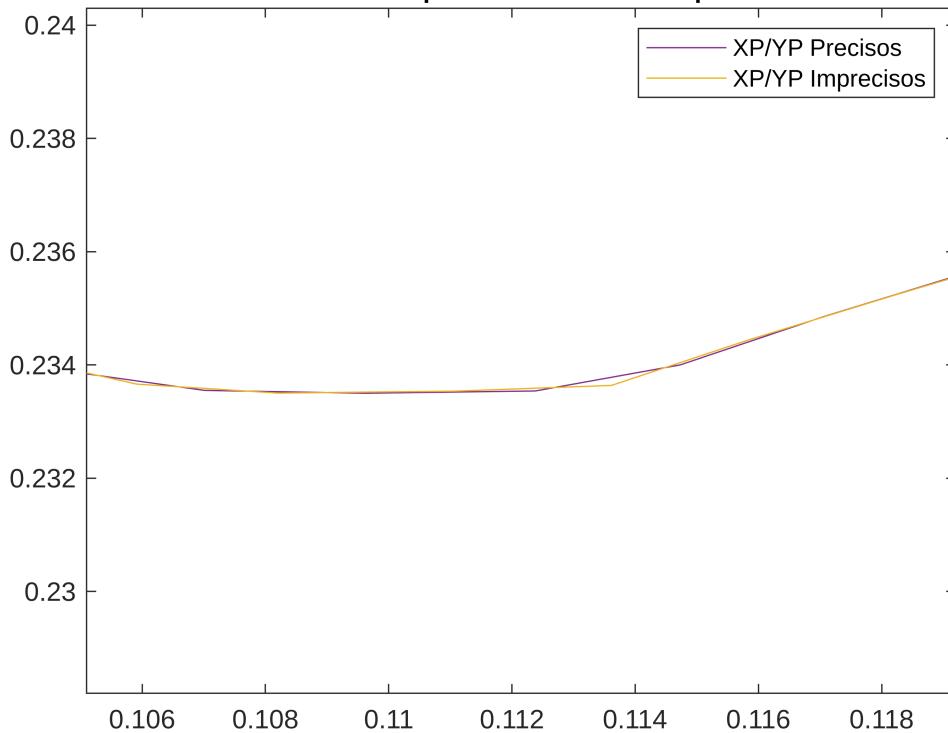
Ahora comparamos las posiciones X e Y del polo, no hay variación aparente.

```
figure;
plot(tablaI.X,tablaI.Y,'Color',[0.4940 0.1840 0.5560])
hold on
plot(tablaP.X,tablaP.Y,'Color',[0.9290 0.6940 0.1250])
legend("XP/YP Precisos","XP/YP Imprecisos")
sgtitle('Coordenadas del polo de ambas precisiones')
hold off
```



```
figure;
plot(tablaI.X,tablaI.Y,'Color',[0.4940 0.1840 0.5560])
hold on
plot(tablaP.X,tablaP.Y,'Color',[0.9290 0.6940 0.1250])
legend("XP/YP Precisos","XP/YP Imprecisos")
shtitle('Coordenadas del polo de ambas precisiones')
hold off
xlim([0.1051 0.1192])
ylim([0.2282 0.2403])
```

Coordenadas del polo de ambas precisiones

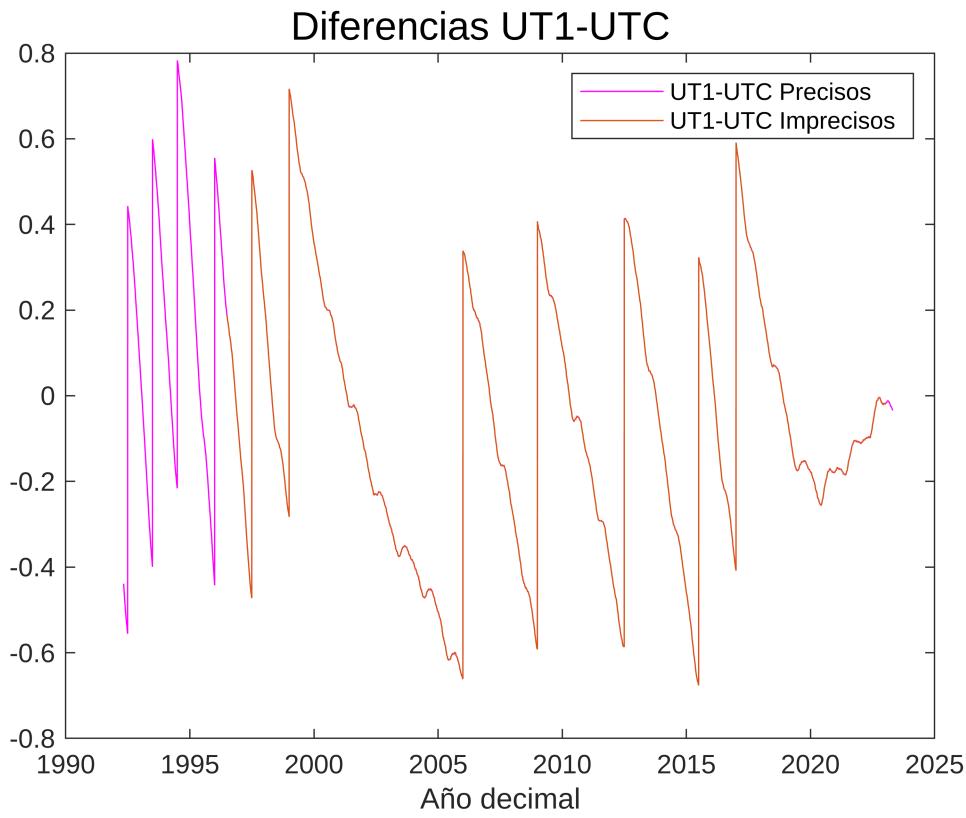


Y como era de esperar, al unir las gráficas donde estén X e Y siguen habiendo diferencias al acercarnos lo suficiente.

Variación UT1-UTC respecto del tiempo

En esta última gráfica la diferencia es más evidente, ya que en el inicio de ella hay un escalón de los datos precisos, así que podemos concluir que los otros datos no son tan acertados como los precisos.

```
figure;
plot(tablaI.Fecha,tablaI.UT, "magenta")
hold on
plot(tablaP.Fecha,tablaP.UT)
legend( "UT1-UTC Precisos", "UT1-UTC Imprecisos" )
xlabel( 'Año decimal' )
sgtitle( 'Diferencias UT1-UTC' )
hold off
```



Aqui obsaervamos las irregularidades.

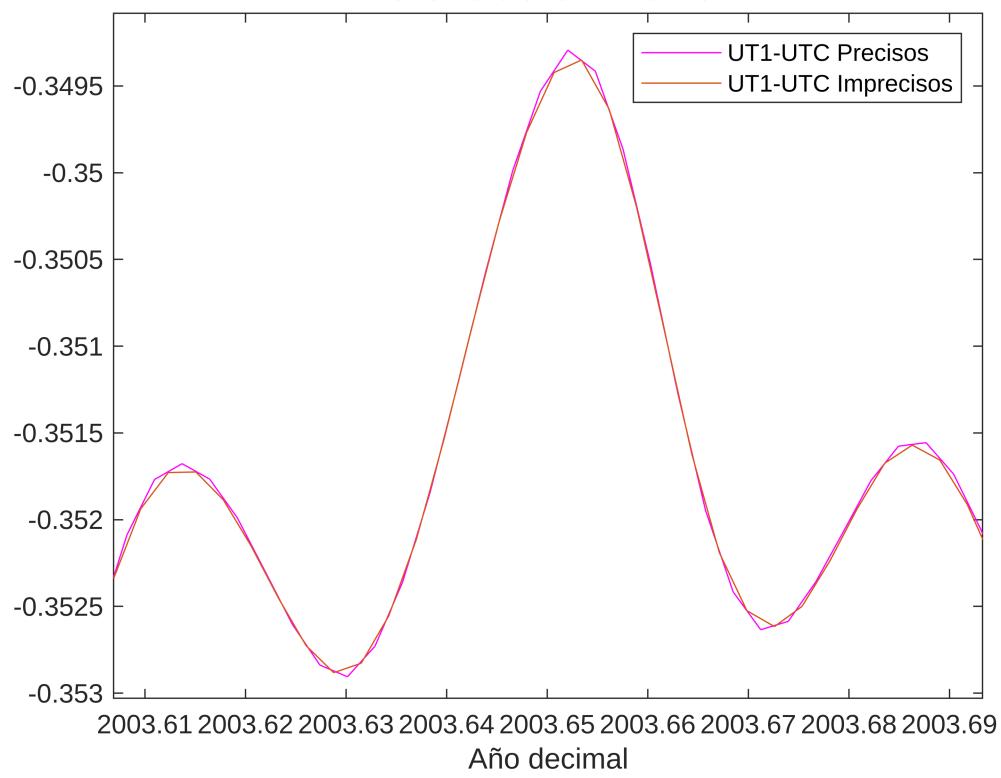
```

figure;
plot(tablaI.Fecha,tablaI.UT, "magenta")
hold on
plot(tablaP.Fecha,tablaP.UT)
legend( "UT1-UTC Precisos", "UT1-UTC Imprecisos" )
xlabel( 'Año decimal' )
sgtitle( 'Diferencias UT1-UTC' )
hold off

xlim([2003.6069 2003.6933])
ylim([-0.35303 -0.34908])

```

Diferencias UT1-UTC



Adrián Fernández Tejada - Adrián Sánchez Loureiro