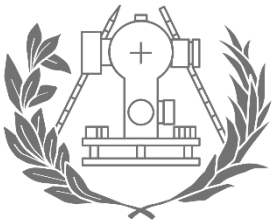
DESARROLLO DE ALGORITMOS USANDO DATOS GNSS



Trabajo Final de Grado en Ingeniería Geomática y Topografía

Escuela Técnica Superior de Ingeniería Geodésica, Cartográfica y Topográfica

Autor: Jorge Hernández Olcina

Tutora: Ana Belén Anquela Julián

Julio 2017

Resumen

La tecnología GNSS (*Sistema Global de Navegación por* Satélite) es aquella que comprende el conjunto de sistemas de posicionamiento por satélite que permiten obtener posicionamiento espacial de forma autónoma, con cobertura global.

El desarrollo del GNSS surge inicialmente en los años 70, con el despliegue del sistema militar estadounidense GPS (*Sistema de Posicionamiento Global*), destinado a la localización de objetivos, tropas, guiado de misiles etc. A partir de una constelación formada por una red de satélites, es posible determinar la posición de un receptor GNSS, lo que conlleva una gran multitud de utilidades en la actualidad, tanto aplicaciones militares como aplicaciones civiles.

En el presente trabajo se introducen los conceptos básicos del funcionamiento de los sistemas GNSS, así como el software existente actualmente que permite el tratamiento de la información procedente de los sistemas de posicionamiento por satélite.

Con todo ello se va a proceder a desarrollar una serie de algoritmos de cálculo, basando éstos en scripts y bibliografía existentes, que permitan el cálculo de la posición de un receptor en la superficie terrestre. A su vez, esta serie de procedimientos permitirá conocer la posición de los satélites en el cielo, referenciando su posición con respecto a la Tierra.

Por último se analizarán los resultados obtenidos tanto para la posición del receptor como para las posiciones de los satélites y se hará énfasis en la aplicación del procedimiento seguido, además de detallar los costes que generarían el desarrollo de este trabajo.

Palabras clave

GNSS, algoritmos, interpolación Lagrange, Matlab, Easy Suite.

Abstract

GNSS (Global Navigation Satellite System) technology is that which includes the set of satellite positioning systems that allows obtaining spatial positioning in an autonomous way, with global coverage.

The development of the GNSS emerged initially in the 70s, with the deployment of the US military system GPS (Global Positioning System), destined to the location of objectives, troops, guided missiles etc. From a constellation formed by a network of satellites, it is possible to determine the position of a GNSS receiver, which entails a large multitude of utilities at present, both military applications and civil applications.

This document introduces the basic concepts of the operation of GNSS systems, as well as existing software that allows the processing of information from satellite positioning systems.

With this, we will proceed to develop a serie of calculation algorithms, based on existing scripts and bibliography, which allow the calculation of the position of a receiver on the Earth's surface. In turn, these series of procedures will allow to know the position of the satellites in the sky, referencing their position with respect to the Earth.

Finally, the results obtained for the position of the receiver and the positions of the satellites will be analyzed, emphasizing the application of the procedure followed, as well as detailing the costs that would generate the development of this work.

Key words

GNSS, algorithms, Lagrange’s interpolation, Matlab, Easy Suite.

Resum

La tecnologia GNSS (Sistema Global de Navegació per Satèl·lit) és aquella que comprén el conjunt de sistemes de posicionament per satèl·lit que permeten obtindre posicionament espacial de forma autònoma, amb cobertura global.

El desenvolupament del GNSS sorgix inicialment als anys 70, amb el desplegament del sistema militar nord-americà GPS (Sistema de Posicionament Global), destinat a la localització d'objectius, tropes, guiat de míssils etc. A partir d'una constel·lació formada per una xarxa de satèl·lits, és possible determinar la posició d'un receptor GNSS, la qual cosa comporta una gran multitud d'utilitats en l'actualitat, tant aplicacions militars com aplicacions civils.

En el present treball s'introduïxen els conceptes bàsics del funcionament dels sistemes GNSS, així com els programes existents actualment que permeten el tractament de la informació procedent dels sistemes de posicionament per satèl·lit.

Amb tot això se'n va a procedir a desenvolupar una sèrie d'algoritmes de càlcul, basant aquestos en scripts i bibliografia existents, que permeten el càlcul de la posició d'un receptor en la superfície terrestre. Al seu torn, esta sèrie de procediments permetrà conéixer la posició dels satèl·lits en el cel, referenciant la seua posició respecte a la Terra.

Finalment s'analitzaran els resultats obtinguts tant per a la posició del receptor com per a les posicions dels satèl·lits i es farà èmfasi en l'aplicació del procediment seguit, a més de detallar els costos que generarien el desenrotllament d'aquest treball.

Paraules clau

GNSS, algoritmes, interpolació Lagrange, Matlab, Easy Suite.

Índice

Índice de figuras, tablas y gráficos

Figuras

Gráficas

Tablas

MEMORIA

1. Introducción

Una de las tecnologías más utilizadas en la actualidad son los **sistemas de navegación global por satélite** (GNSS), tanto a nivel de usuario como a nivel profesional, ya que permiten desarrollar una gran cantidad de aplicaciones, como pueden ser sistemas de navegación y guiado, localización, cartografiado, gestión del territorio etc. Éstos son ejemplos que cada vez están más presentes en las vidas cotidianas de las personas, pues las infraestructuras globales creadas para el uso del GNSS permiten la aparición de multitud de explotaciones profesionales.

La motivación de este trabajo es profundizar en el funcionamiento del GNSS, además de comprender y adquirir los conocimientos básicos de cálculo que llevan integrados algunos dispositivos que usan esta tecnología.

En el presente trabajo se pondrán en conocimiento los conceptos básicos sobre el posicionamiento terrestre empleando la tecnología de posicionamiento por satélite, además de profundizar en la metodología de cálculo a partir de algoritmos matemáticos que permiten obtener las coordenadas de un dispositivo que incorpore la tecnología GNSS.

1.1 Situación actual

La necesidad de saber nuestra posición en la superficie de la Tierra cada vez es mayor, y una de las formas de conocerla a tiempo real y de manera rápida es mediante el posicionamiento GNSS, por tanto, en la actualidad, los sistemas de posicionamiento por satélite se han convertido en una herramienta indispensable, por tanto, su explotación genera multitud de aplicaciones.

Todo esto es posible a las constelaciones de satélites que se encuentran en órbita, las cuales son gestionadas por distintos organismos y entidades que permiten la conexión de cualquier dispositivo a las redes de satélites para obtener la posición en la superficie. En definitiva, se necesita de una red de satélites operativos, que cubran la totalidad de la Tierra, y permitan la conexión libre a los mismo por cualquier receptor.

En la actualidad existen diversas constelaciones en activo, de las cuales destacan:

- ***NAVSTAR – GPS:*** (Navigation System and Ranging - Global Position System). Red de satélites gestionada por el Departamento de Defensa de los Estados Unidos. Es la más utilizada actualmente y cuenta con una red de 32 satélites activos.
- ***GLONASS:*** Red de satélites gestionada por el Ministerio de defensa de la Federación Rusa. Cada vez se utiliza más en la actualidad y cuenta con 24 satélites activos en órbita.
- ***Galileo:*** Sistema de posicionamiento por satélite desarrollado por la Unión Europea. Se encuentra en fase de despliegue y desarrollo en la actualidad, pero está operativo desde finales de 2016.

1.2 Objetivos

La finalidad de este trabajo es profundizar en la metodología de cálculo de la posición de un receptor GNSS en la superficie terrestre y la empleada para el cálculo de la posición de los satélites en el cielo. Para ello se va a partir de una rutina de algoritmos existente, a partir de la cual se van a modificar dichos programas además de desarrollar nuevas rutinas de cálculo que se adapten a las necesidades deseadas.

Resumiendo, el trabajo consta de las siguientes partes:

- Modificar la estructura de algoritmos y scripts existentes con el fin de adaptarlos a los datos actuales. La rutina dada fue programada con objeto de cálculo de la posición de un receptor con efemérides transmitidas, en el año 2001. Actualmente los ficheros de observables y navegación han cambiado, por consiguiente, será necesario adaptar la rutina a los nuevos ficheros.
- Generar nuevos scripts y algoritmos que realicen el mismo procedimiento de cálculo que la rutina dada solo que en este caso, los datos de partida son efemérides precisas.
- Con todo esto, se va a generar un código que calcule la posición de un receptor con ambas soluciones (efemérides transmitidas y precisas), y que, a su vez, calcule la posición de los satélites que intervienen en la observación.

1.3 Estado del arte

Actualmente existe una gran diversidad de software para el post-proceso de datos GNSS, la gran mayoría de ellos desarrollados por instituciones, universidades y pequeñas organizaciones para su propio uso, aunque también las empresas privadas que fabrican tecnología para la observación GNSS de alta precisión, cuentan con software privativo propio, adaptado a sus propias necesidades.

Debido a que existe una gran cantidad de software en el mercado, tanto libre como de pago, en el presente trabajo se van a comentar aquellos que se han considerado oportunos. Por un lado se van a comentar aquellos programas más relevantes en la actualidad y por otro lado se van a comentar una serie de librerías de código Matlab [3], de las cuales se ha sacado la rutina que se va a emplear en el desarrollo de este trabajo.

Los programas científicos para el tratamiento de los datos GNSS son:

- **Bernese:** *[fuentes: Unavco y Nagarvil]*
  Software científico multi-GNSS de procesamiento de datos de alta precisión desarrollado por el Instituto de Astronomía de la Universidad de Berna (AIUB). Bernese es un software comercial de alto rendimiento.
  Los usuarios más habituales de este programa son:
  - Usuarios científicos de investigación y educación.
  - Organismos responsables de levantamientos GNSS de alta precisión (por ejemplo, redes de primer orden).
  - Organismos responsables de mantener las redes de estaciones permanentes.
  - Usuarios comerciales con aplicaciones GNSS complejas que exigen una alta precisión, fiabilidad y alta productividad.

  El programa Bernese está perfectamente adaptado para:

- Procesamiento rápido de redes de frecuencia simple y doble.
- Procesamiento automático de redes de estaciones permanentes.
- Procesamiento de los datos de un gran número de receptores.
- Combinación de diferentes tipos de receptores, teniendo en cuenta variaciones de fase del centro de la antena.
- Procesamiento combinado de observaciones GPS y GLONASS.
- Resolución ambigüedades en líneas base de larga distancia (2000 km y más).
- Monitoreo del comportamiento de la ionosfera y la troposfera.
- Estimación del reloj y tiempo de transferencia.
- Generación de soluciones de red de mínima restricción.
- Determinación de la órbita y estimación de los parámetros de orientación de la Tierra.

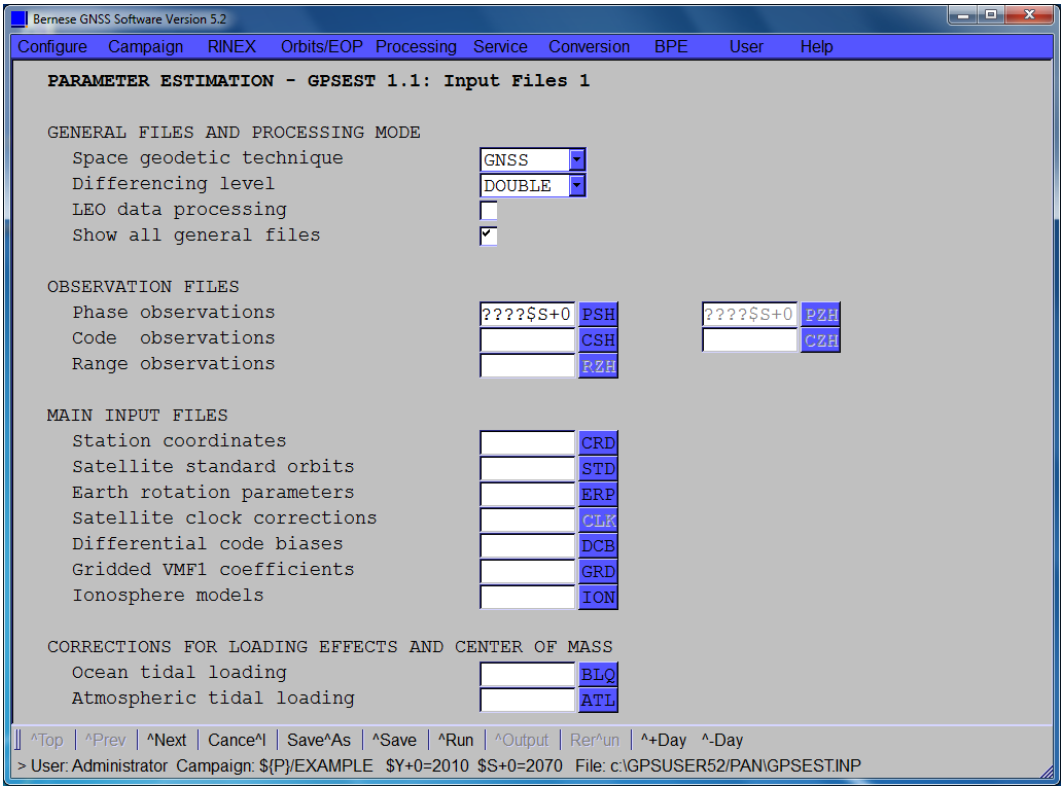


Figura 1. *Interfaz del programa Bernese. [fuente: gsc-europa]*

- **GAMIT/GLOBK/TRACK:** *[fuentes: Unavco y Nagarvil]*
Software desarrollado por el Department of Earth Atmospheric and Planetary Sciences (Massachusetts Institute of Technology). Programa desarrollado para el análisis de mediciones GPS, principalmente empleado para estudiar la deformación cortical. El software ha sido desarrollado por el MIT, el Instituto Scripps de Oceanografía y la Universidad de Harvard con el apoyo de la Fundación Nacional de Ciencia.

    - **GAMIT** es una colección de programas utilizados para el análisis de los datos GPS. Utiliza observables de fase y de pseudodistancia GPS para estimar posiciones tridimensionales relativas de las estaciones de tierra y órbitas de los satélites, retardo s cenitales atmosféricos, y parámetros

de orientación tierra. El software está diseñado para ejecutarse en cualquier sistema operativo UNIX.

- **GLOBK** es un filtro de Kalman, cuyo principal objetivo es combinar varias soluciones geodésicas tales como GPS, VLBI, y los experimentos SLR. Se aceptan como datos, o "cuasi-observaciones" las estimaciones y matrices de covarianza para las coordenadas de la estación, los parámetros de orientación de la Tierra, los parámetros orbitales, y las posiciones de la fuente generadas a partir del análisis de observaciones primarias.
- **TRACK** es un programa de posicionamiento cinemático GPS de fase diferencial. **TrackRT** es un programa de procesamiento cinemático GPS en tiempo real.

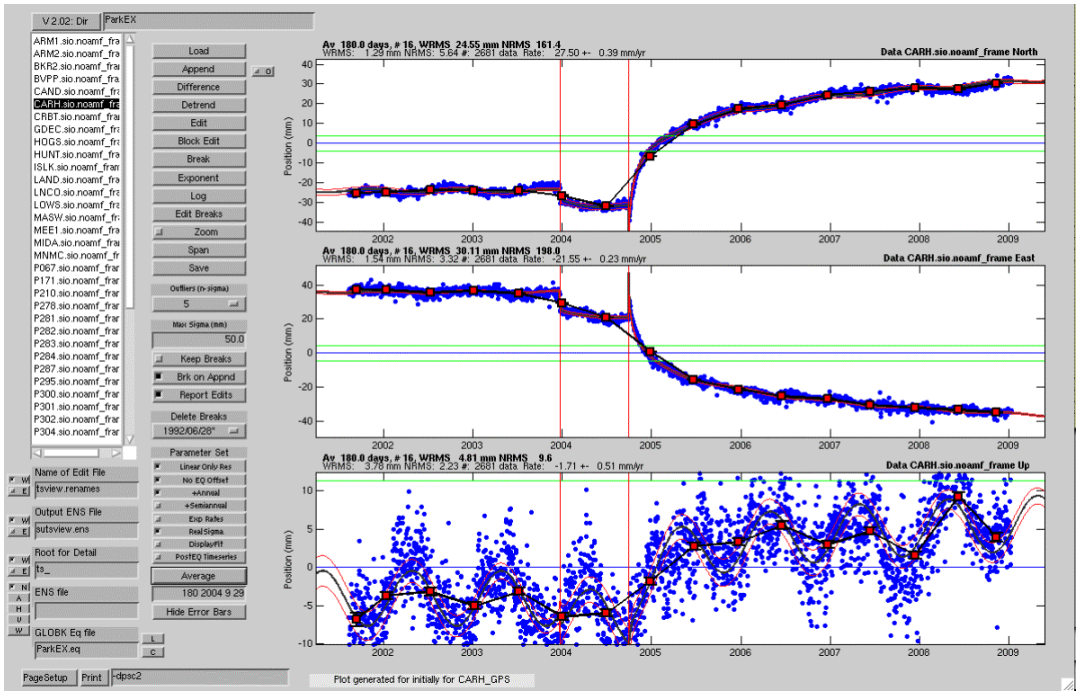


Figura 2. *Interfaz del programa Gamit/Globk/Track. [fuente: gpsg.mit]*

- **GIPSY-OASIS II (Jet Propulsion Laboratory – NASA)** *[fuentes: Unavco y Nagarvil]*

Paquete de software para el procesamiento de datos GPS automatizado de alta precisión.

Características de GOA II:

- Precisión a nivel centimétrico (tierra y espacio).
- Operaciones de bajo costo sin vigilancia automatizadas.
- Análisis de capacidades innovador de GPS y non-GPS.
- Capacidad en tiempo real (con RTG - Real-Time GIPSY)
- Filtro único / más suave es una de las capacidades de estimación GPS y de la precisión.

- Adaptable a: orbitadores non-GPS; programas que no son de la NASA (FAA, militar, comercial).

Con GOA II se puede hacer:

- Determinación de la órbita - naves espaciales individuales o constelaciones de satélites.
- Posicionamiento preciso y en el momento: tierra, mar y aire
- Operaciones automatizadas (no vigilado).
- Capacidad en tiempo real.
- Capacidad en tiempo real con RTG

Figura 3. *Web del programa Gipsy-Oasis II. [fuente: ggenluz]*

Éstos son los programas más empleados en la actualidad para el tratamiento en post-proceso de observaciones GNSS. Sin embargo, como se ha comentado, se pueden encontrar librerías de scripts de código abierto que llevan programadas rutinas para el post-proceso de la información geográfica. Entre ellas encontramos:

- **The GPS Easy Suite–Matlab code for the GPS newcomer, Kai Borre:** *[fuente: Artano]*
  The Easy Suite es una colección de scripts de Matlab que tienen como finalidad comprender los aspectos básicos del sistema GPS. Vienen dados en dos colecciones:
  - **GPS Easy Suite I (2003) (10 scripts):** Permiten realizar operaciones básicas como calcular la posición de los satélites con efemérides transmitidas, cambios de formato de fechas, cálculo de coordenadas a partir de pseudodistancias etc. En ellos se encuentra la rutina que se va a emplear en el presente trabajo:
    - *"easy3.m"*. Cálculo de la posición del receptor mediante posicionamiento absoluto con código (SPP - standard point positioning).
  - **GPS Easy suite II (8 scripts):** Realizan cálculos más complejos, como detección de saltos de ciclos, cálculos de retardo ionosférico, líneas base a partir de mediciones de fase etc.

Figura 4. *Logo de la librería The Easy Suite. [fuente: EasySuiteII]*

1.4 Entorno de programación

Dado que la rutina a emplear viene programada en código Matlab, en este apartado se va a comentar cómo funciona el entorno de programación de Matlab, software que va a ser empleado para el desarrollo de este trabajo.

- **MATLAB:** *[fuente: neredia]*

  MATLAB es un entorno de programación y cálculo científico de altas prestaciones para cálculo numérico y visualización. Acrónimo de ``MATrix LABoratory'' (Laboratorio de Matrices).
  Fue creado en sus inicios con el fin de proporcionar un acceso fácil al software matricial creado en los proyectos LINPACK y EISPACK, los cuales son los programas más avanzados en cálculo matricial.
  Es un sistema interactivo de trabajo que emplea como elemento básico las matrices sin necesidad de dimensionamiento, lo cual permite resolver problemas de manera más rápida que con otros lenguajes de programación.
  MATLAB proporciona cajas de herramientas denominadas TOOLBOXES, las cuales contienen conjuntos de funciones que se utilizan para resolver diversidad de problemas.

  Para simplificar el funcionamiento en cuanto a programación en el entorno de MATLAB, se van a diferenciar los dos tipos de scripts que se van a tratar y utilizar en este trabajo. Entre ellos encontramos:

  - Programas:
    Son scripts que recogen todos los procedimientos de cálculo deseados, y que se ejecutan directamente desde la ventana de comandos de MATLAB. Los datos de entrada para el cálculo pueden ser cargados desde archivos, pedir por pantalla o estar incluidos en el mismo código. La extensión de los scripts es *".m"*, formato propio de MATLAB.
  - Funciones:
    Muy similares a los programas. Ambos son scripts solo que en este caso especial los datos y valores son definidos previamente, y al llamar a la función es necesario introducirle los datos necesarios para que realice los cálculos oportunos. Se emplean para reducir y reutilizar código. La extensión de los scripts es *".m"*, formato propio de MATLAB.

Figura 5. *Logo del programa Matlab. [fuente: Mathworks]*

2. Fundamentos GNSS

En los siguientes apartados se van a definir a modo resumen los fundamentos y teorías en los que se basa la tecnología GNSS. Se van a desarrollar también conceptos que son empleados en este trabajo, para que el lector entre un poco en contexto con el funcionamiento general del GNSS antes de pasar a la explicación de la metodología empleada en el mismo.

La mayor parte de la información de los siguientes apartados ha sido extraída a modo resumen del libro *GNSS. GPS: fundamentos y aplicaciones en Geomática* [1].

2.1 Introducción

Según la ESA (*Navipedia*) se define GNSS como:

> *"GNSS (Sistema Global de Navegación por Satélite) es el término genérico estándar para sistemas de navegación por satélite que proporcionan un posicionamiento geoespacial autónomo con cobertura global. Este término incluye, por ejemplo, GPS, GLONASS, Galileo, Beidou y otros sistemas regionales. "*

A modo resumen, el GNSS es la tecnología que permite posicionarnos en la superficie terrestre. A partir de observaciones a satélites, se pueden obtener las coordenadas de un punto en la superficie.

El fundamento del GNSS es obtener la posición de los satélites, es decir, obtener sus coordenadas en un sistema de referencia concreto, con el fin de poder calcular a partir de ellas la posición del receptor.

Por tanto, para poder emplear esta tecnología se necesita definir las órbitas de los satélites con el fin de poder conocer su posición, la cual viene definida en las efemérides, y un sistema de referencia con el que trabajar. En este caso, el sistema de referencia empleado es WGS84, coordenadas cartesianas geocéntricas (ECEF).

El sistema de referencia ECEF queda definido como:

- Origen: Centro de la Tierra (geocéntrico).
- Plano Fundamental: Ecuador terrestre.
- Eje vertical: Perpendicular al plano fundamental en dirección al norte geográfico.
  Eje principal: definido por la dirección del centro de la Tierra al punto Aries, dirección que une el centro de la Tierra con el Sol en el equinoccio Vernal.

2.2 Teoría de orbitas

Como se ha mencionado en el apartado anterior, se deben definir las órbitas de los satélites con el fin de poder conocer su posición en el espacio. Dicha teoría se basa en las tres leyes de Kepler y en la ley de gravitación universal de Newton.

Leyes de Kepler:

- Primera ley de Kepler: Los planetas describen órbitas elípticas alrededor del Sol, siendo éste uno de sus focos.

- Segunda ley de Kepler: Los planetas en su recorrido elíptico barren áreas iguales en tiempos iguales.
- Tercera ley de Kepler: Los cuadrados de los periodos de revolución son proporcionales a los cubos de las distancias medias de los planetas al sol.

Estas leyes tienen como fundamento definir la forma de la elipse, que vendrá dada por su semieje mayor y la excentricidad, teniendo como uno de sus focos a la Tierra (1ª ley de Kepler); determinar el vector posición del satélite (2ª ley de Kepler) y determinar la velocidad del satélite, que disminuye con la altura (3ª ley de Kepler).

A dichas leyes hay que añadirles la ley de Gravitación y los efectos que sufren las órbitas por las perturbaciones gravitacionales, presión e irregularidades del campo gravitatorio.

Ley de gravitación Universal:

- Un objeto material del universo atrae a otro objeto con una fuerza directamente proporcional al producto de sus masas e inversamente proporcional al cuadrado de la distancia que los separa. Se considera que la masa de la Tierra queda concentrada en su centro de masas, se desprecia la atmósfera y que solo está sometida a la fuerza gravitatoria.

Por consiguiente, siguiendo las leyes de Kepler, la posición de un satélite viene determinada por seis parámetros:

- Tamaño de la órbita:
  - **a:** Semieje mayor, define el tamaño de la elipse.
  - **e:** Excentricidad, define la forma. (Permite obtener el semieje menor **b**).
- Plano orbital:
  - **Ω:** Ascensión del nodo, indica la dirección de la elipse.
  - **I:** Inclinación.
- Orientación de la órbita en el plano:
  - **w:** Argumento del perigeo, sirve para situar el satélite a lo largo del tiempo.
- Tiempo de paso por el perigeo t_0:
  - **V:** Anomalía, indica la posición instantánea dentro de la órbita.

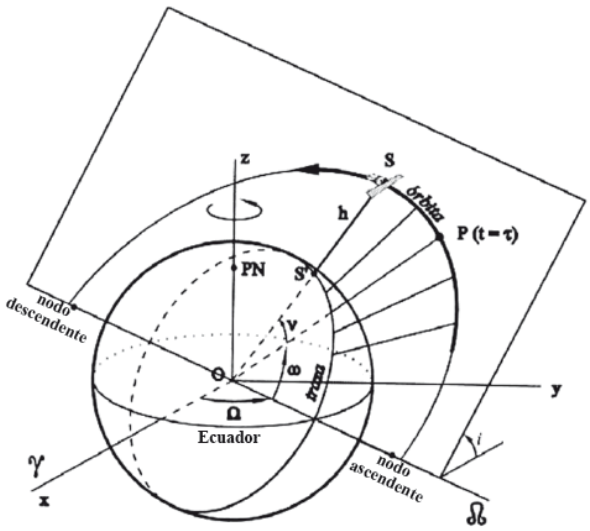


Figura 6. *Parámetros orbitales de un satélite. [fuente: scielo]*

Con todos estos parámetros orbitales se va a poder calcular la posición de los satélites, los cuales transmiten dicha información en tiempo real (efemérides), como parte del mensaje. Estos parámetros se pueden obtener a posteriori a partir de las distintas instituciones que trabajan en el tema del GNSS, y son las llamadas efemérides precisas.

En siguientes apartados se va a explicar el funcionamiento de las redes de estaciones permanentes, con el fin de ejemplificar el funcionamiento del GNSS, ya que en este trabajo se va a trabajar con datos observados desde una estación; y se van a explicar los ficheros de efemérides, donde se hará reseña a la parte teórica desarrollada en este apartado.

Para mayor detalle consultar el *Capítulo 3. Órbitas de los Satélites* [1].

2.3 Mediciones GNSS. Estaciones permanentes

En este apartado se van a desarrollar los conceptos básicos del posicionamiento GNSS, y posteriormente se va a proceder a comentar el funcionamiento de las redes GNSS y de las estaciones permanentes.

2.3.1 Posicionamiento GNSS

El posicionamiento GNSS es un sistema pasivo donde el receptor capta la señal del satélite decodificada, a partir de la cual calcula la posición del mismo. Conocidas las posiciones de los satélites observados en el momento de la toma de datos (efemérides), para obtener la posición del receptor será suficiente con medir las distancias entre el receptor y los satélites.

Cada medición (observable) es la distancia entre el satélite y el receptor y se basa en la propagación de las ondas electromagnéticas. Considerando que la señal se propaga en el vacío a la velocidad de la luz ($3*10^5$ Km/s) se puede obtener la distancia conocido el tiempo de viaje de la señal, método conocido como cálculo de la pseudodistancia. Debido a que no es fácil determinar el tiempo de la señal, para mediciones de mayor precisión se emplea otro método.

Por tanto, se van a definir dos métodos de cálculo de la distancia, medición en código y medición en fase.

- **Medición en código:** Se determina el tiempo transcurrido entre emisión de la señal y recepción de la misma. Para determinarlo, el satélite emite una determinada marca de forma que, cuando llega la señal que ha emitido el satélite, las compara y determina el incremento de tiempo que ha tardado en recibirla.
- **Medición en fase:** La medida de la distancia puede calcularse midiendo el nº entero N de longitudes de onda y la parte no entera, como se muestra en la figura siguiente, y a partir de ello calcular la distancia. No es un método fácil ya que la determinación de N no es sencilla, problema que se conoce como determinación de ambigüedades.

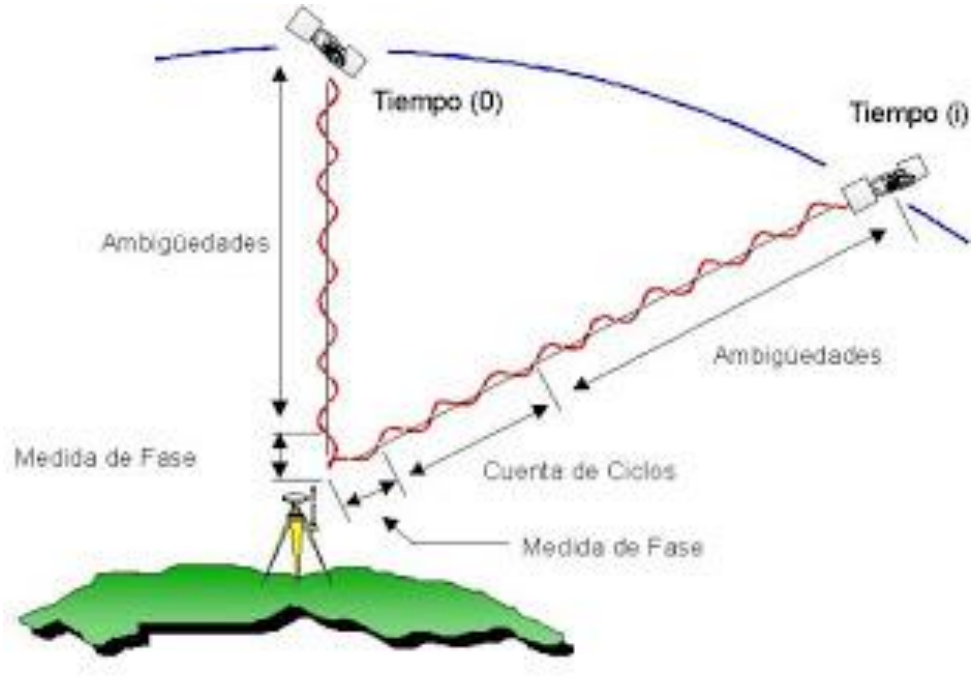


Figura 7. *Esquema de las mediciones GNSS. [fuente: notasdegeodesia]*

En resumen, conocida la distancia entre el receptor y los satélites se va a poder calcular la posición del receptor, conocidas previamente las coordenadas de los satélites observados en el momento de la medición.

Para mayor detalle consultar el *Capítulo 4. GNSS. Sistema Global de Navegación por Satélite* [1].

2.3.2 Redes de Estaciones Permanentes

Una de las aplicaciones de la tecnología GNSS son las Redes de Estaciones Permanentes. Éstas son un conjunto de receptores GNSS que están instalados de forma fija en puntos de coordenadas conocidas. Están conectados a una red que da servicios en tiempo diferido o en tiempo real, tanto datos brutos como datos tratados. Su propósito es realizar mediciones de forma continua a los satélites. Se emplean, entre otras cosas, como redes geodésicas.

En la actualidad existen distintas redes de estaciones permanentes a diferente escala; continentales (Red EUREF – Europa), estatales (Red ERGNSS- España), autónomas (Red ARAGEA – Aragón) etc. Para la metodología seguida se ha trabajado con datos de la Red ERVA (Comunidad Valenciana), que cuenta con varias estaciones permanentes distribuidas por toda la comunidad (red autonómica).

- **Red ERVA:**

  La Red de Estaciones de Referencia de Valencia tiene como fin facilitar datos GNSS mediante una red de medición continua y posicionamiento por satélite. Es una herramienta imprescindible para vuelos fotogramétricos, cartografía, geodesia, navegación...

  En la siguiente imagen se pueden apreciar las distintas estaciones que se encuentran distribuidas alrededor de toda la Comunidad Valenciana.

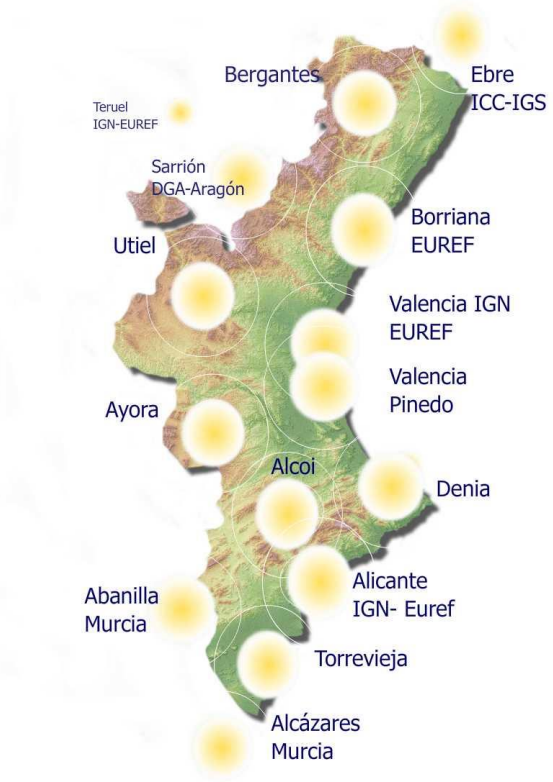


Figura 8. *Mapa distribución de la red ERVA. [fuente: gvacartografic]*

En este trabajo se van a emplear datos de una estación permanente, concretamente de la estación de VCIA (Pinedo), por tanto, es importante conocer a qué red pertenecen, como funciona y qué organismo la gestiona.

Para mayor detalle sobre redes permanentes y sobre otras redes consultar el *Capítulo 16* [1].

2.4 Ficheros de datos. Observables y efemérides

Hay que diferenciar dos tipos de datos con los que se va a trabajar, por un lado están las efemérides transmitidas (ficheros RINEX) y por otro lado están las efemérides precisas.

A continuación se va a explicar cada uno de los tipos mencionados y los ficheros asociados a cada dato.

2.4.1 Efemérides Transmitidas

Son datos que nos permiten determinar la posición del satélite. Los datos vienen en el mensaje de navegación, por tanto, son instantáneos y se obtienen a partir de la observación.

El formato de intercambio de dicha información son los ficheros RINEX, los cuales contienen los observables, los datos de navegación y los datos atmosféricos.

- **Ficheros RINEX de observación:**

En este archivo se encuentran los tres observables básicos (Doppler, Pseudodistancia y fase) y el término tiempo.

```
     2.11           OBSERVATION DATA    M (MIXED)           RINEX
VERSION / TYPE
teqc  2011Oct11     IGN-E               20170202 02:24:30UTCPGM / RUN
BY / DATE
Linux 2.4.21-27.ELsmp|Opteron|gcc -static|Linux x86_64|=+   COMMENT
BIT 2 OF LLI FLAGS DATA COLLECTED UNDER A/S CONDITION       COMMENT
VALE                                                        MARKER
NAME
13439M001                                                   MARKER
NUMBER
Area de Geodesia    Instituto Geografico Nacional           OBSERVER /
AGENCY
1700071             LEICA GR10          3.11/6.403          REC # /
TYPE / VERS
10190012            LEIAR25.R3      LEIT                    ANT # /
TYPE
  4929534.0457   -29050.6755  4033709.9250                  APPROX
POSITION XYZ
        3.0390        0.0000        0.0000                  ANTENNA:
DELTA H/E/N
     1     1                                                WAVELENGTH
FACT L1/2
     8    L1    L2    L5    C1    C2    P2    S1    S2      # / TYPES
```

Figura 9. *Cabecera de un RINEX de observación. [fuente: propia]*

- **Ficheros RINEX de navegación:**

Contiene los datos orbitales, los parámetros del reloj y la precisión de las medidas de pseudodistancia de los satélites observados. Puede contener opcionalmente datos como los parámetros del modelo ionosférico de Klobuchar.

```
     2.10           NAVIGATION DATA                         RINEX
VERSION / TYPE
    GPSNet 2.62 3098                                        PGM / RUN
BY / DATE
    8.3819D-09 -7.4506D-09 -5.9605D-08  5.9605D-08          ION ALPHA
    8.8064D+04 -3.2768D+04 -1.9661D+05  1.9661D+05          ION BETA
    0.000000000000D+00 2.664535259100D-15   503808     1934 DELTA-UTC:
A0,A1,T,W
    18                                                      LEAP
SECONDS
                                                            END OF
HEADER
```

Figura 10. *Cabecera de un RINEX de navegación. [fuente: propia]*

2.4.2 Efemérides Precisas

Son determinaciones orbitales XYZ de alta precisión, que se transmiten ya en coordenadas cartesianas geocéntricas. Se emplean los datos de pseudodistancia y fase registrados por estaciones permanentes, siendo el proceso de cálculo inverso al del GNSS, se calculan las coordenadas de los satélites a partir de las coordenadas de la Tierra, siendo éstas muy precisas.

- **Ficheros *".sp3"*:**

Las efemérides precisas son facilitadas en ficheros *".sp3"*. El formato básico del archivo es una cabecera, seguido de una serie de épocas que contienen los registros de posición y de reloj para cada satélite que aparece en el encabezado.

En este trabajo se han empleado ficheros de efemérides precisas de solución final, para la constelación de NAVSTAR (GPS), donde aparecen las coordenadas de los 32 satélites cada 15 minutos durante un día completo.

```
#cP2017  2  1  0  0  0.00000000      96 ORBIT IGS14 HLM  IGS
## 1934 259200.00000000   900.00000000 57785 0.0000000000000
+   32   G01G02G03G04G05G06G07G08G09G10G11G12G13G14G15G16G17
+        G18G19G20G21G22G23G24G25G26G27G28G29G30G31G32  0  0
+          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
+          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
+          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
++         2  2  1  0  2  2  2  2  2  2  2  2  2  2  2  2
++         2  2  2  2  2  2  2  2  2  2  2  2  2  2  0  0
++         0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
++         0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
++         0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
%c G  cc GPS ccc cccc cccc cccc cccc ccccc ccccc ccccc ccccc
%c cc cc ccc ccc cccc cccc cccc cccc ccccc ccccc ccccc ccccc
%f  1.2500000  1.025000000  0.00000000000  0.000000000000000
%f  0.0000000  0.000000000  0.00000000000  0.000000000000000
%i    0    0    0    0      0      0      0      0         0
%i    0    0    0    0      0      0      0      0         0
/* FINAL ORBIT COMBINATION FROM WEIGHTED AVERAGE OF:
/* cod emr esa gfz grg jpl mit ngs sio
/* REFERENCED TO IGS TIME (IGST) AND TO WEIGHTED MEAN POLE:
/* PCV:IGS14_1930 OL/AL:FES2004  NONE     Y  ORB:CMB CLK:CMB
*  2017  2  1  0  0  0.00000000
```

Figura 11. *Cabecera de un fichero ".sp3". [fuente: propia]*

- **Ficheros *".clk"*:**

Contienen la información del estado del reloj de cada uno de los satélites en una época concreta.

```
     3.00           C                                       RINEX
VERSION / TYPE
CCLOCK              IGSACC @ GA and MIT                     PGM /
RUN BY / DATE
GPS week: 1934   Day: 3   MJD: 57785                        COMMENT
THE COMBINED CLOCKS ARE A WEIGHTED AVERAGE OF:              COMMENT
  cod emr esa gfz grg                                       COMMENT
THE FOLLOWING REFERENCE CLOCKS WERE USED BY ACs:            COMMENT
  TWTF AMC2 BRUX WAB2                                       COMMENT
THE COMBINED CLOCKS ARE ALIGNED TO GPS TIME                 COMMENT
USING THE SATELLITE BROADCAST EPHEMERIDES                   COMMENT
All clocks have been re-aligned to the IGS time scale: IGST COMMENT
    18                                                      LEAP
SECONDS
     2    AR    AS                                         # /
TYPES OF DATA
IGS  IGSACC @ GA and MIT                                    ANALYSIS
CENTER
```

Figura 12. *Cabecera de un fichero ".clk". [fuente: propia]*

- **Interpolación de Lagrange:**

En el presente trabajo uno de los objetivos, como se ha comentado, es obtener las coordenadas de los satélites cada 30 segundos. Debido a que los ficheros de efemérides precisas solo facilitan las posiciones de los satélites cada 15 minutos, es preciso realizar una interpolación entre épocas para poder obtener la posición de los mismos cada 30 segundos. Para ello, la mejor forma es aplicando el polinomio de Lagrange, el cual nos dará una solución muy aproximada de la órbita del satélite, donde la incógnita del polinomio será la variable tiempo, la cual se podrá sustituir para obtener la posición del satélite en el momento deseado.

En este apartado se va a explicar el procedimiento teórico del polinomio. Más adelante, en la aplicación de la metodología, se expondrá el problema planteado para las efemérides precisas y cómo el polinomio lo resuelve, así como un pequeño ejemplo numérico a modo de demostración del funcionamiento del polinomio.

Por tanto, asumimos valores funcionales (t_j) que definen las épocas:

$t_j, j = 0,1, \dots, n$ Entonces:

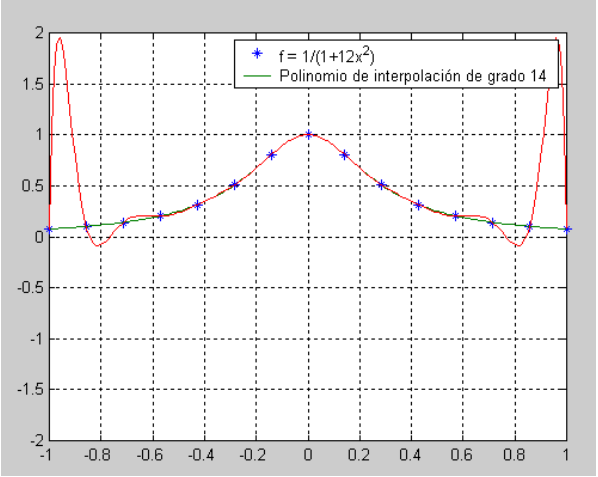
$$l_j(t) = \frac{(t-t_0)(t-t_1)\dots(t-t_{j-1})(t-t_{j+1})\dots(t-t_n)}{(t_j-t_0)(t_j-t_1)\dots(t_j-t_{j-1})(t_j-t_{j+1})\dots(t_j-t_n)} \quad ecuación\ 1.1$$

Es la definición de las funciones de correspondencia base ($l_j(t)$) de grado "n" relacionado con una época "t" arbitraria.

El valor funcional interpolado en la época "t" sigue el sumatorio:

$$f(t) = \sum_{j=0}^{n} f(t_j) * l_j(t) \quad ecuación\ 1.2$$

Ésta es la definición matemática del polinomio. En la siguiente figura se puede apreciar un ejemplo del comportamiento del polinomio. En las partes extremas el polinomio aparece sesgado, pero en la parte central el polinomio se aproxima bastante a la función dada. También se puede observar que el polinomio siempre coincide en los puntos utilizados para la definición del mismo $f(t_j)$.



Gráfica 1. *Comportamiento de la función de interpolación de Lagrange. [fuente: uam]*

Éste apartado ha sido extraído del libro de Hoffmann [2]. (Págs. 52-53), y ha sido traducido para el presente trabajo.

2.4.3 Precisión de las efemérides

En la siguiente tabla aparecen los distintos tipos de soluciones de efemérides que existen, junto a sus precisiones y el intervalo de tiempo de observación de cada una de ellas. Aparecen tanto para GPS como para GLONASS. Además también se puede apreciar la precisión de los datos del estado del reloj asociado a cada solución y su intervalo de tiempo asociado.

Products (delay)	**Broadcast** (Real-Time)	**Ultra-Rapid** **Predicted** (Real-Time)	**Ultra-Rapid** **Observed** (3-9 h)	**Rapid** (17-41 h)	**Final** (12-18 d)
Orbit GPS (sampling)	∼100 cm (∼2 h)	∼5 cm (15 min)	∼3 cm (15 min)	∼2.5cm (15 min)	∼ 2.5 cm (15 min)
GLONASS (sampling)					∼5 cm (15 min)
Clock GPS (sampling)	∼5 ns (daily)	∼3 ns (15 min)	∼150 ps (15 min)	∼75 ps (5 min)	∼75 ps (30 sec)
GLONASS (sampling)					∼ TBD (15 min)

Tabla 1. *Precisiones de los tipos de efemérides. [fuente: navipedia]*

3. Metodología

Se va a proceder a explicar la metodología empleada para el desarrollo del trabajo. Este apartado se va a dividir en dos grandes bloques:

- Desarrollo de la interpolación de Lagrange: Se van a explicar los procedimientos seguidos para el desarrollo de scripts que calculen la interpolación de Lagrange para efemérides precisas.
- Modificación de la rutina de *"easy3"*: Se va a explicar cómo se han modificado los distintos scripts que forman parte de la rutina para que funcionen con ficheros RINEX de la actualidad. También se ha modificado el programa para que calcule las coordenadas de una estación en distintas épocas y en distinto periodo de tiempo al programado inicialmente por *Kai Borre*, con ambas soluciones (efemérides transmitidas y precisas, estas últimas necesitan de la interpolación).

3.1 Datos de partida

Se va a trabajar con los datos de la estación de *"VCIA"*, perteneciente a la red de estaciones permanentes *ERVA*, a día 01-02-2017, por tanto, ficheros de partida son:

Ficheros RINEX: [*fuente de descarga: ICV*]

- ***VCIA0320.17d*:** Fichero que contiene las observaciones del día seleccionado. Viene comprimido en *Hatanaka*, por tanto, se debe descomprimir con los ejecutables de *Hatanaka* para obtener el fichero *VCIA0320.17o*, que ya será válido para trabajar con él.
- ***VCIA0320.17n*:** Fichero de navegación que contiene los parámetros orbitales de los satélites (efemérides transmitidas).

Ficheros IGS: [*fuente de descarga: igs*]

- ***igs19342.sp3, igs19343.sp3, igs19344.sp3*:** Ficheros de efemérides precisas. Contienen las coordenadas de los satélites en solución final. Se corresponden con los días 31-01-2017, 01-02-2017 y 02-02-2017 respectivamente, con un intervalo de 15 minutos cada época.
- ***igs19343.clk_30s*:** Fichero que contiene los valores del estado del reloj satélite a satélite en un intervalo de actualización de 30 segundos. Se corresponde con el día 01-02-2017.

3.2 Interpolación de Lagrange

Como se ha explicado en el apartado 2.4.2, las efemérides precisas vienen dadas ya en coordenadas cada 15 minutos durante un día completo. Se necesita por tanto una forma de poder obtener las coordenadas de los satélites en distintas épocas del día. Para ello se va a aplicar la interpolación por el método de polinomios de Lagrange, que nos devolverá una función que interpola la posición de los satélites respecto de la variable tiempo, por tanto, a partir de ella se podrán obtener las posiciones de los satélites en cualquier época del día.

A continuación se va a desarrollar un pequeño ejemplo práctico del funcionamiento y comportamiento de la interpolación por el método de Lagrange:

- Se va a aplicar la metodología de Lagrange teniendo como datos de partida tres valores procedentes de una función cualquiera, por tanto, se va a interpolar a partir de tres épocas distintas. La función resultante será, en este caso, una función de segundo grado. Los datos de partida son: (Ver formulación en apartado 2.4.2)

$$f(t_0) = f(-3) = 13$$
$$f(t_1) = f(1) = 17$$
$$f(t_2) = f(5) = 85$$

Dados los valores de la función para las tres épocas mencionadas (t_0, t_1, t_2) respectivamente, se calculan los polinomios base ($l_j(t)$ ecuación 1.1) para las distintas épocas:

$$l_0(t) = \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} = \frac{1}{32}(t^2 - 6t + 5)$$

$$l_1(t) = \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} = -\frac{1}{16}(t^2 - 2t - 15)$$

$$l_2(t) = \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} = \frac{1}{36}(t^2 + 2t - 3)$$

Por tanto si aplicamos el sumatorio (ecuación 1.2), el resultado queda:

$$f(t) = 2t^2 + 5t + 10$$

Con esta función resultante se puede interpolar a lo largo del tiempo y obtener resultados muy próximos y precisos a los que la función inicial $f(t_0)$ nos daría.
Como se ha comentado en el apartado de teoría la función se sesga en los extremos de la misma, pero en las partes centrales se aproxima bastante, donde los puntos de partida de la función inicial a la que se quiere aproximar siempre coinciden.
Esto último servirá para tener como punto de control si la función de interpolación obtenida para los satélites es buena o no, pues cada 15 minutos la posición de los satélites deberá coincidir con la posición dada inicialmente. También se comprobará y demostrará que la función se sesga en algún caso en las épocas iniciales y finales. (Ejemplo extraído de Hoffmann [2])

Ejemplificado el procedimiento a utilizar, se va a aplicar lo mismo para la interpolación de los satélites, con el fin de poder obtener la posición de los 32 satélites que componen la constelación de *"NAVSTAR"* en las mismas épocas que vienen dadas las observaciones del receptor (día completo cada 30 segundos).

La metodología seguida para este trabajo ha conllevado la realización de tres scripts que realicen la interpolación, los cuales se van adaptando a las necesidades que surgen a lo largo del desarrollo de la misma.

3.2.1 Programa "Inter_Lagrange.m"

Inicialmente se deseó obtener un programa que calculara la interpolación de Lagrange durante un día completo, es decir, elaborar un script que calculara la función interpoladora cada 30 segundos durante 24 horas.

Para ello es necesario partir de los datos de efemérides precisas de tres días seguidos (31 de enero, 1 y 2 de febrero), donde el día central será el día seleccionado para obtener la función interpoladora. Esto es debido a que, como se ha comentado anteriormente, la formulación de Lagrange obtiene funciones que se sesgan al inicio y final de las mismas, por tanto, será necesario partir de más datos iniciales y finales de efemérides para que la función resultante no se sesgue en los extremos y el día seleccionado quede centrado en la función resultante. Para ello el programa leerá los datos del día seleccionado completo, las 12 últimas horas del día anterior y las 12 primeras horas del día siguiente.

Estructura y funcionamiento del script:

- El programa se inicia leyendo los tres ficheros de efemérides precisas (*".sp3"*). Debido a que Matlab está orientado a trabajar con matrices y los ficheros de efemérides no tienen estructura de matriz, se ha programado que, una vez cargados los ficheros, el script almacena los datos que se necesitan para la interpolación. Dichos archivos tendrán la estructura que se adapte mejor para el cálculo. Los archivos son:
    - ***"MatrizCoorSat.txt"*:** Archivo de tipo texto que almacena las coordenadas de los 32 satélites GPS de forma continuada. El programa lee las coordenadas desde las 12 horas del 31-01-2017 hasta las 23:45 horas del mismo día, las coordenadas del día 01-01-2017 de todo el día completo y las coordenadas del día 02-01-2017 desde las 0 horas hasta las 12 horas; y almacena éstas en dicho fichero, quedando así el día deseado centrado. Su estructura queda:

MatrizCoorSat.txt* | Epocas.txt

1	PG01	-3481.577306	16924.983886	-20202.695993	48.153681	6	4	6	78
2	PG02	21407.005964	-14847.669058	4555.269538	484.097453	9	5	6	87
3	PG03	6077.873678	19154.741882	-17386.654150	-106.930139	3	3	7	91
4	PG04	-22126.868548	-4874.597253	13866.042656	999999.999999				
5	PG05	13454.327600	-9097.282784	20961.348918	-63.215765	2	4	4	80
6	PG06	24710.946997	-5374.574600	-8107.204123	306.418033	6	5	6	92
7	PG07	11929.615823	10925.467401	21299.128000	387.271587	5	7	6	96
8	PG08	-5397.304378	25518.038866	4856.240477	-47.928579	6	4	4	82
9	PG09	9154.667400	19838.115894	15068.159987	294.346745	4	4	6	78
10	PG10	-23498.345205	-9037.815124	-8607.274308	-109.776198	5	6	3	103
11	PG11	-5889.155551	22347.748672	-13686.218280	-668.072675	6	8	5	75
12	PG12	-3650.898604	-18620.188777	-18818.852788	388.534990	5	4	5	84
13	PG13	12251.026628	-20575.340000	11422.534821	-67.619038	5	5	5	90
14	PG14	-15445.606628	6229.254976	-20393.294770	-56.720739	9	4	6	84
15	PG15	3092.667738	-25932.447248	3427.732992	-341.999830	6	5	7	59
16	PG16	-13902.592486	6849.738778	21523.255412	24.856810	8	4	3	125
17	PG17	14265.803786	3677.258916	-21749.241453	-192.668000	10	3	7	87
18	PG18	-21246.298970	-16005.371384	2638.832546	595.643269	10	6	5	101
19	PG19	15549.162015	-4585.111192	-21354.258396	-514.377485	10	5	8	81
20	PG20	-4157.887403	-18403.526705	18537.534000	457.980585	4	3	4	64
21	PG21	-16893.260156	-7704.852097	19828.832846	-523.315025	7	4	4	90
22	PG22	-526.645737	17427.055269	-19777.102361	115.605962	4	5	6	76
23	PG23	639.464943	26067.795038	3836.276083	-204.150257	8	3	6	78
24	PG24	6040.315260	-17981.289349	-18572.817317	-29.815409	6	4	4	110
25	PG25	-16271.986095	-19076.090754	-9214.501571	-327.642282	5	5	6	84
26	PG26	-20855.371568	-291.297393	16495.478032	-519.145228	8	3	6	98
27	PG27	-12194.543618	17449.262671	15717.063434	202.336778	6	5	3	94
28	PG28	22023.780541	13489.178774	-5567.088370	586.240633	9	6	6	84
29	PG29	-5757.977484	-22279.957733	13292.907837	629.752750	5	6	2	73
30	PG30	21943.339568	2792.079000	14781.488032	160.578489	4	3	4	71
31	PG31	-25269.451533	5921.576596	-6468.760692	221.849120	9	6	6	115
32	PG32	-14945.852350	-3766.596384	-21624.793641	-302.570056	4	5	3	76
33	PG01	-5524.199156	17812.116566	-18915.679612	48.154389	6	4	5	102
34	PG02	21835.814895	-14721.725366	1739.897748	484.091431	9	5	6	98
35	PG03	4263.521857	18134.123494	-18949.311724	-106.930668	3	3	7	102
36	PG04	-23254.367276	-5830.056733	11510.636768	999999.999999				
37	PG05	15415.263482	-7763.245129	20152.767892	-63.213891	4	4	4	83

Figura 13. *Fichero "MatrizCoorSat.txt". [fuente: propia]*

Esta figura muestra las primeras líneas del archivo obtenido, en azul se puede observar que, una vez almacenada la posición de los 32 satélites en una época determinada, se almacena seguidamente la posición de los mismos en la época siguiente.

- ***"Epocas.txt"*:** Archivo de tipo texto que almacena la fecha de las épocas correspondientes a las coordenadas almacenadas en el fichero anterior, en el mismo orden. El programa lee todas las épocas deseadas y cuando detecta un cambio de época almacena los datos de ésta en dicho archivo. Su estructura queda:

MatrizCoorSat.txt | Epocas.txt

1	2017	1	31	12	0	0.00000000
2	2017	1	31	12	15	0.00000000
3	2017	1	31	12	30	0.00000000
4	2017	1	31	12	45	0.00000000
5	2017	1	31	13	0	0.00000000
6	2017	1	31	13	15	0.00000000
7	2017	1	31	13	30	0.00000000
8	2017	1	31	13	45	0.00000000
9	2017	1	31	14	0	0.00000000
10	2017	1	31	14	15	0.00000000
11	2017	1	31	14	30	0.00000000
12	2017	1	31	14	45	0.00000000
13	2017	1	31	15	0	0.00000000
14	2017	1	31	15	15	0.00000000
15	2017	1	31	15	30	0.00000000
16	2017	1	31	15	45	0.00000000
17	2017	1	31	16	0	0.00000000

Figura 14. *Fichero "Epocas.txt". [fuente: propia]*

- Una vez almacenados los datos necesarios, se procede a la programación de la metodología de interpolación. Para ello, previamente se han de convertir las épocas almacenadas a una misma unidad, en este caso, todo a segundos.

```
for i=1:m
    % Crea una matriz de tiempos (en segundos)
    if i<=48 % Para las epocas del dia anterior
        hora = (tmp(i,4)-24)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    elseif i>=145 % Para las epocas del dia siguiente
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60 + tmp(146,5)*60;
        a(i,1) = a(144,1) + minutos + hora;
    else % Para las epocas del dia
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    end
end
```

Figura 15. *Bucle de conversión de tiempos. [fuente: propia]*

Como se puede apreciar en la figura anterior, el programa lee las épocas almacenadas y las convierte a segundos, almacenando los valores resultantes en una matriz de tiempos. En este caso particular los tiempos se adaptan a las necesidades del programa. Se parte de que el segundo 0 es la primera época del día que se desea interpolar, por consiguiente, las épocas pertenecientes al día anterior vendrán dadas de forma que se restarán los segundos que tiene la época en cuestión a los segundos que tiene un día, quedando estos valores en negativo y, por tanto, por el otro lado, las épocas pertenecientes al día siguiente vendrán dadas como la suma de los segundos de éstas más los segundos que tiene un día.

-5400	81000
-4500	81900
-3600	82800
-2700	83700
-1800	84600
-900	85500
0	86400
900	87300
1800	88200
2700	89100
3600	90000
4500	90900
5400	91800

En estas figuras se puede apreciar lo explicado anterior mente. Anteriormente la época 0 segundos el valor del tiempo viene dado como la época (en segundos) menos la duración de un día (86.400 segundos).

Lo mismo para las épocas del día siguiente, donde se suman los segundos de un día al valor de la época (en segundos)

Se puede comprobar a simple vista que es correcto ya que, si las efemérides vienen dadas cada 15 minutos, que equivalen a 900 segundos, cada época será un múltiplo de $\pm$ 900.

Figura 16. *Matriz de tiempos [fuente: propia]*

- Con los datos del tiempo (en segundos) preparados, ya se puede proceder a la interpolación del polinomio de Lagrange. Volviendo al apartado de teoría, los valores de las coordenadas serán $f(t_n)$, y los valores de las épocas serán las (t_n) con las que se obtendrán los polinomios $l_n(t)$. La estructura será elaborar un bucle que lea las coordenadas (X, Y, Z) de cada satélite y elabore tres funciones de interpolación (una para la coordenada X, otra para la Y y otra para la Z), donde, tras obtenerlas, se hará una sustitución de la variable tiempo cada 30 segundos, obteniendo así la posición de cada uno de los satélites cada 30 segundos durante un día completo.

```
% Coordenada X
ftjx = str2double(lin(1,2));
% Coordenada Y
ftjy = str2double(lin(1,3));
% Coordenada Z
ftjz = str2double(lin(1,4));
for j=1:m
    if j~=cont
        L=L*(t-a(j))/(a(cont)-a(j));
    end
end
% Polinomio X
L1=L*ftjx;
ftx=ftx+L1;
% Polinomio Y
L2=L*ftjy;
fty=fty+L2;
% Polinomio Z
L3=L*ftjz;
ftz=ftz+L3;
```

Figura 17. *Bucle de interpolación. [fuente: propia]*

En la figura anterior se puede apreciar la parte del código que realiza la interpolación. Primero se lee y se almacenan las coordenadas del satélite a interpolar en su época correspondiente. Con la variable tiempo se calcula el polinomio correspondiente a esa época ($l_n(t)$), el cual se multiplicará por cada coordenada ($f(t_n)$), almacenando el resultado en otra variable. Por último se realizará un sumatorio para obtener la función final, para cada coordenada. Este procedimiento se repetirá tantas veces como épocas tiene un día completo, y tantas veces como satélites se interpolen.

Una vez obtenida la función interpoladora de cada coordenada, para un satélite, se procede a la sustitución de la variable tiempo, en un intervalo de un día completo, cada 30 segundos. En la siguiente figura se puede apreciar el código empleado para la sustitución, la cual será almacenada en un fichero de texto satélite a satélite.

```
% Almacenar en archivo la interpolacion cada 30 seg
cont1=0;
for k=1:2881
    % Sustituir en la variable tiempo
    cx = subs(ftx,cont1);
    cy = subs(fty,cont1);
    cz = subs(ftz,cont1);
    % Convertir a 'double'
    coorx = double(cx);
    coory = double(cy);
    coorz = double(cz);
    % Imprime
    fprintf(interpolacion,'%1.0f',h);
    fprintf(interpolacion,'%20.6f',coorx);
    fprintf(interpolacion,'%20.6f',coory);
    fprintf(interpolacion,'%20.6f',coorz);
    fprintf(interpolacion,'%10.0f\n',cont1);
    % Cambio de epoca
    cont1=cont1+30;
end
```

Figura 18. *Bucle de sustitución del tiempo. [fuente: propia]*

El fichero de salida final, *"Interpolacion.txt"*, tendrá por tanto, el número de satélite GPS interpolado, las coordenadas X, Y, Z y la época a la que corresponden, ordenados respectivamente en columnas como se muestra en la figura siguiente.

```
1        3768.091140      -17042.908733      -20047.644848          0
1        3848.090235      -16933.453401      -20076.774736         30
1        3927.101635      -16835.210673      -20100.051623         60
1        4005.073288      -16748.903316      -20117.118469         90
1        4081.966710      -16675.065562      -20127.710838        120
1        4157.756467      -16614.049929      -20131.653591        150
1        4232.429551      -16566.035665      -20128.856772        180
1        4305.984642      -16531.038619      -20119.310795        210
1        4378.431302      -16508.922314      -20103.081036        240
1        4449.789086      -16499.410013      -20080.301929        270
1        4520.086607      -16502.097583      -20051.170675        300
1        4589.360559      -16516.466934      -20015.940652        330
1        4657.654715      -16541.899851      -19974.914635        360
1        4725.018912      -16577.692033      -19928.437914        390
1        4791.508037      -16623.067158      -19876.891383        420
1        4857.181025      -16677.190810      -19820.684704        450
```

Figura 19. *Fichero "Interpolacion.txt". [fuente: propia]*

Una vez programado el código necesario para la interpolación de la posición de los satélites, se procede a la ejecución del código. Debido a que la cantidad de datos a procesar en enorme, y que la función interpoladora resultante es tan grande, el programa tarda unas 5 horas en ejecutarse por completo. Es, por tanto, un programa pesado, que dependiendo del procesador de la máquina que lo compute tardará más o menos tiempo.

Análisis de los resultados que devuelve el programa.

- Tras la obtención de los resultados, ha sido necesaria una comprobación de los mismos para ver si la computación ha sido buena o no, y para ver si los valores son válidos para posteriormente utilizarlos en el cálculo de la posición del receptor.
  Para ello, se han comprobado los resultados obtenidos por la interpolación con los resultados que la función *"satpos.m"* de la librería de *Easy Suite II* (se explicará su funcionamiento más adelante). Esta función devuelve la posición de los satélites a partir de los parámetros de efemérides transmitidas. La diferencia entre la posición obtenida por efemérides transmitidas y por efemérides precisas debe estar en torno a $\pm$ 1 metro.
  Tras la comprobación se ha apreciado que la posición obtenida por ambas soluciones varía mucho con respecto a la precisión estimada, en torno a los 40 metros. Este comportamiento solo se produce en las épocas iniciales y finales del día, pues en las horas centrales de mismo las coordenadas respetan ese metro de diferencia indicado. Debido a que se ha comprobado que los puntos de control coinciden (las coordenadas cada 15 minutos obtenidas en la interpolación deben ser las mismas que las iniciales dadas en el fichero *".sp3"*), se ha llegado a la conclusión de que la interpolación obtenida es correcta, solo que la función interpoladora se sesga al inicio y final del día interpolado, por tanto, los resultados obtenidos no valen para los objetivos de este trabajo.
  Una posible solución sería partir de las 24 horas del día anterior y de las 24 horas del día siguiente en la interpolación, en vez de coger solo las 12 últimas y las 12 primeras horas del día anterior y siguiente respectivamente, solo que este procedimiento implicaría un alto nivel de computación de datos, que muchos procesadores no aguantarían, o tardarían muchas más horas en realizarlo.

En resumen, será necesario adaptar el problema a las necesidades del trabajo, es decir, se realizará la interpolación en ventanas horarias, en este caso, a una ventana de dos horas, pues como se explicará más adelante, para el cálculo de la posición del receptor se interpolará con los observables de las dos primeras horas del día; por consiguiente se va a reducir la interpolación a las dos primeras horas del día en cuestión, reduciendo así el tiempo de ejecución del programa y arreglando el problema de los sesgos que se producen en los extremos de la función interpoladora.

Para consultar el código completo consultar el **Anexo I**.

3.2.2 Programa "Inter_Lagrange2.m"

Siguiendo lo mencionado al final del apartado anterior, se va a modificar el código del programa *"Inter_Lagrange.m"* con el fin de reducir el problema, obteniendo la interpolación en las dos primeras horas del día con un intervalo de 30 segundos entre época y época. Para ello se han llevado a cabo una serie de cambios en el código del programa anterior:

- Se crearán dos archivos nuevos al inicio del programa, *"MatrizCoorSat2.txt"* y *"Epocas2.txt"*, con la misma estructura mencionada en el apartado anterior. En estos archivos solo se almacenarán las coordenadas y las épocas de las dos últimas horas del día anterior y las 4 primeras horas del día a interpolar, siendo el intervalo de las dos primeras horas del día el deseado. Con esto, dicho intervalo queda centrado en la función.
- Con los datos almacenados, se realiza el mismo proceso que en el apartado anterior. Primero se convertirán los tiempos a una misma unidad, segundos.

```
for i=1:m
    % Crea una matriz de tiempos (en segundos)
    if i<=8 % Para las epocas del dia anterior
        hora = (tmp(i,4)-24)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    else % Para las epocas del dia
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    end
end
```

Figura 20. *Bucle de conversión de tiempos. [fuente: propia]*

- Con los datos de tiempo obtenidos, se realiza el mismo proceso que en el apartado anterior; se calcula una función interpoladora para cada coordenadas, donde se sustituirá la variable tiempo cada 30 segundos durante las dos primeras horas del día, como se ha indicado anteriormente.
- Tras ejecutar el programa, cuya duración de procesamiento se ve reducida bastante, menos de una hora, se procede comparar las coordenadas obtenidas de la misma forma que con el programa anterior, concluyendo que los resultados obtenidos respetan ese intervalo de variación de ± 1 metro, incluso menor en algunos casos, por tanto, los resultados obtenidos se dan por aceptables y los sesgos desaparecen interpolando en ventanas horarias.

Debido a que, para el cálculo de la posición del receptor, es necesario saber cuál es el estado del reloj, en este programa se ha añadido el código necesario para la lectura del fichero de estados del reloj (*".clk"*) y almacenamiento del mismo en un nuevo archivo, con los estados del reloj de los satélites cada 30 segundos.

- ***"Estado_Reloj2.txt"*:** En este archivo se almacenará el estado del reloj de los satélites interpolados, para su posterior uso en el cálculo de la posición del receptor con la solución de efemérides precisas.
  Para ello se ha leído el archivo *".clk"* línea por línea almacenando los valores deseados, pues en este archivo viene dado el estado del reloj de un día completo cada 30 segundos, y en el programa se ha añadido el código necesario para almacenar solo los valores de las dos primeras horas. La estructura del archivo queda:

Estado_Reloj2.txt

1	1	0	0.000048191662422070	0.000000000028584270963600
2	2	0	0.000483807739436800	0.000000000031081110609200
3	3	0	-0.000106955393445200	0.000000000016602445365600
4	5	0	-0.000063125963830430	0.000000000053343580194000
5	6	0	0.000306631826848800	0.000000000039995732875000
6	7	0	0.000387108531823800	0.000000000030891086050200
7	8	0	-0.000047983988741930	0.000000000026631704912500
8	9	0	0.000294670645901800	0.000000000030436927305500
9	10	0	-0.000109648262675200	0.000000000026080477819800
10	11	0	-0.000668120148803200	0.000000000022360521111900
11	12	0	0.000388521259175800	0.000000000026731317776900
12	13	0	-0.000067697111940330	0.000000000047946651382700
13	14	0	-0.000056793820019430	0.000000000040932062291700
14	15	0	-0.000342044484071200	0.000000000041598831441500
15	16	0	0.000024908292528670	0.000000000037595724813200
16	17	0	-0.000192616117885200	0.000000000031508840014600
17	18	0	0.000595756465596800	0.000000000031729869119400
18	19	0	-0.000514335259359200	0.000000000028612472290700
19	20	0	0.000458053705559800	0.000000000038520807968800
20	21	0	-0.000523251049295200	0.000000000030954868904600
21	22	0	0.000115110801716800	0.000000000014102736151900
22	23	0	-0.000204194476718200	0.000000000039790466504000
23	24	0	-0.000029838940388730	0.000000000026309440210300
24	25	0	-0.000327933674525200	0.000000000041572179194100
25	26	0	-0.000519183315763200	0.000000000051676215381500
26	27	0	0.000202585647893800	0.000000000034014124916000
27	28	0	0.000586368379894800	0.000000000035999607902500
28	29	0	0.000629651282492800	0.000000000023319706026000
29	30	0	0.000160582556817800	0.000000000025857927356300
30	31	0	0.000221751388550800	0.000000000027114313508300

Figura 21. *Fichero "Estado_Reloj2.txt". [fuente: propia]*

Donde la primera columna se corresponde con el nº del satélite, la segunda indica la época (en segundos) a la que corresponde y la tercera y cuarta se corresponden con los valores del estado del reloj (*"Bias"* y *"Bias Sigma"* respectivamente).

En resumen, los resultados obtenidos interpolando en ventanas horarias, centrando éstas en el medio de la función, son bastante precisos y, por consiguiente, se da por bueno el procedimiento para la obtención de coordenadas por interpolación de Lagrange.

Surge un problema a la hora de obtener la posición del receptor, pues las coordenadas del satélite vienen dadas en el momento de recepción de la señal, y es necesario saber la posición de éstos en el momento de salida de la señal. A consecuencia de esto, será necesario adaptar el método para que el tiempo de sustitución no sea cada 30 segundos, que es el tiempo de recepción, sino que se necesitará el tiempo de salida de señal, el cual se obtiene (se explicará mejor en apartados siguientes) restando el tiempo de viaje y corrigiendo del estado del reloj. Por tanto, será necesario leer el fichero de observaciones para obtener dicho tiempo. Como la rutina del *easy3* realiza este procedimiento, se va a aprovechar para utilizar esos observables también para la interpolación, necesitando así adaptar el código en forma de función de Matlab, que recibirá las coordenadas y el tiempo como entrada.

Para consultar el código completo consultar el **Anexo I**.

3.2.3 Función "Inter_Lgrge_Easy3.m"

El procedimiento de esta función es el mismo que el desarrollado en los apartados anteriores, solo cambia la forma de entrada de los datos, pues como se ha explicado, la función recibe unos valores y devuelve unos resultados específicos. La función queda

adaptada de forma que cada vez que se llama a ésta, devolverá solo la interpolación de ese satélite en la época indicada. Queda, por tanto, definida como:

```
function [pos_int] = Inter_Lgrge_Easy3(satelite,t_GPS,coorsat,tmp)
```

Donde las variables de entrada son:

- ***satelite*:** Es el número del satélite a interpolar.
- ***t_GPS*:** Es el tiempo de emisión de la señal por el satélite, es decir, la época en la que se desea obtener la posición del satélite.
- ***coorsat*:** Matriz de coordenadas del satélite. Está compuesta por las coordenadas del satélite del cual se desea obtener la interpolación, que se almacenan fila por fila por cada una de las épocas. Proceden del archivo *"MatrizCoorSat2.txt".*
- ***tmp*:** Matriz de las épocas de interpolación. Proceden del archivo *"Epocas2.txt".*

Devuelve:

- ***pos_int*:** Vector columna con las coordenadas del satélite interpolado en la época definida.

Con esta función se podrá obtener la posición de cada uno de los satélites dependiendo de la observación dada en el fichero de observables, en la época de salida de la señal del satélite.

En el apartado siguiente se explicará su implementación a la rutina *easy3* y su funcionamiento dentro de élla.

Para consultar el código completo consultar el **Anexo I**.

3.3 Modificación y ejecución de la rutina *easy3*

Como se ha mencionado al inicio del punto 3, y en los objetivos de este trabajo, el propósito final es obtener las coordenadas de la estación permanente con las soluciones facilitadas, efemérides precisas (interpolación) y efemérides transmitidas (RINEX de navegación), y ver la diferencia que hay entre la posición de los satélites obtenida a partir de ambas soluciones, además de calcular y mostrar las coordenadas y órbitas que éstas generan para cada satélite que interviene en la observación.

Para ello se va a partir de la rutina de *easy3*, que como se ha indicado anteriormente, se concibió con el fin de calcular la posición de un receptor con efemérides transmitidas, para 20 épocas de un segundo de intervalo. Solo para uso de satélites GPS (*"NAVSTAR"*).

En este trabajo se va a modificar toda la rutina, tanto el programa principal como las funciones asociadas al mismo, con el fin de poder calcular las coordenadas del receptor con las dos soluciones en un intervalo de observación de 2 horas, con observables cada 30 segundos, un total de 240 épocas.

Los ficheros RINEX de entrada han cambiado, su forma es distinta a la empleada cuando se programó la rutina *easy3*, además los ficheros de ejemplo que vienen adjuntos al programa están modificados para dicho código. En este trabajo se va a modificar toda la lectura de los datos, con el fin de poder generalizar un poco más esta rutina, que trabaje con todo el fichero y que lea solo los datos que sean necesarios.

En resumen, el primer paso será modificar la entrada de los datos, tanto de navegación (efemérides transmitidas) como de observables. La parte de cálculo con efemérides precisas se explicará más adelante, pues se ha llevado a cabo la creación y modificación de nuevas funciones para su cálculo.

3.3.1 Estructura del programa "easy3.m"

Es el programa (script) principal de la rutina. Será, por tanto, donde se carguen los datos de entrada y se preparen para el cálculo de la posición del receptor. El programa contiene una serie de funciones que serán las encargadas de llevar el procedimiento de cálculo. En este caso, se van a diferenciar dentro de la rutina, dos tipos de funciones: funciones principales, aquellas que se llaman directamente desde el programa principal y funciones secundarias, las cuales son llamadas dentro de las funciones principales.

A continuación se va a comentar cada uno de los tipos de funciones y cuáles de éstas están comprendidas dentro de dichos tipos, así como su funcionalidad y si han sido modificadas o no.

A) Funciones principales:

Dentro de este grupo se encuentran aquellas funciones que son llamadas directamente en el programa principal, y que son las encargadas de procesar los datos y calcular la posición del receptor. En orden de aparición en el código se encuentran:

- ***"rinexe.m"*:** Recibe el archivo de navegación (".**n"*) y el "archivo de efemérides" (archivo de salida de los datos de navegación con un formato específico definido (.dat)). Lee el mensaje del RINEX de navegación y reformatea los datos en una matriz que se guarda en el "archivo de efemérides". No ha sido modificada.
- ***"get_eph"*:** Recibe el "archivo de efemérides", las cuales son reformadas en una matriz de 21 filas (parámetros orbitales) y tantas columnas como efemérides de satélites hay. No ha sido modificada.
- ***"aheader.m"*:** Recibe el "archivo de observables". Analiza la cabecera del archivo RINEX y saca una lista de tipos de observación y antena offset, y devuelve los tipos de observables. No ha sido modificada.
- ***"fepoch_0.m"*:** Recibe el "archivo de observables". Busca dentro del fichero de observables la siguiente época (en la primer iteración, primera época) y devuelve el tiempo en segundos de la época y el nº identificador de los satélites GPS observados.
  En esta función se han modificado varias partes:
  - Evitados saltos de línea al aparecer mayor nº de satélites observados.
  - Añadida funcionalidad para evitar la lectura de observables de Glonass.
  - Cambio del año de referencia programado inicialmente.
- ***"grabdata.m"*:** Recibe el fichero de observables, el número de observables de la época en cuestión (procedente de la función anterior, tamaño del vector de satélites) y los tipos de observables. Devuelve los observables que aparecen en esa época para cada satélite. Se ha modificado:
  - Añadida sentencia para evitar la segunda línea de satélites observados Glonass.
- ***"fobs_typ.m"***: Recibe los tipos de observables y el nombre del observable deseado, y devuelve la columna de la matriz de observación que contiene dicho observable. No ha sido modificada.

- ***"recpo_lsa.m"*:** Recibe los observables, los satélites, el tiempo de la época en segundos, las efemérides transmitidas y el número de época correspondiente, y calcula la posición del receptor con pseudodistancias empleando mínimos cuadrados. Devuelve también la posición de los satélites en la época de cálculo. En esta función se han modificado varias partes:
    - Añadida funcionalidad para que elimine aquellos observables de los cuales no se tiene dato y aparecen en el vector de observables como *"NaN" (Not A Number).*

```
% Elimininamos los satelites de los que no se tienen datos
a = size(obs);
obs1 = obs;
for i=1:a
    b = obs(i,1);
    if isnan(b)
       obs1(i) = [];
       sats(i) =[];
       a = a - 1;
    end
end
obs = obs1;

global H
```

Figura 22. *Eliminación de valores "NaN" [fuente: propia]*

    - Añadida funcionalidad para que tome las efemérides transmitidas en distintas épocas, dependiendo de la época en la que se haya tomado el observable.

```
for i = 1:m
    if q >= 120
        k = col_Eph(i);
        k= k + 32;
    else
        k = col_Eph(i);
    end
```

Figura 23. *Cambio de época en efemérides transmitidas. [fuente: propia]*

    - Añadido almacenamiento de las coordenadas obtenidas para los satélites en esa época.
    - Modificados los valores de la troposfera y añadida función de ionosfera (se explicará en el apartado de funciones secundarias).

B) Funciones secundarias:

Dentro de este grupo se encuentran aquellas funciones que son llamadas dentro de las funciones principales. Tienen como fin realizar cálculos internos que las funciones anteriores necesitan para computar sus funcionalidades.

- ***"check_t.m"*:** Recibe el tiempo GPS. Repara los saltos de semanas GPS. No ha sido modificada.

- ***"e_r_corr.m"*:** Recibe el tiempo de viaje de la señal y la posición del satélite, y calcula las coordenadas ECEF del satélite rotado por el efecto de rotación de la Tierra durante el tiempo de recorrido de la señal. No ha sido modificada.
- ***"find_eph.m"*:** Recibe las efemérides transmitidas, el nº de satélite y la época y devuelve las columnas donde se encuentran almacenadas las efemérides de cada satélite que interviene en una observación. No ha sido modificada.
- ***"julday.m"*:** Recibe el año, mes, día y hora de la observación y devuelve el día en el calendario Juliano. No ha sido modificada.
- ***"gps_time.m"*:** Recibe el día en el calendario Juliano y devuelve la semana GPS y los segundos de la semana. No ha sido modificada.
- ***"topocent.m"*:** Recibe la posición del receptor y la posición del satélite y transforma la posición del satélite a un sistema de coordenadas topocéntrico con origen en la posición del receptor. No ha sido modificada.
- ***"tropo.m"*:** Recibe los parámetros necesarios para el cálculo del error troposférico. No ha sido modificada.
- ***"ion.m"*:** Recibe la posición del satélite en el sistema topocéntrico y el tiempo GPS, y con ello calcula el error ionosférico que se empleará en el ajuste mínimo cuadrático. Esta función no estaba implementada en el código inicial del programa, ha sido añadida con el fin de mejorar los resultados del cálculo de la posición del receptor. Para ello es necesario previamente añadir en la función los parámetros de la ionosfera que aparecen en la cabecera del fichero de navegación. En este caso el modelo aplicado en el cálculo es el *Modelo de Klobuchar*, el cual consiste en un algoritmo de corrección ionosférica cuyos parámetros se transmiten, como se ha indicado, en el fichero de navegación, siendo éste un modelo simple; supone que todos los electrones están concentrados en una capa delgada a 350 Km de altura. (Mas información en *Capítulo 8: La ionosfera* [1])

```
function dion = ion(az,el,lat,lon,tgps)

alpha(1)=8.3819D-09;
alpha(2)=-7.4506D-09;
alpha(3)=-5.9605D-08;
alpha(4)=5.9605D-08;
beta(1)=8.8064D+04;
beta(2)=-3.2768D+04;
beta(3)=-1.9661D+05;
beta(4)=1.9661D+05;
```

Figura 24. *Parámetros ionosféricos. [fuente: propia]*

Todas estas funciones son las que implementan el cálculo de la posición del receptor, en el siguiente apartado se explicará el procedimiento de cálculo que se sigue para la obtención, en este caso, de las coordenadas de la antena por medio de la pseudodistancia.

Previamente a dicha explicación, se van a comentar los cambios añadidos a la rutina para que calcule la posición con la solución de efemérides precisas. Como se ha explicado anteriormente, la posición del satélite en este caso proviene de la interpolación de Lagrange, por tanto, será necesario añadir la función de interpolación al programa, y además, modificar la función que calcula la posición del receptor, para que tome la posición de los satélites que intervienen en el cálculo de la función de interpolación. Además se añadirá como corrección del tiempo el estado del reloj procedente del

archivo que se generó con el programa *"Inter_Lagrange2.m"* (apartado 3.2.2). Para el cálculo con la solución de efemérides precisas se ha llevado a cabo el siguiente procedimiento:

- Se necesita una función que calcule la posición del receptor. Como la estructura de cálculo es la misma que con efemérides transmitidas, se va a partir de la función *"recpo_lsa.m"* a partir de la cual se va a crear la función *"recpo_efp.m"*, que realizará el mismo procedimiento de cálculo, solo que la posición de los satélites la tomará de la función de interpolación que se incorporará al código de esta. La estructura de la función queda:

```
function [Xe,Ye,Ze,pos] = recpo_efp(obs,sats,time,Clk,q,coorsat,tmp)
```

  Donde:
  - ***obs*:** Vector de observables (pseudodistancias).
  - ***sats*:** Vector de satélites.
  - ***time*:** Época de la observación
  - ***Clk*:** Estados del reloj (fichero *"Estado_Reloj2.txt"*)
  - ***q*:** Nº de época que interpola.
  - ***coorsat*:** Matriz de coordenadas procedente del fichero *"MatrizCoorSat.txt".*
  - ***tmp*:** Matriz de épocas procedente del archivo *"Epocas2.txt".*

  Devuelve:

  - ***Xe, Ye, Ze*:** Vectores de coordenadas de los satélites.
  - ***pos*:** Vector de posición con las coordenadas del receptor calculadas.

- Una vez añadidas las variables necesarias para el cálculo de la posición del receptor, se procede a modificar el código para incorporar la función de interpolación:

  1) Añadido bucle que lee la matriz de estados del reloj y almacena éstos dependiendo del satélite que entre en ese momento
  2) Modificación de la obtención del tiempo GPS (estado del reloj añadido) y bucle que lee la matriz de coordenadas y selecciona aquellas que pertenecen al satélite que se desea interpolar, las cuales almacena en una nueva matriz que será la entrada de la función de interpolación.
  3) Añadida en la primera iteración del sistema la función de interpolación, que devolverá la posición del satélite interpolada en el instante tiempo GPS.
  4) En la rutina inicial para efemérides transmitidas, la posición del satélite se calcula en cada iteración. Como se añadió la funcionalidad de almacenar las coordenadas del satélite (Xe, Ye, Ze), éstas se recuperan para las siguientes iteraciones, evitando así que se calcule de nuevo la posición de los satélites. Esto se ha hecho con el fin de reducir el tiempo de cálculo, ya que la interpolación es un proceso de computación costoso, de esta forma se reduce bastante el tiempo de cálculo.

```
for iter = 1:no_iterations
    A = [];
    H = [];
    omc = []; % observed minus computed observation
    dir_vector = [];
    for i = 1:m
        satelite = sats(i,1);
        for l = 1:m2
            % Obtencion del estado del reloj
            satel1 = Clk (l,1);
            tmp1 = Clk (l,2);
            if ((satelite == satel1) && (((q-1)*30 == tmp1)))
               CLK(1,1) = Clk (l,3);
               CLK(2,1) = Clk (l,4);
            end
        end
        % Coordenadas del satélite
        tx_RAW = ((q-1)*30) - obs(i)/v_light;
        tcorr = CLK(1,1);
        tx_GPS = tx_RAW-tcorr;
        conta = 0;
        for z=1:352
            if coorsat(z,1)==satelite
                conta = conta + 1;
               cs(conta,:) = coorsat(z,:);
            end
        end
        %X = Inter_Lgrge_Easy3(satelite,tx_GPS,cs,tmp);
        if iter == 1
            X = Inter_Lgrge_Easy3(satelite,tx_GPS,cs,tmp);
            traveltime = 0.072;
            Rot_X = X;
            trop = 0;
            dion=0;
        else
            X = [Xe(i,1);Ye(i,1);Ze(i,1)];
            rho2 = (X(1)-pos(1))^2+(X(2)-pos(2))^2+(X(3)-pos(3))^2;
            traveltime = sqrt(rho2)/v_light;
            Rot_X = e_r_corr(traveltime,X);
            rho2 = (Rot_X(1)-pos(1))^2+(Rot_X(2)-pos(2))^2+(Rot_X(3)-pos(3))^2;
            [az,el,dist,phi,lambda] = topocent(pos(1:3,:),Rot_X-pos(1:3,:));
```

1

2

3

4

Figura 25. *Estructura de la función "recpo_efp". [fuente: propia]*

Con todas estas modificaciones el programa ya está preparado para calcular la posición del receptor con ambas soluciones. Como se ha indicado al inicio el intervalo de tiempo elegido es dos horas (las dos primeras horas del día), que equivalen a 240 épocas, con una diferencia de tiempo de 30 segundos entre época y época (diario 30s).

3.3.2 Procedimiento de cálculo

Una vez conocida la estructura de scripts que componen la rutina de *easy3*, en este apartado se va a explicar el procedimiento de cálculo que se sigue para el tratamiento de los datos GNSS.

El cálculo de las coordenadas sigue el método de mediciones de fase, el cual nos permite obtener la posición mediante la pseudodistancia. En este caso se van a tratar

los observables como si se tratara de un cinemático, aunque los datos procedan de una observación estática.

Por tanto se van a calcular las coordenadas observación a observación, y al final de todas las épocas se va a realizar una media aritmética de toda las posiciones obtenidas por cada observable, con el fin de dar una solución media de la posición del receptor.

El procedimiento de cálculo es el mismo en ambas soluciones, tanto en efemérides transmitidas como en efemérides precisas, cuya única diferencia es la obtención de la posición de los satélites, la cual se ha explicado en apartados anteriores. El proceso es el siguiente.

1) Preparación de los datos: Se preparan los datos necesarios para el cálculo. Para ello se emplean las funciones mencionadas en el apartado anterior.
2) Una vez preparados los datos éstos entran en la función que realiza el cálculo, *"recpo_lsa.m"* para efemérides transmitidas y *"recpo_efp.m"* para efemérides transmitidas. Como la obtención de la posición de los satélites ya se ha explicado, se va a generalizar el cálculo que realizan estas dos funciones, pues es el mismo procedimiento.
3) La forma de cálculo es simple, por cada época de observación se va a generar un sistema de ecuaciones que tenga como incógnitas las coordenadas de la estación y el estado del reloj. En cada una de las épocas aparece un número aleatorio de satélites observados, los cuales van a generar las ecuaciones del sistema, es decir, por cada satélite se añadirá una ecuación nueva.
   El sistema de ecuaciones se resolverá siguiendo un ajuste pos mínimos cuadrados, el cual devuelve una posición aproximada del receptor y el estado del reloj del mismo.
   Para la obtención de la matriz de coeficientes (matriz A) se necesitan los valores de la posición del satélite, el observable de la pseudodistancia y una posición previa del receptor. Lo mismo se necesita para obtener el vector de observables (vector K), solo que en este caso se añadirán las correcciones del error troposférico e ionosférico.
   Como se ha comentado, es necesario disponer de una posición previa del receptor, pero como al principio ésta se desconoce el procedimiento seguido es simple:
   - Se calcula una primera iteración donde se parte que la posición del receptor es 0, y se calculan unas coordenadas previas, las cuales tendrán mucho error. Las nuevas coordenadas se almacenan en el vector de posición, y se vuelve a iterar realizan el mismo cálculo pero en este caso se empleará la nueva posición obtenida. Este proceso se repite tantas veces hasta que el error se reduce, es decir, se itera el proceso hasta obtener unas matrices estables y por tanto una posición del receptor buena. En la rutina de *esay3* se itera hasta seis veces, pues son suficientes para obtener una posición buena.
4) Con todas las posiciones obtenidas por cada época observada (concretamente 240 posiciones calculadas), se realiza una media aritmética de todas ellas obteniendo unas coordenadas promediadas de la estación.
5) Las coordenadas obtenidas se encuentran el sistema de coordenadas ECEF. Para una mejor comparación con la solución oficial, se ha realizado una transformación de las mismas a coordenadas geográficas y posteriormente a coordenadas UTM.

6) El programa no solo calcula la posición del receptor, sino que inicialmente ya venía programado un ploteo de la variación de las coordenadas en las distintas épocas de cálculo. Además se ha añadido un ploteo de las órbitas de los satélites y el cálculo de la diferencia que se produce entre ambas soluciones, en las épocas de trabajo, con el fin de poder ver la diferencia entre calcular las coordenadas de los satélites de una forma o de otra.

3.4 Resultados

Como ya se ha indicado, el programa devolverá las coordenadas calculadas siguiendo el procedimiento explicado, y además, devolverá una serie de gráficas con la comparativa de las órbitas de los satélites calculadas con las soluciones de efemérides, así como el cálculo de la diferencia existente entre ambas.

3.4.1 Coordenadas

El programa calcula la posición de la estación en coordenadas ECEF. Para una mejor comparativa con la solución oficial, que viene dada en ETRS89, se han aplicado algoritmos de transformación de coordenadas del sistema ECEF a geográficas y posteriormente a cartesianas locales (UTM). Los resultados obtenidos son:

```
Coordenadas de la cabecera del fichero de observaciones:

   X:   4932702.944     Y:   -29607.788  Z:  4029833.040

   Lat: 39.26085455   Lon:  -0.20380578  Helip:   62.952

   Xutm: 728594.664  Yutm:  4368496.315  Helip:   62.952

Posición media para 240 épocas, con Efemérides Transmitidas:

   X:   4932698.041     Y:   -29605.212  Z:  4029832.804

   Lat: 39.26086409   Lon:  -0.20379513  Helip:   59.003

   Xutm: 728597.123  Yutm:  4368499.331  Helip:   59.003

Posición media para 240 épocas, con Efemérides Precisas:

   X:   4932698.160     Y:   -29605.046  Z:  4029832.995

   Lat: 39.26086433   Lon:  -0.20379443  Helip:   59.215

   Xutm: 728597.288  Yutm:  4368499.409  Helip:   59.215
```

Figura 26. *Coordenadas obtenidas. [fuente: propia]*

Antes de analizar los resultados, hay que tener en cuenta que las coordenadas de los satélites en efemérides precisas vienen dadas en el sistema de referencia IGS14, por tanto, para una mejor aproximación a la solución final previamente se ha de llevar una transformación al sistema de referencia adecuado, en este caso ETRS89(ETRF89). Para ello se ha empleado la web de *EUREF*, que cuenta con una calculadora:

Figura 27. *Transformación de las coordenadas. [fuente: propia/EUREF]*

Por tanto, las coordenadas transformadas quedan:

```
Coordenadas de la cabecera del fichero de observaciones:

   X:   4932702.944     Y:   -29607.788  Z:  4029833.040

   Lat: 39.26085455   Lon:  -0.20380578  Helip:   62.952

   Xutm: 728594.664  Yutm:  4368496.315  Helip:   62.952

Posición media para 240 épocas, con Efemérides Transmitidas:

   X:   4932698.041     Y:   -29605.212  Z:  4029832.804

   Lat: 39.26086409   Lon:  -0.20379513  Helip:   59.003

   Xutm: 728597.123  Yutm:  4368499.331  Helip:   59.003

Posición media para 240 épocas, con Efemérides Precisas:

   X:   4932698.446     Y:   -29605.379  Z:  4029832.633

   Lat: 39.26086282   Lon:  -0.20379582  Helip:   59.208

   Xutm: 728596.969  Yutm:  4368498.936  Helip:   59.208
```

Figura 28. *Coordenadas obtenidas tras la transformación. [fuente: propia]*

Como se puede apreciar, tras la transformación, las coordenadas obtenidas con efemérides precisas obtienen mejor resultado que las calculadas con efemérides transmitidas, aunque la diferencia es de apenas unos 20 - 30 cm, tanto en X, en Y como en la altura elipsoidal.

En aspectos generales, las coordenadas obtenidas para dos horas de observación, con intervalo de 30 segundos varían unos 3 metros con respecto a las coordenadas aproximadas dadas en el fichero de observaciones. Como se está trabajando con medidas de código, con observables de pseudodistancia, las precisiones a obtener no pueden ser muy altas, como podría ocurrir con mediciones en código, pero aun así los resultados obtenidos son aceptables, ya que si comparamos los resultados obtenidos con este procedimiento con las precisiones que puede obtener un navegador comercial (que utiliza un procedimiento similar), en este caso las precisiones han mejorado bastante, por tanto, a priori se puede concluir que los resultados obtenidos son aceptables y han cumplido con las expectativas.

3.4.2 Posición de los satélites

En este apartado se van a comparar las órbitas de los satélites. Los resultados de calcular una media y desviación típica de los vectores de diferencia (distancia) entre la posición calculada con efemérides transmitidas y la calculada con precisas son:

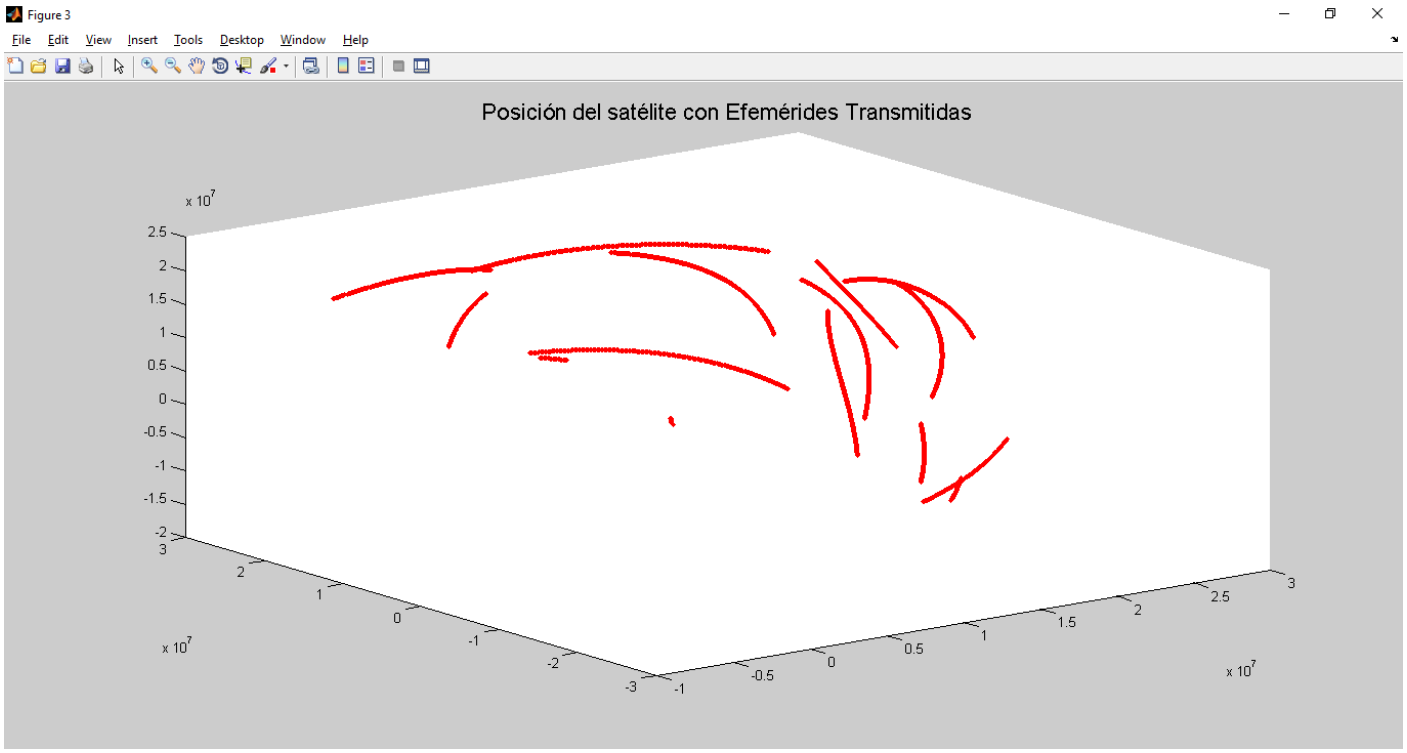
```
Módulo del vector diferencial entre la posición del satélite con Ef Precisas y con Ef Transmitidas:

   Satélite:  1    Diferencia (m): 0.776    Desviación Típica: 0.015
   Satélite:  7    Diferencia (m): 1.069    Desviación Típica: 0.208
   Satélite:  8    Diferencia (m): 1.517    Desviación Típica: 0.186
   Satélite: 10    Diferencia (m): 1.121    Desviación Típica: 0.052
   Satélite: 11    Diferencia (m): 1.597    Desviación Típica: 0.020
   Satélite: 16    Diferencia (m): 1.534    Desviación Típica: 0.041
   Satélite: 18    Diferencia (m): 1.796    Desviación Típica: 0.079
   Satélite: 20    Diferencia (m): 2.391    Desviación Típica: 0.377
   Satélite: 21    Diferencia (m): 1.491    Desviación Típica: 0.070
   Satélite: 25    Diferencia (m): 1.301    Desviación Típica: NaN
   Satélite: 26    Diferencia (m): 0.939    Desviación Típica: 0.033
   Satélite: 27    Diferencia (m): 1.059    Desviación Típica: 0.078
   Satélite: 29    Diferencia (m): 0.872    Desviación Típica: 0.033
   Satélite: 30    Diferencia (m): 1.680    Desviación Típica: 0.029
   Satélite: 31    Diferencia (m): 1.323    Desviación Típica: 0.448
   Satélite: 32    Diferencia (m): 2.457    Desviación Típica: 0.045
```

Figura 29. *Diferencia entre posiciones y error asociado. [fuente: propia]*

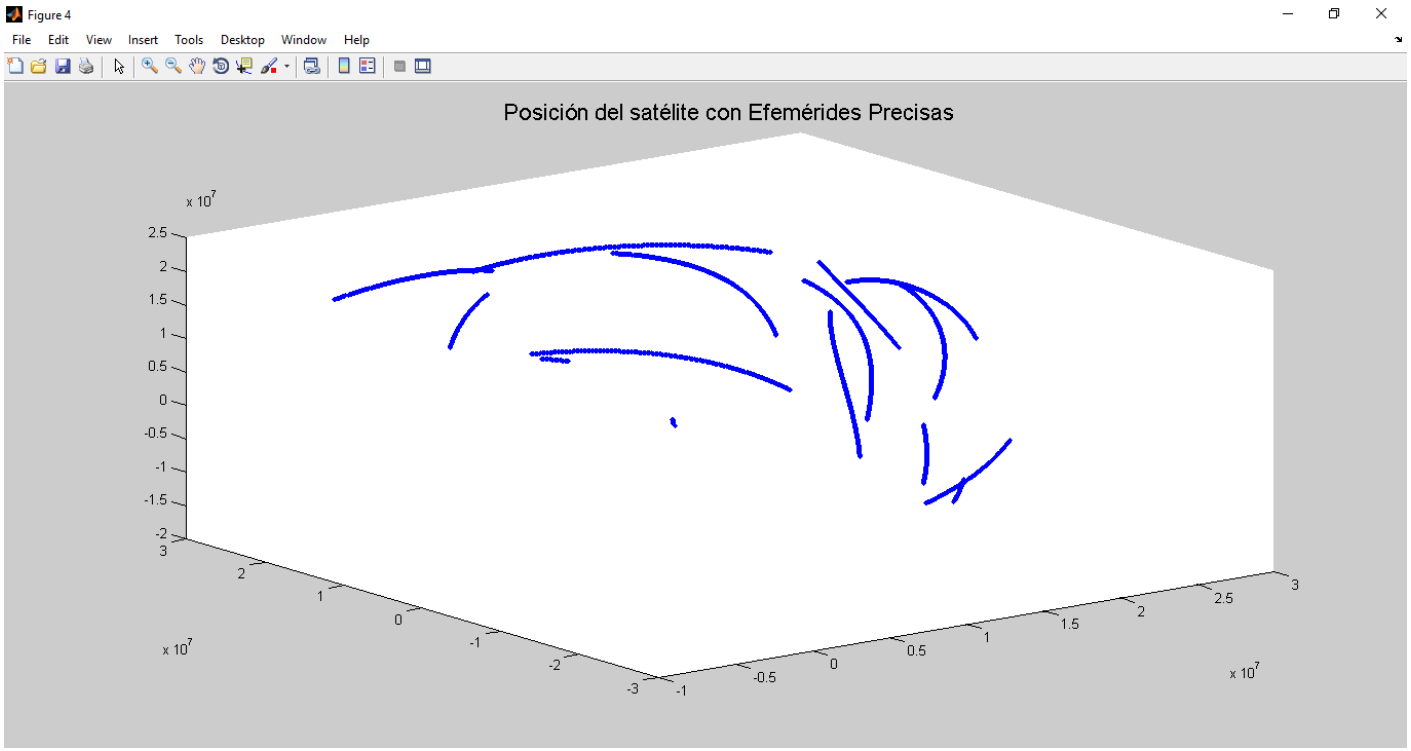
Como se puede apreciar, la diferencia promedio por cada satélite se encuentra entre 0’7 metros y 2’3 metros, lo cual es buen indicativo de que el procedimiento seguido es bueno, ya que la diferencia entre ambas soluciones suele rondar ese intervalo de diferencia. Además se ha añadido la desviación típica asociada a cada observable (al 68%). El error cometido como máximo está en unos 30 – 40 cm, por tanto, se puede concluir que los resultados obtenidos son aceptables.

En este caso cabe destacar que el satélite G25 ha devuelvo un valor de desviación típica nulo. Esto es debido a que dicho satélite solo ha sido observado en una época y por consiguiente no se puede calcular su desviación típica, y su media será solo la diferencia de distancia en esa observación. (Ver gráfica de detalle en el **Anexo II**).

Para una apreciación más visual del comportamiento de las órbitas, las gráficas que devuelve el programa son:

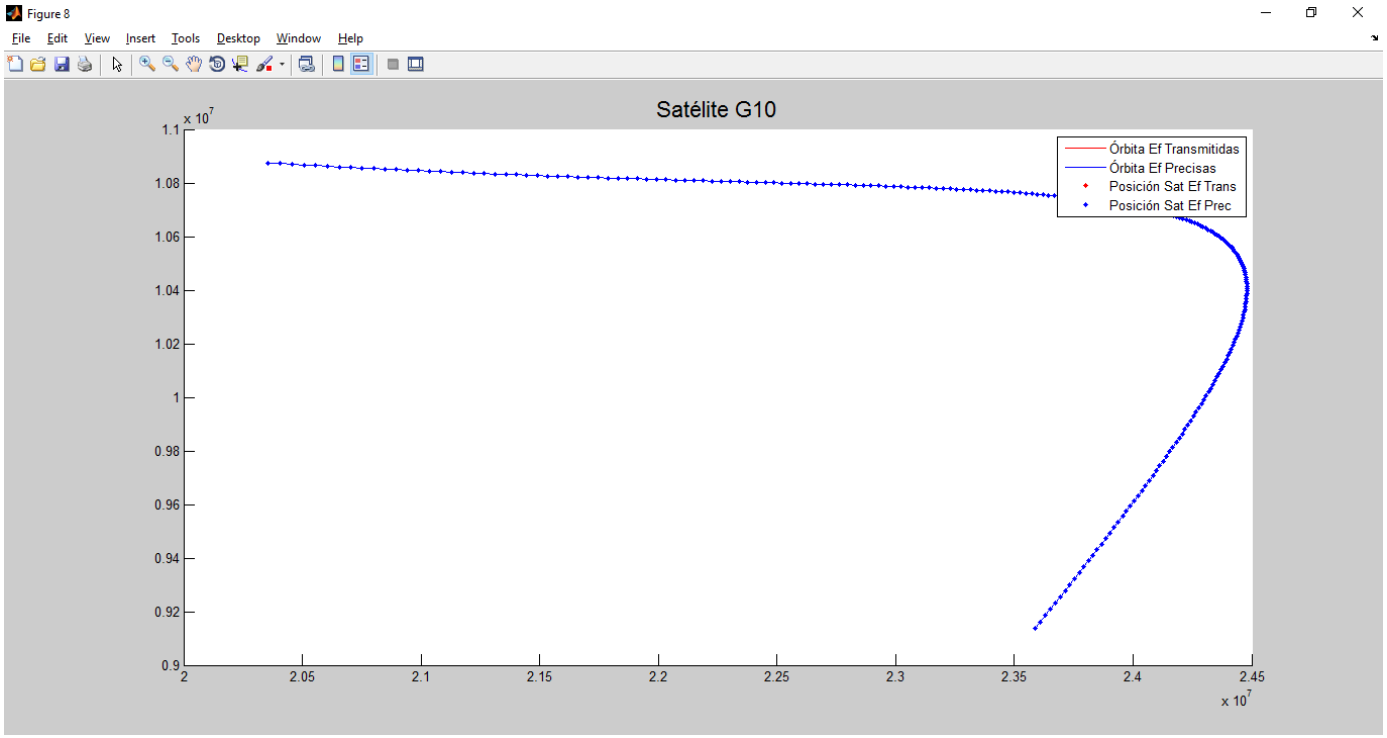


Gráfica 2. *Posición de los satélites observados en órbita con Efemérides Transmitidas en 3D. [fuente: propia]*

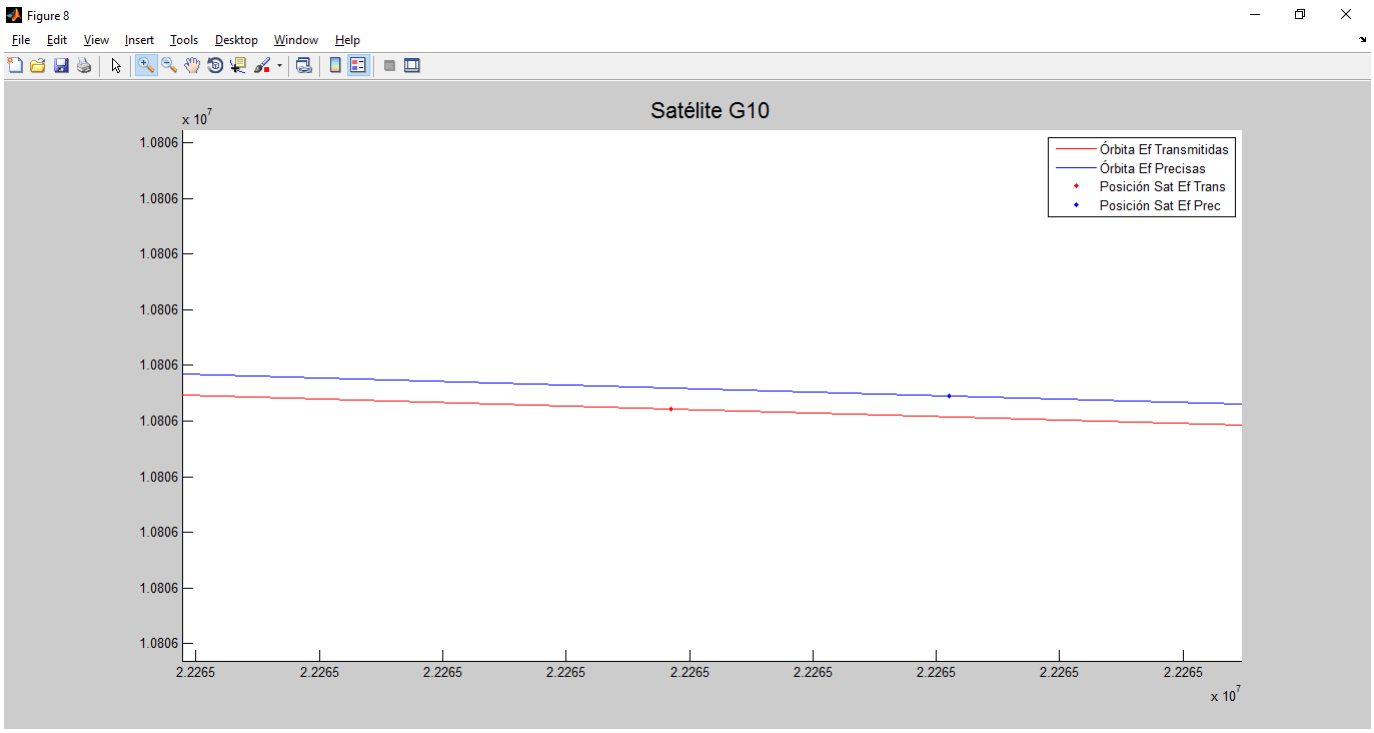


Gráfica 3. *Posición de los satélites observados en órbita con Efemérides Precisas en 3D. [fuente: propia]*

Estas dos figuras muestran las órbitas definidas con cada solución para el periodo de dos horas, de todos los satélites observados, en 3D. Como se puede apreciar la diferencia visual es mínima, por tanto se procede a analizar una a una las órbitas. Como son muchas las órbitas, y su comportamiento es similar, se va a analizar un ejemplo, y el resto encuentran en el **Anexo II**.



Gráfica 4. *Posición del satélite G10 con ambas soluciones. [fuente: propia]*



Gráfica 5. *Diferencia de posición del satélite G10 con ambas soluciones. [fuente: propia]*

Como se puede apreciar en la *Gráfica 3*, la diferencia entre ambas soluciones es inapreciable, se superponen las órbitas, por tanto será necesario hacer zoom hasta diferenciarlas. En la *Gráfica 4* se puede apreciar la diferencia existente, la cual se corresponde con los valores numéricos obtenidos.

4. Conclusiones

En este trabajo se ha profundizado en los principios fundamentales del GNSS así como en la metodología de cálculo que se emplea para las mediciones en estaciones permanentes. Además se han aplicado una serie de conceptos de cálculo de posicionamiento mediante mediciones de fase, cálculo de coordenadas de satélites con efemérides transmitidas en las épocas de observación y cálculo de la posición de los mismos con la solución final de efemérides precisas, a partir de un método de interpolación.

Con todo ello, se puede concluir que:

- Para la interpolación de Lagrange es necesario el uso de algoritmos y scripts de programación, ya que la complejidad del cálculo solo permite realizarlo por medio de la computación.
- La diferencia de cálculo de la posición de los satélites con ambas soluciones es de unos metros, por tanto, se puede concluir que ambas metodologías de cálculo son correctas, pues respetan el margen de error que las instituciones indican.
- Las coordenadas del receptor obtenidas, con ambas soluciones, se han mejorado con la aplicación de esta metodología, con respecto al cálculo que realizan algunos dispositivos a partir de la pseudodistancia, además destacar la mejora de unos 30-40 cm que las efemérides precisas generan con respecto a las efemérides transmitidas.

En resumen, se puede concluir que se ha conseguido alcanzar los objetivos iniciales, que era mejorar lo ya existente, además de profundizar y comprender mejor la metodología de cálculo y los principios del GNSS.

Como líneas futuras de estudio, a partir de este trabajo, se podrían considerar:

- Al estar trabajando con metodología de cinemático con observables de estático, se podría estudiar la posibilidad de modificar el código con el fin de generar un solo sistema de ecuaciones que comprendiera todos los observables de todas las épocas, en vez de realizar el promedio de varios ajustes.
- Otra posibilidad sería, a partir de este código, modificarlo para que el cálculo lo realizase con mediciones en fase, lo cual generaría altísimas precisiones, además de modificar los modelos ionosférico y sobre todo troposférico, con el fin de afinar un poco más.

Presupuesto

Una vez finalizado el desarrollo del trabajo, en este apartado se va a realizar un desglose de los costes que conllevaría la realización de las actividades pertinentes para el desarrollo de la metodología explicada. Por tanto, se va a detallar un presupuesto valorativo donde se van a desglosar las distintas actividades a realizar y los costes que éstas generan, el tiempo de ejecución y los medios empleados.

Se van a diferenciar dos partes:

- Mano de obra.
- Alquiler de software.

Mano de obra

Para el desarrollo de este trabajo solo será necesario como recursos humanos un Ingeniero en Geomática y Topografía. Los costes de mano de obra vendrán dados en el Convenio de OODD de la provincia de Valencia en el año 2017.

Vista la tabla de convenio, considerando al técnico de ejecución como titulado medio se obtiene que:

	Coste	Nº Pagas	Total
Sueldo Base	1,393.49 €	12+2	19,508.86 €
Plus de Convenio	66.00 €	12+2	924.00 €
	Sueldo Bruto Anual		20,432.86 €
	Seguridad Social (40%)		8,173.14 €
	Coste Total (anual)		28,606.00 €

Tabla 2. *Salario anual. [fuente: propia]*

Donde:

- Sueldo Base y Plus de Convenio: Tabla de Convenio [4].
- Seguridad Social [5].

Como es una contrata parcial, se necesita conocer el coste del trabajador por horas. Sabiendo que en la Comunidad Valenciana en nº de días laborables son 249 [6], a jornada completa, 8 horas, el nº de horas de trabajo al año asciende a 1992 horas. Por tanto, se dividirá el coste total entre el número de horas, quedando:

Precio:	14.36 €/h

Tabla 3. *Coste del trabajador a la hora. [fuente: propia]*

Alquiler de software

El único software necesario para el desarrollo del proyecto es el programa Matlab, cuyo coste es:

- **Licencia Matlab:** $\frac{2.000€/año}{12\ meses} = 166{,}66\ €/mes$

Presupuesto por contrata [7]

Partiendo de que:

$$PEC = PEM + GGE + BI$$

Donde:

- PEC: Presupuesto Ejecución por Contrata.
- PEM: Presupuesto de Ejecución de Material.
- GGE: Gastos Generales.
- BI: Beneficio Industrial.

Se estiman, por tanto, el tiempo de trabajo para el desarrollo de la metodología, siendo entonces:

Actividad	Tiempo (días)	Coste Trabajador (€/h)	Coste Trabajador (€/día)	Total
Desarrollo de algoritmos	7	14.36 €	114.88 €	804.18 €
Actividad	Tiempo (días)	Software	Precio Alquiler (€/día)	Total
Desarrollo de algoritmos	7	166.66 €	5.56 €	38.89 €
			Total PEM	843.07 €

Tabla 4. *Cálculo del PEM. [fuente: propia]*

Aplicando la fórmula:

PEC	Coste
PEM	843.07 €
GGE (13% de PEM)	109.60 €
BI (6% de PEM)	50.58 €
Total	1,003.26 €
IVA (21%)	210.68 €
Total	1,213.94 €

Tabla 5. *Cálculo del coste final. [fuente: propia]*

Por tanto, el coste final del trabajo es de:

1,213.94 €

"Mil doscientos trece con noventa y cuatro euros"

Bibliografía

[1] Berné Valero J.L., Anquela Julián A.B., Garrido Villén N. (2016). "GNSS. GPS: fundamentos y aplicaciones en Geomática" Ed. Universidad Politécnica de Valencia. ISBN: 978-84-9048-261-2.

[2] Hofmann-Wellenhof, B., Lichtenegger, H. and Wasle, E. (2008). "GNSS Global Navigation Satellite Systems. GPS, GLONASS, Galileo and more". SpringerWienNewYork. ISBN: 978-3-211-73012-6.

[3] Matworks. Matlab products. https://es.mathworks.com/products/matlab.html

[4] Convenio de OODD de la provincia de Valencia (2017). https://fesmcugtpv.org/cms/phocadownload/oficinas/oficinas-despachos-valencia/tablas-salariales-2017.pdf

[5] Seguridad Social. https://www.gerencie.com/seguridad-social-para-trabajadores-independientes-y-verificacion-del-pago-de-aportes-por-parte-del-contratante.html

[6] Días laborables. http://www.dias-laborables.es/dias_laborables_feriados_2017_Comunidad%20Valenciana.htm

[7] Presupuesto por Contrata. https://itec.es/informacio/ConfeccionPresupuesto-PEC.aspx

Agency Space Europena (ESA). Navipedia. http://www.navipedia.net

Artano. Artano Pérez, K. "Ejercicios Prácticos para el procesado de datos GNSS - PBGNSS". Proyecto Fin de Máster.

EasySuiteII. Kai Borre. "GPS Easy Suite II: A Matlab Companion (Aalborg University)".

EUREF. http://www.epncb.oma.be/_productsservices/coord_trans/

Formato ficheros RINEX. https://emedia.rmit.edu.au/satellite/node/21 http://gage.upc.edu/gFD

ggenluz. http://ggenluz.blogspot.com.es/2015/02/

gpsg.mit. http://www-gpsg.mit.edu/~tah/GGMatlab/

gsc-europa. https://www.gsc-europa.eu/bernese-gnss-software-from-bern-university

gvacartografic. https://gvacartografic.wordpress.com/2012/05/17/red-de-estaciones-de-referencia-de-la-comunitat-valenciana-erva/

ICV. ftp://icvficheros.icv.gva.es

IGS. http://igscb.jpl.nasa.gov

Nagarvil. https://nagarvil.webs.upv.es/bernese-redes-gnss/

neredia. http://nereida.deioc.ull.es/~pcgull/ihiu01/cdrom/matlab/contenido/node2.html

notasdegeodesia.http://notasdegeodesia.blogspot.com.es/2009/08/principio-de-los-navegadores-gps.html

scielo.http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0122-97612012000200001

uam.http://www.uam.es/personal_pdi/ciencias/barcelo/cnumerico/recursos/interpolacion.html

Unavco. https://www.unavco.org/software/data-processing/postprocessing

ANEXOS

ANEXO 1: Scripts y Funciones

- **Funciones de Interpolación**

Inter_Lagrange.m

```
% Programa  : Inter_Lagrange.m
% Objeto   : Cálculo de la posición de los satélites mediante
%            interpolación por el polinomio de Lagrange a partir de
%            efemérides precisas.
% Recibe   : Fichero de efemérides '.sp3' que contiene:
%            - Posición de los satélites cada 15 minutos en solución
final.
% Devuelve : Posición de los satélites interpolando cada 't' segundos.
% Autor    : Jorge Hernández Olcina

% function [pos_int] = Inter_Lagrange(ef_ant,ef_prec,ef_sig)
Inicio = datestr(now) % Muestra por pantalla la hora de inicio
%-------------------------------------------------------
%              PREPARACIÓN DE LOS DATOS
%-------------------------------------------------------

% Archivos de salida
    satelites=fopen('MatrizCoorSat.txt','w');
    epocas=fopen('Epocas.txt','w');
    interpolacion=fopen('Interpolacion.txt','w');
% Carga de los datos:
    % Archivo día anterior
    ef_ant = 'igs19342.sp3';
    ea = fopen(ef_ant,'rt');
    % Archivo día completo
    ef_prec = 'igs19343.sp3';
    ep = fopen(ef_prec,'rt');
    % Archivo día siguiente
    ef_sig = 'igs19344.sp3';
    es = fopen(ef_sig,'rt');
% Almacenado de los datos archivos:
    % Datos día anterior
    cont = 0;
    for i=1:3190
        lin = fgets(ea);
        % Archivo de coordenadas de los satelites
        if i>=1608 && cont~=32
           fprintf(satelites,lin);
           cont = cont +1;
        % Archivo Epocas
        else
           if i>=1607
              cont = 0;
              lin = strtok(lin,'*'); % Elimina el '*' que hay delante
de las epocas
              fprintf(epocas,lin);
           end
        end
    end
    % Datos día completo
    cont = 0;
    for i=1:3190
        lin = fgets(ep);
```

```
        % Archivo de coordenadas de los satelites
        if i>=24 && cont~=32
           fprintf(satelites,lin);
           cont = cont +1;
        % Archivo Epocas
        else
           if i>=23
              cont = 0;
              lin = strtok(lin,'*'); % Elimina el '*' que hay delante
de las epocas
              fprintf(epocas,lin);
           end
        end
    end
    % Datos día siguiente
    cont = 0;
    for i=1:1639
        lin = fgets(es);
        % Archivo de coordenadas de los satelites
        if i>=24 && cont~=32
           fprintf(satelites,lin);
           cont = cont +1;
        % Archivo Epocas
        else
           if i>=23
              cont = 0;
              lin = strtok(lin,'*'); % Elimina el '*' que hay delante
de las epocas
              fprintf(epocas,lin);
           end
        end
    end

%-------------------------------------------------------
%                    INTERPOLACIÓN
%-------------------------------------------------------

% Tiempo de interpolacion
t = 30;
% Conversion del tiempo
tmp = load('Epocas.txt');
[m,n]=size(tmp);
a=zeros(m,1);
for i=1:m
    % Crea una matriz de tiempos (en segundos)
    if i<=48 % Para las epocas del dia anterior
        hora = (tmp(i,4)-24)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    elseif i>=145 % Para las epocas del dia siguiente
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60 + tmp(146,5)*60;
        a(i,1) = a(144,1) + minutos + hora;
    else % Para las epocas del dia
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    end
end
```

```
% Interpolacion
coorsat = 'MatrizCoorSat.txt';
%coor = load('MatrizCoorSat.txt'); no funciona
%[m1,n1]=size(coor); no funciona
syms t; % Define la variable 't'
% Apertura Datos
cr = fopen(coorsat,'rt');

for h=1:32
    ftx=0; % Polinomio para las X
    fty=0; % Polinomio para las Y
    ftz=0; % Polinomio para las Z
    cont=0; % Contador matriz tiempo
    Satelite = h % Muestra por pantalla el satelite con el que esta
interpolando
    cr1 = fopen(coorsat,'rt'); % Apertura Datos
    % Carga del satelite
    linsat = fgets(cr);
    div = strsplit(linsat);
    % Nombre del satelite
    sat = div(1,1);
    for i=1:32:197632
        lin = fgets(cr1);
        lin = strsplit(lin);
        if (strcmp(lin(1,1),sat)==1)
            L=1;
            cont=cont+1;
            % Coordenada X
            ftjx = str2double(lin(1,2));
            % Coordenada Y
            ftjy = str2double(lin(1,3));
            % Coordenada Z
            ftjz = str2double(lin(1,4));
            for j=1:m
                if j~=cont
                    L=L*(t-a(j))/(a(cont)-a(j));
                end
            end
            % Polinomio X
            L1=L*ftjx;
            ftx=ftx+L1;
            % Polinomio Y
            L2=L*ftjy;
            fty=fty+L2;
            % Polinomio Z
            L3=L*ftjz;
            ftz=ftz+L3;
        else
            lin='';
        end
    end
    % Almacenar en archivo la interpolacion cada 30 seg
    cont1=0;
    for k=1:2881
        % Sustituir en la variable tiempo
        cx = subs(ftx,cont1);
        cy = subs(fty,cont1);
        cz = subs(ftz,cont1);
        % Convertir a 'double'
```

```
        coorx = double(cx);
        coory = double(cy);
        coorz = double(cz);
        % Imprime
        fprintf(interpolacion,'%1.0f',h);
        fprintf(interpolacion,'%20.6f',coorx);
        fprintf(interpolacion,'%20.6f',coory);
        fprintf(interpolacion,'%20.6f',coorz);
        fprintf(interpolacion,'%10.0f\n',cont1);
        % Cambio de epoca
        cont1=cont1+30;
    end
end
Fin = datestr(now) % Muestra por pantalla la hora de fin
```

Inter_Lagrange2.m

```
% Programa : Inter_Lagrange2.m
% Objeto   : Cálculo de la posición de los satélites mediante
%            interpolación por el polinomio de Lagrange a partir de
%            efemérides precisas.
% Recibe   : Fichero de efemérides '.sp3' que contiene:
%            - Posición de los satélites cada 15 minutos en solución
final.
% Devuelve : Posición de los satélites interpolando cada 't' segundos.
% Autor    : Jorge Hernández Olcina

% function [pos_int] = Inter_Lagrange(ef_ant,ef_prec,ef_sig)
Inicio = datestr(now) % Muestra por pantalla la hora de inicio
%-------------------------------------------------------
%              PREPARACIÓN DE LOS DATOS
%-------------------------------------------------------

% Archivos de salida
    satelites=fopen('MatrizCoorSat2.txt','w');
    epocas=fopen('Epocas2.txt','w');
    interpolacion=fopen('Interpolacion2.txt','w');
    est_clk=fopen('Estado_Reloj2.txt','w');
% Carga de los datos:
    % Archivo día anterior
    ef_ant = 'igs19342.sp3';
    ea = fopen(ef_ant,'rt');
    % Archivo día completo
    ef_prec = 'igs19343.sp3';
    ep = fopen(ef_prec,'rt');
    % Archivo estado del reloj
    reloj = 'igs19343.clk_30s';
    clk = fopen(reloj,'rt');
% Almacenado de los datos archivos:
    % Datos día anterior
    cont = 0;
    for i=1:3190
        lin = fgets(ea);
        % Archivo de coordenadas de los satelites
        if i>=2928 && cont~=32
           fprintf(satelites,lin);
           cont = cont +1;
        % Archivo Epocas
        else
```

```
            if i>=2927
                cont = 0;
                lin = strtok(lin,'*'); % Elimina el '*' que hay delante
de las epocas
                fprintf(epocas,lin);
            end
        end
    end
    % Datos día 2 horas
    cont = 0;
    for i=1:583
        lin = fgets(ep);
        % Archivo de coordenadas de los satelites
        if i>=24 && cont~=32
            fprintf(satelites,lin);
            cont = cont +1;
        % Archivo Epocas
        else
            if i>=23
                cont = 0;
                lin = strtok(lin,'*'); % Elimina el '*' que hay delante
de las epocas
                fprintf(epocas,lin);
            end
        end
    end

%-------------------------------------------------------
%                    INTERPOLACIÓN
%-------------------------------------------------------

% Tiempo de interpolacion
t = 30;
% Conversion del tiempo
tmp = load('Epocas2.txt');
[m,n]=size(tmp);
a=zeros(m,1);
for i=1:m
    % Crea una matriz de tiempos (en segundos)
    if i<=8 % Para las epocas del dia anterior
        hora = (tmp(i,4)-24)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    else % Para las epocas del dia
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    end
end

% Interpolacion
coorsat = 'MatrizCoorSat2.txt';
%coor = load('MatrizCoorSat.txt'); no funciona
%[m1,n1]=size(coor); no funciona
syms t; % Define la variable 't'
% Apertura Datos
cr = fopen(coorsat,'rt');

for h=1:32
```

```
    ftx=0; % Polinomio para las X
    fty=0; % Polinomio para las Y
    ftz=0; % Polinomio para las Z
    cont=0; % Contador matriz tiempo
    Satelite = h % Muestra por pantalla el satelite con el que esta
interpolando
    cr1 = fopen(coorsat,'rt'); % Apertura Datos
    % Carga del satelite
    linsat = fgets(cr);
    div = strsplit(linsat);
    % Nombre del satelite
    sat = div(1,1);
    for i=1:32:25600
        lin = fgets(cr1);
        lin = strsplit(lin);
        if (strcmp(lin(1,1),sat)==1)
            L=1;
            cont=cont+1;
            % Coordenada X
            ftjx = str2double(lin(1,2));
            % Coordenada Y
            ftjy = str2double(lin(1,3));
            % Coordenada Z
            ftjz = str2double(lin(1,4));
            for j=1:m
                if j~=cont
                    L=L*(t-a(j))/(a(cont)-a(j));
                end
            end
            % Polinomio X
            L1=L*ftjx;
            ftx=ftx+L1;
            % Polinomio Y
            L2=L*ftjy;
            fty=fty+L2;
            % Polinomio Z
            L3=L*ftjz;
            ftz=ftz+L3;
        else
            lin='';
        end
    end
    % Almacenar en archivo la interpolacion cada 30 seg
    cont1=0;
    for k=1:241
        % Sustituir en la variable tiempo
        cx = subs(ftx,cont1);
        cy = subs(fty,cont1);
        cz = subs(ftz,cont1);
        % Convertir a 'double'
        coorx = double(cx);
        coory = double(cy);
        coorz = double(cz);
        % Imprime
        fprintf(interpolacion,'%2.0f',h);
        fprintf(interpolacion,'%20.6f',coorx);
        fprintf(interpolacion,'%20.6f',coory);
        fprintf(interpolacion,'%20.6f',coorz);
        fprintf(interpolacion,'%10.0f\n',cont1);
        % Cambio de epoca
```

```
        cont1 = cont1 + 30;
    end
end

%-------------------------------------------------------
%                    ESTADO DEL RELOJ
%-------------------------------------------------------

% Estado del Reloj para cada epoca (30 seg)
cseg = 0;
csat = 0;
csat2 = 0;
for l=1:13458
    linclk = fgets(clk);
    linclk = strsplit(linclk);
    if l>=472 && csat~=31 && cseg<=7200
       Bias = str2double(linclk(1,10));
       Bias_Sigma = str2double(linclk(1,11));
       satelite = strtok(linclk(1,2),'G');
       fprintf(est_clk,'%2.0f',str2double(satelite));
       fprintf(est_clk,'%10.0f',cseg);
       fprintf(est_clk,'%30.18f',Bias);
       fprintf(est_clk,'%30.24f\n',Bias_Sigma);
       % Contador satelites
       csat = csat + 1;
       csat2 = csat2 + 1;
    elseif csat2 == 319
        if cseg == 6870;
            for m=1:216
              linclk = fgets(clk);
              csat2 = -1;
            end
        else
          for m=1:217
              linclk = fgets(clk);
              csat2 = -1;
              if m==217
                 linclk = strsplit(linclk);
                 if strcmp(linclk(1,2),'YEL2') == 1
                    for i=1:6
                        linclk = fgets(clk);
                        linclk = strsplit(linclk);
                    end
                 else
                    if strcmp(linclk(1,2),'YELL') == 1
                        for i=1:5
                            linclk = fgets(clk);
                            linclk = strsplit(linclk);
                        end
                    else
                        if strcmp(linclk(1,2),'YKRO') == 1
                            for i=1:4
                                linclk = fgets(clk);
                                linclk = strsplit(linclk);
                            end
                        else
                           if strcmp(linclk(1,2),'YSSK') == 1
                              for i=1:3
                                  linclk = fgets(clk);
                                  linclk = strsplit(linclk);
```

```
                                end
                            else
                                if strcmp(linclk(1,2),'ZAMB') == 1
                                    for i=1:2
                                        linclk = fgets(clk);
                                        linclk = strsplit(linclk);
                                    end
                                else
                                    if strcmp(linclk(1,2),'ZECK') == 1
                                        linclk = fgets(clk);
                                        linclk = strsplit(linclk);
                                    else
                                        if strcmp(linclk(1,2),'ZIM2') ==
0
                                            linclk = fgets(clk);
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    elseif csat==31
        csat = 0;
        csat2 = csat2 + 1;
        % Cambio de epoca
        cseg = cseg + 30;
    else
    end
end
Fin = datestr(now) % Muestra por pantalla la hora de fin
```

Inter_Lgrge_Easy3.m

```
% Función  : Inter_Lgrge_Easy3.m
% Objeto   : Cálculo de la posición de los satélites mediante
%            interpolación por el polinomio de Lagrange a partir de
%            efemérides precisas.
% Recibe   : Fichero de efemérides '.sp3' que contiene:
%            - Posición de los satélites cada 15 minutos en solución
final.
% Devuelve : Posición de los satélites interpolando cada 't' segundos.
% Autor    : Jorge Hernández Olcina

function [pos_int] = Inter_Lgrge_Easy3(satelite,t_GPS,coorsat,tmp)

%-------------------------------------------------------
%                    INTERPOLACIÓN
%-------------------------------------------------------

% Conversion del tiempo
[m,n]=size(tmp);
a=zeros(m,1);
for i=1:m
    % Crea una matriz de tiempos (en segundos)
```

```
    if i<=1 % Para las epocas del dia anterior
        hora = (tmp(i,4)-24)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    else % Para las epocas del dia
        hora = tmp(i,4)*3600;
        minutos = tmp(i,5)*60;
        a(i,1) = hora + minutos;
    end
end

% Interpolacion
syms t; % Define la variable 't'
% Apertura Datos
ftx=0; % Polinomio para las X
fty=0; % Polinomio para las Y
ftz=0; % Polinomio para las Z
cont=0; % Contador matriz tiempo
%div = strsplit(linsat);
% Nombre del satelite
sat = satelite;
for i=1:11
    lin = coorsat(i,:);
    satel = lin(1);
    if satel==sat
        L=1;
        cont=cont+1;
        % Coordenada X
        ftjx = lin(1,2);
        % Coordenada Y
        ftjy = lin(1,3);
        % Coordenada Z
        ftjz = lin(1,4);
        for j=1:m
            if j~=cont
                L=L*(t-a(j))/(a(cont)-a(j));
            end
        end
        % Polinomio X
        L1=L*ftjx;
        ftx=ftx+L1;
        % Polinomio Y
        L2=L*ftjy;
        fty=fty+L2;
        % Polinomio Z
        L3=L*ftjz;
        ftz=ftz+L3;
    end
end
% Sustituir en la variable tiempo
cx = subs(ftx,t_GPS);
cy = subs(fty,t_GPS);
cz = subs(ftz,t_GPS);
% Convertir a 'double'
pos_int(1,1) = double(cx)*1000;
pos_int(2,1) = double(cy)*1000;
pos_int(3,1) = double(cz)*1000;
```

- **Rutina Easy3 modificada**

Easy3.m (modificado)

```
% EASY3   Read RINEX navigation file reformat into Matlab Eph matrix.
%         Open a RINEX observation file analyse the header and
identify
%         observation types. The function fepoch_0 finds epoch time
%         and observed PRNs in an OK epoch (digit 0, RTK observations
%         will have a 2 in this place). Next we read the observations
%         and use recpo_ls to get a least-squares estimate for the
%         (stand alone) receiver position.

%Kai Borre 31-10-2001
%Copyright (c) by Kai Borre
%$Revision: 1.0 $  $Date: 2001/10/31  $

Inicio = datestr(now) % Muestra por pantalla la hora de inicio
% Read RINEX ephemerides file and convert to
% internal Matlab format
rinexe('VCIA0320mod2.17n','eph.dat');
Eph = get_eph('eph.dat');
coorsat = load('MatrizCoorSat2.txt');
tmp = load('Epocas2.txt');
%Ef_Prec = load('Interpolacion2.txt','rt');
Clk = load('Estado_Reloj2.txt','rt');

% We identify the observation file and open it
ofile1 = 'VCIA0320.17o';
fid1 = fopen(ofile1,'rt');
[Obs_types1, ant_delta1, ifound_types1, eof11] = anheader(ofile1);
NoObs_types1 = size(Obs_types1,2)/2;
Post = [];
Pose = [];
Xtt = [];
Ytt = [];
Ztt = [];
Xee = [];
Yee = [];
Zee = [];

% There are 240 epochs of data in ofile1
for q = 1:240
    [time1, dt1, sats1, eof1] = fepoch_0(fid1);
    NoSv1 = size(sats1,1);
    % We pick the observed P2 pseudoranges
    obs1 = grabdata(fid1, NoSv1, NoObs_types1);
    i = fobs_typ(Obs_types1,'P2');
    [Xt,Yt,Zt,post] = recpo_lsa(obs1(:,i),sats1,time1,Eph,q);% Calcula
la posicion con Ef Transmitidas
    [Xe,Ye,Ze,pose] =
recpo_efp(obs1(:,i),sats1,time1,Clk,q,coorsat,tmp);
    Post = [Post post];
    Pose = [Pose pose];
    % Almacen de coordenadas
        % Efemerides Transmitidas
        Xtt = [Xtt;Xt];
        Ytt = [Ytt;Yt];
        Ztt = [Ztt;Zt];
```

```
        % Efemerides Precisas
        Xee = [Xee;Xe];
        Yee = [Yee;Ye];
        Zee = [Zee;Ze];
end
c = input('Indica el satélite que desee dibujar (0 para dibujar
todos): ');
elipsoide = parelip('GRS80');
% Coordenadas Cabecera
[latc,longc,helc]=trigeo(4932702.9440,-
29607.7882,4029833.0400,elipsoide);
[xpc,ypc]=geoutm(latc,longc,elipsoide);
fprintf('\nCoordenadas de la cabecera del fichero de observaciones:')
fprintf('\n\n   X: %13.3f     Y: %12.3f  Z: %12.3f\n', 4932702.9440,-
29607.7882,4029833.0400)
fprintf('\n   Lat: %11.8f   Lon: %12.8f  Helip: %8.3f\n',
rad_psdo(latc),rad_psdo(longc),helc)
fprintf('\n   Xutm: %10.3f  Yutm: %12.3f  Helip: %8.3f\n',
xpc,ypc,helc)
% Efemerides Transmitidas
me = mean(Post,2);
[lat,long,hel]=trigeo(me(1,1),me(2,1),me(3,1),elipsoide);
[xp,yp]=geoutm(lat,long,elipsoide);
fprintf('\nPosición media para 240 épocas, con Efemérides
Transmitidas:')
fprintf('\n\n   X: %13.3f     Y: %12.3f  Z: %12.3f\n', me(1,1),
me(2,1), me(3,1))
fprintf('\n   Lat: %11.8f   Lon: %12.8f  Helip: %8.3f\n',
rad_psdo(lat),rad_psdo(long),hel)
fprintf('\n   Xutm: %10.3f  Yutm: %12.3f  Helip: %8.3f\n', xp,yp,hel)
% Efemerides Precisas
me1 = mean(Pose,2);
[lat1,long1,hel1]=trigeo(me1(1,1),me1(2,1),me1(3,1),elipsoide);
[xp1,yp1]=geoutm(lat1,long1,elipsoide);
fprintf('\nPosición media para 240 épocas, con Efemérides Precisas:')
fprintf('\n\n   X: %13.3f     Y: %12.3f  Z: %12.3f\n', me1(1,1),
me1(2,1), me1(3,1))
fprintf('\n   Lat: %11.8f   Lon: %12.8f  Helip: %8.3f\n',
rad_psdo(lat1),rad_psdo(long1),hel1)
fprintf('\n   Xutm: %10.3f  Yutm: %12.3f  Helip: %8.3f\n',
xp1,yp1,hel1)
% Dibujo Efemerides Transmitidas
plot((Post(1:3,:)-Post(1:3,1)*ones(1,q))','linewidth',2)
title('Posición a lo largo del Tiempo','fontsize',16)
legend('X','Y','Z')
xlabel('Épocas [Intervalo de 30 s]','fontsize',16)
ylabel('Variación en Coordenadas, respecto a la 1ª época
[m]','fontsize',16)
set(gca,'fontsize',16)
legend
figure
% Dibujo Efemerides Precisas
plot((Pose(1:3,:)-Pose(1:3,1)*ones(1,q))','linewidth',2)
title('Posición a lo largo del Tiempo','fontsize',16)
legend('X','Y','Z')
xlabel('Épocas [Intervalo de 30 s]','fontsize',16)
ylabel('Variación en Coordenadas, respecto a la 1ª época
[m]','fontsize',16)
set(gca,'fontsize',16)
legend
```

```
% Órbitas Satélites
[a,b] = size(Xtt);
cont = 0;
Xtt1 = [];
Ytt1 = [];
Ztt1 = [];
Xee1 = [];
Yee1 = [];
Zee1 = [];
for i=1:a
    sat = Xtt(i,2);
    if sat==c
       cont = cont + 1;
       Xtt1(cont,1) = Xtt(i,1);
       Ytt1(cont,1) = Ytt(i,1);
       Ztt1(cont,1) = Ztt(i,1);
       Xee1(cont,1) = Xee(i,1);
       Yee1(cont,1) = Yee(i,1);
       Zee1(cont,1) = Zee(i,1);
    elseif c==0
       Xtt1(i,1) = Xtt(i,1);
       Ytt1(i,1) = Ytt(i,1);
       Ztt1(i,1) = Ztt(i,1);
       Xee1(i,1) = Xee(i,1);
       Yee1(i,1) = Yee(i,1);
       Zee1(i,1) = Zee(i,1);
    end
end
if isempty(Xtt1)
   fprintf('Satélite no observado o valor introducido incorrecto\n')
else
    figure
    plot3(Xtt1,Ytt1,Ztt1,'r.')
    title('Posición del satélite con Efemérides
Transmitidas','fontsize',16)
    figure
    plot3(Xee1,Yee1,Zee1,'b.')
    title('Posición del satélite con Efemérides
Precisas','fontsize',16)
end

% Cálculo de la diferencia vectorial entre posicion transmitidas y
precisas
fprintf('\nMódulo del vector diferencial entre la posición del
satélite con Ef Precisas y con Ef Transmitidas:\n\n')
figure
for j=1:32
    cnt = 0;
    VecDif = [];
    VecDif2 = 0;
    Xt = [];
    Yt = [];
    Zt = [];
    Xe = [];
    Ye = [];
    Ze = [];
    for k=1:a
        satelite = Xtt(k,2);
        if satelite==j
           % Calculo del vector
```

```
            cnt = cnt + 1;
            VecDif(cnt,1) = sqrt((Xtt(k,1)-Xee(k,1))^2+(Ytt(k,1)-
Yee(k,1))^2+(Ztt(k,1)-Zee(k,1))^2);
            VecDif2 = VecDif2 + VecDif(cnt,1);
            % Ploteo de los satelites
            Xt(cnt,1) = Xtt(k,1);
            Yt(cnt,1) = Ytt(k,1);
            Zt(cnt,1) = Ztt(k,1);
            Xe(cnt,1) = Xee(k,1);
            Ye(cnt,1) = Yee(k,1);
            Ze(cnt,1) = Zee(k,1);
        end
    end
    if VecDif ~= 0
        % Media y Desviación típica
        VecDif2 = VecDif2/cnt;
        [c,d] = size(VecDif);
        SumXi_Xmed = 0;
        Xi_Xmed = 0;
        for l=1:c
            Xi_Xmed = (VecDif(l,1) - VecDif2)^2;
            SumXi_Xmed = SumXi_Xmed + Xi_Xmed;
        end
        DesvEst = sqrt(SumXi_Xmed/(cnt-1));
        fprintf('   Satélite: %2.0f    Diferencia (m): %2.3f
Desviación Típica: %2.3f\n', j, VecDif2, DesvEst)
        % Ploteo de los satelites
        l = line(Xt,Yt,'LineWidth',1);
        set(l,'Color','r');
        hold on
        l1 = line(Xe,Ye,'LineWidth',1);
        set(l1,'Color','b');
        hold on
        plot(Xt,Yt,'r.')
        hold on
        plot(Xe,Ye,'b.')
        if j==1
            title('Satélite G01','fontsize',16)
        elseif j==2
            title('Satélite G02','fontsize',16)
        elseif j==3
            title('Satélite G03','fontsize',16)
        elseif j==4
            title('Satélite G04','fontsize',16)
        elseif j==5
            title('Satélite G05','fontsize',16)
        elseif j==6
            title('Satélite G06','fontsize',16)
        elseif j==7
            title('Satélite G07','fontsize',16)
        elseif j==8
            title('Satélite G08','fontsize',16)
        elseif j==9
            title('Satélite G09','fontsize',16)
        elseif j==10
            title('Satélite G10','fontsize',16)
        elseif j==11
            title('Satélite G11','fontsize',16)
        elseif j==12
            title('Satélite G12','fontsize',16)
```

```
        elseif j==13
            title('Satélite G13','fontsize',16)
        elseif j==14
            title('Satélite G14','fontsize',16)
        elseif j==15
            title('Satélite G15','fontsize',16)
        elseif j==16
            title('Satélite G16','fontsize',16)
        elseif j==17
            title('Satélite G17','fontsize',16)
        elseif j==18
            title('Satélite G18','fontsize',16)
        elseif j==19
            title('Satélite G19','fontsize',16)
        elseif j==20
            title('Satélite G20','fontsize',16)
        elseif j==21
            title('Satélite G21','fontsize',16)
        elseif j==22
            title('Satélite G22','fontsize',16)
        elseif j==23
            title('Satélite G23','fontsize',16)
        elseif j==24
            title('Satélite G24','fontsize',16)
        elseif j==25
            title('Satélite G25','fontsize',16)
        elseif j==26
            title('Satélite G26','fontsize',16)
        elseif j==27
            title('Satélite G27','fontsize',16)
        elseif j==28
            title('Satélite G28','fontsize',16)
        elseif j==29
            title('Satélite G29','fontsize',16)
        elseif j==30
            title('Satélite G30','fontsize',16)
        elseif j==31
            title('Satélite G31','fontsize',16)
        else
            title('Satélite G32','fontsize',16)
        end
        legend('Órbita Ef Transmitidas','Órbita Ef Precisas','Posición
Sat Ef Trans','Posición Sat Ef Prec')
        if j<=31
            figure
        end
    end
end
print -deps easy3

Fin = datestr(now) % Muestra por pantalla la hora de fin
clear all
%%%%%%%%%%%%%%%%%%%% end easy3.m %%%%%%%%%%%%%%%%
```

Recpo_lsa.m

```
function [Xt,Yt,Zt,pos] = recpo_lsa(obs,sats,time,Eph,q)
% RECPO_LS Computation of receiver position from pseudoranges
%          using ordinary least-squares principle
```

```
%Kai Borre 31-10-2001
%Copyright (c) by Kai Borre
%$Revision: 1.1 $  $Date: 2002/07/10  $

% easy3 modified for computing the Hessian matrix
% of the observation equations;
% and mean vector for satellite directions
% to be compared with the eigenvalues of inv(A'*A)

% December 11, 2005

% Elimininamos los satelites de los que no se tienen datos
a = size(obs);
obs1 = obs;
for i=1:a
    b = obs(i,1);
    if isnan(b)
       obs1(i) = [];
       sats(i) =[];
       a = a - 1;
    end
end
obs = obs1;

global H

v_light = 299792458;
dtr = pi/180;
m = size(obs,1);  % number of svs
el = zeros(m,1);
% identify ephemerides columns in Eph
for t = 1:m
    col_Eph(t) = find_eph(Eph,sats(t),time);
end
% preliminary guess for receiver position and receiver clock offset
pos = zeros(4,1);
no_iterations = 6;
ps_corr = [];
sat_pos = [];

for iter = 1:no_iterations
    A = [];
    H = [];
    omc = []; % observed minus computed observation
    dir_vector = [];

    for i = 1:m
        satelite = sats(i,1);
        if q >= 120
            k = col_Eph(i);
            k= k + 32;
        else
            k = col_Eph(i);
        end
        tx_RAW = time - obs(i)/v_light;
        t0c = Eph(21,k);
        dt = check_t(tx_RAW-t0c);
```

```
        tcorr = (Eph(2,k)*dt + Eph(20,k))*dt + Eph(19,k);
        tx_GPS = tx_RAW-tcorr;
        dt = check_t(tx_GPS-t0c);
        tcorr = (Eph(2,k)*dt + Eph(20,k))*dt + Eph(19,k);
        tx_GPS = tx_RAW-tcorr;
        X = satpos(tx_GPS, Eph(:,k));
        %X1 = satpos(time, Eph(:,k));
        if iter == 1
            traveltime = 0.072;
            Rot_X = X;
            trop = 0;
            dion=0;
        else
            rho2 = (X(1)-pos(1))^2+(X(2)-pos(2))^2+(X(3)-pos(3))^2;
            traveltime = sqrt(rho2)/v_light;
            Rot_X = e_r_corr(traveltime,X);
            rho2 = (Rot_X(1)-pos(1))^2+(Rot_X(2)-pos(2))^2+(Rot_X(3)-
pos(3))^2;
            [az,el,dist,phi,lambda] = topocent(pos(1:3,:),Rot_X-
pos(1:3,:));

            if iter == no_iterations, El(i) = el; end
            trop = tropo(sin(el*dtr),0.0629513,1016.25,286.8,0.0,...
                0.0,0.0,0.0);
            dion = ion(az,el,phi,lambda,tx_GPS);
            %dion=0;
        end
        % subtraction of pos(4) corrects for receiver clock offset and
        % v_light*tcorr is the satellite clock offset
        if iter == no_iterations
            ps_corr = [ps_corr; obs(i)+v_light*tcorr-trop];
            sat_pos = [sat_pos; X'];
        end
        omc = [omc; obs(i)-norm(Rot_X-pos(1:3),'fro')-
pos(4)+v_light*tcorr-trop-dion];
        A = [A; (-(Rot_X(1)-pos(1)))/obs(i)...
                (-(Rot_X(2)-pos(2)))/obs(i) ...
                (-(Rot_X(3)-pos(3)))/obs(i) 1];
                % Compute a unit vector at receiver position pointing
to
                % satellite
                unit_vec = (Rot_X(1:3,1)-
pos(1:3,1))/norm(Rot_X(1:3,1)-pos(1:3,1));
                dir_vector = [dir_vector unit_vec];
    % Almacen coordenadas satelites
    Xt(i,1)=X(1,1); % Coordenada
    Xt(i,2)=satelite; % Satelite
    Yt(i,1)=X(2,1); % Coordenada
    Yt(i,2)=satelite; % Satelite
    Zt(i,1)=X(3,1); % Coordenada
    Zt(i,2)=satelite; % Satelite
        %end
    end % i
    x = A\omc;
    pos = pos+x;

    if iter == no_iterations, GDOP = sqrt(trace(inv(A'*A)));
        %% two lines that solve an exercise on computing tdop
        % invm = inv(A'*A);
        % tdop = sqrt(invm(4,4))
```

```
    end
end % iter

%    dir_vector = mean(dir_vector,2)
%    Sigma = inv(A(1:3,1:3)'*A(1:3,1:3))
%   [a,b] = eig(Sigma)


basic_obs = [sat_pos ps_corr];

%%%%%%%%%%%%%%%%%%%%%%  recpo_lsa.m  %%%%%%%%%%%%%%%%%%%%%%
```

Recpo_efp.m

```
function [Xe,Ye,Ze,pos] = recpo_efp(obs,sats,time,Clk,q,coorsat,tmp)
% RECPO_LS Computation of receiver position from pseudoranges
%          using ordinary least-squares principle

%Kai Borre 31-10-2001
%Copyright (c) by Kai Borre
%$Revision: 1.1 $  $Date: 2002/07/10  $

% easy3 modified for computing the Hessian matrix
% of the observation equations;
% and mean vector for satellite directions
% to be compared with the eigenvalues of inv(A'*A)

% December 11, 2005

% Elimininamos los satelites de los que no se tienen datos
format long g
a = size(obs);
obs1 = obs;
for i=1:a
    b = obs(i,1);
    if isnan(b)
       obs1(i) = [];
       sats(i) =[];
       a = a - 1;
    end
end
obs = obs1;

global H

v_light = 299792458;
dtr = pi/180;
m = size(obs,1);  % number of svs
[m2,n2] = size(Clk);
el = zeros(m,1);
% identify ephemerides columns in Eph
%for t = 1:m
    %col_Eph(t) = find_eph(Eph,sats(t),time);
%end
% preliminary guess for receiver position and receiver clock offset
pos = zeros(4,1);
no_iterations = 6;
```

```
ps_corr = [];
sat_pos = [];

for iter = 1:no_iterations
    A = [];
    H = [];
    omc = []; % observed minus computed observation
    dir_vector = [];
    for i = 1:m
        satelite = sats(i,1);
        for l = 1:m2
            % Obtencion del estado del reloj
            satel1 = Clk (l,1);
            tmp1 = Clk (l,2);
            if ((satelite == satel1) && (((q-1)*30 == tmp1)))
               CLK(1,1) = Clk (l,3);
               CLK(2,1) = Clk (l,4);
            end
        end
        % Coordenadas del satélite
        tx_RAW = ((q-1)*30) - obs(i)/v_light;
        tcorr = CLK(1,1);
        tx_GPS = tx_RAW-tcorr;
        conta = 0;
        for z=1:352
            if coorsat(z,1)==satelite
                conta = conta + 1;
               cs(conta,:) = coorsat(z,:);
            end
        end
        %X = Inter_Lgrge_Easy3(satelite,tx_GPS,cs,tmp);
        if iter == 1
            X = Inter_Lgrge_Easy3(satelite,tx_GPS,cs,tmp);
            traveltime = 0.072;
            Rot_X = X;
            trop = 0;
            dion=0;
        else
            X = [Xe(i,1);Ye(i,1);Ze(i,1)];
            rho2 = (X(1)-pos(1))^2+(X(2)-pos(2))^2+(X(3)-pos(3))^2;
            traveltime = sqrt(rho2)/v_light;
            Rot_X = e_r_corr(traveltime,X);
            rho2 = (Rot_X(1)-pos(1))^2+(Rot_X(2)-pos(2))^2+(Rot_X(3)-
pos(3))^2;
            [az,el,dist,phi,lambda] = topocent(pos(1:3,:),Rot_X-
pos(1:3,:));

            if iter == no_iterations, El(i) = el; end
            trop = tropo(sin(el*dtr),0.0629513,1016.25,286.8,0.0,...
                0.0,0.0,0.0);
            dion = ion(az,el,phi,lambda,time);
        end
        % subtraction of pos(4) corrects for receiver clock offset and
        % v_light*tcorr is the satellite clock offset
        if iter == no_iterations
            ps_corr = [ps_corr; obs(i)+v_light*tcorr-trop];
            sat_pos = [sat_pos; X'];
        end
        omc = [omc; obs(i)-norm(Rot_X-pos(1:3),'fro')-
pos(4)+v_light*tcorr-trop-dion];
```

```
        A = [A; (-(Rot_X(1)-pos(1)))/obs(i)...
                (-(Rot_X(2)-pos(2)))/obs(i) ...
                (-(Rot_X(3)-pos(3)))/obs(i) 1];
                % Compute a unit vector at receiver position pointing
to
                % satellite
                unit_vec = (Rot_X(1:3,1)-
pos(1:3,1))/norm(Rot_X(1:3,1)-pos(1:3,1));
                dir_vector = [dir_vector unit_vec];

    % Almacen coordenadas satelites
    Xe(i,1)=X(1,1); % Coordenada
    Xe(i,2)=satelite; % Satelite
    Ye(i,1)=X(2,1); % Coordenada
    Ye(i,2)=satelite; % Satelite
    Ze(i,1)=X(3,1); % Coordenada
    Ze(i,2)=satelite; % Satelite
        %end
    end % i
    x = A\omc;
    pos = pos+x;

    if iter == no_iterations, GDOP = sqrt(trace(inv(A'*A)));
        %% two lines that solve an exercise on computing tdop
        % invm = inv(A'*A);
        % tdop = sqrt(invm(4,4))
    end
end % iter

%    dir_vector = mean(dir_vector,2)
%    Sigma = inv(A(1:3,1:3)'*A(1:3,1:3))
%   [a,b] = eig(Sigma)


basic_obs = [sat_pos ps_corr];

%%%%%%%%%%%%%%%%%%%%%  recpo_lsa.m  %%%%%%%%%%%%%%%%%%%%%
```

Fepoch_0.m

```
function [time, dt, sats, eof, datee] = fepoch_0(fid)
% FEPOCH_0   Finds the next epoch in an opened RINEX file with
%             identification fid. From the epoch line is produced
%             time (in seconds of week), number of sv.s, and a mark
%             about end of file. Only observations with epoch flag 0
%             are delt with.

%Kai Borre 09-14-96; revised 03-22-97; revised Sept 4, 2001
%Copyright (c) by Kai Borre
%$Revision: 1.0 $  $Date: 1997/09/22  $
%fide = fopen(fid,'rt');

global sat_index;
time = 0;
dt = 0;
sats = [];
NoSv = 0;
eof = 0;
```

```
while 1
   lin = fgets(fid); % earlier fgetl
   if (numel(lin) > 69)% Añadido para evitar saltos diferencias en
numero de satelites
      lin = fgets(fid);
   end
   answer = findstr(lin,'COMMENT');

   if ~isempty(answer);
      lin = fgetl(fid);
   end;


   if (feof(fid) == 1);
      eof = 1;
      break
   end;
    if ((numel(lin) == 15) || (numel(lin) == 1)) % Añadido para que
salte lineas con satelites Glonass
       lin = '     2.10           OBSERVATION DATA    G (GPS)
RINEX VERSION / TYPE';
    end
   if ((strcmp(lin(29),'0') == 0) && (size(deblank(lin),2) == 29))
      eof = 1;
      break
   end; % We only want type 0 data
   % Lectura satelites
   if ((strcmp(lin(2),'1') == 1)  &&  (strcmp(lin(29),'0') == 1))
%cambio 0 por 1 por el año
      ll = length(lin)-2;
      if ll > 60, ll = 60; end;
      linp = lin(1:ll);
      %fprintf('%60s\n',linp);
      [year, lin] = strtok(lin);
      %year;
      [month, lin] = strtok(lin);
      [day, lin] = strtok(lin);
      %month;
      day;
      [hour, lin] = strtok(lin);
      %hour
      [minute, lin] = strtok(lin);
      %minute
      [second, lin] = strtok(lin);
      %second
      [OK_flag, lin] = strtok(lin);
      h = str2num(hour)+str2num(minute)/60+str2num(second)/3600;
      jd = julday(str2num(year)+2000, str2num(month), str2num(day),
h);
      [week, sec_of_week] = gps_time(jd);
      jd;
      time = sec_of_week;
      lin = strtok(lin,'R'); % Quita los satelites de Glonass
      [NoSv, lin] = strtok(lin,'G');

      for k = 1:str2num(NoSv) % Arreglada para que meta solo satelites
GPS
          if isempty(lin) == 0
```

```
            [sat, lin] = strtok(lin,'G');
            prn(k) = str2num(sat);
         end
      end

      sats = prn(:);
      dT = strtok(lin);
      if isempty(dT) == 0
         dt = str2num(dT);
      end
      break

   end
end;
datee=[str2num(year) str2num(month) str2num(day) str2num(hour)
str2num(minute) str2num(second)];

%%%%%%%% end fepoch_0.m %%%%%%%%%
```

Grabdata.m

```
function Obs = grabdata(fid, NoSv, NoObs)
%GRABDATA Positioned in a RINEX file at a selected epoch
%       reads observations of NoSv satellites

%Kai Borre 09-13-96
%Copyright (c) by Kai Borre
%$Revision: 1.0 $  $Date: 1997/09/23  $

global lin

Obs = zeros(NoSv, NoObs);

if NoObs <= 5     % This will typical be Turbo SII data
   for u = 1:NoSv
      lin = fgetl(fid);
      for k = 1:NoObs
         Obs(u,k) = str2num(lin(2+16*(k-1):16*k-2));
      end
   end
else                % This will typical be Z12 data
   Obs = Obs(:,[1 2 3 4 5]); % We cancel the last two columns 6 and 7
   NoObs = 5;
   for u = 1:NoSv
      lin = fgetl(fid);
      if (numel(lin)<= 62)% Se añade para evitar la segunda linea que
indica satelites de observación
        lin = fgetl(fid);
      end
      lin_doppler = fgetl(fid);
      for k = 1:NoObs  %%-1
        if isempty(str2num(lin(1+16*(k-1):16*k-2))) == 1,
            Obs(u,k) = nan;
        else                                    %
        Obs(u,k) = str2num(lin(1+16*(k-1):16*k-2));
        end
     % Obs(u,NoObs) = str2num(lin(65:78));
      end
```

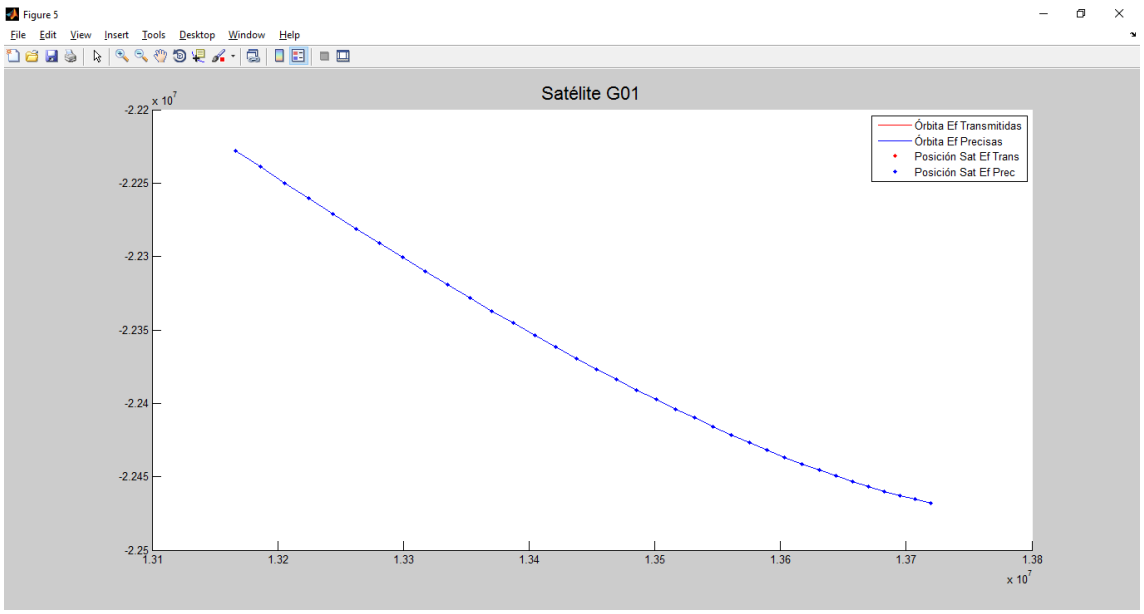
```
    end
end
%%%%%%%%% end grabdata.m %%%%%%%%%
```

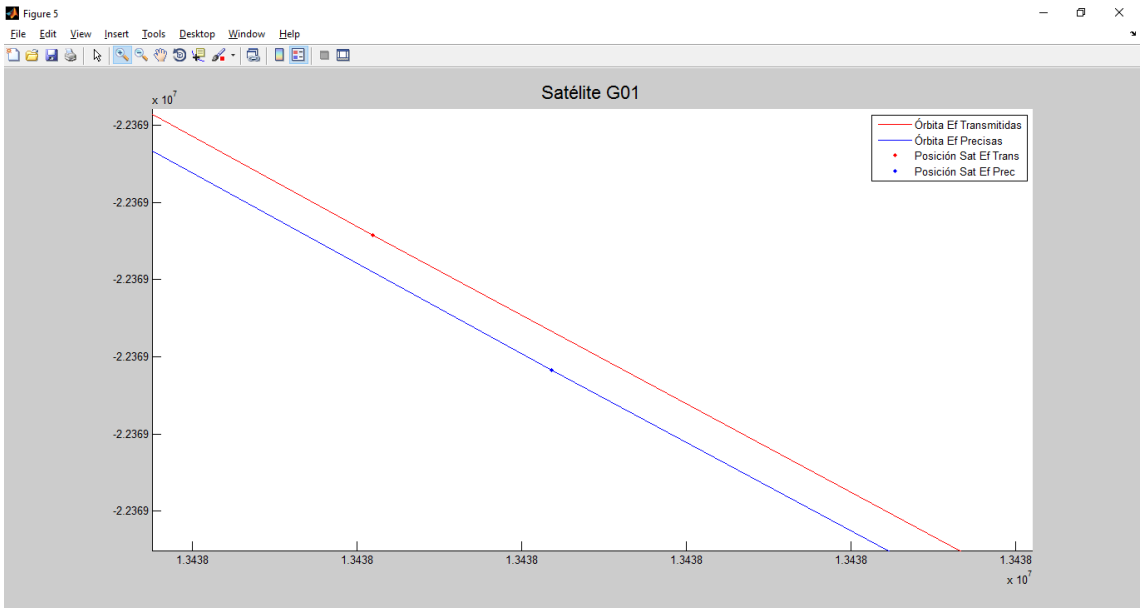
Ion.m

```
function dion = ion(az,el,lat,lon,tgps)

alpha(1)=8.3819D-09;
alpha(2)=-7.4506D-09;
alpha(3)=-5.9605D-08;
alpha(4)=5.9605D-08;
beta(1)=8.8064D+04;
beta(2)=-3.2768D+04;
beta(3)=-1.9661D+05;
beta(4)=1.9661D+05;

%wgs84con;
% global constants used: c_speed
c_speed = 2.99792458e+8;
% Calculate angles in semicircles
elsm = (el*pi/180) / pi;
azsm = (az*pi/180) / pi;
latsm = (lat*pi/180) / pi;
lonsm = (lon*pi/180) / pi;
% Compute the earth-centered angle
psi = 0.0137 / (elsm + 0.11) - 0.022;
% Compute the subionospheric latitude
iono_lat = latsm + psi * cos(az);
if (iono_lat > 0.416)
iono_lat = 0.416;
elseif (iono_lat < -0.416)
iono_lat = -0.416;
end
% Compute the subionospheric longitude
iono_lon = lonsm + (psi * sin(az)) / (cos(iono_lat * pi));
% Find the local time at the subionospheric point
localt = 43200. * iono_lon + tgps;
kk = 0;
while ( (localt >= 86400.) && (kk < 10) )
localt = localt - 86400.;
kk = kk + 1;
end
while ( (localt < 0.) && (kk < 10) )
localt = localt + 86400.;
kk = kk + 1;
end
if (kk == 10)
error('IONOC.m - error in local time computation');
end
% Calculate the geomagnetic latitude of the earth projection of the
% ionospheric intersection point
latm = iono_lat + 0.064 * cos((iono_lon - 1.617) * pi);
% Calculate the period by using the beta terms from the GPS message
per = ((beta(4)*latm + beta(3))*latm + beta(2))*latm + beta(1);
if (per < 72000.)
per = 72000.;
end
% Calculate the argument of the cosine term
```

```
x = (pi + pi) * (localt - 50400.) / per;
% Calculate the slant factor
slantf = 0.53 - elsm;
slantf = 1. + 16. * slantf * slantf * slantf;
% Calculate the amplitude by using the alpha terms from the GPS
message
amp = ((alpha(4)*latm + alpha(3))*latm + alpha(2))*latm + alpha(1);
if (amp < 0.)
amp = 0.;
end
% Calculate the L1 ionospheric correction (in meters)
% paratro para pasar a P2 en Mhz

multi=(1575.45/1227.60)^2;

if (abs(x) < 1.57)
xx = x*x;
dion = multi*(slantf * (5.e-9 + amp * ( 1. - 0.5*xx + (xx*xx/24) )
)*c_speed);
else
dion = multi*( slantf * 5.e-9 * c_speed);
end
end
```

ANEXO 2: Órbitas de los Satélites. Gráficas





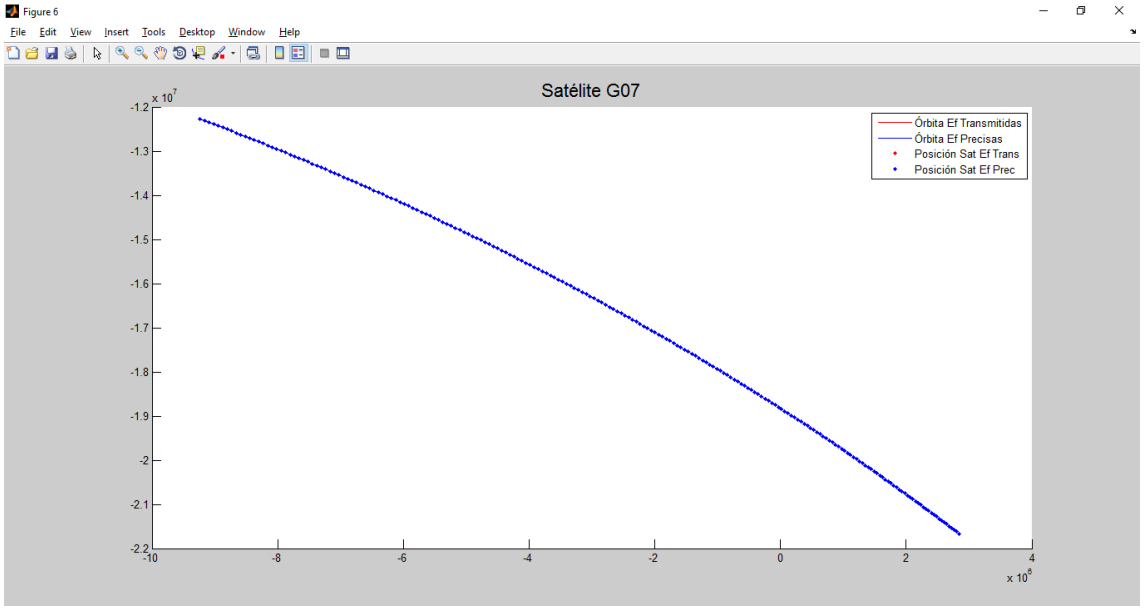


Figure 6
File Edit View Insert Tools Desktop Window Help
Satélite G07
x 10^7
-1.2
-1.3
-1.4
-1.5
-1.6
-1.7
-1.8
-1.9
-2
-2.1
-2.2
-10
-8
-6
-4
-2
0
2
4
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

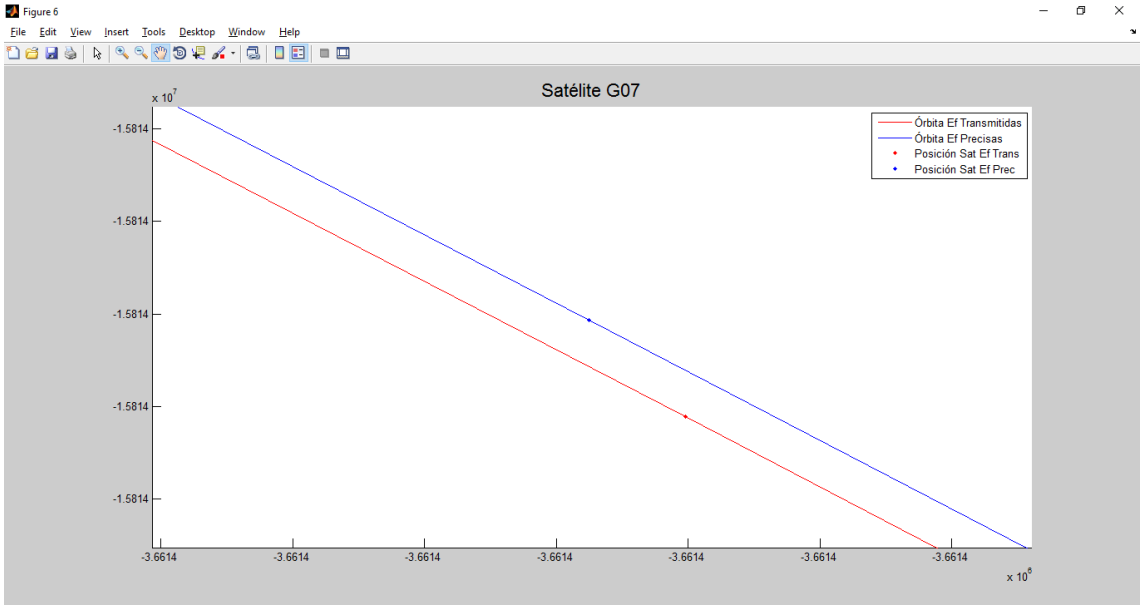


Figure 6
File Edit View Insert Tools Desktop Window Help
Satélite G07
x 10^7
-1.5814
-1.5814
-1.5814
-1.5814
-1.5814
-3.6614
-3.6614
-3.6614
-3.6614
-3.6614
-3.6614
-3.6614
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

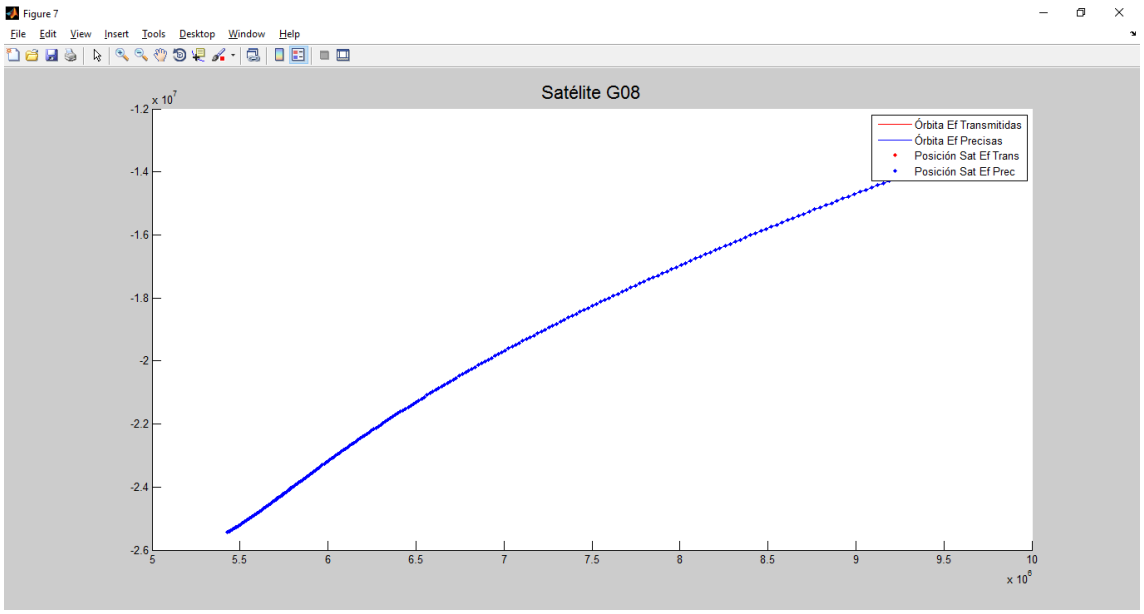


Figure 7
File Edit View Insert Tools Desktop Window Help
Satélite G08
x 10^7
-1.2
-1.4
-1.6
-1.8
-2
-2.2
-2.4
-2.6
5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

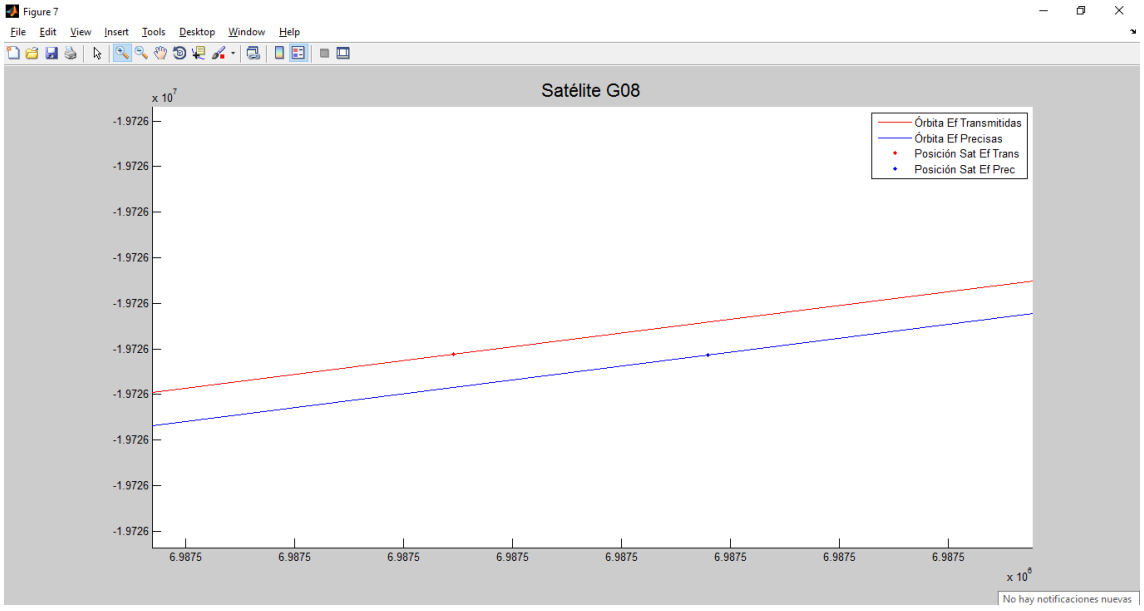


Figure 7
File Edit View Insert Tools Desktop Window Help
Satélite G08
x 10^7
-1.9726
-1.9726
-1.9726
-1.9726
-1.9726
-1.9726
-1.9726
-1.9726
-1.9726
6.9875 6.9875 6.9875 6.9875 6.9875 6.9875 6.9875 6.9875
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec
No hay notificaciones nuevas

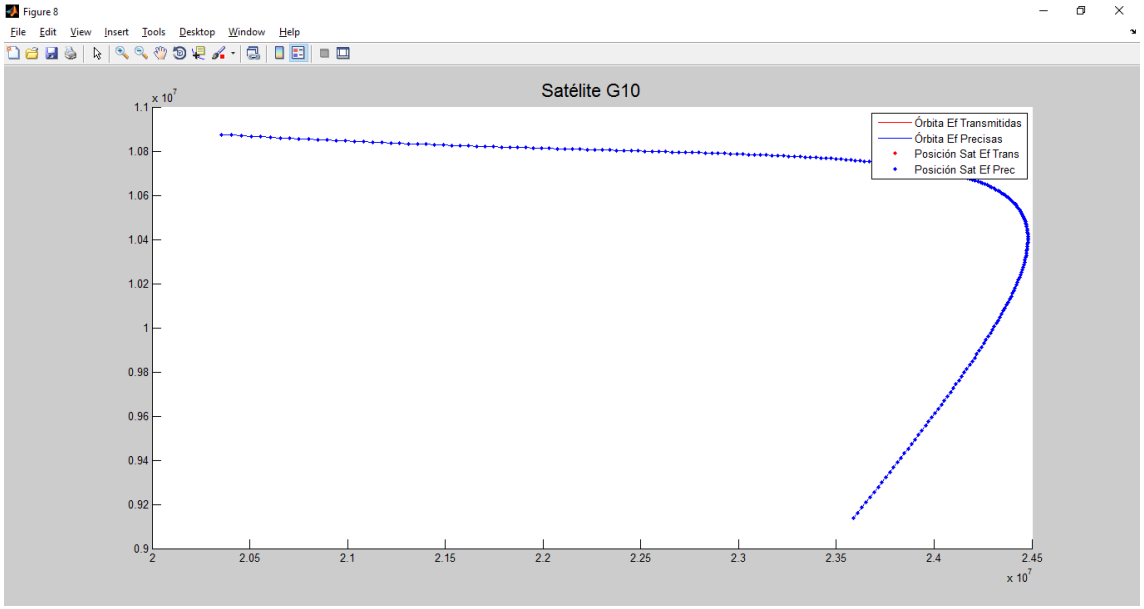


Figure 8
File Edit View Insert Tools Desktop Window Help
Satélite G10
x 10^7
1.1
1.08
1.06
1.04
1.02
1
0.98
0.96
0.94
0.92
0.9
2
2.05
2.1
2.15
2.2
2.25
2.3
2.35
2.4
2.45
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

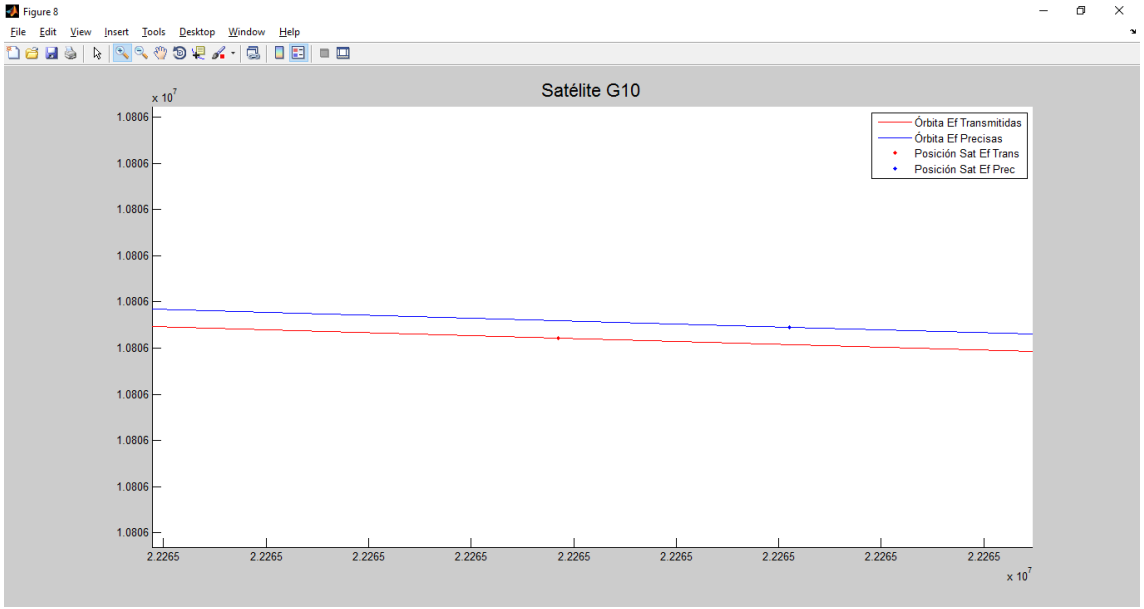


Figure 8
File Edit View Insert Tools Desktop Window Help
Satélite G10
x 10^7
1.0806
1.0806
1.0806
1.0806
1.0806
1.0806
1.0806
1.0806
1.0806
2.2265
2.2265
2.2265
2.2265
2.2265
2.2265
2.2265
2.2265
2.2265
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

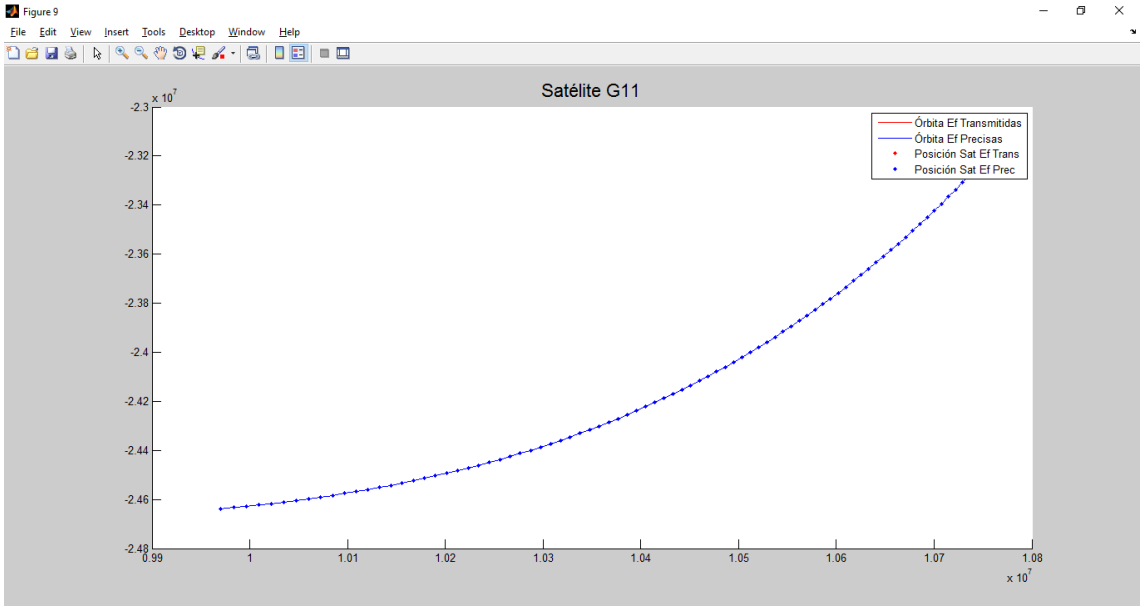


Figure 9
File Edit View Insert Tools Desktop Window Help
Satélite G11
x 10^7
-2.3
-2.32
-2.34
-2.36
-2.38
-2.4
-2.42
-2.44
-2.46
-2.48
0.99
1
1.01
1.02
1.03
1.04
1.05
1.06
1.07
1.08
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

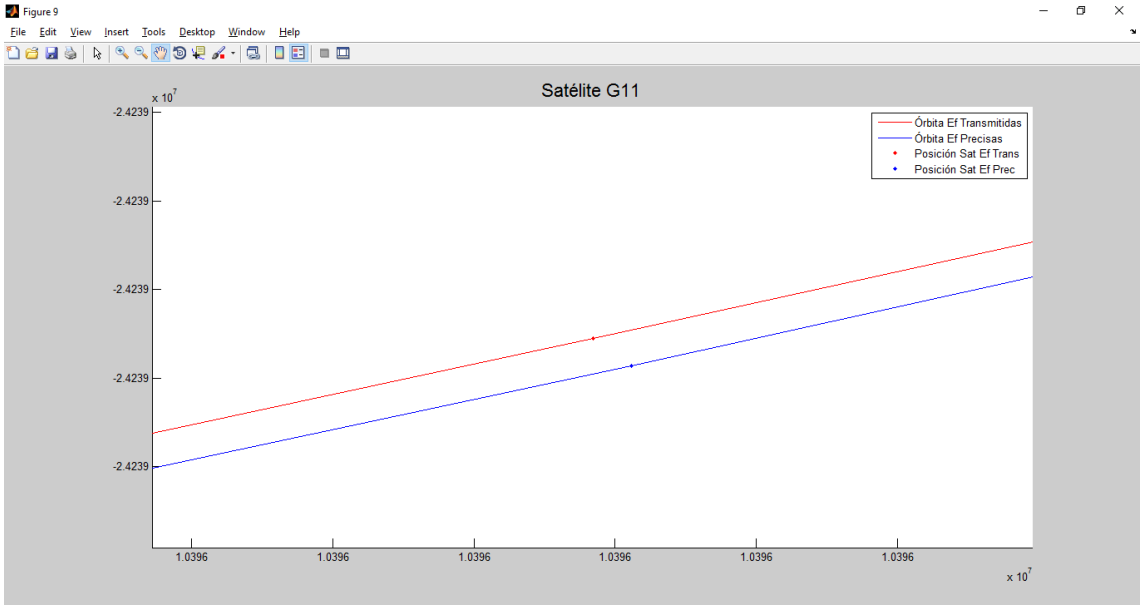


Figure 9
File Edit View Insert Tools Desktop Window Help
Satélite G11
x 10^7
-2.4239
-2.4239
-2.4239
-2.4239
-2.4239
1.0396
1.0396
1.0396
1.0396
1.0396
1.0396
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

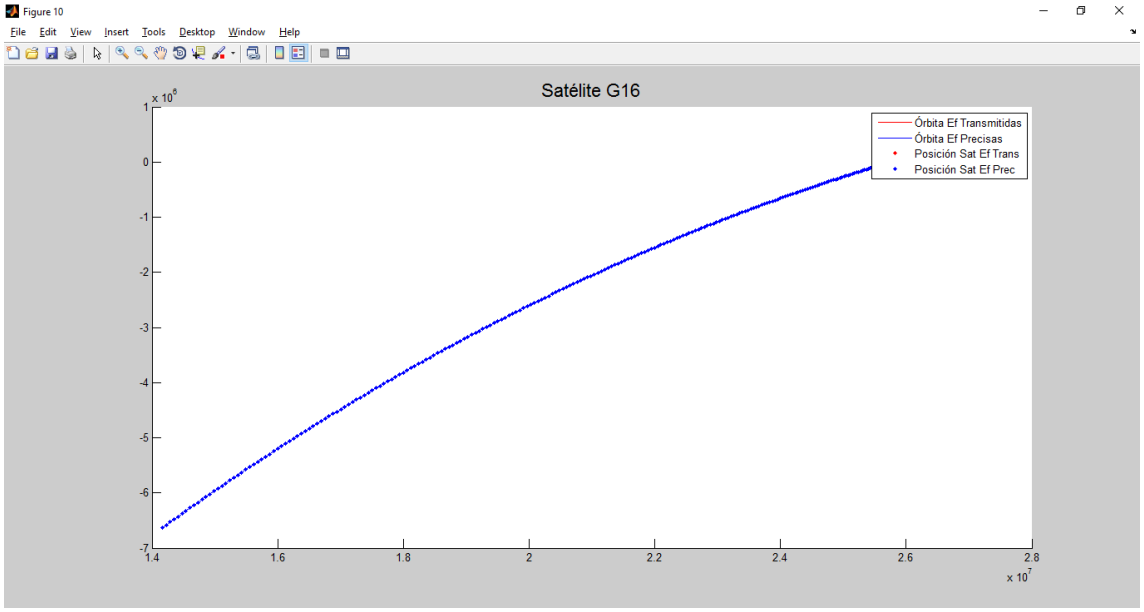


Figure 10
File Edit View Insert Tools Desktop Window Help
Satélite G16
x 10^6
1
0
-1
-2
-3
-4
-5
-6
-7
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

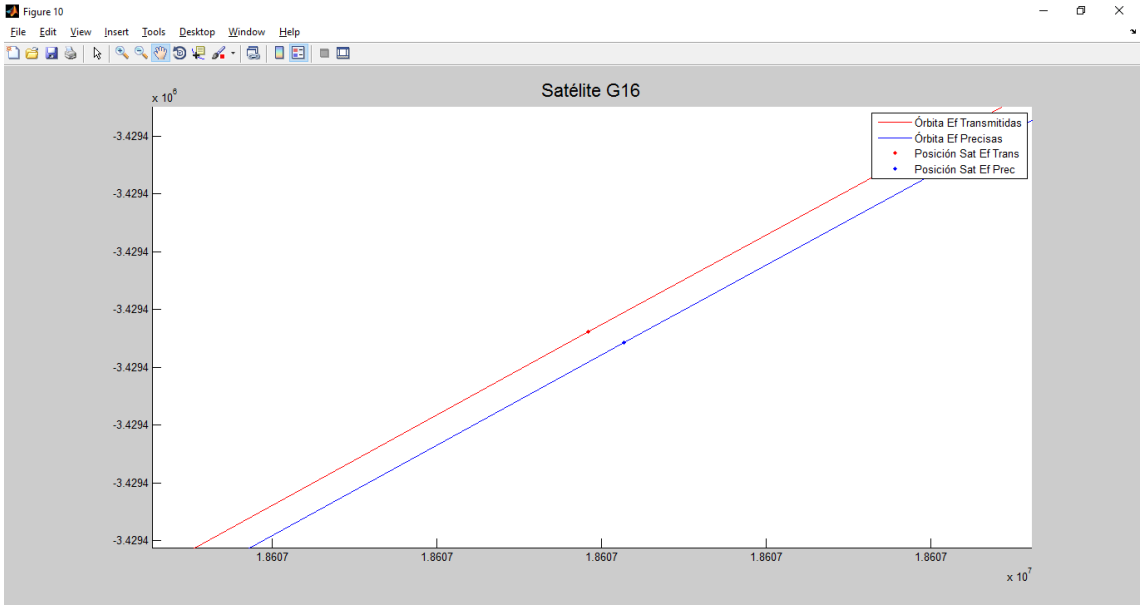


Figure 10
File Edit View Insert Tools Desktop Window Help
Satélite G16
x 10^6
-3.4294
-3.4294
-3.4294
-3.4294
-3.4294
-3.4294
-3.4294
-3.4294
1.8607
1.8607
1.8607
1.8607
1.8607
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

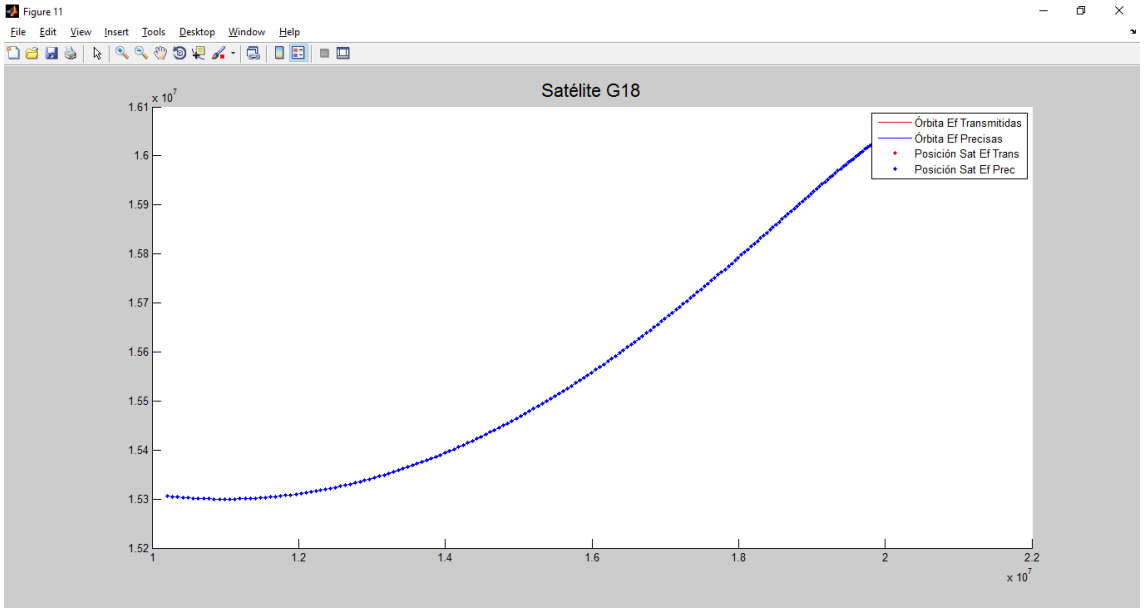


Figure 11
File Edit View Insert Tools Desktop Window Help
Satélite G18
x 10^7
1.61
1.6
1.59
1.58
1.57
1.56
1.55
1.54
1.53
1.52
1
1.2
1.4
1.6
1.8
2
2.2
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

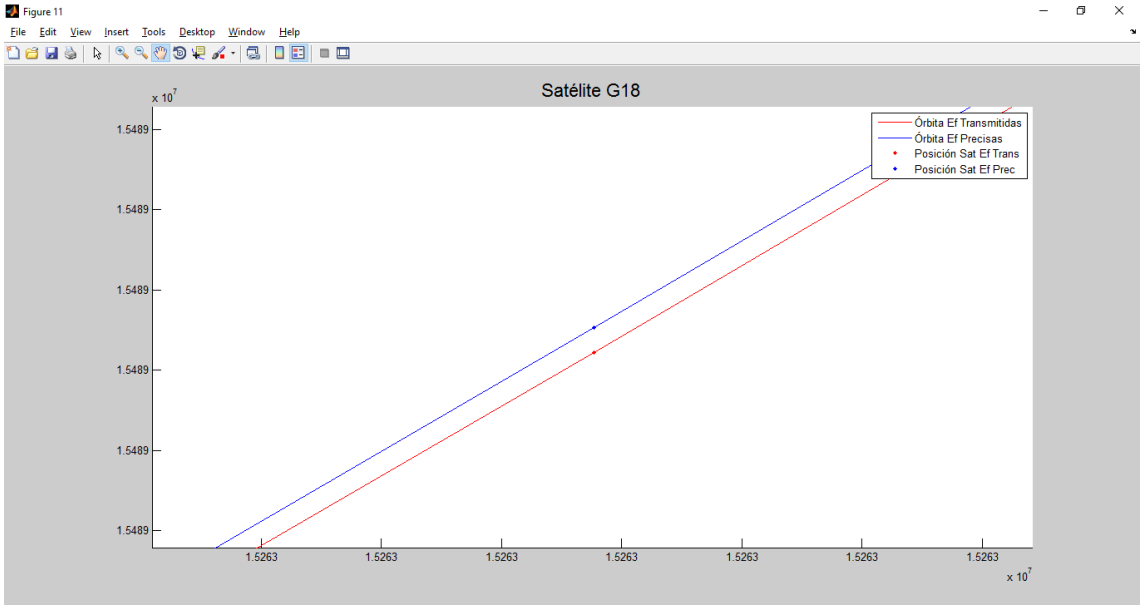


Figure 11
File Edit View Insert Tools Desktop Window Help
Satélite G18
x 10^7
1.5489
1.5489
1.5489
1.5489
1.5489
1.5489
1.5263
1.5263
1.5263
1.5263
1.5263
1.5263
1.5263
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

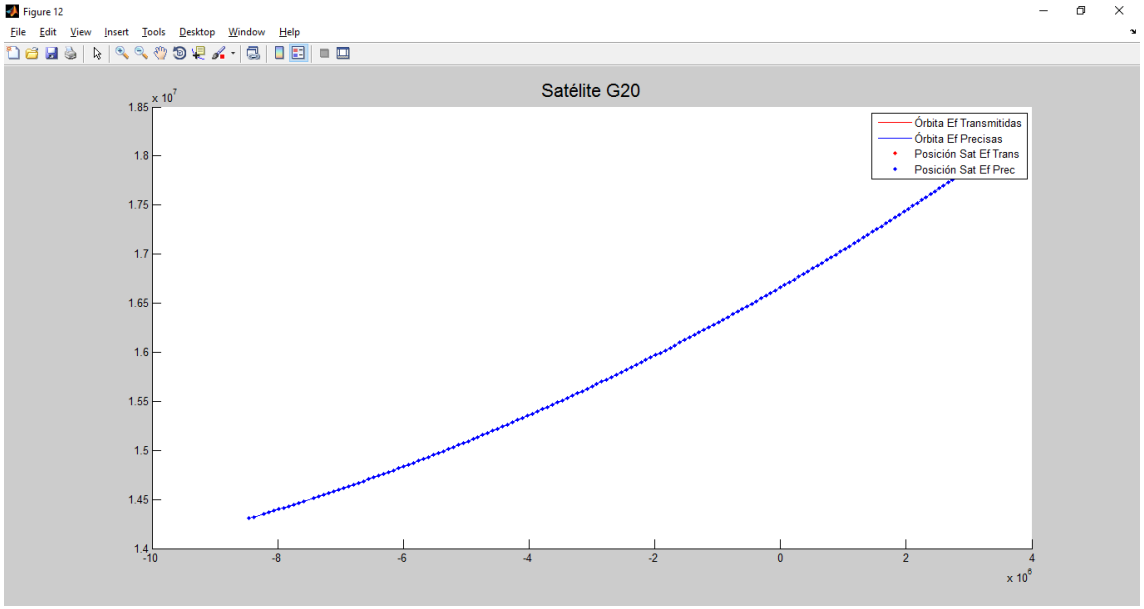


Figure 12
File Edit View Insert Tools Desktop Window Help
Satélite G20
x 10^7
1.85
1.8
1.75
1.7
1.65
1.6
1.55
1.5
1.45
1.4
-10
-8
-6
-4
-2
0
2
4
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

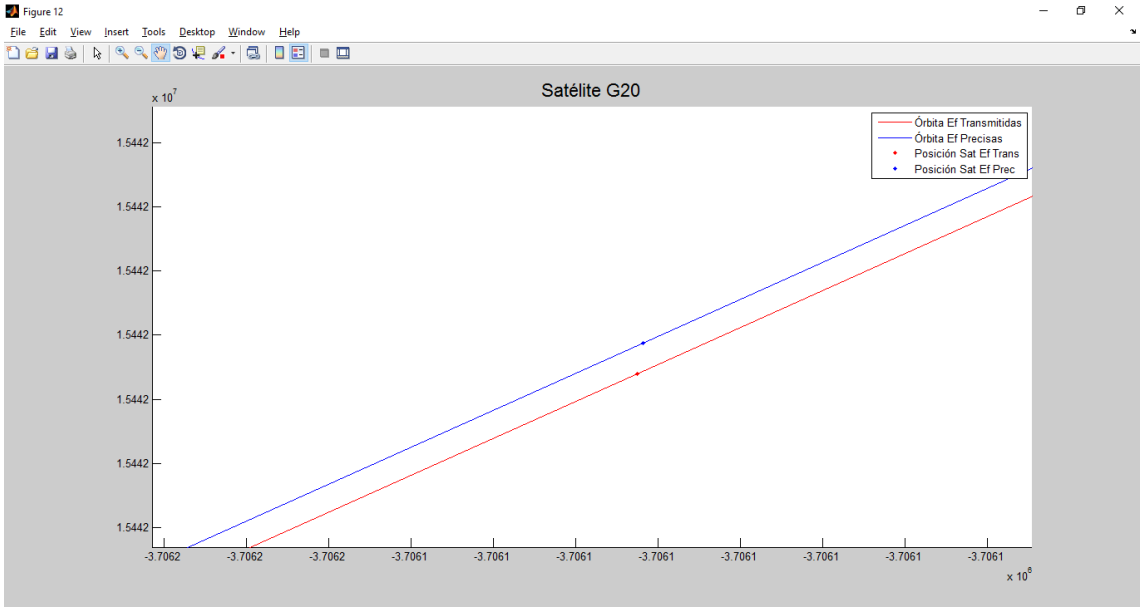


Figure 12
File Edit View Insert Tools Desktop Window Help
Satélite G20
x 10^7
1.5442
1.5442
1.5442
1.5442
1.5442
1.5442
1.5442
-3.7062
-3.7062
-3.7062
-3.7061
-3.7061
-3.7061
-3.7061
-3.7061
-3.7061
-3.7061
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

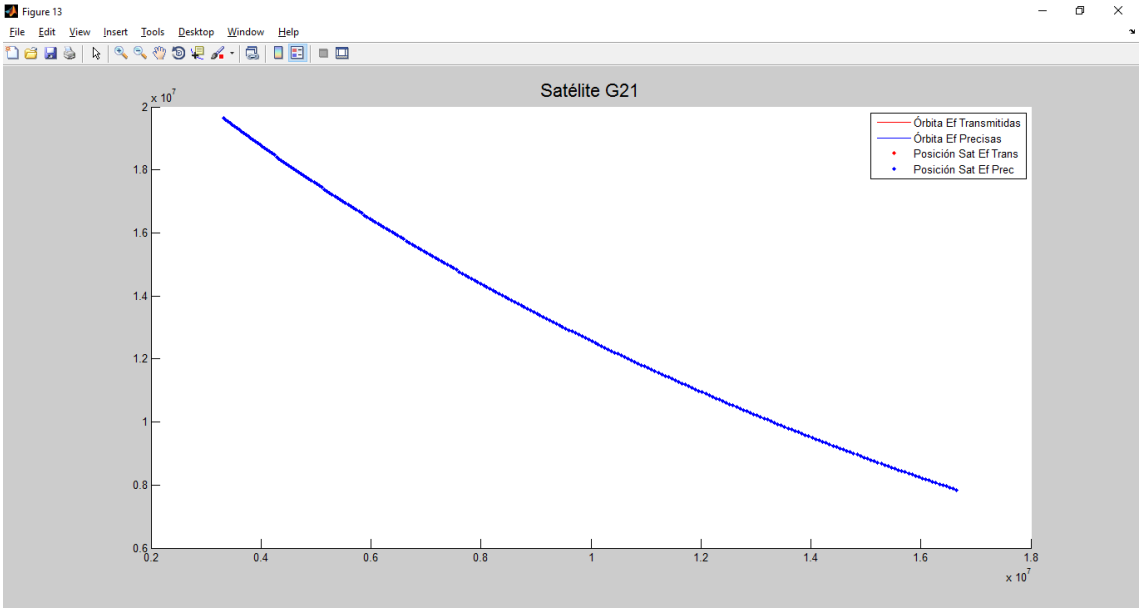


Figure 13
File Edit View Insert Tools Desktop Window Help
Satélite G21
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8
x 10^7

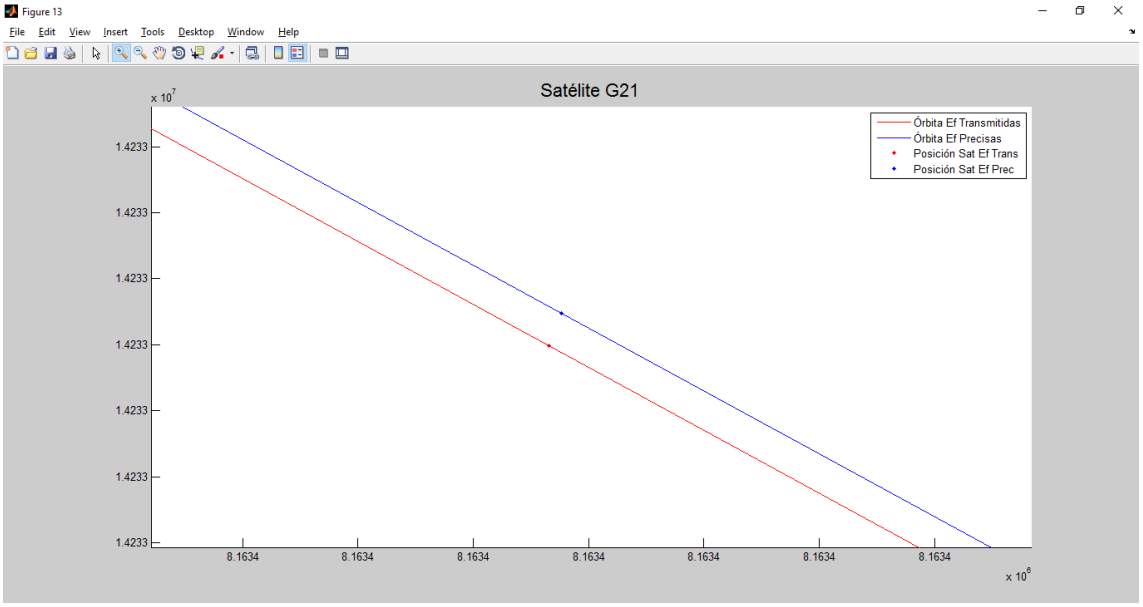


Figure 13
File Edit View Insert Tools Desktop Window Help
Satélite G21
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec
1.4233
1.4233
1.4233
1.4233
1.4233
1.4233
1.4233
8.1634 8.1634 8.1634 8.1634 8.1634 8.1634 8.1634
x 10^6

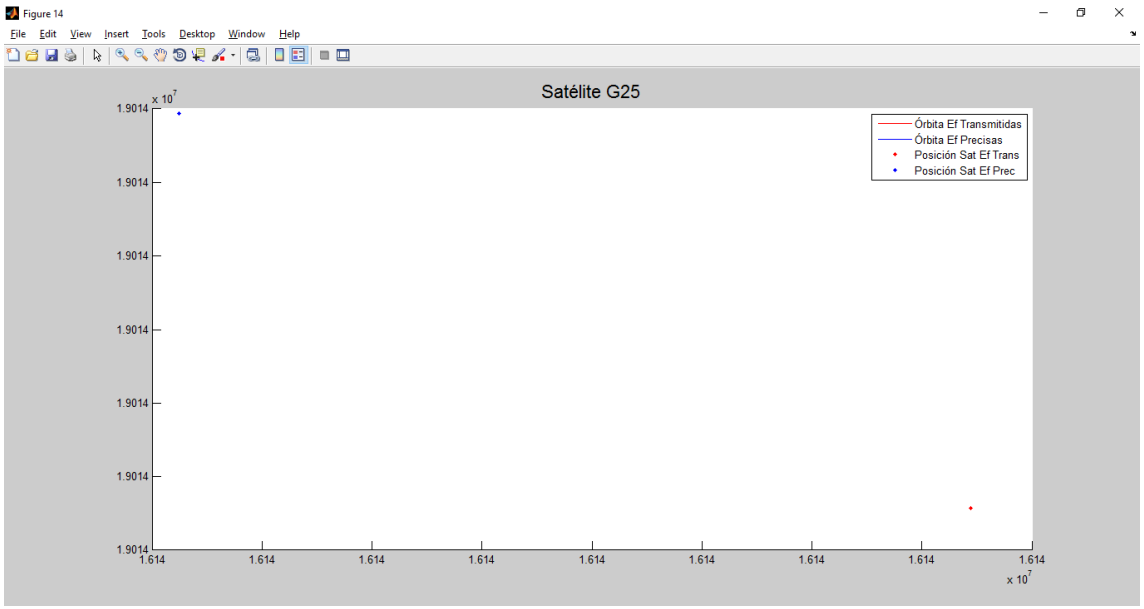


Figure 14
File Edit View Insert Tools Desktop Window Help
Satélite G25
x 10^7
1.9014
1.9014
1.9014
1.9014
1.9014
1.9014
1.614
1.614
1.614
1.614
1.614
1.614
1.614
1.614
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

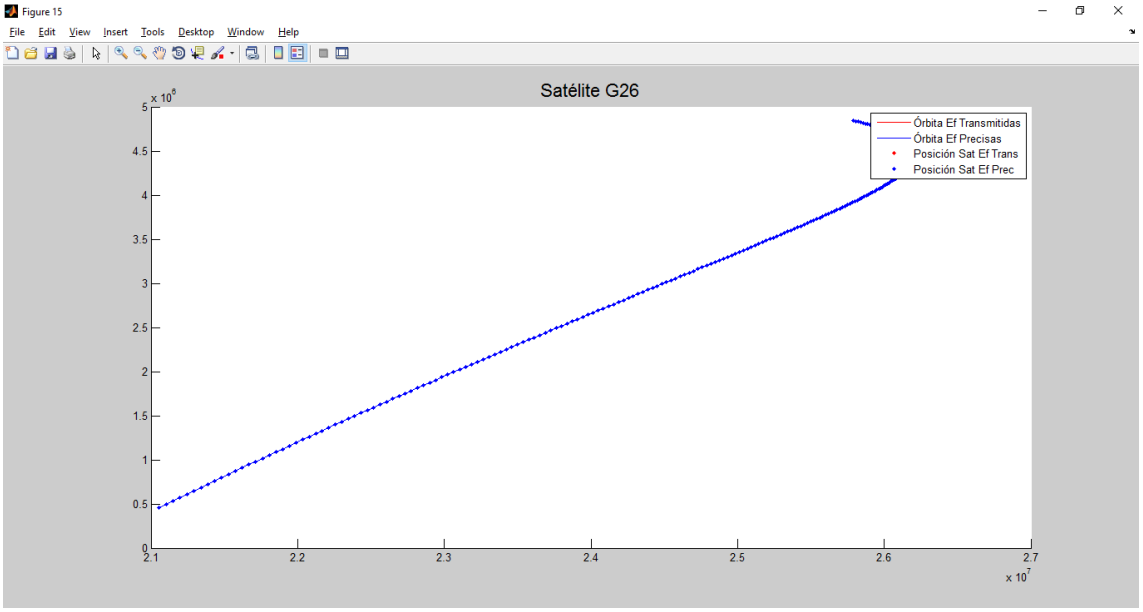


Figure 15
File Edit View Insert Tools Desktop Window Help
Satélite G26
x 10^6
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
2.1
2.2
2.3
2.4
2.5
2.6
2.7
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

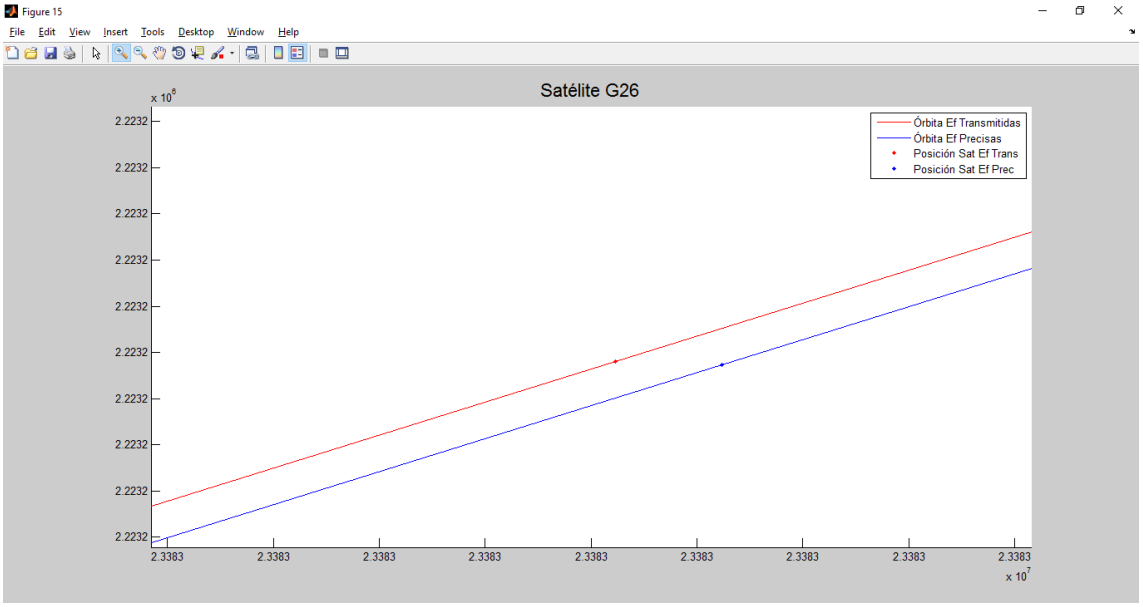


Figure 15
File Edit View Insert Tools Desktop Window Help
Satélite G26
x 10^6
2.2232
2.2232
2.2232
2.2232
2.2232
2.2232
2.2232
2.2232
2.2232
2.3383
2.3383
2.3383
2.3383
2.3383
2.3383
2.3383
2.3383
2.3383
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

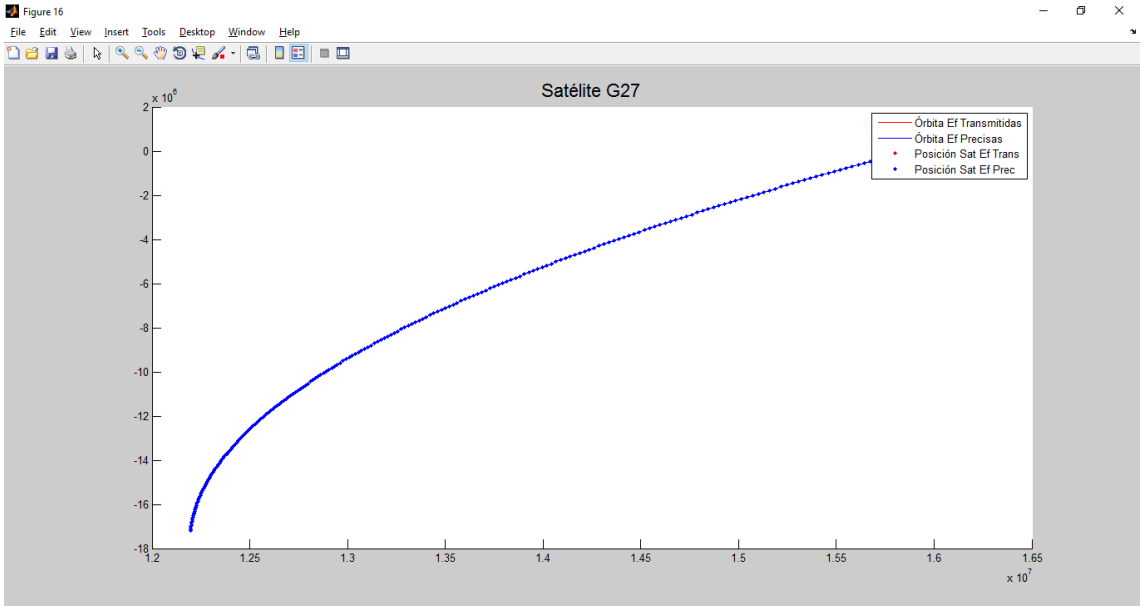


Figure 16
File Edit View Insert Tools Desktop Window Help
Satélite G27
x 10^6
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
1.2
1.25
1.3
1.35
1.4
1.45
1.5
1.55
1.6
1.65
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

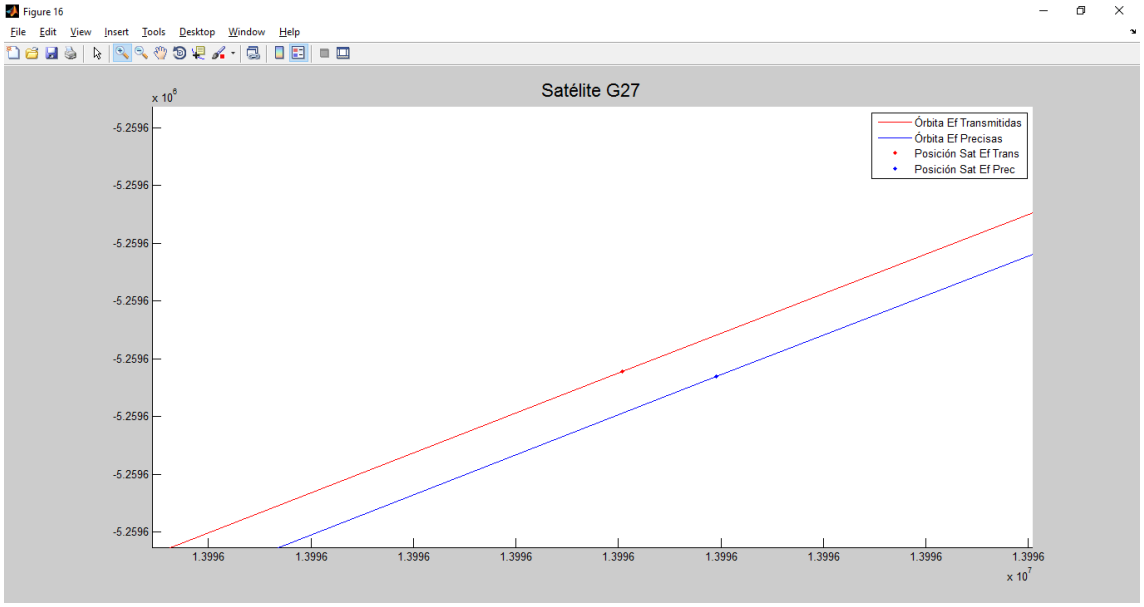


Figure 16
File Edit View Insert Tools Desktop Window Help
Satélite G27
x 10^6
-5.2596
-5.2596
-5.2596
-5.2596
-5.2596
-5.2596
-5.2596
-5.2596
1.3996
1.3996
1.3996
1.3996
1.3996
1.3996
1.3996
1.3996
1.3996
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

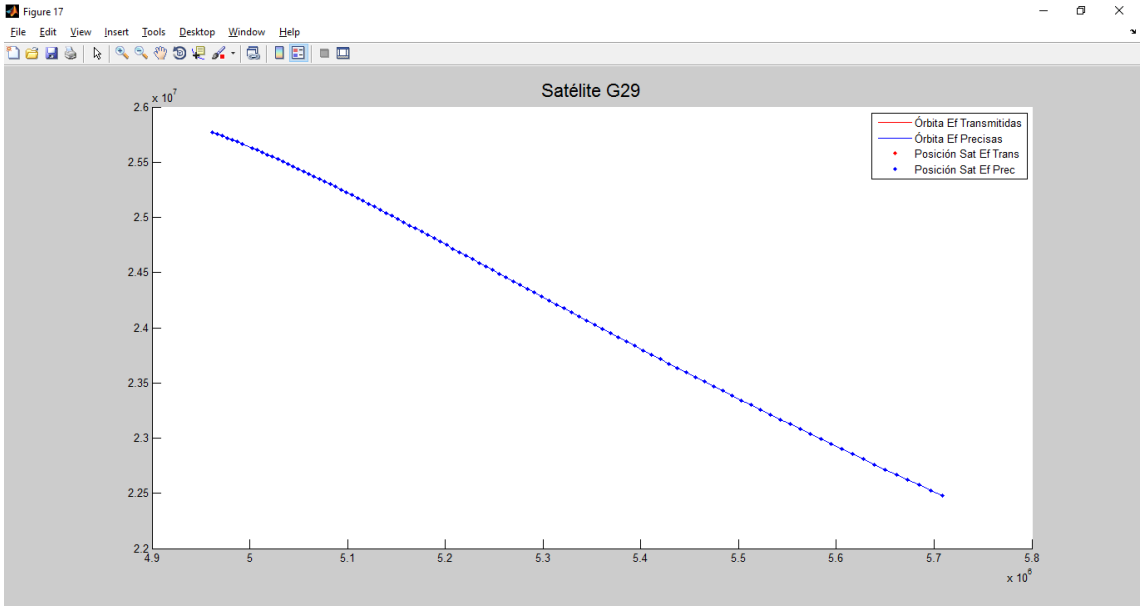


Figure 17
File Edit View Insert Tools Desktop Window Help
Satélite G29
x 10^7
2.6
2.55
2.5
2.45
2.4
2.35
2.3
2.25
2.2
4.9
5
5.1
5.2
5.3
5.4
5.5
5.6
5.7
5.8
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

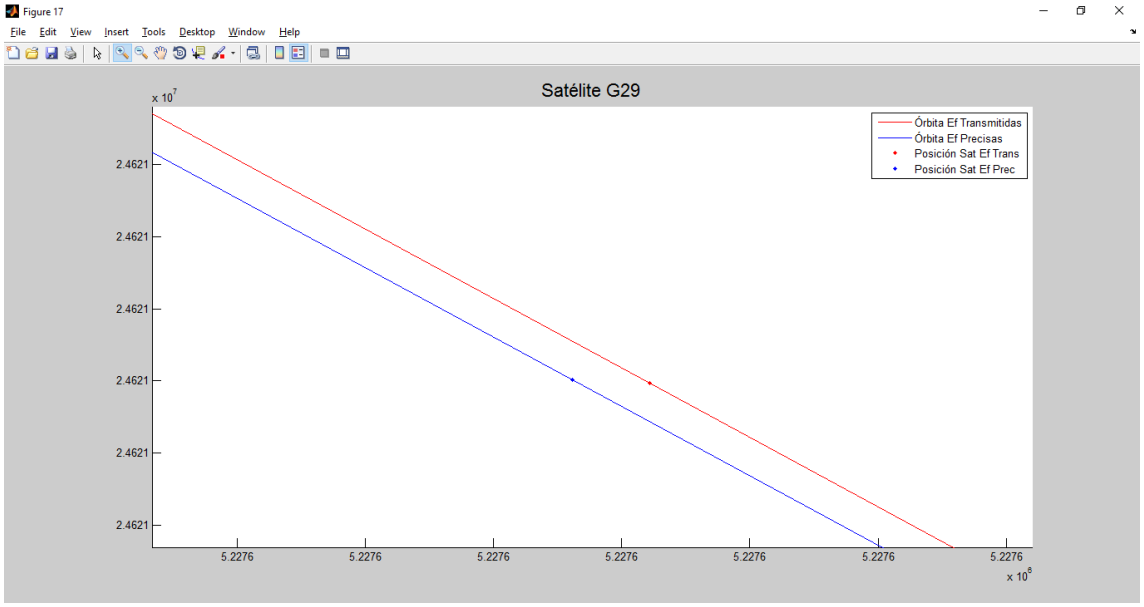


Figure 17
File Edit View Insert Tools Desktop Window Help
Satélite G29
x 10^7
2.4621
2.4621
2.4621
2.4621
2.4621
2.4621
5.2276
5.2276
5.2276
5.2276
5.2276
5.2276
5.2276
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

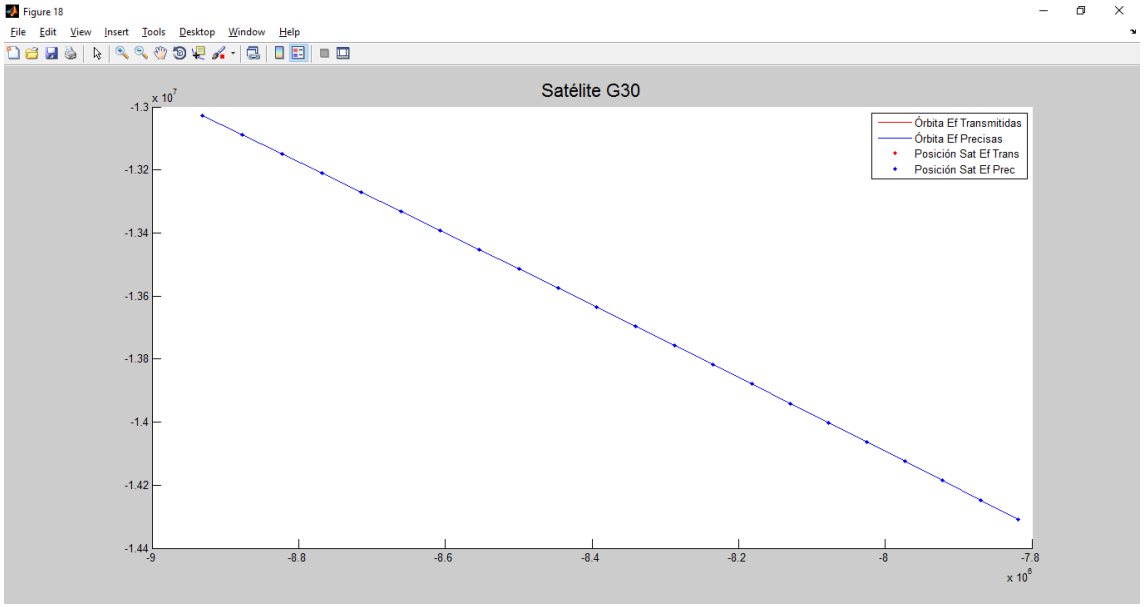


Figure 18
File Edit View Insert Tools Desktop Window Help
Satélite G30
x 10^7
-1.3
-1.32
-1.34
-1.36
-1.38
-1.4
-1.42
-1.44
-9
-8.8
-8.6
-8.4
-8.2
-8
-7.8
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

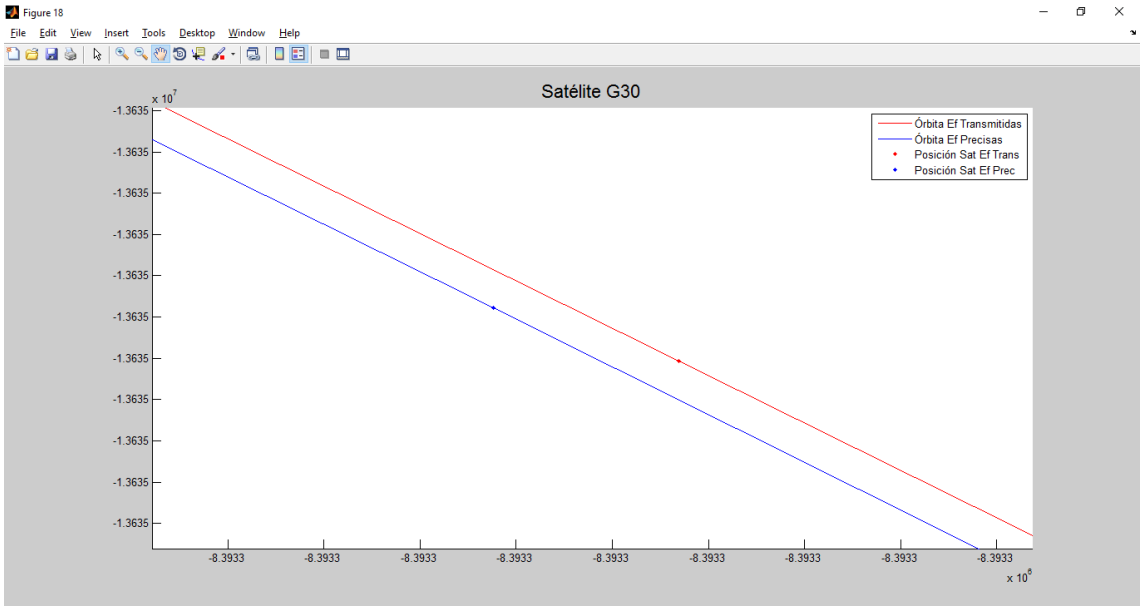


Figure 18
File Edit View Insert Tools Desktop Window Help
Satélite G30
x 10^7
-1.3635
-8.3933
x 10^6
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

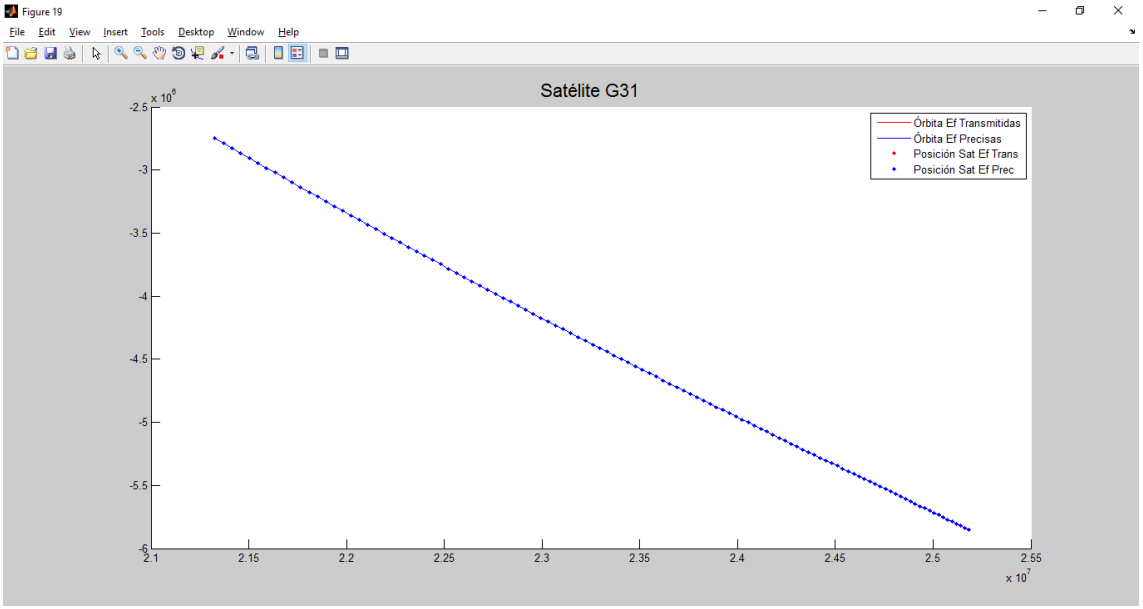


Figure 19
File Edit View Insert Tools Desktop Window Help
Satélite G31
x 10^6
-2.5
-3
-3.5
-4
-4.5
-5
-5.5
-6
2.1
2.15
2.2
2.25
2.3
2.35
2.4
2.45
2.5
2.55
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

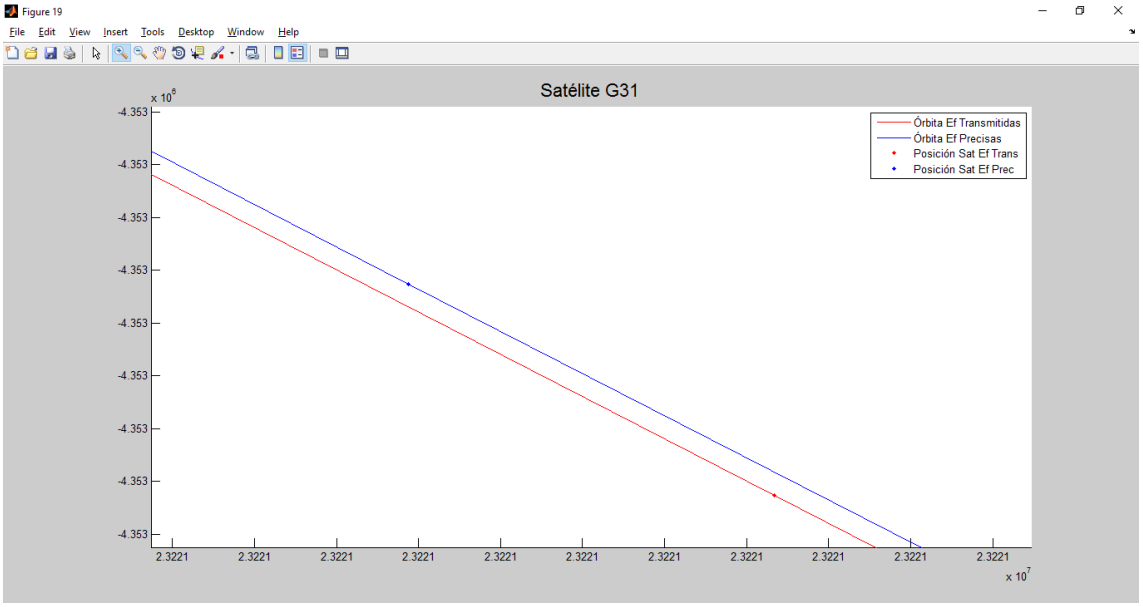


Figure 19
File Edit View Insert Tools Desktop Window Help
Satélite G31
x 10^6
-4.353
-4.353
-4.353
-4.353
-4.353
-4.353
-4.353
-4.353
2.3221
2.3221
2.3221
2.3221
2.3221
2.3221
2.3221
2.3221
2.3221
2.3221
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

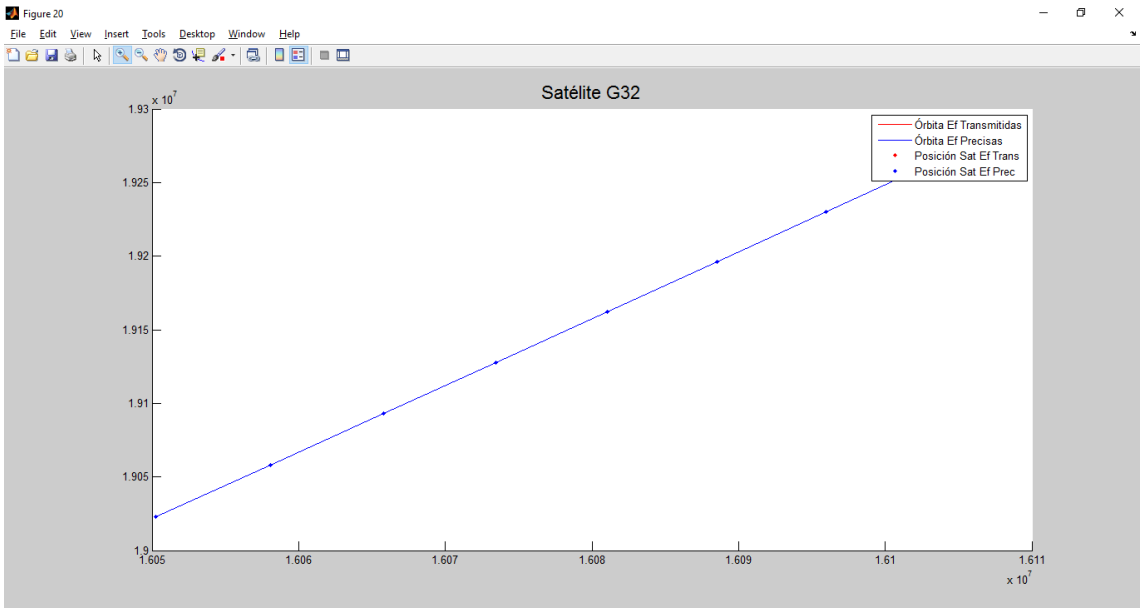


Figure 20
File Edit View Insert Tools Desktop Window Help
Satélite G32
x 10^7
1.93
1.925
1.92
1.915
1.91
1.905
1.9
1.605
1.606
1.607
1.608
1.609
1.61
1.611
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec

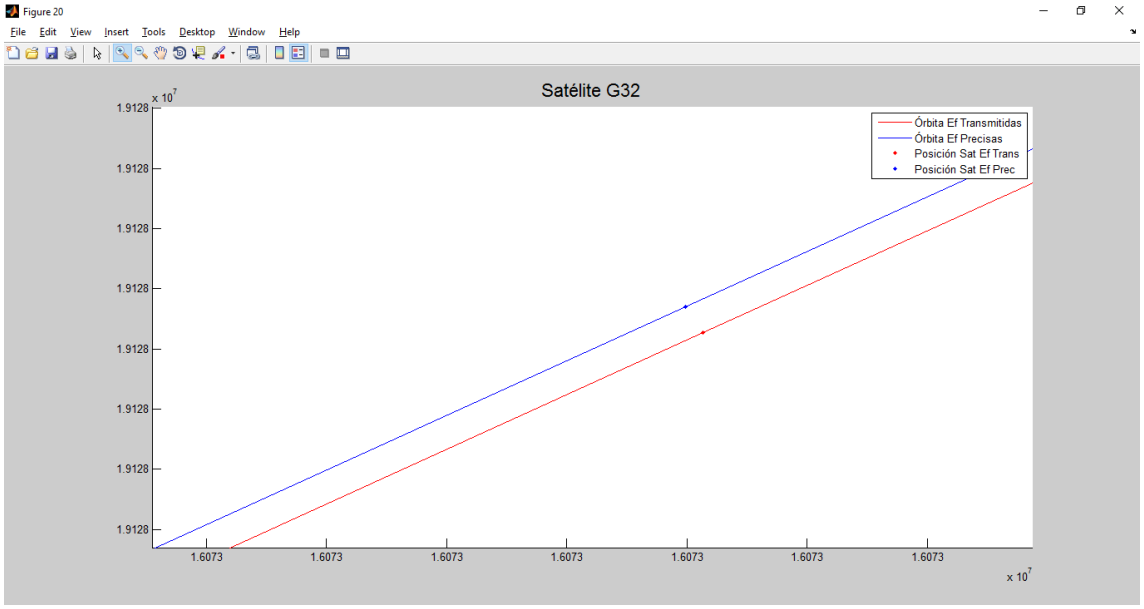


Figure 20
File Edit View Insert Tools Desktop Window Help
Satélite G32
x 10^7
1.9128
1.9128
1.9128
1.9128
1.9128
1.9128
1.9128
1.6073
1.6073
1.6073
1.6073
1.6073
1.6073
1.6073
x 10^7
Órbita Ef Transmitidas
Órbita Ef Precisas
Posición Sat Ef Trans
Posición Sat Ef Prec