

Escuela Politécnica Superior de Ávila.
Universidad de Salamanca

Trabajo Fin de Grado

Grado en Ingeniería Geomática y Topografía

**Desarrollo de un complemento de QGIS para la
planificación de trabajos con receptores GNSS**

GNSSplanning

Autor

Francesc Masdeu Ferrer

Tutoras

Manuela Chaves Tolosa

Ana Belén Gonzalo Calderón

Marzo de 2017

TABLA DE CONTENIDOS

| | |
|---|----|
| INTRODUCCIÓN | 5 |
| I. CONTEXTO | 5 |
| II. ESTADO DEL ARTE..... | 6 |
| III. OBJETIVOS..... | 7 |
| IV. METODOLOGÍA..... | 8 |
| V. ESTRUCTURA DE LA MEMORIA | 9 |
| 1. FUNDAMENTOS TEÓRICOS..... | 11 |
| 1.1. MOVIMIENTO ORBITAL..... | 11 |
| 1.1.1. DETERMINACIÓN DE LAS ÓRBITAS GNSS | 12 |
| 1.1.1.1. Causa del movimiento orbital..... | 12 |
| 1.1.1.2. Parámetros orbitales | 13 |
| 1.1.2. SISTEMAS DE REFERENCIA | 20 |
| 1.1.2.1. Sistema de referencia orbital..... | 20 |
| 1.1.2.2. Sistema de referencia Ecuatorial Cartesiano (SEC)..... | 21 |
| 1.1.2.3. Sistema de referencia Convencional Terrestre (CTS) | 22 |
| 1.1.2.4. Relación entre sistemas de referencia..... | 22 |
| 1.2. PERTURBACIONES EN LAS ÓRBITAS | 24 |
| 1.2.1. EFECTOS PERTURBADORES | 25 |
| 1.2.1.1. Efecto de la atracción gravitatoria del Sol y de la Luna | 26 |
| 1.2.1.2. Perturbación por la no esfericidad de la Tierra | 28 |
| 1.2.1.3. Perturbación por fuerzas no gravitatorias | 34 |
| 1.3. MODELOS DE PROPAGACIÓN | 38 |
| 1.3.1. ¿QUÉ ES UN PROPAGADOR? | 39 |
| 1.3.2. TIPO DE PROPAGADORES | 39 |
| 1.3.2.1. De dos cuerpos | 39 |
| 1.3.2.2. Propagadores J2 y J4..... | 40 |
| 1.3.2.3. Propagadores de los receptores GNSS | 40 |
| 1.3.2.4. Conjuntos SGP (Simplifield General Perturbations) | 42 |
| 1.4. EFEMÉRIDES: los datos de partida de los modelos de propagación | 44 |
| 1.4.1. ALMANAQUES | 44 |
| 1.4.2. EFEMÉRIDES RADIODIFUNDIDAS (broadcast ephemerides)..... | 44 |

| | | |
|----------|---|----|
| 1.4.3. | EFEMÉRIDES PRECISAS | 47 |
| 1.4.4. | LOS CONJUNTOS TLE (Two-Line Elements set)..... | 49 |
| 1.5. | FACTORES QUE LIMITAN LA PRECISIÓN | 53 |
| 1.5.1. | ERRORES CAUSADOS POR LA ATMÓSFERA | 54 |
| 1.5.2. | ERRORES CAUSADOS POR EL MULTITRAYECTO DE LA SEÑAL | 55 |
| 1.5.3. | ERRORES CAUSADOS POR SEÑALES INTERFERENTES..... | 56 |
| 1.5.4. | ERRORES CAUSADOS POR LA DISTRIBUCIÓN GEOMÉTRICA DE LOS SATÉLITES EN EL CIELO. | 56 |
| 2. | SELECCIÓN DE UN PROPAGADOR PARA GNSSplanning | 61 |
| 2.1. | ESTUDIO DE CARACTERÍSTICAS | 61 |
| 2.1.1. | ¿QUÉ PERTURBACIONES DEBE CORREGIR EL PROPAGADOR? | 61 |
| 2.1.2. | UNIFORMIDAD Y ACCESIBILIDAD DE LOS DATOS DE PARTIDA | 62 |
| 2.1.3. | ADAPTACIÓN AL LENGUAJE DE PROGRAMACIÓN..... | 63 |
| 2.2. | PROPIUESTA DE LOS PROPAGADORES CANDIDATOS | 64 |
| 2.3. | EVALUACIÓN DE LOS PROPAGADORES CANDIDATOS | 65 |
| 2.3.1. | PROP-RECEPTOR | 65 |
| 2.3.2. | PROP-SDP4 | 68 |
| 2.3.3. | POSICIONES REALES Y CALCULADAS DE LOS SATÉLITES | 69 |
| 2.4. | VALORACIÓN DE LOS CANDIDATOS | 74 |
| 2.5. | LIMITACIONES TEMPORALES | 75 |
| 3. | EL COMPLEMENTO GNSSplanning | 77 |
| 3.1. | ENTORNO DE TRABAJO..... | 77 |
| 3.1.1. | QGIS: EL ENTORNO SIG..... | 77 |
| 3.1.2. | PYTHON: EL ENTORNO DE PROGRAMACIÓN | 78 |
| 3.1.3. | HERRAMIENTAS PARA CREAR EL COMPLEMENTO | 79 |
| 3.1.4. | LIBRERÍAS DE PYTHON INTEGRADAS EN GNSSplanning..... | 80 |
| 3.2. | INTERFAZ GRÁFICA DEL USUARIO..... | 80 |
| 3.2.1. | DISEÑO DE LA INTERFAZ GRÁFICA..... | 80 |
| 3.3. | FUNCIONAMIENTO DEL COMPLEMENTO | 83 |
| 3.3.1. | INSTALACIÓN DEL COMPLEMENTO | 83 |
| 3.3.2. | UTILIZACIÓN DEL COMPLEMENTO | 85 |
| 3.4. | MÓDULOS Y FUNCIONES PRINCIPALES..... | 89 |
| 3.4.1. | ESTRUCTURA DE FICHEROS | 89 |
| 3.4.2. | DESCRIPCIÓN DE LOS MÓDULOS Y DE LAS FUNCIONES..... | 90 |
| 3.4.2.1. | gnss_planning.py | 90 |
| 3.4.2.2. | ImportTLE.py..... | 93 |
| 3.4.2.3. | Satellite.py | 94 |

| | | |
|----------|--|-----|
| 3.4.2.4. | Propagate.py..... | 96 |
| 3.4.2.5. | DOP.py..... | 97 |
| 3.4.2.6. | Capture_coords.py..... | 98 |
| 3.4.2.7. | transforms.py..... | 99 |
| 3.5. | COMPARACIÓN CON OTRAS APLICACIONES..... | 102 |
| 4. | CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO | 105 |
| 4.1. | CONCLUSIONES..... | 105 |
| 4.2. | FUTURAS LÍNEAS DE TRABAJO..... | 107 |
| | REFERENCIAS BIBLIOGRÁFICAS | 109 |
| | ANEXO 1. DIAGRAMAS DE FLUJO | 113 |
| a. | MÓDULO gnss_planning.py..... | 113 |
| b. | MÓDULO ImportTLE.py | 114 |
| c. | CLASE Satellite.py | 115 |
| d. | MÓDULO Propagate.py | 116 |
| e. | MÓDULO DOP.py | 117 |
| | ANEXO 2. CÓDIGO | 119 |
| a. | Código gnss_planning.py..... | 119 |
| b. | Código ImportTLE.py | 129 |
| c. | Código Satellite.py..... | 131 |
| d. | Código Propagate.py | 135 |
| e. | Código DOP.py | 137 |
| f. | Código Capture_coords.py | 140 |
| g. | Código transform.py..... | 141 |
| h. | Código PROP-Receptor.py | 145 |

ÍNDICE DE TABLAS

| | |
|---|----|
| TABLA 1. PARÁMETROS DE LA ELÍPSE DE LOS SATÉLITES GPS. | 15 |
| TABLA 2. PERTURBACIONES POR EFECTO DEL SOL Y LA LUNA..... | 27 |
| TABLA 3. VALOR NUMÉRICO DE J_2 PARA CADA PLANETA DEL SISTEMA SOLAR Y PARA LA LUNA..... | 30 |
| TABLA 4. FUERZAS PERTURBADORAS EN LOS SATÉLITES DE LAS CONSTELACIONES GNSS. [3] | 37 |
| TABLA 5. PARÁMETROS RADIODIFUNDIDOS EN EL MENSAJE DE NAVEGACIÓN [3]..... | 45 |
| TABLA 6 DESCRIPCIÓN DE LOS PARÁMETROS DE LOS FICHEROS TLE [20]. | 51 |
| TABLA 7. TABLA DE LOS PARÁMETROS DE LAS ÓRBITAS PARA LAS CONSTELACIONES GNSS[23][24][25][26][27][28]. | 61 |
| TABLA 8. DATOS DE LOS SATÉLITES UTILIZADOS EN LA EVALUACIÓN DE LOS PROPAGADORES..... | 65 |
| TABLA 9. DATOS DE LA ESTACIÓN DE REFERENCIA BELLOOESP DE LA RED PERMANENTE EUREF..... | 66 |
| TABLA 10. EFEMÉRIDES PRECISAS. POSICIÓN REAL DE LOS SATÉLITES..... | 69 |
| TABLA 11. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-RECEPTOR Y LA COMPARACIÓN CON LA POSICIÓN REAL (G12). | 70 |
| TABLA 12. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-SDP4 Y LA COMPARACIÓN CON LA POSICIÓN REAL (G12)..... | 70 |
| TABLA 13. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-RECEPTOR Y LA COMPARACIÓN CON LA POSICIÓN REAL (G15). | 71 |
| TABLA 14. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-SDP4 Y LA COMPARACIÓN CON LA POSICIÓN REAL (G15)..... | 71 |

| | |
|--|-----|
| TABLA 15. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-RECEPTOR Y LA COMPARACIÓN CON LA POSICIÓN REAL (G19) | 72 |
| TABLA 16. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-SDP4 Y LA COMPARACIÓN CON LA POSICIÓN REAL (G19)..... | 72 |
| TABLA 17. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-RECEPTOR Y LA COMPARACIÓN CON LA POSICIÓN REAL (G24) | 73 |
| TABLA 18. RESULTADOS OBTENIDOS CON EL PROPAGADOR PROP-SDP4 Y LA COMPARACIÓN CON LA POSICIÓN REAL (G24)..... | 73 |
| TABLA 19. VARIACIÓN ENTRE LAS POSICIONES CALCULADAS Y POSICIONES REALES (DISTANCIA, ÁNGULO Y TIEMPO)..... | 76 |
| TABLA 20. VALORES PARA EL ESTUDIO DE COMPARACIÓN..... | 102 |
| TABLA 21.TABLA COMPARATIVA DE VALORES DE DOP..... | 102 |

ÍNDICE DE FIGURAS

| | |
|--|-----|
| FIGURA 1. ÓRBITA DE UN SATÉLITE..... | 14 |
| FIGURA 2. PARÁMETROS ORBITALES | 16 |
| FIGURA 3. 2 ^a LEY DE KEPLER | 16 |
| FIGURA 4. ANOMALÍA EXCÉNTRICA. | 18 |
| FIGURA 5. SISTEMA DE REFERENCIA ORBITAL..... | 20 |
| FIGURA 6. SISTEMA DE REFERENCIA ECUATORIAL CARTESIANO. | 21 |
| FIGURA 7. SISTEMA DE REFERENCIA CTS..... | 22 |
| FIGURA 8. RELACIÓN ENTRE SISTEMA ORBITAL Y CONVENCIONAL TERRESTRE. | 23 |
| FIGURA 9. EFECTOS DE LARGA DURACIÓN DE LAS PERTURBACIONES [57]...... | 25 |
| FIGURA 10. DESPLAZAMIENTO DEL CENTRO DE MASAS DE LA TIERRA. | 28 |
| FIGURA 11. EFECTO DE J_2 SOBRE LA ÓRBITA DE UN SATÉLITE..... | 31 |
| FIGURA 12. TASA DE LA REGRESIÓN NODAL [7]..... | 32 |
| FIGURA 13. TASA DE ROTACIÓN DEL PERIGEO [7]..... | 33 |
| FIGURA 14. ÓRBITA MOLNIYA..... | 34 |
| FIGURA 15. EFECTO DE LA FRICCIÓN ATMOSFÉRICA EN LA ÓRBITA DE UN SATÉLITE..... | 35 |
| FIGURA 16. DISTRIBUCIÓN DE LOS ESCOMBROS ESPACIALES [8]..... | 38 |
| FIGURA 17. PRECISIONES DE LAS EFEMÉRIDES PRECISAS (GPS Y GLONASS)[16]...... | 48 |
| FIGURA 18. DATOS DE LOS FICHEROS TLE [20]...... | 52 |
| FIGURA 19. EFEMÉRIDES TRANSMITIDAS A LA ESTACIÓN BELLOOESP. | 66 |
| FIGURA 20. OBSERVABLES DE LA ESTACIÓN BELLOOESP. | 67 |
| FIGURA 21. FICHERO TLE..... | 68 |
| FIGURA 22. VARIACIONES ENTRE LA POSICIÓN REAL Y LAS PREDICHAS POR LOS DOS PROPAGADORES (G12)..... | 70 |
| FIGURA 23 VARIACIONES ENTRE LA POSICIÓN REAL Y LAS PREDICHAS POR LOS DOS PROPAGADORES (G15)..... | 71 |
| FIGURA 24. VARIACIONES ENTRE LA POSICIÓN REAL Y LAS PREDICHAS POR LOS DOS PROPAGADORES (G19). | 72 |
| FIGURA 25. VARIACIONES ENTRE LA POSICIÓN REAL Y LAS PREDICHAS POR LOS DOS PROPAGADORES (G24)..... | 73 |
| FIGURA 26. VARIACIÓN ENTRE LAS POSICIONES CALCULADAS Y POSICIONES REALES. | 76 |
| FIGURA 27. PLUGIN BUILDER | 79 |
| FIGURA 28. INTERFAZ GRÁFICA DEL USUARIO. | 81 |
| FIGURA 29. ADMINISTRADOR DE COMPLEMENTOS DE QGIS..... | 83 |
| FIGURA 30. GNSSPLANNING EN EL ADMINISTRADOR DE COMPLEMENTOS DE QGIS. | 84 |
| FIGURA 31.GNSSPLANNING EN EL MENÚ DE TAREAS DE QGIS. | 84 |
| FIGURA 32. VENTANA PRINCIPAL DE GNSSPLANNING. | 85 |
| FIGURA 33. INSERTAR POSICIÓN DEL OBSERVADOR..... | 85 |
| FIGURA 34. INSERTAR PRIMERA ÉPOCA..... | 86 |
| FIGURA 35. INSERTAR OBSTRUCCIONES. | 86 |
| FIGURA 36. RESULTADOS CON Y SIN OBSTRUCCIONES. | 87 |
| FIGURA 37. SECUENCIA DE RESULTADOS. | 88 |
| FIGURA 38. ESTRUCTURA DE DIRECTORIOS DEL COMPLEMENTO GNSSPLANNING. | 89 |
| FIGURA 39. COORDENADAS GLOBALES Y LOCALES..... | 100 |
| FIGURA 40. SISTEMA LOCAL. | 101 |
| FIGURA 41. COMPARACIÓN DE RESULTADOS CON OTRA APLICACIÓN EXISTENTE EN EL MERCADO. | 103 |

INTRODUCCIÓN

En este Trabajo Fin de Grado se presenta el desarrollo de un complemento (*plugin*) para el software QGIS. El objetivo principal es crear una herramienta ágil y eficaz, que sirva de soporte en la fase de planificación de los trabajos con receptores GNSS. Con la creación de este complemento, denominado **GNSSplanning**, se pretende demostrar que las aplicaciones utilizadas para la predicción de la posición de los satélites en épocas futuras, pueden mejorar su rendimiento y sus resultados, si se integran en un entorno SIG.

I. CONTEXTO

Con la incorporación de nuevas constelaciones de satélites de posicionamiento, los receptores GNSS han visto aumentada su operatividad en zonas donde hace pocos años les era imposible trabajar. Al poder recibir señales de más satélites a la vez, también se ha mejorado la calidad de las observaciones, pudiendo seleccionar, entre más candidatos, aquellos satélites que ofrezcan mejores resultados.

Aun así, sigue habiendo ciertos factores que minimizan este aumento de la operatividad. Las condiciones atmosféricas imponen restricciones en la calidad de la señal, y lo hacen de forma variable dependiendo del momento del día. Por otro lado, las obstrucciones físicas dentro de la zona de trabajo (casas, muros, montañas, etc.), dificultan la recepción nítida de la señal o anulan completamente la posibilidad de trabajar con algún satélite, aunque esté por encima del horizonte.

Para asegurarnos de alcanzar las precisiones esperadas en nuestras observaciones, tenemos que controlar con qué escenario vamos a trabajar: desde cuántos satélites podremos utilizar y cuál será su distribución en el cielo en ese instante, hasta conocer cómo es el entorno sobre el que estacionaremos nuestro receptor. Por eso, sigue siendo crucial empezar cualquier trabajo con receptores GNSS con una buena planificación.

Si queremos conocer las posiciones futuras de los satélites debemos recurrir a modelos predictivos. Con un seguimiento continuo de su situación y conociendo cómo es su movimiento teórico, podemos calcular cuál será su posición aproximada.

De forma simplificada, podría decirse que los satélites GNSS se desplazan en órbitas alrededor de la Tierra siguiendo las leyes de los movimientos orbitales de Kepler. Pero la realidad es mucho más compleja. Los satélites están sometidos a fuerzas que perturban este movimiento teórico kepleriano (desde la influencia de la gravedad de la Luna o el Sol, hasta el efecto de la presión de la radiación solar) y modifican constantemente su trayectoria.

Los modelos de propagación de órbitas, incorporan estas perturbaciones en sus algoritmos y permiten obtener la posición y velocidad de un satélite a partir de las efemérides dadas, en un instante concreto.

Utilizando estos modelos de propagación en la fase de planificación de nuestros trabajos, podemos conocer qué satélites estarán por encima del horizonte en el momento de las mediciones y, con ello, determinar la calidad de las observaciones mediante el cálculo de los valores DOP. Pero para obtener un valor de DOP realista, necesitamos conocer exactamente qué satélites podremos utilizar. Debemos descartar todos aquellos que estén ocultos por las obstrucciones físicas del entorno.

Para definir lo mejor posible ese entorno de trabajo, necesitamos incorporar a nuestra planificación modelos descriptivos que lo representen. Por ejemplo, los modelos digitales del terreno reproducen a escala la topografía de la superficie terrestre y los modelos de elevación aportan más información tridimensional sobre construcciones antrópicas, como edificios, muros o presas. Los mapas de cobertura arbórea (de densidad y de altura) pueden añadir otro input en estos modelos descriptivos, informando de dónde no se recibirá la señal de los satélites por culpa de la vegetación.

Así pues, utilizando todos estos factores de análisis en nuestra fase de planificación podremos tomar mejores decisiones y mejorar nuestro rendimiento en los trabajos de campo.

Y para ello necesitamos una herramienta eficaz, capaz de fusionar modelos descriptivos, modelos predictivos y modelos de decisión. Los sistemas de información geográfica (SIG) se presentan como la herramienta más idónea en el ámbito del estudio de la geometría del espacio y lo que sucede en él,

II. ESTADO DEL ARTE

Todos los programas incorporados en las controladoras de los receptores GNSS incluyen alguna aplicación o módulo que nos ofrece información sobre la posición de los satélites. Normalmente, mediante alguna tabla o un gráfico polar se muestran los satélites que se encuentran por encima del horizonte. Estas posiciones se determinan mediante los datos de almanaque que incorpora el mensaje de navegación emitido por cualquiera de los satélites observados. Pero estas posiciones no dejan de ser una aproximación calculada a partir de los parámetros orbitales, sin ningún tipo de corrección. Además, no se tiene en cuenta las obstrucciones físicas del entorno y, por lo tanto, el escenario que se muestra no es real.

Los programas para la planificación de trabajos con GNSS, que suelen utilizarse en oficina, también incorporan aplicaciones similares. Los modelos de propagación que utilizan dependen de cada programa, pero todos ellos parten de datos precisos que sí incorporan los valores de las perturbaciones, por lo que permiten obtener una mejor estimación de la posición. Además, la mayoría presentan opciones para descartar los satélites situados en posiciones bajas sobre el horizonte, pudiendo filtrar algunos satélites no deseados.

Pero son muy pocas las aplicaciones que permiten definir las obstrucciones que nos vamos a encontrar. Y ninguna de ellas consigue modelarlas de forma realista. Por ejemplo, la aplicación Trimble® GNSS

Planning On-line permite definir al usuario un perfil de horizonte, pero mediante la tediosa tarea de tener que introducir manualmente cada par azimut-altura que lo define. No existe actualmente ninguna aplicación que permita incorporar algún modelo tridimensional descriptivo para automatizar este proceso.

Por otro lado, existen multitud de software SIG en el mercado, más o menos similares entre ellos, y todos con un mismo objetivo: poder integrar gran cantidad de datos con una variable espacial en un entorno geográfico. En los SIG que encontramos hoy en día en el mercado, existen algunas aplicaciones enfocadas a trabajar con datos procedentes de observaciones GNSS. Sobretodo están orientados a la captura de datos en tiempo real o a la visualización de estos datos a posteriori. Pero no existe ninguna aplicación que permita trabajar con las efemérides de los satélites para poder planificar escenarios futuros.

La fusión de herramientas de planificación de trabajos con GNSS en un entorno SIG, es un campo que no se ha explorado todavía y resulta muy interesante evaluar. Este trabajo pretende utilizar el potencial de los SIG en la planificación de los trabajos con receptores GNSS.

Esta aplicación se hará sobre QGIS [1], desarrollando un complemento con el lenguaje de programación Python [2]. Se han analizado muchas de las librerías de código Python publicadas, con la intención de encontrar alguna aplicación que cumpliera con los mismos objetivos propuestos en este proyecto. Existen aplicaciones que ejecutan, algunas tareas muy útiles y reutilizables, pero no recorren todo el circuito de operaciones que este proyecto pretende desarrollar de forma conjunta.

III. OBJETIVOS

La novedad que se ofrece en este proyecto, no es la de crear una aplicación de predicción como las que ya existen, sino que recae en el hecho de que esta se desarrolla en un entorno SIG aprovechando gran parte del potencial en el análisis geoespacial que nos ofrece este entorno. No sólo se pretende crear una herramienta útil por sí misma, sino que además sea capaz de interactuar con información geográfica en formato vectorial o *raster*.

Con este fin, se pretende desarrollar un complemento para QGIS que tendrá que ser capaz de integrar en el modelo de predicción aquellas obstrucciones que impidan al receptor recibir, de forma directa, la señal de los satélites. Se pretende utilizar los modelos digitales de elevación para afinar el indicador DOP ofreciendo un escenario lo más parecido a la realidad. Para indicar el error de la propagación de la señal a través de la atmósfera, se intentará aprovechar el entorno SIG para incorporar datos procedentes de modelos del estado de la ionosfera.

El complemento tendrá que ser capaz de:

- Utilizar un modelo de propagación teniendo en cuenta las leyes del movimiento orbital y las

perturbaciones que influyen en él.

- Mediante la introducción de una fecha y hora concreta, tendrá que predecir la posición aproximada de todos los satélites de navegación que estén operativos (en el estudio se incluirán satélites de las constelaciones GPS, GLONASS, GALILEO y BEIDOU).
- Deberá utilizar las efemérides como datos de partida para dichas predicciones, ya sea aportadas por el propio usuario o adquiridas de forma automática.
- Mediante la interactuación con el lienzo de QGIS, tendrá que poder extraer las coordenadas aproximadas de la posición del usuario y definir las obstrucciones que se encuentran a su alrededor.
- Incorporar información de otros modelos o mapas temáticos para calcular la calidad de la señal recibida de los satélites.
- Dada la posición de un observador, el complemento tendrá que representar la situación de los satélites visibles sobre el cielo del observador.
- Finalmente, tendrá que ofrecer un indicador de la calidad de las observaciones que se obtendrán con ese escenario, así como presentar los resultados en un formato sencillo y fácil de interpretar.

Otra de las finalidades de este proyecto es aprender a desarrollar la herramienta en sí misma. Con este trabajo se describirá cómo crear un complemento para QGIS programado íntegramente en código Python. Se van a mostrar qué librerías de Python se utilizan para relacionarse con QGIS y cómo pueden implementarse *scripts* de creación propia.

IV. METODOLOGÍA

Primero, será necesario recordar aquellos conceptos teóricos que nos ayuden a entender cómo funciona un modelo de propagación. Debemos conocer, por ejemplo: cuál es el movimiento teórico que rige la trayectoria de los satélites, qué fuerzas externas influyen en esta trayectoria, cómo se corrigen o qué modelos existen para predecir la posición de un satélite.

Para escoger el propagador que se utilizará en el complemento GNSSplanning, se evaluará mediante distintas simulaciones numéricas, el comportamiento de algunos de los propagadores estudiados. Por un lado, se hará un estudio para determinar qué características queremos que tenga nuestro modelo, analizando la accesibilidad, la uniformidad y simplicidad de los formatos de los datos de partida o qué perturbaciones se deben incluir según la tipología de las órbitas de los satélites.

Sabiendo qué requisitos deberá cumplir nuestro modelo, se propondrán dos candidatos. Para cada uno de ellos, utilizando varios satélites, se realizarán simulaciones numéricas. Para realizar estos casos

prácticos será necesario programar dos *scripts* independientes para cada modelo. Estos scripts tendrán que calcular las posiciones aproximadas de los satélites durante 12 días a partir de la época inicial. Finalmente, según sean los resultados obtenidos, se argumentará qué propagador será el escogido para incluirse en la aplicación.

Elegido el modelo de propagación, se desarrollará el complemento GNSSplanning, utilizando íntegramente el lenguaje de programación Python. Se programarán múltiples librerías, módulos y funciones que permitan realizar cada una de las funcionalidades expuestas en los objetivos, además de crear una interfaz gráfica amigable para interactuar con el usuario. Para mejorar GNSSplanning, se contempla la posibilidad de aprovechar parte de código existente en las librerías publicadas.

En la fase de desarrollo, el complemento será implementado en un solo ordenador personal, dónde se irán testeando cada uno de los módulos programados. En una segunda fase, se cargará la aplicación en otros dispositivos para asegurarse de su buen funcionamiento.

Con la aplicación finalizada, se hará una comparativa con alguna otra herramienta similar existente en el mercado. Así, se evaluarán los resultados que se obtienen con nuestra aplicación y se verificará si las ganancias que se pretenden ofrecer, son reales o no.

V. ESTRUCTURA DE LA MEMORIA

En el Capítulo 1 se profundizará en la descripción del movimiento orbital y los parámetros que definen la órbita de un satélite GNSS, estudiando primero el caso teórico de los dos cuerpos, donde no se tienen en cuenta las influencias externas al sistema Tierra-satélite. Descrito este movimiento teórico, se definirán las perturbaciones a las que se somete este movimiento, qué elementos las provocan y qué efectos tienen en la trayectoria final.

Se hará un estudio de los modelos de propagación, describiendo cómo funcionan y qué precisiones se obtienen con ellos. Se pondrá especial atención en dos conceptos: sus datos de partida, o sea, qué efemérides utilizan y qué precisión nos ofrecen a medida que nos alejemos de la época inicial. Así, se hará un repaso explicativo de las distintas efemérides existentes, describiendo su contenido y su accesibilidad.

Por otro lado, para determinar el indicador de calidad que la aplicación presentará al usuario, debemos conocer cuáles son las causas que disminuyen la precisión de las observaciones con receptores GNSS y qué parámetros son los que ofrecen información sobre esta calidad. Concretamente, se estudiarán los indicadores que informen de los posibles errores debido a la distribución geométrica de los satélites en el cielo (como los indicadores DOP) y del error de propagación de la señal a través de la atmósfera, entre otros.

En el Capítulo 2 se mostrará el estudio realizado para elegir el propagador que se utilizará en nuestra

aplicación y se expondrán los resultados y conclusiones.

En el Capítulo 3 se hará una descripción de la aplicación creada. Los primeros apartados se dedicarán a describir el entorno de trabajo, los programas utilizados, las librerías de Python que se han utilizado para implementar la aplicación en QGIS y las que se han utilizado para mejorar el propio código de la aplicación. En este capítulo también se hará una descripción de la interfaz con la que el usuario interactuará para usar la aplicación.

Finalmente, se hará una descripción del contenido y del funcionamiento de cada uno de los módulos y de las funciones programadas íntegramente en este proyecto.

En los Anexos se mostrarán los diagramas de flujo que la aplicación sigue en cada proceso y el código Python de todos los módulos y funciones creados en este proyecto.

1. FUNDAMENTOS TEÓRICOS

1.1. MOVIMIENTO ORBITAL

Uno de los problemas que se van a tratar en este trabajo es determinar la posición, en un instante concreto, de un cuerpo en movimiento a través de su órbita alrededor de la Tierra.

Los métodos utilizados en los sistemas GNSS para determinar la posición de un receptor sobre la superficie terrestre no difieren, en lo esencial, con las mediciones que los topógrafos realizaban hasta hace tan sólo medio siglo. Para conocer la posición de un punto cualquiera, sólo era necesario un aparato que midiera valores angulares y tener visuales a una serie de puntos con coordenadas conocidas con precisión. Los vértices geodésicos o topográficos materializados sobre el terreno, nos proporcionaban estos puntos de coordenadas conocidas sobre los que se ejecutaban las observaciones de distancias y ángulos.

En el sistema GNSS se substituyen los vértices por otros puntos de coordenadas conocidas: los satélites artificiales. Conociendo sus posiciones con exactitud podemos situar nuestro receptor sobre un marco de referencia. Pero si los vértices son puntos “inamovibles” sobre el terreno, los satélites son referencias en constante movimiento orbitando alrededor de la Tierra. Así pues, la posición de cada satélite irá variando durante el tiempo en que estemos realizando nuestras observaciones.

Uno de los problemas más importantes tanto en la planificación de misiones con satélites como en los trabajos topográficos con receptores GNSS es poder predecir con exactitud esta posición. No sólo necesitamos saber dónde está cada satélite a cada momento, sino que también necesitamos saber dónde estará en un futuro cercano.

Pero, ¿cómo podemos determinar esta posición? Para ello necesitamos conocer su movimiento y definir un sistema de referencia sobre el que poder describir su posición. Además, será fundamental determinar y entender qué perturbaciones afectan a dicho movimiento, determinar su magnitud y corregir las desviaciones que provocan en la trayectoria. Será necesario considerar el efecto de la no esfericidad terrestre, los efectos de atracción de terceros cuerpos y otras fuerzas no gravitacionales.

Pero vamos primero a centrarnos en una simplificación del sistema para entender cómo se mueven los satélites y cómo podemos determinar su posición y velocidad en cualquier instante.

Esta descripción simplificada del movimiento orbital la apoyaremos en las tres leyes del movimiento planetario de Kepler (1609-1918) respaldadas por la segunda ley de Newton y su ley de la Gravitación

Universal (1687). La teoría de la relatividad no influye demasiado en los movimientos de los planetas (exceptuando en el caso de Mercurio, el que está más próximo al Sol, donde sí se aprecia y es medible el efecto de la relatividad), pero sí que se tendrá que tener en cuenta en los cálculos de los movimientos de los satélites de navegación ya que afecta, por ejemplo, a sus osciladores.

1.1.1. DETERMINACIÓN DE LAS ÓRBITAS GNSS

1.1.1.1. Causa del movimiento orbital

Como ya hemos dicho, conocer la posición de un satélite implica determinar su movimiento. Para describir este movimiento necesitamos acudir, primero, a las leyes de Newton. Vamos a utilizar su segunda ley del movimiento, donde nos relaciona la fuerza de un cuerpo con su masa y su aceleración y la ley de Gravitación Universal, sobre la fuerza de atracción entre dos cuerpos. De hecho, estamos retrocediendo a Johannes Kepler, pues la Ley de Gravitación Universal fue el resultado de la formulación matemática que Newton hizo de las leyes de los movimientos de los planetas de Kepler, y que veremos más adelante.

➤ **Ley de Gravitación Universal:** Un objeto material del universo atrae a otro objeto material con una fuerza directamente proporcional al producto de sus masas e inversamente proporcional al cuadrado de la distancia que los separa [3].

$$\vec{F} = m \cdot \vec{a} \quad \vec{F} = G \frac{m_1 m_2}{r^2} \hat{r} \quad (1) (2)$$

Tenemos que el movimiento de un satélite a través de su órbita puede escribirse como:

$$\ddot{\vec{r}} + \frac{G(M_T + m_S)}{r^3} \vec{r} = \vec{0} \quad (3)$$

siendo

M_T, m_S las masas de los dos cuerpos, Tierra y satélite, respectivamente,

G la constante gravitatoria terrestre ($G = 6,67 * 10^{-11} \text{ N m}^2/\text{Kg}^2$),

r la distancia que separa el centro de la Tierra y el del satélite,

\vec{r} el vector posición relativo,

$\ddot{\vec{r}} = \frac{d^2\vec{r}}{dt^2}$ el vector aceleración relativo

En la atracción Tierra-Luna no se puede despreciar la masa de la Luna (M_L) frente a la masa de la Tierra (M_T), puesto que la primera no es lo suficientemente pequeña comparada con la segunda. Pero en el estudio del efecto de la gravedad entre la Tierra y los satélites artificiales se puede considerar la masa del satélite (alrededor de 700 kg en los satélites GNSS) como despreciable en comparación con la masa de la Tierra ($M = 5,98 * 10^{24}$ Kg) por lo que la ecuación quedaría como:

$$\frac{d^2\vec{r}}{dt^2} = -\mu \frac{\vec{r}}{r_{Ts}^3} \quad (4)$$

donde μ es la constante gravitacional terrestre ($\mu = G * M_T = 3986005 * 10^8$ m³/s²) y uno de los parámetros que definen el sistema de referencia WGS84.

El signo negativo de la ecuación indica es que se trata de una fuerza atractiva entre dos cuerpos. El desplazamiento del satélite, en el estudio clásico de la mecánica celeste, es conocido como el de los dos cuerpos, con la particularidad que depende solamente de la distancia entre el centro de la Tierra y el satélite y es independiente de la masa del satélite.

1.1.1.2. Parámetros orbitales

La ecuación diferencial de segundo orden vectorial (4) necesita de seis constantes de integración para su resolución. Estas constantes se corresponden con los seis parámetros orbitales que definen una órbita kepleriana clásica. Vamos a estudiar estos parámetros. Para ello, vamos a ver qué nos dice Kepler sobre los movimientos de los planetas [3].

- *1^a Ley de las órbitas, de 1609: La órbita de cada planeta es una elipse, con el Sol en uno de sus focos.*
- *2^a Ley de las áreas, de 1618: La línea que une el Sol con un planeta barre áreas iguales en tiempos iguales.*
- *3^a Ley de los periodos, de 1618: El cuadrado del periodo de un planeta es proporcional al cubo del semieje mayor de la órbita*

¿Pero, como podemos aplicar estas leyes al movimiento de los satélites artificiales?

- ***1^a Ley de Kepler:***

De la primera ley de Kepler podemos deducir que los satélites, en su revolución alrededor de la Tierra, recorren una trayectoria elíptica cuyo foco es la misma Tierra. Esta elipse viene definida por su semieje mayor (a) y por su excentricidad (e). En el caso de los satélites de las constelaciones GNSS, la excentricidad es muy pequeña dando a lugar a órbitas casi circulares. Pero este “casi circulares” no es lo mismo que “circulares”, por lo que siguen mandando las leyes del movimiento elipsoidal.

- Perigeo: punto de la órbita del satélite en el que la distancia entre el satélite y la Tierra es mínima, o en el que el satélite se encuentra más próximo a la Tierra.
- Apogeo: punto de la órbita del satélite en el que la distancia entre el satélite y la Tierra es máxima, o en el que el satélite se encuentra más alejado a la Tierra.
- Línea de ápsides: línea que une el perigeo y el apogeo pasando por el foco de la elipse.
- Línea nodal: intersección del plano orbital y del plano ecuatorial terrestre
- Nodo ascendente/nodo descendente: puntos de la órbita en que el satélite pasa del hemisferio sur al norte, y viceversa.
- Punto Aries o equinoccio vernal: es el nodo ascendente de la órbita de la Tierra alrededor del Sol.

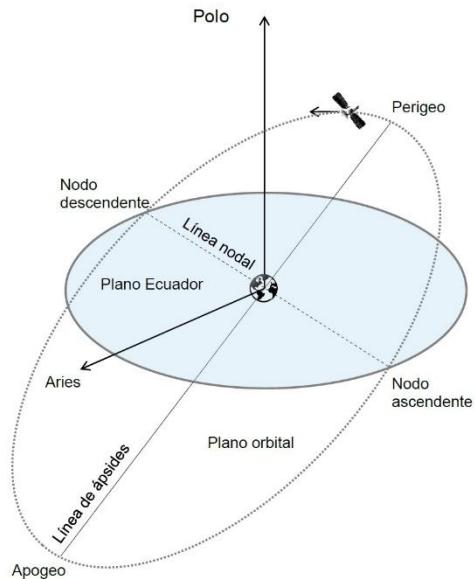


Figura 1. Órbita de un satélite

La situación de una órbita en el espacio viene determinada por cinco de los seis parámetros orbitales (o elementos keplerianos). El sexto parámetro define la posición del satélite dentro de la órbita. Estos elementos son:

- Los que definen la forma de la propia órbita:

a: semieje mayor, que define el tamaño de la órbita.

$$a = \frac{d_{apogeo} + d_{perigeo}}{2} \quad (5)$$

e: excentricidad, que define la forma de la órbita. En los satélites GNSS, aunque pequeña, la excentricidad es mayor que 0 y por lo tanto la órbita es una curva elíptica.

$$e = \sqrt{\frac{a^2 - b^2}{a^2}} = \sqrt{1 - \left(\frac{b}{a}\right)^2} \quad (6)$$

- Los que definen la posición del plano orbital:

Ω : Ascensión recta del nodo ascendente, que define el ángulo medido en el plano ecuatorial entre la dirección hacia el punto vernal y la dirección hacia el nodo ascendente. En las efemérides del mensaje de navegación se facilita la longitud del nodo ascendente medido desde el meridiano de Greenwich (o su equivalente). La diferencia entre la ascensión recta y la longitud es el ángulo que forma la dirección del punto vernal con la dirección hacia el meridiano de Greenwich, que es el tiempo sidéreo de Greenwich.

i : inclinación del plano de la órbita, es el ángulo que forma el plano orbital y el plano ecuatorial terrestre respecto al plano del ecuador.

- Los que definen la orientación de la órbita sobre su plano:

ω : argumento del perigeo, es el ángulo, medido en el plano orbital, entre la línea de los nodos y la línea de los ápsides que contiene al perigeo. Como puede verse en la Figura 2, ω es el ángulo formado entre la línea nodal y la línea de ápsides, sobre el plano orbital.

Tabla 1. Parámetros de la elipse de los satélites GPS.

| Elemento orbital | Satélite GPS |
|------------------|--------------|
| a | 26560 km |
| e | <0,02 |
| i | 55° |

Con estos cinco parámetros orbitales, hemos definido el marco por donde se moverán los satélites. Podríamos decir que hemos trazado el carril o la pista por donde se moverá el vehículo y lo hemos situado en un sistema de referencia global. Ya sabemos por dónde se va a mover el satélite y que de este carril no se va a salir, ¿pero en qué punto lo situamos dentro de esta pista? ¿cómo podemos situar la posición del satélite en un instante determinado?

Como los parámetros orbitales no cambian con el tiempo (ignorando las perturbaciones), sólo necesitamos referenciar la posición del satélite a uno de estos parámetros para conocer su posición.

La posición instantánea del satélite sobre la órbita se describe con una cantidad angular que define el sexto elemento kepleriano y que es el único dependiente del tiempo:

$\nu(t)$: **anomalía verdadera**, es el ángulo medido desde el centro de la Tierra sobre el plano orbital, entre el perigeo y la posición instantánea del satélite. Este ángulo se mide positivamente desde el perigeo a la posición del satélite y, a diferencia de los parámetros orbitales, varía con el tiempo, $\nu(t)$.

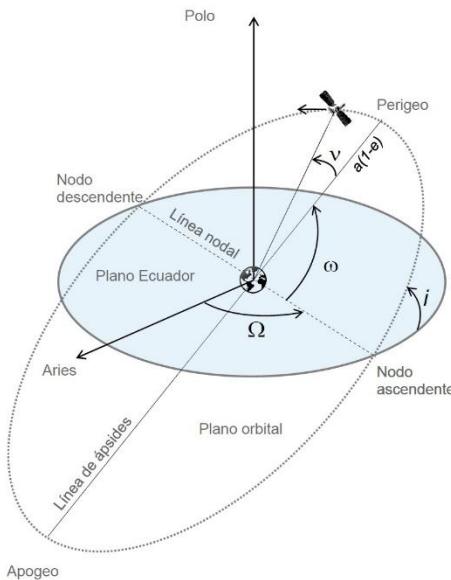


Figura 2. Parámetros orbitales

¿Pero cómo podemos calcular este ángulo en función del tiempo? Veamos que nos dice Kepler en su segunda ley.

➤ **2^a ley de Kepler:**

Con la segunda ley sabemos que los satélites GNSS se mueven a distinta velocidad a lo largo de su trayectoria, moviéndose más rápidamente en su perigeo y más lentamente en su apogeo. Así, la línea que une el centro de la Tierra con el satélite barre áreas iguales en tiempos iguales. Esta línea es la que determina el vector posición del satélite en cada instante.

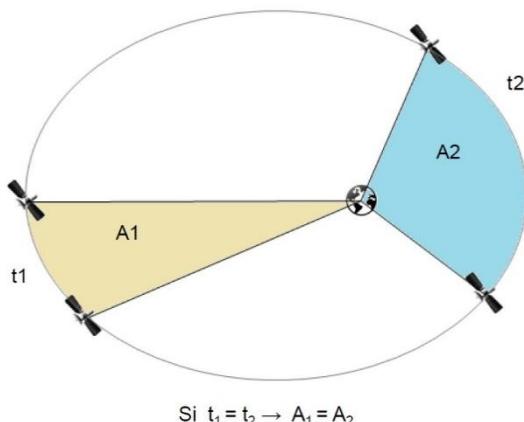


Figura 3. 2^a ley de Kepler

Por lo tanto, el satélite se mueve por su órbita a una velocidad no constante lo que imposibilita el cálculo de la anomalía verdadera de forma sencilla sin conocer una velocidad constante. Para determinar esta velocidad constante nos basamos en la tercera ley de Kepler.

➤ **3^a ley de Kepler:**

La tercera ley aplicada a los satélites, nos dice que el cuadrado de su periodo orbital (tiempo que tardan en dar una vuelta entera alrededor de la Tierra), es proporcional al cubo de su distancia media al centro de la Tierra. Por lo tanto, su velocidad disminuye al aumentar la altura.

$$P^2 = k \cdot r^3 \quad (7)$$

Donde:

P : periodo del satélite. Su unidad de medida en el Sistema Internacional es el segundo (s).

k : constante de proporcionalidad. Su unidad de medida en el Sistema Internacional es el segundo al cuadrado partido por metro cúbico (s²/m³).

r : distancia media a la Tierra. Por las propiedades de la elipse se cumple que su valor coincide con el del semieje mayor de la elipse, *a*. Su unidad de medida en el Sistema Internacional es el metro (m).

Para el cálculo de *k*, volvemos a la ley de Gravitación Universal de Newton. Como hemos dicho anteriormente, para órbitas circulares:

$$F = G \frac{M_T m_s}{r^2} \quad \text{y} \quad F = m_s \cdot a \quad (8)$$

entonces,

$$G \frac{M_T m_s}{r^2} = m_s \cdot \frac{v^2}{r} \quad \text{donde} \quad v = \omega \cdot r \quad (9) \quad (10)$$

por lo que

$$\mu \frac{m_s}{r^2} = m_s \cdot \omega^2 \cdot r \quad (11)$$

y sabiendo que

$$\omega = 2 \frac{\pi}{P} \quad (12)$$

nos queda

$$\mu \frac{m_s}{r^2} = m_s \cdot \frac{4\pi^2}{P^2} \cdot r \quad (13)$$

con lo que podemos relacionar el valor de la constante k con la constante gravitacional

$$\frac{P^2}{r^3} = \frac{4\pi^2}{\mu} = k \quad (14)$$

Kepler propuso relacionar el movimiento verdadero de un planeta sobre su órbita elíptica, con el movimiento aparente y constante que este realizaría sobre una órbita circular con radio igual al semieje mayor de la elipse. Esta velocidad constante, llamada **velocidad angular media (n)** o **movimiento medio** se calcula en función del periodo P y de la altura media del satélite cuyo valor coincide, como hemos dicho, con el semieje mayor de la elipse a .

La velocidad angular media para una vuelta entera es:

$$n = \frac{2\pi}{P} = \sqrt{\frac{\mu}{a^3}} \quad (15)$$

Podemos ahora definir la **anomalía media $M(t)$** como el ángulo que formaría con el eje de la elipse un satélite que girase con movimiento uniforme sobre una circunferencia cuyo diámetro coincidiese con el eje principal de la elipse (llamada circunferencia principal). Podemos expresarla, para un instante t , en función de la velocidad media.

$$M(t) = n \cdot (t - t_0) \quad (16)$$

donde t_0 es el tiempo de paso por el perigeo del satélite.

Pero la anomalía media, no se puede expresar como una magnitud geométrica, es un valor ficticio en función de una velocidad no real, sino de una velocidad media. La **anomalía excéntrica $E(t)$** es el ángulo medido desde el centro de la órbita (no desde el centro de la Tierra, como en la anomalía verdadera) comprendido entre el perigeo y el punto de proyección del satélite sobre la circunferencia concéntrica a la órbita y de radio igual su semieje mayor.

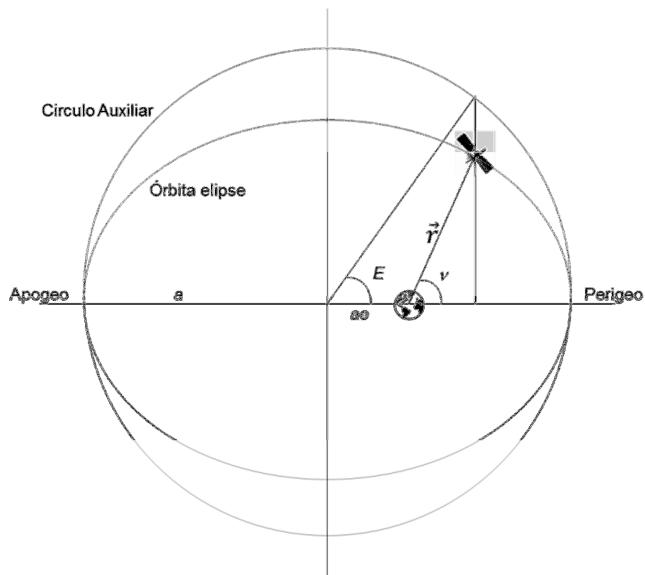


Figura 4. Anomalía excéntrica.

Para relacionar estas dos anomalías tenemos la **ecuación de Kepler**:

$$E(t) = M(t) + e \cdot \operatorname{sen} E(t) \quad (17)$$

Como vemos, esta es una ecuación trascendente dónde aparece la incógnita $E(t)$ en ambos lados de la igualdad debiéndose resolver por métodos iterativos. En las órbitas de los satélites GNSS, al tener una excentricidad muy reducida, la iteración convergerá muy rápidamente a la solución buscada. En una circunferencia, la excentricidad es $e = 0$, y por tanto la ecuación se reduce a

$$E(t) = M(t) \quad (18)$$

Se debe tener en cuenta que $e \cdot \operatorname{sen} E(t)$ no es un ángulo, sino un arco medido en radianes y que, por tanto, tendremos que transformar a grados sexagesimales:

$$\frac{360^\circ}{2\pi} = \frac{x}{e \cdot \operatorname{sen} E(t)} \quad \rightarrow \quad x = \frac{360^\circ}{2\pi} e \cdot \operatorname{sen} E(t) \quad (19) \quad (20)$$

que substituyendo en la ecuación de Kepler

$$E(t) = M(t) + 57.29577954 \cdot e \cdot \operatorname{sen} E(t) \quad (21)$$

Con la anomalía excéntrica podemos ya determinar la anomalía verdadera para posicionar el satélite sobre su órbita:

$$\tan \nu = \frac{\operatorname{sen} E \sqrt{1 - e^2}}{\cos E - e} \quad (22)$$

La suma de la anomalía verdadera y el argumento del perigeo dan el **argumento de la latitud (ν)**:

$$\nu = \omega + \nu \quad (23)$$

De acuerdo con lo expuesto podemos definir la posición del satélite en el espacio con los parámetros orbitales:

- $(a, e, i, \Omega, \omega, \nu(t))$ con la anomalía verdadera
- $(a, e, i, \Omega, \omega, M(t))$ con la anomalía media
- $(a, e, i, \Omega, \omega, E(t))$ con la anomalía excéntrica

Sabiendo ya la posición del satélite podemos determinar nuestra posición. Pero para ello debemos conocer las coordenadas del satélite con respecto a un sistema de referencia fijo terrestre (XYZ_{ECEF}). Vamos a ver como determinar estas coordenadas a partir de los parámetros orbitales. Para ello tenemos que definir los sistemas de referencia adecuados.

1.1.2. SISTEMAS DE REFERENCIA

1.1.2.1. Sistema de referencia orbital

Podemos expresar la posición y velocidad de un satélite dentro de su órbita en función de la anomalía excéntrica y la anomalía verdadera mediante las coordenadas cartesianas $[e_1, e_2]$ respecto a un sistema geocéntrico. Este sistema está definido por el triángulo con origen en el centro de gravedad terrestre, con un plano Oe_1e_2 coincidente con la órbita del satélite y con el eje Oe_3 perpendicular a este plano. El eje Oe_1 se orienta de forma que contenga el perigeo de la órbita tal que el vector de posición sea [4]:

$$\vec{r} = a \begin{bmatrix} \cos E - e \\ \sqrt{1 - e^2} \sin E \end{bmatrix} = r \begin{bmatrix} \cos v \\ \sin v \end{bmatrix} \quad (24)$$

Por lo que

$$x_{so} = r \cdot \cos v \quad (25)$$

$$y_{so} = r \cdot \sin v \quad (26)$$

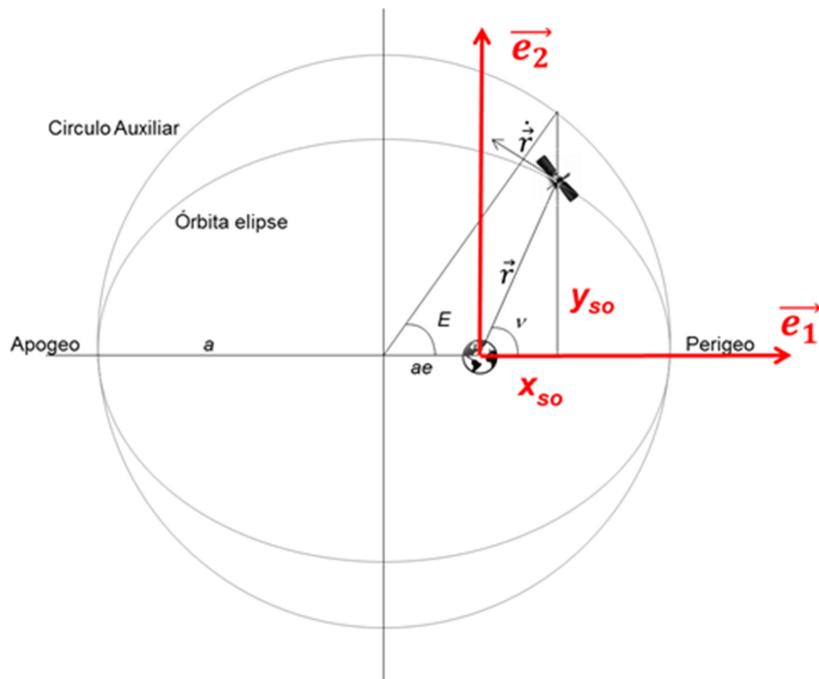


Figura 5. Sistema de referencia orbital

1.1.2.2. Sistema de referencia Ecuatorial Cartesiano (SEC)

Se basa en los sistemas inerciales centrados en la Tierra (ECI). Un sistema se denomina inercial cuando está en reposo o en movimiento rectilíneo uniforme (con velocidad constante y aceleración igual a cero). Los sistemas de referencia centrados y fijos en la Tierra (ECEF) no son sistemas inerciales ya que se mueven solidarios con la Tierra y rotan con ella, con lo que la aceleración del sistema no es nula. Para los objetos en el espacio, las ecuaciones que describen el movimiento orbital son más simples en un marco no giratorio como ECI. Pero como veremos, un sistema ECI no es exactamente inercial ya que el centro de masas de la Tierra se acelera a medida que viaja en su órbita alrededor del Sol y se tendrá que tener en cuenta en los cálculos de la perturbación del satélite debido a las influencias gravitacionales.

Un sistema ECI es independiente del movimiento de rotación terrestre y se caracteriza por:

- Su origen O es el centro de masas de la Tierra.
- Su Plano fundamental es el Ecuador terrestre.
- Los tres ejes del sistema son:
 - El eje vertical OZ es perpendicular al Plano fundamental en la dirección del eje de rotación medio de la Tierra,
 - El eje OX va desde el centro de la Tierra al punto Aries que es el punto de intersección del Plano ecuatorial con al elíptica del Sol en su movimiento ascendente.
 - Eje OY es perpendicular a los otros dos, de manera que se forme un sistema dextrógiro.

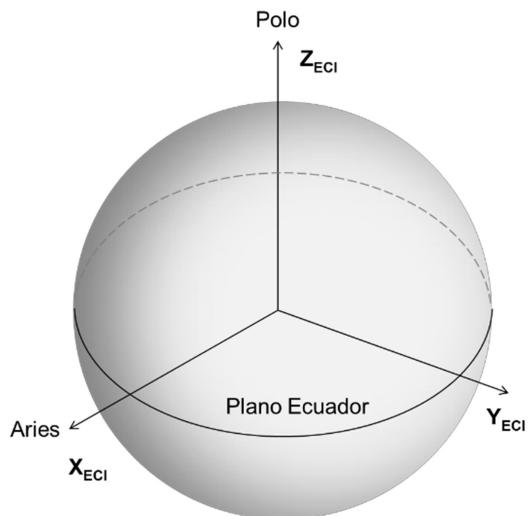


Figura 6. Sistema de referencia ecuatorial cartesiano.

Los distintos sistemas de referencia ECI representan la posición del eje OX basándose en criterios temporales:

- J2000: define el ecuador medio y el equinoccio medio a la hora terrestre 12:00 el 1 de enero de 2000. Puede ser referido como J2000 o EME2000.
- M50: Este marco es similar a J2000, pero se define con el equinoccio medio a las 12:00 el 1 de enero de 1950.

- GCRF: *Geocentric Celestial Reference Frame* (GCRF) fue adoptado por la Unión de Astronomía Internacional desde enero del 1998 utilizando un equinoccio medio convencional definido por múltiples observaciones a cuerpos extra galácticos (cuásares).
- MOD (del inglés, *Mean of Date*): está definido usando el ecuador medio y el equinoccio en una fecha en particular.
- TME (del inglés, *True Equator, Mean Equinox*): es el marco ECI utilizado para los formatos de transferencia de datos orbitales *Two-line elements* de NORAD [5] (que se verán más adelante), se denomina a veces ecuador verdadero, equinoccio medio (TME), aunque no utiliza el equinoccio medio convencional.

1.1.2.3. Sistema de referencia Convencional Terrestre (CTS)

Las coordenadas obtenidas en los receptores GNSS son coordenadas cartesianas geocéntricas en un sistema ligado a la Tierra (ECEF). En los receptores GPS, el sistema es el WGS84. Tiene las siguientes características:

- Su origen O_0 es el centro de masas de la Tierra.
- Su Plano fundamental es el ecuador terrestre.
- Los tres ejes del sistema son:
 - El eje vertical OZ_0 es perpendicular al Plano fundamental en la dirección del eje de rotación medio de la Tierra,
 - El eje OX_0 va desde el centro de la Tierra en dirección al meridiano de Greenwich
 - Eje OY_0 es perpendicular a los otros dos.

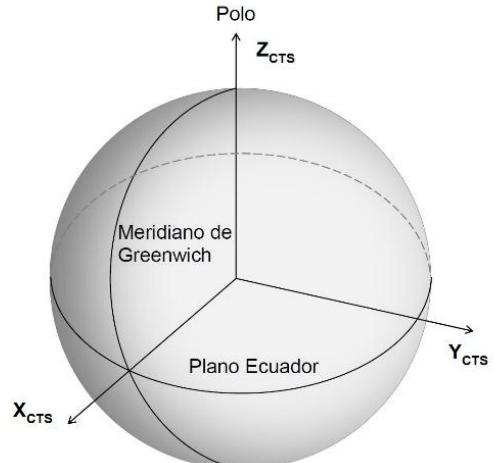


Figura 7. Sistema de referencia CTS.

1.1.2.4. Relación entre sistemas de referencia

La relación entre el Sistema de Referencia Ecuatorial Cartesiano y el Sistema orbital la determinan los ángulos Ω , i , ω tal y como se ve en la figura.

Para pasar las coordenadas al SEC necesitamos hacer tres giros:

- 1º) uno con argumento del perigeo ($-\omega$) lo que nos lleva la línea de ápsides hasta coincidir con la línea nodal, respecto a e_3 ,
- 2º) otro giro respecto al eje e_1 y con el ángulo de inclinación ($-i$) para hacer coincidir los planos del ecuador y el de la órbita, y
- 3º) otro con el ángulo de la longitud ascensión recta del nodo ascendente ($-\Omega$) para hacer coincidir la línea de ápsides con la línea del equinoccio vernal (punto Aries), respecto a e_3 .

Por último, para pasar a un sistema fijo a la Tierra, tenemos que realizar un último giro respecto al eje OZ. El ángulo de giro se denomina **Hora sidérea aparente en Greenwich ($-\theta_0$)** y es la diferencia angular sobre el plano del ecuador entre la posición del meridiano de Greenwich y el punto Aries. Por lo que el paso de coordenadas de un sistema orbital a un sistema CTS vendrá dado por la matriz de rotación:

$$\vec{R} = \vec{R}_3(-\theta_0) \cdot \vec{R}_3(-\Omega) \cdot \vec{R}_1(-i) \cdot \vec{R}_3(-\omega) \quad (25)$$

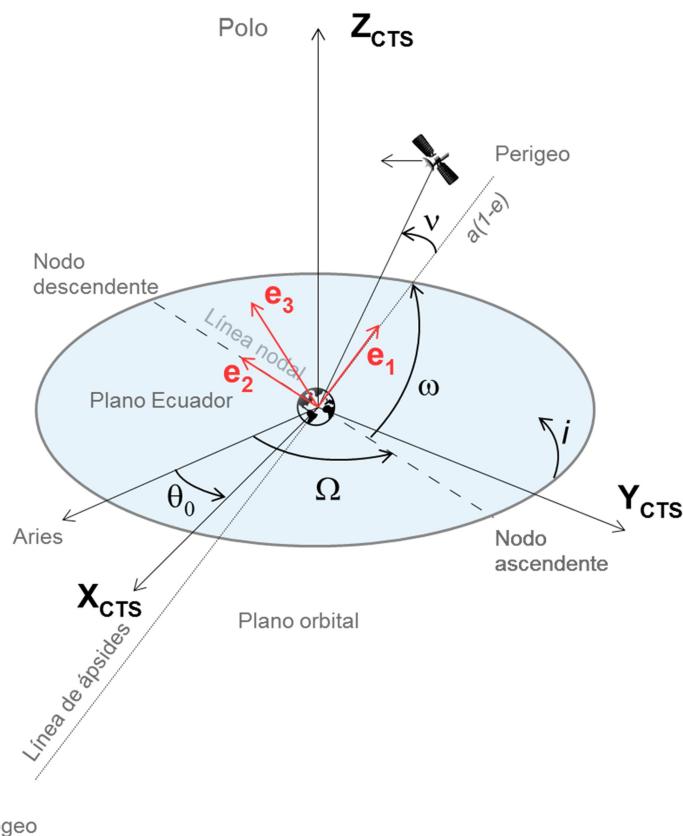


Figura 8. Relación entre sistema orbital y convencional terrestre.

1.2. PERTURBACIONES EN LAS ÓRBITAS

La trayectoria de un satélite alrededor de la Tierra no sigue el movimiento definido por las leyes de Kepler, sino que se mueve con ciertas oscilaciones alrededor de su eje de la órbita teórica como consecuencia de la influencia de otras fuerzas distintas de la fuerza de atracción gravitatoria.

Estas fuerzas externas al sistema de los dos cuerpos (Tierra-satélite), llamadas fuerzas perturbadoras, en el caso de los satélites GNSS son debidas principalmente a la deformidad de la Tierra, a la proximidad de otros cuerpos celestes (Sol y Luna) o a la presión de la radiación solar. La influencia de la atmósfera es importante en vehículos espaciales de órbitas bajas, pero inapreciable en los satélites GNSS ya que circulan a distancias muy alejadas de la atmósfera.

Algunos de los efectos perturbadores alteran la velocidad del satélite, con más o menos magnitud, en función de la altura de su órbita, pero otros varían de forma irregular independientemente de esta altura. Como consecuencia, tenemos que el movimiento resultante no es una órbita plana, sino que, en cada momento, en cada posición del satélite, se establece una órbita plana distinta denominada elipse osculatrix instantánea.

Con el seguimiento continuo de la posición de los satélites a partir de las bases de observación terrestres, se determina la posición instantánea sobre cada una de las elipses osculatrices pudiéndose entonces ajustar la trayectoria a una órbita kepleriana media, o trayectoria teórica más aproximada. Como se verá, esta aproximación sólo será suficientemente ajustada a la realidad en un intervalo de tiempo determinado a partir de la observación (futuro o pasado) que variará en función de la precisión de las observaciones y los modelos de predicción aplicados.

Hasta ahora hemos estudiado las órbitas de los satélites artificiales poniendo como premisas que:

- el sistema estaba formado únicamente por dos cuerpos: el del satélite y el de la Tierra,
- la Tierra era esférica,
- las masas de dichos cuerpos se concentraban en su centro, y
- no actuaba ninguna otra fuerza externa al sistema que no fuese la gravitacional entre ambos.

Así, obteníamos la expresión general de la órbita en función de sus seis elementos clásicos:

$$a, e, i, \Omega, \omega, \nu(t)$$

de los cuales los cinco primeros permanecían constantes y sólo la anomalía verdadera (ν) variaba con el tiempo. Veremos ahora que, como consecuencia de estas perturbaciones, también los otros elementos

sufren variaciones durante el periodo de rotación.

Como sabemos, la realidad es mucho más compleja que la simplificación que nos propuso Kepler. Nos vemos obligados a tener en cuenta muchos más factores externos si queremos predecir la trayectoria de un satélite con cierta precisión. No podemos obviar la presencia de otros cuerpos celestes, sobretodo la Luna y el Sol. Tampoco podemos olvidar que la deformidad de la Tierra, tanto en geometría como en estructura, obligan a considerarla como una masa no uniforme con un centro de gravedad variable e inestable.

Todos estos factores, y otros que veremos a continuación dan lugar a una serie de perturbaciones en la órbita de un satélite que, aunque el objetivo final de este trabajo no requiera de un posicionamiento preciso de cada satélite, no se pueden despreciar.

1.2.1. EFECTOS PERTURBADORES

Podemos clasificar las perturbaciones atendiendo a distintos criterios. Por ejemplo, según sea la naturaleza de la fuerza que las origina, tendríamos:

- las que son causa de la influencia de otros cuerpos celestes con masa suficientemente grande,
- las de considerar la Tierra como un elemento no esférico y con una masa no homogénea, y
- las resultantes de fuerzas no gravitacionales.

Según el periodo de influencia:

- seculares: su efecto es acumulativo y hacen que los elementos keplerianos varíen linealmente con el tiempo.
- Periódicas: producen una variación sinusoidal en los elementos keplerianos a lo largo del tiempo.
 - de periodo corto: con un periodo menor al periodo de la órbita
 - de periodo largo: con un periodo mayor al de la órbita

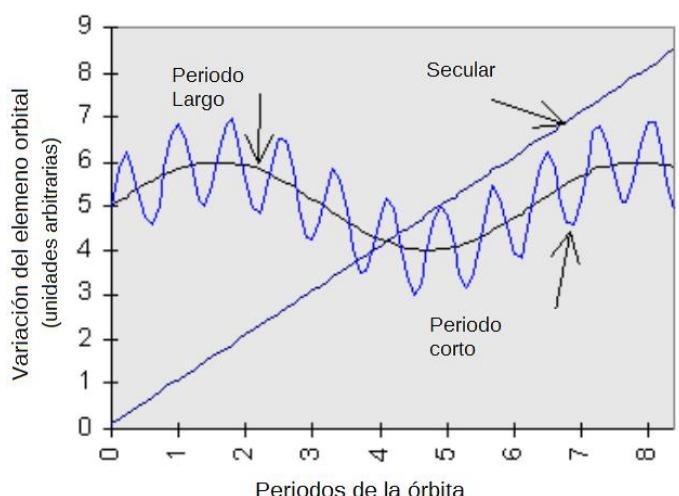


Figura 9. Efectos de larga duración de las perturbaciones [57].

1.2.1.1. Efecto de la atracción gravitatoria del Sol y de la Luna

Este efecto da lugar a variaciones de largo periodo y se obtiene a partir de los elementos orbitales de los astros correspondientes. Principalmente afecta a los ángulos i , Ω , ω .

Cuando el satélite está más cerca del Sol, o sea, cuando está en la cara de día de la Tierra, la fuerza de la gravedad del Sol es mayor. Por el contrario, cuando está más lejos del astro, en la cara de noche de la Tierra, su influencia será menor. Esta variación de la gravedad del Sol sobre el satélite produce dos efectos en su órbita.

Por una parte, provoca que el eje de la elipse se acorte en la dirección Sol-Tierra. Por otra, produce un par que provoca un giro en la inclinación de la órbita. Según [6] esta variación viene dada por la expresión:

$$\Delta i_{sol} = \frac{3 \mu_s r^2}{4 h r_s^3} \sin\Omega \sin\hat{i}_s \cos\hat{i}_s \quad (26)$$

Donde

$$\mu_s = 1,32686 \cdot 10^{11} \text{ km}^3/\text{s}^2 \text{ constante gravitacional de Sol,}$$

$$r = \text{altura de la órbita,}$$

$$r_s = 1,49592 \cdot 10^8 \text{ km es la distancia aproximada entre el Sol y la Tierra,}$$

$$h = 129,640 \text{ km}^2/\text{s}$$

$$\hat{i}_s = 23,45^\circ \text{ es la inclinación del Sol,}$$

$$\Omega = \text{ascensión recta del nodo ascendente del satélite.}$$

Por otra parte, la variación de la ascensión recta del nodo ascendente y del argumento del perigeo del satélite son, según [6]:

$$\Delta\Omega_{sol} = -\frac{0,00154\cos(i)}{\eta} \quad (27)$$

$$\Delta\omega_{sol} = \frac{0,00077(4 - 5\sin^2(i))}{\eta} \quad (28)$$

La influencia de la Luna sobre la inclinación de la órbita del satélite tampoco es despreciable y podría asimilarse a la perturbación que recibe la órbita terrestre debido a la proximidad de otros planetas mayores del sistema solar.

$$\Delta i_{luna} \approx \frac{3 \mu_l r^2}{4 h r_l^3} \sin(\Omega - \Omega_l) \sin\hat{i}_l \cos\hat{i}_l \quad (29)$$

Donde

$\mu_l = 4,9028 \cdot 10^3 \text{ km}^3/\text{s}^2$ es la constante gravitacional de la Luna.

$r_l = 3,844 \cdot 10^5 \text{ km}$ es la distancia aproximada entre la Luna y la Tierra

i_l = entre $18,3^\circ$ a $28,6^\circ$ es la inclinación de la Luna

Ω_l = ascensión recta del nodo ascendente de la Luna.

La variación de la ascensión recta del nodo ascendente y del argumento del perigeo del satélite por influencia lunar son [6]:

$$\Delta\Omega_{luna} = -\frac{0,00338\cos(i)}{\eta} \quad (30)$$

$$\Delta\omega_{luna} = \frac{0,00169(4 - 5\sin^2(i))}{\eta} \quad (31)$$

Según un estudio expuesto por el Centro Nacional de Información Geográfica en una de sus publicaciones (Geodesia Superior Vol.II, J.B.M.B) tras una observación de 4 horas sobre satélites de la constelación GPS, los efectos sobre los elementos orbitales, expresados en metros de desviación respecto a la órbita teórica fueron:

Tabla 2. Perturbaciones por efecto del Sol y la Luna.

| Elemento orbital | Luna y Sol |
|------------------------------|------------|
| Δa | 220 m |
| Δe | 140 m |
| Δi | 80 m |
| $\Delta\Omega$ | 80 m |
| $\Delta\omega$ y $\Delta\nu$ | 500 m |

Se puede decir que las influencias del Sol y de la Luna son de similar magnitud ya que, aunque la distancia entre ellos y el satélite sean muy distintas, también lo son las masas de dichos cuerpos. La Luna ejerce sobre el satélite una aceleración media de $5 \cdot 10^{-6} \text{ m/s}^2$, y el Sol ejerce sobre el satélite una aceleración media de $2 \cdot 10^{-6} \text{ m/s}^2$. El efecto de los otros planetas es casi despreciable (y normalmente así se considera) incluso en las mediciones para la Geodesia espacial ($3 \cdot 10^{-10} \text{ m/s}^2$).

1.2.1.2. Perturbación por la no esfericidad de la Tierra

Como hemos visto, la no esfericidad de la Tierra nos imposibilita considerarla como una masa puntual. Como todo planeta en movimiento de rotación, la Tierra se ensancha en la zona ecuatorial por el efecto de la fuerza centrífuga. El eje polar es 21 km menor que el eje ecuatorial que a la vez tampoco es de igual magnitud en todo su plano: en la longitud 165°E mide 20 m menos que en la longitud 75°E. Esto provocará una deriva del satélite Este-Oeste hacia dos puntos de equilibrio estable (en ausencia de otras acciones) que corresponden a los extremos del eje mayor de la elipse ecuatorial (75°E y 105°W).

Los achatamientos de los polos y la forma más abultada en el ecuador, dan lugar a un campo de gravedad totalmente asimétrico. No podemos considerar este campo como un valor fijo, sino que vendrá dado en función de la posición del satélite. No sólo dependerá de su distancia respecto al centro de la Tierra, como veíamos hasta ahora, sino que también de la longitud (λ) y la latitud (ϕ).

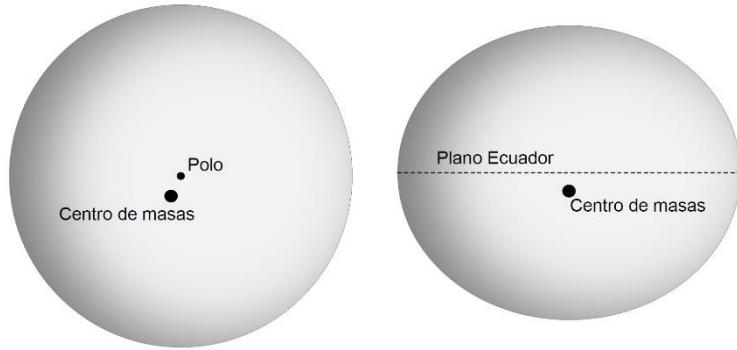


Figura 10. Desplazamiento del centro de masas de la Tierra.

Pasamos entonces de tener un valor de la gravedad que sólo estaba en función de la constante gravitacional de la Tierra μ_T y de la distancia entre su centro de masa y el satélite (r) a un valor que es función también de su posición relativa a la Tierra.

$$U = -\frac{\mu}{r} + B(r, \phi, \lambda) \quad (32)$$

La aceleración gravitacional en cualquier punto de la Tierra se expresa comúnmente como función expresada en términos polinomios de Legendre y coeficientes adimensionales J_n [6]:

$$B(r, \phi, \lambda) = \frac{\mu}{r} \left\{ \sum_{n=2}^{\infty} \left[\left(\frac{R_T}{r}\right)^n J_n P_{n0} \cos \phi + \sum_{m=1}^n \left(\frac{R_T}{r}\right)^n (C_{nm} \cos m\lambda + S_{nm} \sin m\lambda) P_{nm} \cos \phi \right] \right\} \quad (33)$$

donde

R_T es el radio de la Tierra en el ecuador (aproximadamente 6,378 km),

J_n son los coeficientes armónicos zonales,

P_{nm} son los polinomios de Legendre,

C_{nm} y S_{nm} son los coeficientes armónicos teserales (para $n \neq m$) y los coeficientes armónicos sectoriales (par $n = m$), respectivamente.

Los armónicos zonales dependen de la latitud y los teserales dependen de la longitud, por lo que se tienen que tener en cuenta cuando la órbita del satélite es geosíncrona, cuando permanece casi fijo respecto a la Tierra. Se considera que para la mayoría de las órbitas, a excepción de las geosíncrona, los ejes terrestres son similares y por lo tanto sólo se considera las correcciones armónicas zonales reduciendo la expresión a una dependencia sólo de la distancia al centro de masas y a la latitud:

$$U = -\frac{\mu}{r} \left[1 - \sum_{n=2}^{\infty} J_n \left(\frac{R_T}{r} \right)^2 P_n(\cos\phi) \right] \quad (34)$$

Si consideramos sólo hasta el término de segundo orden del polinomio de Legendre P_2 tenemos:

$$U \approx -\frac{\mu}{r} \left[1 - J_2 \left(\frac{R_T}{r} \right)^2 \frac{3\cos^2\phi - 1}{2} + \dots \right] \quad (35)$$

Los armónicos zonales, cuyos valores son adimensionales, son el resultado de observar el movimiento de los satélites alrededor de los planetas y son característicos de cada uno de ellos. Para el caso que nos interesa, el del planeta Tierra, estos valores son:

$$J_0 = 0 \text{ (sistema de coordenadas = centro de masas de la Tierra)}$$

$$J_2 = 0,00108263$$

$$J_3 = -2,33936 \cdot 10^{-3} \cdot J_2$$

$$J_4 = -1,49601 \cdot 10^{-3} \cdot J_2$$

...

De estos, J_2 es el más importante y se conoce como el coeficiente de aplanamiento de la Tierra siendo su valor mil veces mayor que J_3 y J_4 . Sin embargo, para un modelado más preciso del aplanamiento de la Tierra se tienen en cuenta para el cálculo de la función potencial.

Tabla 3. Valor numéricico de J_2 para cada planeta del Sistema Solar y para la Luna.

| Planeta | Aplanamiento (f) | J_2 |
|----------|----------------------|--------------------------|
| Mercurio | 0,000 | 60×10^{-6} |
| Venus | 0,000 | $4,458 \times 10^{-6}$ |
| Tierra | 0,003353 | $1,08263 \times 10^{-3}$ |
| Marte | 0,00648 | $1,96045 \times 10^{-3}$ |
| Júpiter | 0,06487 | $14,736 \times 10^{-3}$ |
| Saturno | 0,9796 | $16,298 \times 10^{-3}$ |
| Urano | 0,02293 | $3,34343 \times 10^{-3}$ |
| Neptuno | 0,01708 | $3,411 \times 10^{-3}$ |
| Luna | 0,0012 | $202,7 \times 10^{-6}$ |

Para cada uno de estos valores tendríamos que:

$$\begin{aligned}
 U_0 &= -1 \\
 U_{J2} &= \left(\frac{a}{r}\right)^2 J_2 \frac{1}{2} (3\cos^2\phi - 1) \\
 U_{J3} &= \left(\frac{a}{r}\right)^3 J_3 \frac{5}{3} (\cos^3\phi - \frac{3}{5}\cos\phi) \\
 U_{J4} &= \left(\frac{a}{r}\right)^4 J_4 \frac{35}{8} (\cos^4\phi - \frac{6}{7}\cos^2\phi + \frac{3}{35})
 \end{aligned} \tag{36}$$

Lo que supondría que

$$U = \frac{\mu}{r} [U_0 + U_{J2} + U_{J3} + U_{J4} + \dots] \tag{37}$$

Si nos centramos sólo en una primera aproximación (J_2), ¿qué efectos ejerce el aplanamiento sobre la órbita de un satélite? La protuberancia en el ecuador produce una tensión en la órbita que modifica varios elementos orbitales a lo largo del tiempo.

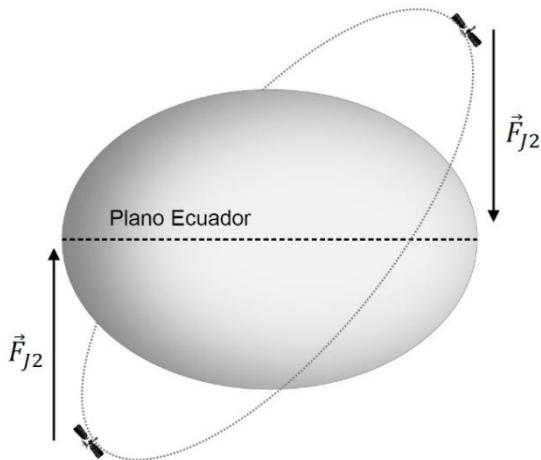


Figura 11. Efecto de J_2 sobre la órbita de un satélite.

Esta tensión no produce cambios ni en la inclinación de la órbita (i) ni en sus dimensiones (a, e). Sin embargo, sí que afecta a la ascensión recta del nodo ascendente (Ω) y al argumento del perigeo (ω) dentro del plano de la órbita.

Si imaginamos una peonza, cuando esta está parada, la gravedad la hace caer, pero si se mantiene girando sobre su punta, aunque la gravedad intenta derribarla, el movimiento angular lo impide. Dicho movimiento, denominado de precesión influye pues en Ω y ω , con lo que con la influencia de J_2 la fuerza ya no proviene sólo del centro de la Tierra y se producen variaciones siguientes:

- *Regresión de los nodos:*

Produce una rotación del plano orbital. La fuerza gravitatoria terrestre provoca un movimiento de rotación perpendicular al plano del ecuador lo que transmite un giro de la línea de nodos parecido al movimiento de precesión terrestre. Este giro depende, en primer lugar, del ángulo de inclinación (i) de la órbita. Su efecto es mayor cuando más cerca de la protuberancia (ecuador) viaje el satélite. Por lo tanto, en las órbitas de baja inclinación su efecto será mayor y en las órbitas polares su efecto será casi nulo.

En segundo lugar, la regresión nodal está también relacionada con la altura de la órbita. Cuando mayor sea esta, menor será el efecto de la gravedad, disminuyendo dicho efecto en una proporción del cuadrado inverso a la distancia (Ley Gravitacional de Newton).

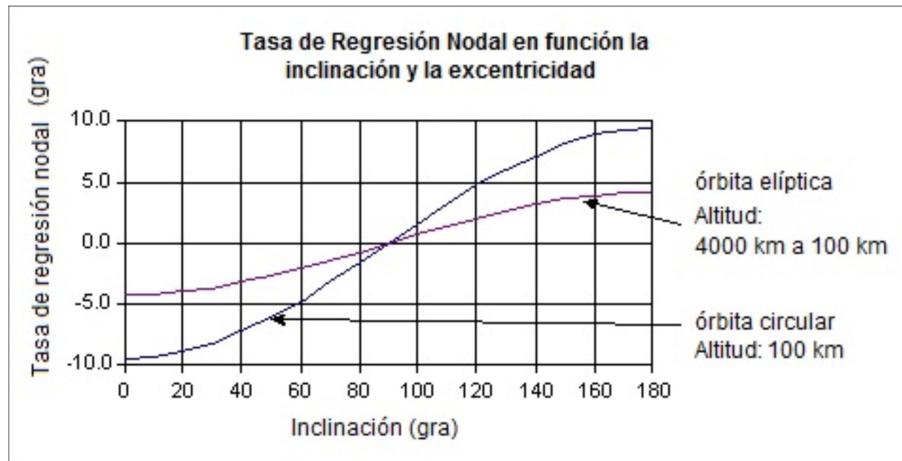


Figura 12. Tasa de la Regresión Nodal [7].

Su efecto puede llegar a ser de unos 9° por día en órbitas de pequeña inclinación y altitud menores a 200 km.

El satélite sufre una desviación hacia el Oeste para las órbitas directas con inclinaciones inferiores a 90° y hacia el Este con órbitas retrógradas con inclinación superior a 90°.

$$\Delta\Omega_{J2} = -\frac{9,9641}{(1-e^2)^2} \cdot \left(\frac{R_T}{a}\right)^{3.5} \cdot \cos i \quad \text{grados/día} \quad (38)$$

- *Rotación de la línea de los ápsides dentro del plano orbital:*

El efecto que se genera es un giro en la línea de ápsides en cada plano orbital instantáneo, lo que provoca una variación del argumento del perigeo (ω). Esta rotación se produce en el mismo sentido del movimiento del satélite en órbitas con inclinaciones menores a 63,4° o mayores de 116,6° y en sentido opuesto si la inclinación está entre estos dos valores.

$$\Delta\omega_{J2} = -\frac{4,9821}{(1-e^2)^2} \cdot \left(\frac{R_T}{a}\right)^{3.5} \cdot (5\cos^2 i - 1) \quad \text{grados/día} \quad (39)$$

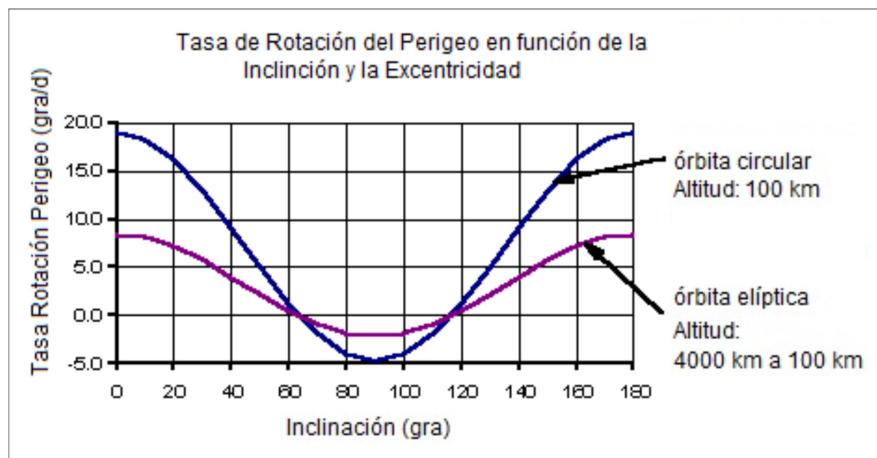


Figura 13. Tasa de Rotación del Perigeo [7].

Estos dos efectos se aprovechan actualmente en dos de las órbitas más utilizadas: la sincrónica con el Sol y la molniya. Las órbitas heliosíncronas (sincrónicas con el Sol), tienen en cuenta la regresión hacia el Este (órbitas mayores a 90º) que es de aproximadamente un grado por día. Esta variación coincide con la rotación de la Tierra alrededor del Sol cuyo valor también es aproximadamente de un grado por día. Dejando actuar esta perturbación se consigue que el satélite esté cada día con la misma inclinación respecto al Sol siempre que pasa por el mismo punto sobre la Tierra. Los estudios que utilizan imágenes tomadas desde estos satélites, consiguen obtener series de datos diarios con la misma inclinación solar para cada día. Son de vital importancia para el estudio de modificaciones de elevaciones (mediante sombra) o para el estudio meteorológico.

La órbita molniya (denominación rusa de relámpago) tiene una inclinación de 63,4º con una alta excentricidad ($e = 0,7$) y un periodo de 12 horas, de las cuales, durante 11 horas, el satélite se encuentra sobre el hemisferio norte antes que se “dispare como un rayo” hacia su perigeo en el hemisferio sur. Con esta inclinación se consigue que el perigeo no rote. La idea de esta órbita fue concebida por la inviabilidad de poner en órbita geosincrónica los satélites de comunicación rusos. La posición del satélite en esta órbita se mantiene en un pequeño intervalo de +/- 15º durante 8 horas con un movimiento aparente geoestacionario. Con la colocación de 3 satélites en órbitas molniyanas se consigue una cobertura diaria total, tal y como lo haría un satélite geosincrónico.

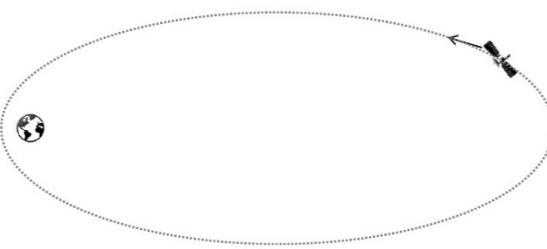


Figura 14. Órbita molniya.

Tal y como veremos en el cuadro resumen, el efecto del achatamiento terrestre sobre el potencial gravitatorio, es el que genera mayor perturbación, del orden de 1000 veces mayor que cualquiera del resto de efectos.

1.2.1.3. Perturbación por fuerzas no gravitatorias

- *Fricción atmosférica:*

Como sabemos, todos los cuerpos en movimiento dentro de la atmósfera, además de la fuerza gravitatoria terrestre también sufren una desaceleración debido a la influencia de la fricción con la densidad del aire. De forma similar los satélites más cercanos a la Tierra, con órbitas bajas (por debajo de los 100 km) perciben la resistencia aerodinámica debida a la fricción con las partículas de la atmósfera. De hecho, este es el segundo efecto más perturbador en las órbitas bajas. Dependiendo del grado de aproximación que estemos utilizando a la hora de calcular las órbitas, este efecto también se incluirá o no, en órbitas elevadas (órbitas de los satélites GNSS).

Si no se desprecia el pequeño efecto debido a la propia rotación de la atmósfera, se considera que esta fuerza es tangente a la órbita y opuesta a la dirección del vector velocidad del satélite. Además, esta fricción es proporcional a la densidad de la atmósfera. Como esta densidad disminuye con la altura, el efecto sobre el satélite será mayor cuanto más cercano se encuentra este de la Tierra, o sea en el perigeo.

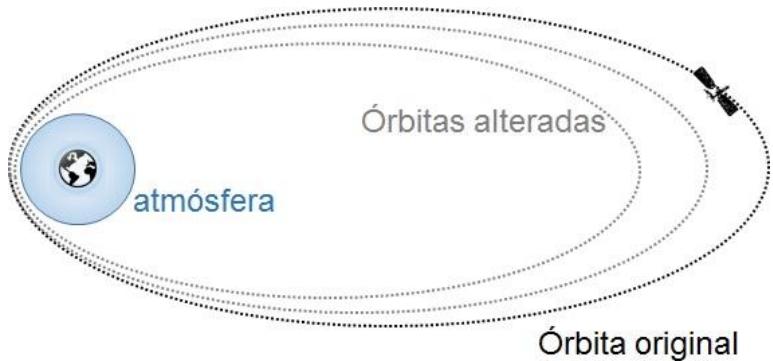


Figura 15. Efecto de la fricción atmosférica en la órbita de un satélite.

Por lo tanto, en cada rotación, se va reduciendo el semieje mayor de la órbita, así como su excentricidad volviendo la órbita cada vez menos elíptica (más circular). Efectivamente, un satélite con órbita elíptica cuando está en el perigeo tiene mayor velocidad que en cualquier otro momento del periodo, pero esta será mayor que si se encontrara en ese mismo punto siendo su órbita circular. La resistencia al avance debido a la fricción con la atmósfera, lo aproxima a cada paso por el perigeo a la velocidad de la órbita circular. No se altera ni la inclinación ni el nodo ya que sólo actúa en el plano de la órbita.

El modelado de este efecto sobre la velocidad del satélite es muy difícil de conseguir. Hay múltiples factores que influyen en el coeficiente de fricción de la atmósfera superior: el ciclo día-noche, el ciclo estacional, la fluctuación del campo magnético, las manchas solares, entre otros. Y como no, el coeficiente de fricción del propio satélite que depende de entre otros de su forma, materiales u orientación. Por consiguiente, la mayoría de veces se recurre a las observaciones empíricas para su determinación.

Su efecto desacelerador puede llegar a ser de 10^{-3} m/s² a 10^{-9} m/s² en las órbitas bajas y despreciable en las altas

- *Efecto de las mareas (terrestres y oceánicas):*

El efecto que las mareas ejercen en las perturbaciones es pequeño en órbitas de gran altura como las de los satélites GNSS, pero de gran importancia cuando estas disminuyen. Al analizar el efecto de las mareas terrestres se estudia si estas están producidas por el Sol o por la Luna. La influencia de las mareas oceánicas es muy compleja de estudiar y se recurre a modelos de áreas para transformar, a incremento de masa, la altitud de un punto de la superficie del mar respecto a una superficie media. Con el incremento de masa final se

puede recalcular la variación del potencial terrestre y así la perturbación que este genera sobre el satélite.

Siguiendo el ejemplo anterior sobre las observaciones hechas en un satélite de la constelación GPS, la desaceleración que se produce por el efecto de las mareas terrestres es de $2 \cdot 10^{-9} \text{ m/s}^2$ y por las mareas oceánicas de $5 \cdot 10^{-10} \text{ m/s}^2$ lo que se traduce en menos de 1 m en el intervalo de 1 día. Por lo que a nuestro trabajo se refiere, sería un efecto despreciable.

- *Presión de la Radiación solar:*

Para los satélites con órbitas altas (como las de las constelaciones GNSS) las dos fuerzas no gravitatorias anteriores actúan en valores casi despreciables, pero los efectos que la radiación solar no pueden ser ignorados, sobre todo en aquellos satélites con grandes paneles solares. La perturbación es debida a la presión electromagnética que ejercen los fotones (ondas electromagnéticas viajando a la velocidad de la luz) procedentes del Sol sobre la superficie del satélite situada de cara al astro.

Debido a la continua radiación de partículas procedentes del Sol se producen dos efectos perturbadores: uno, al chocar estas partículas sobre el satélite, y el otro, denominado efecto albedo, al chocar las partículas que se han reflejado previamente en la superficie terrestre. El efecto directo es más importante y está en función del área efectiva del satélite. Se hace especialmente importante cuando el satélite viaja en dirección a la radiación solar pudiendo generar una aceleración de 10^{-7} m/s^2 que se traduciría a unos 10 m después de pocas horas de observación.

Para los cálculos muy precisos también se considera los efectos de:

- Rozamiento de otras partículas cargadas eléctricamente,
- Resistencia debido al viento solar,
- Resistencia debido al polvo galáctico,
- Interacción del satélite con el campo magnético terrestre,
- Efecto de la radiación térmica del propio satélite.
- Impulsos inesperados causados por gases de escape o mal funcionamiento.

Finalmente, considerando todas las perturbaciones, tenemos que la expresión general de la órbita será:

$$(a, e, i, \Omega, \omega, v)(t) = (a, e, i, \Omega, \omega, v)(t_0) + (\Delta a, \Delta e, \Delta i, \Delta \Omega, \Delta \omega, \Delta v)(t-t_0)$$

Tabla 4. Fuerzas Perturbadoras en los satélites de las constelaciones GNSS. [3]

| Fuerza perturbadora | Aceleración (m/s ²) | Error en órbita tras 3h metros | Error en órbita tras 24h metros |
|----------------------------------|------------------------------------|-----------------------------------|------------------------------------|
| Campo gravitatorio (armónico J2) | $5 \cdot 10^{-5}$ | 2 000 | 10 000 |
| Atracción lunar | $5 \cdot 10^{-6}$ | 5 – 75 | 3 000 |
| Atracción solar | $2 \cdot 10^{-6}$ | 5 – 150 | 1 500 |
| Efecto mareas | 10^{-9} | - | 0,3 |
| Presión Radiación Solar | 10^{-7} | 5 – 10 | 2 |
| Fricción atmosférica | $< 10^{-9}$ | - | - |

Teniendo en cuenta las perturbaciones que hemos descrito hasta ahora, podemos conocer una trayectoria aproximada de un satélite a través de un modelo que tenga en cuenta dichas perturbaciones y esperar una aproximación más correcta de la posición y velocidad buscadas. Cuantas más variables perturbadoras se tengan en cuenta, mejor será la aproximación obtenida a través del modelo. Pero cualquier modelo sólo será, al fin y al cabo, un intento de aproximar matemáticamente un fenómeno real y como consecuencia, sólo se podrá saber con exactitud su trayectoria mediante observaciones directas sobre el mismo satélite con un seguimiento en tiempo real.

1.3. MODELOS DE PROPAGACIÓN

Una de las principales actividades de la astrodinámica es entender y modelar las perturbaciones de las órbitas de los miles de cuerpos que se mueven alrededor de la Tierra. Según algunas estimaciones de la Agencia Espacial Europea (ESA) (Erwan Matton, 2016), actualmente hay más de 29.000 cuerpos artificiales de más de 10 cm de longitud orbitando alrededor de la Tierra, entre los aún operativos y los ya no operativos. En la Iniciativa *Clean Space* de la ESA se trabaja para dar respuesta a preguntas como: ¿Cuántos hay?, ¿Dónde están?, ¿Qué daño pueden ocasionar?, ¿Cómo eliminarlos?, o ¿Cómo evitar crear más “escombros espaciales”?

La mayoría de esos cuerpos están monitorizados desde las estaciones terrestres cuyas observaciones consiguen determinar su posición y su trayectoria en cada momento. Pero para evitar colisiones entre estos cuerpos, es necesario predecir su posición con ciertas garantías y así, corregir su trayectoria (en el caso que sea posible).

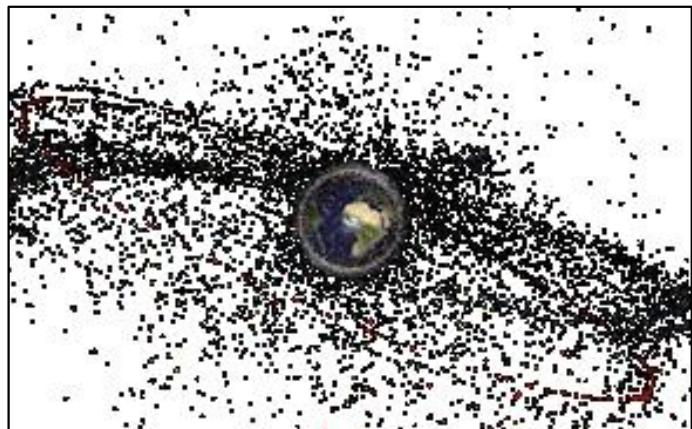


Figura 16. Distribución de los escombros espaciales [8].

Uno de nuestros objetivos en este trabajo, es también determinar la posición de algunos de estos cuerpos, los satélites GNSS que se mueven en órbitas altas, entre 20.000 km y 23.000 km, dónde los “escombros espaciales” no son tan abundantes. Aun así, en los últimos 10 años se ha producido un incremento considerable en la cantidad de satélites artificiales, aumentando también el número de objetos inutilizados en órbita. Este aumento requiere de mayor capacidad de cálculo, lo que se consigue mediante los conjuntos de algoritmos denominados propagadores.

Si bien para nuestro objetivo, no es necesario determinar la posición de cada satélite con precisión métrica, sí que necesitamos conocer esta posición aproximada en el cielo visible para poder determinar si podemos contar con él en el momento concreto de nuestras observaciones en el campo. Como veremos, los propios receptores GNSS, llevan incorporados propagadores que, mediante las efemérides recibidas, les permiten determinar la posición aproximada de cada satélite.

1.3.1. ¿QUÉ ES UN PROPAGADOR?

Llamaremos propagador al esquema numérico que reúne las ecuaciones y los métodos de resolución que permiten el cálculo aproximado de la órbita del satélite. Mediante diversos algoritmos matemáticos, permite obtener efemérides futuras, posición y velocidad de un satélite a partir de los elementos dados en un instante concreto. Conocidas las ecuaciones del movimiento orbital de los satélites y las perturbaciones que actúan sobre él, los propagadores obtendrán la mejor aproximación posible de la trayectoria real del satélite.

Básicamente, dado $(a_0, e_0, i_0, \Omega_0, \omega_0, M_0)$ en un instante t_0 , se calcula $(a, e, i, \Omega, \omega, M)$ en el instante t . En ausencia de perturbaciones el propagador básico sería, tal y como ya hemos visto, el modelo kepleriano o de los dos cuerpos.

Pero como sabemos, estas perturbaciones no podemos obviarlas. Los modelos más complejos usan las ecuaciones de Lagrange o Gauss integrándolas con el modelo más completo posible de perturbaciones. Entre el modelo básico y los más complejos, existen modelos intermedios, semianalíticos, que permiten obtener resultados muy aceptables para muchas aplicaciones.

Normalmente tendemos a considerar que el propagador que incluya más perturbaciones será el que nos dará mayor precisión. Pero no siempre es así. Por ejemplo, los mejores modelos pueden no ser los más adecuados para una predicción a muy largo plazo.

La selección del propagador se tiene que hacer en base al escenario en que nos moveremos. Por ejemplo, los satélites geoestacionarios orbitan a unos 40.000 km y, por lo tanto, no están afectadas por las perturbaciones atmosféricas y la influencia J_2 es relativamente baja. En estos satélites la diferencia a corto plazo entre usar propagadores básicos o complejos podría ser relativamente pequeña. Pero para un satélite en órbita baja, estas diferencias a corto plazo, serían muy significativas.

Por último, es importante tener en cuenta el nivel de precisión de los datos iniciales del propagador, ya que marcarán o influirán, en la precisión de la trayectoria obtenida.

1.3.2. TIPO DE PROPAGADORES

1.3.2.1. De dos cuerpos

Como hemos visto, el propagador básico es el dos cuerpos o propagador de movimiento kepleriano, que utiliza la misma técnica básica descrita en la ecuación de dos cuerpos del desarrollo del movimiento. Se supone que la Tierra es una esfera perfecta y considera solamente, para la

determinación de la órbita, la fuerza de la gravedad de la Tierra, la cual es modelada como un punto de masa.

$$(a, e, i, \Omega, \omega) = (a_0, e_0, i_0, \Omega_0, \omega_0) \quad y \quad (a, e, i, \Omega, \omega) = M_0 + n(t - t_0)$$

1.3.2.2. Propagadores J2 y J4

Ambos propagadores tienen en cuenta las variaciones en la órbita debidas a la irregularidad geométrica de la Tierra.

El propagador J2 considera sólo los efectos de primer orden de J_2 , pero no modela el roce atmosférico o la fuerza gravitacional solar o lunar. Este efecto provoca cambios seculares en los elementos orbitales a lo largo del tiempo. Es un propagador simple de usar y útil en órbitas bajas, donde la influencia de J_2 es grande.

$$a = a_0 \quad e = e_0 \quad i = i_0$$

$$\Omega = \Omega_0 - \frac{3}{2}n \frac{r_T^2}{r^2} J_2 \cos i(t - t_0) \quad (40)$$

$$\omega = \omega_0 + \frac{3}{4}n \frac{r_T^2}{r^2} J_2 (5 \cos i^2 - 1)(t - t_0) \quad (41)$$

$$M = M_0 + \left(n + \frac{3}{4}n \frac{r_T^2}{r^2} J_2 \sqrt{1 - e^2} (2 - 3 \sin i^2) \right) (t - t_0) \quad (42)$$

El propagador J4 (de segundo orden) también representa variaciones importantes en los elementos orbitales debido a considerar la Tierra como una esfera aplastada en los polos. El propagador J4 representa los efectos J_2 de periódicos primer y segundo orden, así como los efectos J_4 de primer orden. Debido a que los efectos J_2 de segundo orden y J_4 de primer orden son muy pequeños, hay muy poca diferencia entre las órbitas generadas por estos dos propagadores.

1.3.2.3. Propagadores de los receptores GNSS

Los receptores GNSS incorporan los algoritmos de cálculo que, mediante las efemérides recibidas (radiodifundidas) en el mensaje de navegación, les permiten reconstruir la posición de cada uno de los satélites de la constelación. Estas efemérides, actualizadas cada pocas horas, contienen:

- los seis parámetros orbitales keplerianos:

$$(a, e, \Omega_0, \omega_0, i_0, M_0)$$

- Nueve parámetros de corrección de las perturbaciones (tres términos seculares de corrección y seis términos periódicos de corrección)

$$(\Delta n, \Omega', i', C_{uc}, C_{us}, C_{rc}, C_{rs}, C_{ic}, C_{is})$$

Estos términos de corrección consideran los efectos de perturbación debido a la no esfericidad de la Tierra, a los efectos directos de la marea y a los efectos de la presión de la radiación solar.

- Y cinco parámetros de tiempo

$$(t_e, t_0, a_0, a_1, a_2)$$

Además, se requiere de dos parámetros adicionales que se definen como parte del sistema de referencia utilizado. Por ejemplo, en el caso de los satélites de la constelación GPS, el WGS-84 se define por la constante gravitacional ($\mu = \text{GMT} = 3.9860014418 \cdot 10^{14} \text{ m}^3/\text{s}^2$) y la velocidad angular de rotación de la Tierra ($\Omega_e = 7292115 \cdot 10^{-11} \text{ rad/s}$).

Los propagadores de órbitas utilizados en los receptores GNSS, con los datos radiodifundidos, basan sus cálculos en las siguientes ecuaciones [4]:

$$M = M_0 + \left(\sqrt{\frac{\mu}{a^3}} + \Delta n \right) (t - t_{0e}) \quad (43)$$

$$\Omega = \Omega_0 + (\Omega' - \Omega'_{Tierra}) t - \Omega'_{Tierra} t_{0e} \quad (44)$$

$$u = u_0 + C_{uc} \cos(2u_0) + C_{us} \sin(2u_0) \quad (45)$$

$$i = i_0 + i'(t - t_{0e}) + C_{ic} \cos(2u_0) + C_{is} \sin(2u_0) \quad (46)$$

Donde, como ya vimos en capítulos anteriores, una vez determinada M podemos resolver la ecuación de Kepler ($M = E - e \sin E$) y, con ello hallar la anomalía verdadera (v) a partir de E , y así calcular

$$u_0 = \omega + v \quad (47)$$

Para el cálculo de la distancia geocéntrica aproximada del satélite se calcula

$$r = r_0 + C_{rc} \cos(2u_0) + C_{rs} \sin(2u_0) \quad (48)$$

Donde

$$r_0 = a(1 - e \cos E) \quad (49)$$

Si bien las anteriores fórmulas permiten reconstruir con precisión razonable la posición de los satélites GNSS, dicha precisión no es suficiente para muchas de las aplicaciones que requieren de una precisión submétrica. Para estas aplicaciones se pueden obtener efemérides post-procesadas mucho más precisas (de hasta cm). Estas efemérides editadas a posteriori, se basan en las observaciones realizadas a los satélites desde las estaciones terrestres.

1.3.2.4. Conjuntos SGP (*Simplified General Perturbations*)

Este conjunto de propagadores son un total de cinco modelos matemáticos (SGP, SGP4, SDP4, SGP8 y SDP8). Todos ellos calculan los vectores de estado en un sistema de coordenadas inerciales centrado en la Tierra (ECI). Este conjunto de modelos se refiere a menudo colectivamente como SGP4 por ser el más usado por su simplicidad.

Hace más de un cuarto de siglo, el Departamento de Defensa de los Estados Unidos (DoD) publicó las ecuaciones y el código fuente que utilizaba para predecir las posiciones de los satélites a través del documento [9]. Debido a que los datos orbitales disponibles a través de la Administración Nacional de la Aeronáutica y del Espacio (NASA) [10] eran también los más utilizados y el formato de estos estaba pensado para ser usado por dicho modelo de ecuaciones, este código se convirtió en uno de los más usados entre los usuarios que necesitaban resultados precisos. Todos los informes publicados posteriormente sobre SGP4 han sugerido sólo mejoras en el código usado para implementarlo, y no cambios en la teoría subyacente.

En este sentido, se ha visto la necesidad de ajustar el modelo para que pudiese utilizarse conjuntamente para el cada vez mayor número de cuerpos puestos en órbita. Para tal fin, se han propuesto modelos que buscan tener en cuenta las perturbaciones pero que, realizando ciertas simplificaciones, favoreciera el tiempo de cálculo vs. la precisión de los resultados.

Para órbitas cercanas a la Tierra se utilizan los modelos SGP. El modelo SGP fue el primero en ser desarrollado en los años 60, actualizado con el SGP4 en los 70 y modificado en los 80 con el SGP8. Son modelos semianalíticos que tienen en cuenta perturbaciones de hasta el J_3 y rozamiento atmosférico.

Las extensiones SDP4 y SDP8 para los modelos SGP4 y SGP8, respectivamente, se usan para órbitas superiores a 5.000 km y modelan los efectos de la Luna y el Sol así como los efectos de la presión solar.

La precisión que se obtiene con estos propagadores es menor al Kilómetro [11], pero empeora entre 1km y 3km por cada día de predicción. Para reducir el error del algoritmo, se debe tomar observaciones precisas iniciales de los elementos orbitales. La precisión del algoritmo aumenta a medida que disminuyen los incrementos e tiempo entre cálculos.

Al realizar simplificaciones en un modelo de predicción, para aumentar la productividad, se asume un error en los resultados. Para corregir o minimizar la dimensión de este error, estos modelos necesitan puntos de referencia para cada satélite que se pretende estudiar. Estos puntos de referencia se generan mediante observaciones (directa o por radar) realizadas por estaciones terrestres a los satélites.

Los datos de estos puntos de referencia se distribuyen en un formato estándar denominado de dos líneas (TLE, *Two line elements*) que codifica una lista de elementos orbitales para cada satélite en un momento preciso en el tiempo, en una época concreta.

Para este trabajo, se ha utilizado el modelo publicado en [11] y [12] donde se revisan y actualizan las ecuaciones iniciales de [9].

1.4. EFEMÉRIDES: los datos de partida de los modelos de propagación

Los propagadores calculan las posiciones futuras de los satélites a partir de datos conocidos. Estos datos son las efemérides de cada satélite y contienen información sobre la situación actual de los satélites, su trayectoria y las perturbaciones que reciben.

1.4.1. ALMANAQUES

En los almanaques que los satélites GNSS incluyen en el mensaje de navegación se adjuntan los parámetros fundamentales de la órbita y los términos de corrección del reloj del satélite, para predecir la posición aproximada de todos los satélites su constelación.

Estos datos se actualizan cada seis días y su propósito es, por ejemplo, la inicialización del receptor, proporcionar al usuario datos para facilitar al receptor la búsqueda de los satélites o para el planeamiento y visualización de los satélites visibles en cada momento en un punto de coordenadas determinadas.

Además, los almanaques también se pueden descargar de distintas webs con distintos formatos. Por ejemplo el *U.S. Coast Guard Navigation Center* [13] distribuye los almanaques de la constelación GPS en los formatos YUMA y SEM:

YUMA: www.navcen.uscg.gov/?pageName=currentAlmanac&format=yuma-txt

SEM: <http://www.navcen.uscg.gov/?pageName=gpsSem>

1.4.2. EFEMÉRIDES RADIODIFUNDIDAS (*broadcast ephemerides*)

Las efemérides transmitidas desde los satélites en tiempo real, incluidas en el mensaje de navegación, están basadas en los datos de las observaciones de pseudodistancias tomadas desde las estaciones del segmento de control del sistema, y calculadas y enviadas a cada satélite. Los datos transmitidos no dejan de ser predicciones de los parámetros reales y, para garantizar la precisión (de algunos metros), sólo deben usarse durante el periodo aproximado de dos hora antes o dos horas después. A diferencia de los almanaques, en las efemérides transmitidas sólo se envía información del satélite que las transmite.

Tabla 5. Parámetros radiodifundidos en el mensaje de navegación [3].

| Parámetro | Unidad | Descripción |
|-------------------|--------------|---|
| ID | | Número PRN del satélite. |
| week | | Semana GPS actual. |
| t_{0e} | segundos | Tiempo de referencia de efemérides transmitidas, contando desde las 0h0'0" del domingo (valor máximo 604800 segundos = semana). |
| \sqrt{a} | metros | Raíz cuadrada del semieje mayor. |
| e | adimensional | Excentricidad. |
| M_0 | radianes | Anomalía media en el momento de referencia t_{0e} . |
| Ω_0 | radianes | Longitud del nodo ascendente en el plano orbitas en el momento de referencia. |
| ω_0 | radianes | Argumento del perigeo. |
| i_0 | radianes | Ángulo de inclinación en la época de referencia. |
| Δn | radianes/s | Diferencia del movimiento medio. |
| Ω' | radianes/s | cambio en la ascensión recta |
| i' | radianes/s | cambio en el ángulo de la inclinación |
| C_{uc} / C_{us} | radianes/s | coeficiente de corrección del argumento de la latitud (amplitud de armónicos cosenoidal/senoidal) |
| C_{rc} / C_{rs} | metros | coeficiente de corrección de la distancia geocéntrica (amplitud de armónicos cosenoidal/senoidal) |
| C_{ic} / C_{is} | radianes/s | coeficiente de corrección del ángulo de inclinación (amplitud de armónicos cosenoidal/senoidal) |
| t_c | segundos | época de referencia del reloj del satélite |
| a_0 | segundos | compensación del reloj del satélite |
| a_1 | segundos | desviación del reloj del satélite |
| a_2 | segundos | desviación de frecuencia de reloj del satélite |

A continuación se muestra un ejemplo de un fichero (en formato RINEX) de efemérides radiodifundidas, correspondiente a las 20h00m00s del 13 de enero de 2017:

```

 3.02          N GNSS NAV DATA      G GPS          RINEX VERSION TYPE
GR50 V4.02       ICGC           20170112 235942 UTC PGM RUN BY DATE
GPSA  7.4506D-09 -1.4901D-08 -5.9605D-08  1.1921D-07    IONOSPHERIC CORR
GPSB  8.8064D+04 -4.9152D+04 -1.9661D+05  3.2768D+05    IONOSPHERIC CORR
GPUT  3.7252902985D-09 1.509903313D-14  61440 1932    TIME SYSTEM CORR
          18     18   1929      7    LEAP SECONDS
          (...)          END OF HEADER
G12 2017 01 13 20 00 00 3.889612853527D-04-2.273736754432D-13 0.000000000000D+00
     6.200000000000D+01 1.432187500000D+02 3.821944913632D-09 6.281228440663D-01
     7.435679435730D-06 6.261964561418D-03 7.290393114090D-06 5.153606155396D+03
     5.040000000000D+05-3.911554813385D-08-2.057158077280D+00 6.146728992462D-08
     9.897694531506D-01 2.523750000000D+02 7.822014986778D-01-7.958545791091D-09
     2.478674675321D-10 1.000000000000D+00 1.931000000000D+03 0.000000000000D+00
     2.000000000000D+00 0.000000000000D+00-1.257285475731D-08 6.200000000000D+01
     4.968000000000D+05
  (...)
```

Dónde:

| | |
|---------------------|--|
| G12 | Número de satélite |
| 2017 01 13 | 13 enero de 2017 |
| 20 00 00 | 20:00:00 horas:minutos:segundos |
| 3.889612853527D-04 | Coeficiente a0 del polinomio de corrección del estado de reloj |
| -2.273736754432D-13 | Coeficiente a1 del polinomio de corrección del estado de reloj |
| 0.000000000000D+00 | Coeficiente a2 del polinomio de corrección del estado de reloj |
| 6.200000000000D+01 | IODE (Issue Of Data Ephemeris), edición de las efemérides |
| 1.432187500000D+02 | Crs Coeficiente del término seno de corrección al radio orbital (metros) |
| 3.821944913632D-09 | An Variación del movimiento medio (rad / seg) |
| 6.281228440663D-01 | M0 Anomalía media en la época TOE (Time Of Ephemeris) (rad) |
| 7.435679435730D-06 | Cuc Coeficiente del término coseno de corrección al argumento de la latitud, perigeo (rad) |
| 6.261964561418D-03 | e Excentricidad de la órbita |
| 7.290393114090D-06 | Cus Coeficiente del término seno de corrección al argumento de la latitud, perigeo (rad) |
| 5.153606155396D+03 | raíz cuadrada del semieje mayor de la órbita (metros) |
| 5.040000000000D+05 | TOE Tiempo de Referencia para la posición del satélite (segundos de la semana GPS) |
| -3.911554813385D-08 | Cic Coeficiente del término |
| -2.057158077280D+00 | Ω0 Longitud del nodo ascendente de la órbita al comienzo de la semana GPS (rad) |
| 6.146728992462D-08- | Cis Coeficiente del término seno de la corrección a la inclinación (rad) |
| 9.897694531506D-01 | i0 Inclinación de la órbita en la época TOE (rad) |
| 2.523750000000D+02 | Crc Coeficiente del término coseno de corrección al radio orbital (metros) |
| 7.822014986778D-01 | ω Argumento del perigeo (rad) |
| -7.958545791091D-09 | Ω' Variación de la ascensión recta (rad/seg) |
| 2.478674675321D-10 | i' Variación de la inclinación (rad/seg) |
| 1.000000000000D+00 | Códigos en L2 |
| 1.931000000000D+03 | Semana GPS |
| 0.000000000000D+00 | L2 P data flag (0 = OK) |
| 2.000000000000D+00 | Precisión de las efemérides (metros) |
| 0.000000000000D+00 | Salud del satélite (0 = OK) |
| -1.257285475731D-08 | TGD (segundos) |
| 6.200000000000D+01 | IODC Edición de los datos de reloj |
| 4.968000000000D+05 | Hora de transmisión del mensaje (segundos de la semana GPS) |

Las efemérides transmitidas también se pueden obtener del mensaje de navegación recibido por alguna estación permanente. Por ejemplo, de la web del *Institut Cartogràfic i Geològic de Catalunya* (ICGC) [14] se pueden descargar los ficheros de sus estaciones permanentes, o en el *Scripps Orbit and Permanent Array Center* (SOPAC) [15] se encuentran todos los mensajes de navegación recibidos por sus estaciones.

1.4.3. EFEMÉRIDES PRECISAS

Las efemérides precisas proporcionan la posición XYZ de alta precisión. Están realizadas por distintas agencias o instituciones que conforman el segmento terrestre de seguimiento, independiente del sistema de control. A diferencia de las efemérides radiodifundidas, las efemérides precisas transmiten directamente las coordenadas tridimensionales del satélite, en un momento concreto, en un sistema de referencia centrado y fijo a la Tierra (ECEF). Para determinar estas coordenadas se emplean los datos de pseudodistancias y fase que son registradas por estaciones permanentes de observación terrestre distribuidas por todo el planeta.

La adquisición de las efemérides precisas es gratuita y se distribuye por Internet siendo las más utilizadas las distribuidas por el *International GNSS Service* (ICS) [16] que son una combinación de las calculadas por distintas agencias. Otros centros de distribución son, por ejemplo, el *Center of Orbit Determination for Europe* (CODE) [17] o el *National Geodetic Survey* (NGS) [18].



➤ ***International GNSS Service:***

En 1991 se creó este servicio que coordina una red mundial de estaciones de observación GNSS que actualmente se compone de 270 estaciones.

Sus objetivos principales son:

- *Definir, implementar y mejorar el Marco de Referencia Terrestre Internacional (ITRF),*
- *Estudiar la geodinámica terrestre,*
- *Determinar las variaciones de la rotación terrestre y las coordenadas del polo, y*
- *Calcular y distribuir las efemérides precisas de las constelaciones GPS y GLONASS.*

Se distribuyen cuatro tipos de efemérides precisas:

- ultra rápidas, previstas: emitidas en tiempo real y actualizadas cada 6 h,
- ultra rápidas, observadas: publicadas entre 3 y 6 h después del tiempo de estudio y actualizadas cada 6 horas,
- rápidas: publicadas entre 17 y 41 horas después de los pasos del satélite, todos los días a las 17 UTC, y
- finales: publicadas entre 12 y 18 días después de la observación.

GPS Satellite Ephemerides / Satellite & Station Clocks

| Type | Accuracy | Latency | Updates | Sample Interval |
|------------------------------|--|---------------|------------------------------|--------------------------|
| Broadcast | orbits ~100 cm | real time | -- | daily |
| | Sat. clocks ~5 ns RMS ~2.5 ns SDev | | | |
| Ultra-Rapid (predicted half) | orbits ~5 cm | real time | at 03, 09, 15, 21 UTC 15 min | |
| | Sat. clocks ~3 ns RMS ~1.5 ns SDev | | | |
| Ultra-Rapid (observed half) | orbits ~3 cm | 3 - 9 hours | at 03, 09, 15, 21 UTC 15 min | |
| | Sat. clocks ~150 ps RMS ~50 ps SDev | | | |
| Rapid | orbits ~2.5 cm | 17 - 41 hours | at 17 UTC daily | 15 min |
| | Sat. & Stn. clocks ~75 ps RMS ~25 ps SDev | | | 5 min |
| Final | orbits ~2.5 cm | 12 - 18 days | every Thursday | 15 min |
| | Sat. & Stn. clocks ~75 ps RMS ~20 ps SDev | | | Sat.: 30s Stn.: 5 min |

Note 1: Orbit accuracies are 1D mean RMS values over the three XYZ geocentric components. IGS accuracy limits, except for predicted orbits, are based on comparisons with independent laser ranging results and discontinuities between consecutive days. The precision is better.

Note 2: The accuracy (neglecting any contributions from internal instrumental delays, which must be calibrated separately) of all clocks is expressed relative to the IGS timescale, which is linearly aligned to GPS time in one-day segments. The standard deviation (SDev) values are computed by removing a separate bias for each satellite and station clock, whereas this is not done for the RMS values.

GLONASS Satellite Ephemerides

| Type | Accuracy | Latency | Updates | Sample Interval |
|-------|----------|--------------|----------------|-----------------|
| Final | ~3 cm | 12 - 18 days | every Thursday | 15 min |

Figura 17. Precisiones de las efemérides precisas (GPS y GLONASS)[16].

Por otro lado, las efemérides precisas se distribuyen en un formato estándar internacional de intercambio mediante ficheros SP3 (*Standard Product 3*) que contienen datos sobre la posición de los satélites así como información de sus relojes, para satélites de las constelaciones de GPS, GLONASS y Galileo. Los archivos contienen datos diarios en formato ASCII con una cabecera informativa y un listado con las coordenadas (en Km) de cada satélite y el estado del reloj (en microsegundos) cada 1 o 15 minutos empezando por las 00h0m00s del día a que se refieren. A continuación de muestra un ejemplo de archivo de efemérides rápidas publicado por el IGS correspondiente a las 00h00m00s del 1 de enero de 2017:

```
#cP2017 1 1 0 0 0.00000000      96 ORBIT IGB08 HLM IGS
## 1930      0.00000000 900.00000000 57754 0.000000000000
+ 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 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 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
++    2 2 3 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 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
%G cc GPS ccc cccc cccc cccc ccccc ccccc ccccc ccccc
%c cc ccc ccc cccc cccc cccc ccccc ccccc ccccc
%F 1.250000 1.02500000 0.0000000000 0.00000000000000
%F 0.000000 0.00000000 0.0000000000 0.00000000000000
%i 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
%i 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
/* RAPID ORBIT COMBINATION FROM WEIGHTED AVERAGE OF:
/* cod emr esa gfz jpl ngs sio
/* REFERENCED TO IGS TIME (IGST) AND TO WEIGHTED MEAN POLE:
/* PCV:IGS08_1928 OL/AL:FES2004 NONE Y ORB:CMB CLK:CMB
* 2017 1 1 0 0 0.00000000
PG01 -15390.619802 -13649.826456 -17041.356459 45.411898 7 7 7 126
PG02 -8489.055244 14573.677287 20921.030182 501.228535 10 10 4 126
PG03 -12657.847522 -23117.504657 3325.258426 -104.628094 8 7 10 103
(...)
```

Para una descripción más precisa del contenido de estos archivos, se puede consultar:

<ftp://igscb.jpl.nasa.gov/igscb/data/format/sp3.txt>

<ftp://igscb.jpl.nasa.gov/igscb/data/format/sp3c.txt>

1.4.4. LOS CONJUNTOS TLE (*Two-Line Elements set*)

Uno de los formatos más comúnmente utilizados para adquirir parámetros orbitales de cualquier vehículo espacial es el conjunto de elementos de 2 líneas o TLE generados por NORAD. Los TLE fueron desarrollados específicamente para su uso con el modelo orbital SGP4 / SDP4 (su uso con cualquier otro

propagador puede invalidar algunos de los supuestos incorporados).

Está formado por conjuntos de tres líneas de datos para cada satélite. La primera, identifica al vehículo espacial y las otras dos, de 69 caracteres cada una, alojan los elementos orbitales clásicos, junto con algunos parámetros adicionales para fines de identificación y para su uso en el modelado de perturbaciones. Los únicos caracteres válidos son los números 0-9, las mayúsculas A-Z, el punto, el espacio y los signos más y menos. Su esquema simple y lineal es el resultado de la herencia de las tarjetas de datos perforadas utilizadas en las primeras computadoras.

Los datos se generan mediante observaciones directas a los satélites realizadas por estaciones terrestres. NORAD hace un seguimiento de todos los objetos detectables en órbita terrestre, creando el TLE correspondiente para cada uno de ellos. Estos ficheros se pueden obtener, del web CelesTrak [19].

A continuación se muestra un ejemplo de una parte del contenido de un TLE correspondiente a los satélites de la constelación GPS, con los parámetros de cada uno de ellos referidos a las 19h 59m 44s del 13 de enero de 2017:

```

GPS BIIR-2 (PRN 13)
1 24876C 97035A 17013.83090278 .00000000 00000-0 00000-0 0 137
2 24876 55.5900 233.1880 0038581 107.3003 126.8820 2.00561932 13
GPS BIIR-3 (PRN 11)
1 25933C 99055A 17013.83090278 -.00000000 00000-0 00000-0 0 137
2 25933 51.5272 84.4053 0164342 92.4466 70.5881 2.00560285 19
GPS BIIR-4 (PRN 20)
1 26360C 00025A 17013.83090278 -.00000000 00000-0 00000-0 0 131
2 26360 53.0944 161.3299 0044804 87.7117 175.3796 2.00569315 16
GPS BIIR-5 (PRN 28)
1 26407C 00040A 17013.83090278 .00000000 00000-0 00000-0 0 139
2 26407 56.6720 351.1654 0203040 269.2348 275.8949 2.00568140 16
(...)
```

Dónde

| | |
|---------|---|
| Línea 0 | AAAAAAAAAAAAAAAAAAAAAA |
| Línea 1 | NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNN-N +NNNN-N N NNNNN |
| Línea 2 | NNNN NNN.NNN NNN.NNN NNNNNNN NNN.NNN NNN.NNN NN.NNNNNNNNNNNNN |

Tabla 6 Descripción de los parámetros de los ficheros TLE [20].

| Línea 0 | |
|---------|--|
| Columna | Descripción |
| 1-... | Nombre del satélite (nomenclatura del catálogo de satélites de NORAD). |

| Línea 1 | |
|---------|--|
| Columna | Descripción |
| 1 | Número de línea. |
| 3-7 | Número del satélite (según catálogo NORAD). |
| 8 | Clasificación (U=desclasificado; C=corregido). |
| 10-11 | Los últimos dos dígitos del año de lanzamiento. |
| 12-14 | Número de lanzamiento del año. |
| 15-17 | Pieza del lanzamiento |
| 19-20 | Época (los últimos dos dígitos del año). |
| 21-32 | Época (día del año y porción fraccional del día). |
| 34-43 | Primera derivada del movimiento medio. |
| 45-52 | Segunda derivada del movimiento medio (parte decimal). |
| 54-61 | Término de rozamiento BSTAR (parte decimal). |
| 63 | Tipo de efemérides. |
| 65-68 | Número de elemento. |
| 69 | Checksum (Módulo 10) |

| Línea 2 | |
|---------|---|
| Columna | Descripción |
| 1 | Número de línea de los datos. |
| 3-7 | Número de satélite. |
| 9-16 | Inclinación (grados) |
| 18-25 | Ascensión Recta del Nodo Ascendente (grados) |
| 27-33 | Excentricidad (parte decimal). |
| 35-42 | Argumento del Perigeo (grados). |
| 44-51 | Anomalía Media (grados). |
| 53-63 | Movimiento Medio (revoluciones por día). |
| 64-68 | Número de revolución a la época (revoluciones). |
| 69 | Checksum (Módulo 10) |

| Nombre del satélite | Identificador internacional | Año de la época y fracción del día Juliano | 1 ^a derivada del movimiento medio | 2 ^a derivada del movimiento medio | Término de rozamiento | Número de elemento y Checksum |
|---------------------|---|---|---|---|---|---|
| NOAA 6 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 | 1 11416U 84123 A 86 50.28438588 0.00000140 00000-0 67960-4 0 5293 |
| Número de satélite | Inclinación | Ascensión Recta del Nodo Ascendente | Excentricidad | Argumento del Perigeo | Anomalía Media | Movimiento Medio |

Número de revolución a la época y Checksum

Figura 18. Datos de los ficheros TLE [20].

1.5. FACTORES QUE LIMITAN LA PRECISIÓN

Conocidos los fundamentos teóricos que nos ayudarán a seleccionar e integrar un modelo de propagación a nuestra programa, vamos ahora a estudiar otro concepto que también queremos implementar en la aplicación. Además de mostrar las posiciones de los satélites, también se pretende proporcionar al usuario un indicador de calidad, ligado a estas posiciones y a la propia de su receptor. Un indicador que determine qué precisión podemos conseguir con el escenario en que nos encontramos.

Vamos a ver primero, los factores que limitan esta precisión en las observaciones, para después determinar qué valores son los que nos informarán de esta calidad. El término habitual para designar la precisión de una medida GNSS es el UERE (del inglés, *User Equivalent Range Error*) que representa el error resultante de la combinación de múltiples factores. Podríamos resumir estos factores que limitan esta precisión en la navegación como:

- Errores de efemérides, debidos a la posición de los satélites en sus órbitas,
- Errores en los relojes (del satélite y del receptor),
- Errores por variación del centro de fase de la antena receptora,
- Errores debidos a la propagación de la señal a través de la ionosfera y la troposfera,
- Errores causados por el multirayecto de las señales,
- Errores causados por señales interferentes,
- Errores debidos al procesado de datos en el mismo receptor, y
- Errores debidos a la distribución geométrica de los satélites en el cielo.

El primer error se ha estudiado ampliamente en capítulos anteriores, explicando qué factores motivan la indeterminación en la posición de los satélites, como se pueden modelar y cuantificar y, por consiguiente, cómo se intentan corregir.

El error acumulado de los relojes de cada satélite (en el sistema GPS, cuatro relojes atómicos), suele ser menor de 1 nanosegundo cada 3 horas, que transmitido hasta el receptor puede dar un error en la medida de la distancia de unos 30 centímetros [21]. Las estaciones de control en Tierra comparan dichos relojes con los de referencia y determina su desfase y consiguiente corrección que, a su vez, es enviada a cada satélite para que se transmita con la señal de navegación. Como hemos visto, estas correcciones se utilizan también en los modelos de propagación.

Los errores debidos al procesado de datos en el receptor o por variación del centro de fase de la antena receptora, son variables que no tiene ningún sentido analizar en este proyecto, pues no se pretende controlar, para cada caso, qué modelo de receptor se utilizará, con qué algoritmos se harán los cálculos o qué características particulares tendrá su antena.

Por otro lado, el error de reloj del receptor, o bias del reloj del receptor, es una de las incógnitas de las ecuaciones de pseudodistancias. Su valor se obtiene en el cálculo y por eso muchas veces no se considera un error.

Los restantes factores, podríamos considerar que son factores propios de cada una de las mediciones y dependen de la posición y el entorno dónde se encuentra el receptor. Una de las motivaciones para desarrollar la aplicación GNSSPlanning sobre un entorno SIG, viene motivada por el hecho de que, a través de un análisis espacial del entorno del receptor, se pueden detectar alguno los causantes de este tipo de errores y estudiar cómo evitarlos.

1.5.1. ERRORES CAUSADOS POR LA ATMÓSFERA

La distancia entre el satélite y el receptor se mide a través del tiempo que emplea la señal transmitida desde que sale del satélite hasta su llegada a la antena receptora, multiplicando después este tiempo por la velocidad de la luz en el vacío. Pero la atmósfera es un medio distinto del vacío, con propiedades diferentes en función de la altura que estemos considerando. Dichas propiedades hacen que la velocidad de la luz sea diferente en las distintas zonas de la atmósfera.

La parte más elevada de la atmósfera es la ionosfera, entre los 50 y 1.000 km de altura respecto a la superficie de la Tierra. Parte de las moléculas de gas que contiene esta capa son ionizadas por la radiación ultravioleta procedente del Sol. La ionización separa, por una parte los iones con carga positiva y por otra parte los electrones libres.

Las ondas electromagnéticas, como las señales transmitidas por los satélites de navegación, al pasar por un campo ionizado presentan una dispersión no lineal que produce una disminución de la velocidad de la luz y su consiguiente efecto en la medida de la distancia.

La ionosfera no es una capa homogénea, ni en su espesor ni en su densidad por lo que el error que produce no es constante y está en función del momento de la observación y de la posición del satélite en el cielo del observador. En los períodos de máxima insolación (mediodía) el error puede llegar a los 150m, y en los períodos de baja insolación (medianoche) este error se reduce hasta los 5 m.

Por otra parte hay que considerar la elevación del satélite sobre el horizonte lo que hará que la señal recorra más o menos longitud a través de la ionosfera. Si el satélite está en el cenit, la distancia recorrida será menor y la influencia de la ionosfera también. Si el satélite está próximo al horizonte, la distancia recorrida será mayor y la influencia de la ionosfera, también.

Este efecto de la ionosfera depende de la frecuencia de la señal. El uso de dos frecuencias, L1 y L2, permite comparar las medidas realizadas por cada una y estimar el error producido por la ionosfera y minimiza su efecto.

Se ha comprobado (Corbasí, 1998) [6] que la relación entre la pseudodistancia medida y la real viene dada por:

$$R = R_m - \frac{A}{f^2} \quad (50)$$

Donde, en las medidas de código, R es la distancia real, R_m la pseudodistancia medida f la frecuencia de la señal y A el parámetro dependiente de las condiciones particulares causadas por la atmósfera.

Si se mide con dos frecuencias podemos igualar estas condiciones particulares de la ecuación y deducir la distancia real.

$$R_1 = R_{1m} - \frac{A}{f_1^2} \quad R_2 = R_{2m} - \frac{A}{f_2^2} \quad (51) (52)$$

Con lo que se obtiene

$$R_1 = R_2 = \frac{R_{1m}f_1^2 - R_{2m}f_2^2}{f_1^2 - f_2^2} \quad (53)$$

El paso de las ondas electromagnéticas por la capa más baja de la atmósfera, la troposfera, también produce una refracción de la señal, pero esta no depende de la frecuencia de la onda que la atraviesa (siempre que dicha frecuencia sea inferior a 30 GHz) y está influenciada por las condiciones atmosféricas que tienen lugar en esta capa. La presión atmosférica, el contenido de vapor de agua o la temperatura, entre otros, se recogen en modelos matemáticos que, actualmente, reducen hasta un 95% sus efectos en la medida de la distancia (hasta unos 3-4 cm).

1.5.2. ERRORES CAUSADOS POR EL MULTITRAYECTO DE LA SEÑAL.

Este fenómeno ocurre cuando una misma señal llega al receptor por dos caminos distintos. En las mediciones de la distancia a los satélites se supone que la señal llega al receptor directamente, pero en realidad, además, esta también llega de forma indirecta reflejada por objetos cercanos o por el paisaje

de su entorno. Estas otras llegadas de la señal se suman a la señal directa para disminuirla (suma en oposición) o para aumentarla (suma en fase) lo que conlleva una incertidumbre en la determinación del momento exacto de la llegada de la señal.

Este error es difícil de evitar sobretodo en estaciones móviles. En estaciones fijas se puede minimizar mediante el uso de antenas receptoras específicamente diseñadas para restringir las señales de entrada sólo a las procedentes del hemisferio superior. Evitando la llegada de la señal reflejada en superficies por debajo de la antena, se impide la llegada de toda señal reflejada en el suelo.

Para las llegadas de la señal procedentes del rebote en objetos por encima de la antena, se utiliza la técnica de procesado de la señal que funciona correctamente para eliminar multirayos lejanos (diferencia de más de 10 m entre el recorrido de la señal directa y la reflejada) pero que puede empeorar la relación señal-ruido de la señal directa en multirayos cercanos (diferencias de pocos metros) (Olmedillas, 2012) [22].

La aplicación GNSSPlaning, si bien no pretende calcular el efecto multirayos sobre las señales que el usuario recibirá en su receptor, si persigue poner a su disposición una herramienta para mostrar los elementos que existen a su alrededor y que pueden ser causantes de este error. Si el usuario escoge la inclusión de un modelo digital de elevaciones para la máscara, podrá ver qué elementos verticales de obstrucción tendrá alrededor susceptibles de contribuir de forma directa al multirayos de la señal. Se deja al propio usuario evaluar dicho efecto.

1.5.3. ERRORES CAUSADOS POR SEÑALES INTERFERENTES.

La presencia de señales externas al sistema GNSS con frecuencias próximas a las L1 y L2 pueden alterar (o incluso bloquear) su funcionamiento. Las señales que afectan directamente a las del sistema GNSS se denominan interferencias dentro de la banda (*in-band interference*) y las que afectan a bandas cercanas se denominan interferencias fuera de banda (*out-band interference*). Las segundas son fácilmente filtradas por la mayoría de receptores, pero las interferencias dentro de banda son mucho más difíciles de filtrar.

1.5.4. ERRORES CAUSADOS POR LA DISTRIBUCIÓN GEOMÉTRICA DE LOS SATÉLITES EN EL CIELO.

La determinación de la posición del receptor se basa, a grosso modo, en identificar la intersección exacta entre todas las esferas teóricas, definida cada una de ellas por un centro ubicado a la posición del satélite y un radio igual a la pseudodistancia transmitida desde este.

Como las incógnitas que este sistema tridimensional genera son 4 (X, Y, Z y tiempo), como mínimo necesitamos conocer la posición de un número igual de satélites. A su vez, cuando mayor sea el número de satélites que entren en el sistema, mayor será la redundancia de datos y mejor será la precisión esperada. No cabe decir, que es imprescindible conocer el estado de salud de cada satélite para incluirlo o no y evitar añadir datos erróneos al sistema.

Otros factores importantes además de la cantidad y la calidad, son los relativos a la forma en que se distribuyen los satélites en la bóveda celeste en el momento del cálculo de la posición. Estos factores se denominan DOP (del inglés, *Dilution of Precision*) y permiten transformar el error en la medida de pseudodistancia en el error correspondiente a las coordenadas del receptor.

Según sea su distribución en el cielo, la indeterminación de la intersección de las esferas será mayor o menor. Por ejemplo, con el número mínimo de satélites necesarios, según sea la distribución de los cuatro satélites, se podrá conseguir diferentes precisiones en la determinación de la posición del receptor. La mejor distribución posible sería disponer de un satélite en el cenit y los otros tres ligeramente por encima del horizonte separados entre ellos por una distancia angular de 120 grados. En cambio, si estos cuatro satélites estuvieran distribuidos muy próximos entre sí, de la intersección de sus esferas resultaría un área de incertidumbre muy grande y una mala precisión de la posición calculada.

El valor de la DOP indica la calidad del conjunto de satélites escogidos para la medición y cuando menor sea este valor, menor será el error cometido en el cálculo.

Como sabemos, en el momento de las observaciones la posición de los satélites no es fija en la bóveda y por tanto el valor de la DOP para un mismo emplazamiento varía constantemente. Por lo que se tendrá que determinar dicho valor para cada instante.

Para determinar la precisión de los cálculos, se definen diferentes tipos de DOP en función de las coordenadas escogidas para su evaluación.

- PDOP: Dilución de la precisión del posicionamiento 3D.
- HDOP: dilución de la precisión del posicionamiento 2D.
- VDOP: dilución de la precisión de la altura.
- TDOP: dilución de la precisión en el tiempo.
- GDOP: dilución del posicionamiento 3D y el tiempo, también conocido como dilución geométrica.

Los valores de DOP se obtienen a partir de las ecuaciones básicas de navegación:

$$R_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2} + c\Delta t_i \quad (54)$$

Dónde x_i, y_i, z_i corresponden a las coordenadas de cada satélite según las efemérides recibidas, x_r, y_r, z_r a las coordenadas aproximadas del receptor y $c\Delta t_i$ el desfase en tiempo del reloj del receptor y del satélite.

Conocidas las coordenadas aproximadas del receptor, podemos realizar un desarrollo en serie de Taylor para linealizar la ecuación, con lo que la distancia aproximada será:

$$R_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2} \quad (55)$$

Con el conjunto de todos los satélites se obtiene la matriz A , que para el caso de 4 satélites se obtendría la siguiente matriz:

$$A = \begin{bmatrix} -\frac{(x_1 - x_r)}{R_1} & -\frac{(y_1 - y_r)}{R_1} & -\frac{(z_1 - z_r)}{R_1} & 1 \\ -\frac{(x_2 - x_r)}{R_2} & -\frac{(y_2 - y_r)}{R_2} & -\frac{(z_2 - z_r)}{R_2} & 1 \\ -\frac{(x_3 - x_r)}{R_3} & -\frac{(y_3 - y_r)}{R_3} & -\frac{(z_3 - z_r)}{R_3} & 1 \\ -\frac{(x_4 - x_r)}{R_4} & -\frac{(y_4 - y_r)}{R_4} & -\frac{(z_4 - z_r)}{R_4} & 1 \end{bmatrix} \quad (56)$$

Los tres componentes de cada una de las filas de la matriz A se pueden considerar como los tres componentes de un vector unitario (la suma de dichas componentes es 1) cuyo sentido es desde el satélite hacia el receptor. Esta matriz tendrá tantas filas como satélites se estén considerando en el cálculo.

Si consideramos

$$Q = (A^T A)^{-1} \quad (57)$$

Obtenemos la matriz

$$Q = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & \sigma_{xt} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} & \sigma_{yt} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 & \sigma_{zt} \\ \sigma_{xt} & \sigma_{zt} & \sigma_{zt} & \sigma_{ct}^2 \end{bmatrix} \quad (58)$$

Cuya traza es la dilución de la precisión geométrica y dónde los elementos de la diagonal son las varianzas de los estimadores de posición sobre cada eje y del estado del reloj. En el caso de cuatro satélites, la mejor configuración será aquella cuya traza sea mínima, o dicho de otra forma, la que obtenga el tetraedro máximo, formado por el receptor y los cuatro satélites situados en los vértices del tetraedro.

Con las varianzas de los estimadores de la posición podemos calcular los diferentes factores de DOP:

$$PDOP = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \quad (57)$$

$$TDOP = \sqrt{\sigma_{ct}^2} \quad (58)$$

$$GDOP = \sqrt{PDOP^2 + TDOP^2} = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \sigma_{ct}^2} = \sqrt{\text{traza } Q} \quad (59)$$

$$HDOP = \sqrt{\sigma_x^2 + \sigma_y^2} \quad (60)$$

$$VDOP = \sqrt{\sigma_z^2} \quad (61)$$

Hoy en día, con la mejora de las constelaciones, los receptores tienen más de 4 satélites sobre su horizonte y, según sea la capacidad del receptor, pueden hacer medidas a todos los satélites visibles. En general, cuántos más satélites se emplean en la medida menores factores DOP se obtienen. Aunque esta afirmación se refleja en la mayoría de los trabajos, no siempre es así. Puede darse el caso que la distribución geométrica de los satélites sea más influyente que la cantidad utilizada y una determinada combinación de, por ejemplo, 6 satélites proporcione unos factores DOP menores que otra determinada combinación de 7 satélites. Un número menor de satélites que estén distribuidos de forma idónea en el cielo pueden proporcionar una posición más precisa que un grupo mayor de satélites distribuidos en forma de enjambre (agrupados en una pequeña zona de la bóveda).

Así pues, para determinar qué satélites serán los mejores candidatos para ser utilizados en los cálculos, necesitamos calcular la DOP para cada una de las posibles combinaciones de grupos de satélites posibles. Esto nos obliga a hacer combinaciones de grupos de n satélites, empezando por $n = 4$ satélites, continuando con $n+1, n+2\dots$ hasta llegar a n igual al número máximo de satélites. Para cada grupo se calcula la DOP y se escoge la que presente el menor valor.

Por ejemplo, un caso común es tener en el momento de la observación un total de 9 satélites de la constelación GPS sobre la bóveda celeste. Las combinaciones posibles para determinar la menor DOP sería:

- Calcular la DOP para:
 - combinaciones de grupos de 4 satélites:

$$C_m^n = \frac{n!}{m!(n-m)!} \quad C_4^9 = \frac{9!}{4!(9-4)!} = 126 \text{ combinaciones} \quad (62)$$

- combinaciones de grupos de 5 satélites = 126 combinaciones
- combinaciones de grupos de 6 satélites = 84 combinaciones
- combinaciones de grupos de 7 satélites = 36 combinaciones
- combinaciones de grupos de 8 satélites = 9 combinaciones

- Calcular la DOP con los 9 satélites.

Al final tendríamos un total de 382 combinaciones posibles y escogeríamos la que presentara menor DOP para calcular nuestra posición.

2. SELECCIÓN DE UN PROPAGADOR PARA GNSSplanning

En esta sección se evaluarán algunos de los modelos de propagación expuestos en el capítulo anterior con el fin de seleccionar el modelo óptimo para nuestra aplicación. Primero, se hará un estudio para determinar qué características queremos que tenga nuestro modelo, analizando la accesibilidad, la uniformidad y simplicidad de los formatos de los datos de partida o qué perturbaciones se deben incluir según la tipología de las órbitas de los satélites.

Sabiendo cómo debe ser nuestro modelo, se procederá a evaluar dos modelos de propagación que cumplan dichos requisitos. Se expondrá un caso práctico con varios satélites para probar la eficiencia y precisión de cada modelo. Finalmente, según sean los resultados obtenidos, se argumentará qué propagador será el escogido para incluirse en el complemento GNSSplanning.

2.1. ESTUDIO DE CARACTERÍSTICAS

2.1.1. ¿QUÉ PERTURBACIONES DEBE CORREGIR EL PROPAGADOR?

Las características de las órbitas de las constelaciones GNSS, determinan qué tipo de perturbaciones necesitan ser corregidas por el modelo. Los satélites de las constelaciones GNSS se caracterizan por tener órbitas con características similares, con inclinaciones de entre 55° y 65° y excentricidades muy cercanas a cero. Los satélites circulan en alturas entre 20000 y 24000 km sobre la superficie terrestre con períodos aproximados de entre 12 y 14 horas.



| | GPS | GLONASS | GALILEO | BEIDOU |
|------------------------------------|-----------------------------|-------------------------------|-----------------------------|-----------------------------|
| Sistema de referencia ¹ | WGS84(G1762) | PZ-90.11 | GTRF 09 v01 | CGCS2000 |
| μ (m^3/s^2) | $3.986004418 \cdot 10^{14}$ | $3.986\ 004\ 4 \cdot 10^{14}$ | $3.986004418 \cdot 10^{14}$ | $3.986004418 \cdot 10^{14}$ |
| Ω_{Tierra} (rad/ s^2) | $7.292115 \cdot 10^{-5}$ | $7.292115 \cdot 10^{-5}$ | $7.292115 \cdot 10^{-5}$ | $7.292115 \cdot 10^{-5}$ |
| Altura media (km) | 20 180 | 19 130 | 23 222 | 21 150 |
| a (m) | 6 378 137 | 6 378 136 | 6 378 137 | 6 378 137 |
| i ° | 55 | 64.8 | 56 | 55 |
| P | 11h 58min | 11h 16min | 14h 5min | 12h 38min |

Tabla 7. Tabla de los parámetros de las órbitas para las constelaciones GNSS[23][24][25][26][27][28].

¹ El marco de referencia WGS84 coincide con el marco de referencia ITRF2008 calculado por el IERS y el GTRF coincide con el marco de referencia ITRF2005. Las diferencias entre estos marcos y el PZ-90.11 y el CGCS200 son despreciables para este proyecto.

Así pues, teniendo en cuenta la altura de las órbitas, se considera que el efecto del rozamiento de la atmósfera en el movimiento de este tipo de satélites, se puede despreciar por encontrarse muy alejadas de su influencia. Por el mismo criterio, sí se deberán considerar las perturbaciones debidas a la influencia del efecto gravitacional del Sol y de la Luna así como el efecto de la presión de la radiación solar.

Los propagadores propios de los receptores GNSS y el propagador definido por la extensión SDP4 de los conjuntos SGP [12] (a partir de ahora, SDP4) cumplen con estos requisitos. Ambos propagadores incorporan en sus modelos de cálculo las correcciones de las perturbaciones debidas a la influencia del efecto gravitacional del Sol y de la Luna, así como las causadas por la presión de la radiación solar. En este sentido, ambos son válidos para ser utilizado con satélites GNSS.

2.1.2. UNIFORMIDAD Y ACCESIBILIDAD DE LOS DATOS DE PARTIDA

Como hemos visto, no todos los propagadores parten de los mismos datos para iniciar sus cálculos. Vamos a analizar cada uno de ellos para identificar sus características y evaluar si pueden utilizarse en nuestro complemento.

Está previsto que GNSSplanning se utilice, principalmente, para trabajos de planificación en oficina. Por lo que se contempla un escenario de trabajo donde no se recibirán los mensajes de navegación directamente de los satélites. Por lo tanto las efemérides de partida tendrán que obtenerse de lugares web que lo distribuyan o del mismo usuario.

- Almanaques: en los almanaques se distribuyen los parámetros orbitales de los satélites de toda una constelación, y por lo tanto, se podrían trabajar de forma conjunta con todos ellos.

Pero sus datos no aportan ninguna información sobre las perturbaciones, por lo que sólo se pueden utilizar con un propagador basado en un modelo de dos-cuerpos.

- Efemérides transmitidas: en sus datos sí que se incluyen las perturbaciones junto a los parámetros orbitales. En los mensajes de navegación recibidos por un receptor (por ejemplo, de una estación permanente) se incluyen las efemérides transmitidas sólo de aquellos satélites de los que se ha recibido señal. De los demás, no se obtienen datos. Por lo que para conseguir una cobertura de datos total (de todos los satélites y de todas las constelaciones) en una época concreta, no nos bastará con las efemérides recibidas por una sola estación. Tendremos que obtener datos de varias estaciones.

Por ejemplo, podríamos obtener un fichero RINEX del *Scripps Orbit and Permanent Array Center* (SOPAC) [15] donde se encuentran todos los mensajes de navegación recibidos por sus estaciones.

Los mensajes que se distribuyen contienen información de más de un satélite, pero no de todos los de la constelación. Por lo que tendríamos que trabajar con varios ficheros de datos a la vez.

- Efemérides precisas: estas efemérides sí incluyen información de todos los satélites de una misma constelación. Para trabajar con datos de todos los satélites, sólo necesitaríamos obtener un fichero para cada constelación GNSS (GPS, GLONASS, GALILEO, BEIDOU), teniendo en cuenta que su nomenclatura varía según a la época a la que se refieren.

Las efemérides precisas se distribuyen en un formato SP3 que contienen datos sobre la posición en coordenadas cartesianas XYZ (km) en un sistema de referencia ECEF. Un propagador que partiese de estas efemérides se tendría que programar un módulo adicional para una primera transformación de coordenadas cartesianas a parámetros orbitales.

- TLE: en cuanto a uniformidad y accesibilidad, los archivos TLE se presentan como la mejor alternativa. Su formato es igual para todas las constelaciones, ya que NORAD se encarga de recopilar todos los datos y los publica en un formato homogéneo. Esta publicación se actualiza de forma periódica en un enlace inmutable, por lo que siempre que se adquieran los datos desde ese enlace, nos aseguraremos de que correspondan a las últimas actualizaciones. Además, existe un único fichero para cada constelación que incluye los datos de todos sus satélites operativos.

Así pues, los datos de partida que presentan mejor accesibilidad, uniformidad y simplicidad de formato son las efemérides radiodifundidas y los TLE.

2.1.3. ADAPTACIÓN AL LENGUAJE DE PROGRAMACIÓN

El complemento GNSSplanning se programará en lenguaje Python, por lo que el modelo de propagación tendrá que adaptarse a este lenguaje. Se presentan varias opciones: escribir el código des de cero para crear el modelo, aprovechar código existente o crear interpretadores para enlazar nuestro código Python con aplicaciones desarrolladas con otros lenguajes.

Para los propagadores de dos cuerpos, los J2 y J4, o los utilizados en los receptores GNSS se debería programar con código Python el modelo entero (no se han encontrado librerías o módulos que se adapten a nuestro complemento) o crear enlaces a programas (por ejemplo MAPLAB [29]), con aplicaciones que ejecuten el modelo. Esta última opción se descarta al querer evitar que el usuario tenga que instalarse software adicional.

Para el propagador SDP4, existen librerías de código Python que se pueden adaptar para integrarlas a nuestro complemento. Aun así se requeriría escribir código adicional para que esta adaptación sea eficaz. La principal ventaja reside en que el código de estas librerías ha sido ya testeado por varios

usuarios.

Vemos que ninguna de las opciones es sencilla, pero si se quiere usar el propagador SDP4 la mejor alternativa es la reutilización del código existente y si se pretende utilizar cualquiera de los otros, lo mejor sería escribir todo el código.

Una de las funciones que se programará será la lectura de los archivos de efemérides procedentes de enlaces web, para incorporar sus datos como variables del programa. Este módulo tendrá que ser lo más computacionalmente eficiente posible, por lo que se premiará al propagador que parta de efemérides sencillas de implementar con código Python. Los datos de los TLE y las efemérides transmitidas son fácilmente interpretables con métodos de lectura de Python.

Así pues, los propagadores que presentan más posibilidades de ser implementados con éxito en nuestro complemento, una vez más, son los propagadores propios de los receptores GNSS y los SDP4.

2.2. PROPUESTA DE LOS PROPAGADORES CANDIDATOS

Una vez definidas las características que debe cumplir nuestro propagador y evaluadas las características de los distintos propagadores, se perfilan dos opciones como las óptimas.

- Una primera opción sería utilizar un modelo de propagación de órbitas similar al utilizado en los receptores GNSS partiendo de efemérides radiodifundidas recibidas por varias de las estaciones permanentes.
- Una segunda opción sería utilizar un propagador basado en la extensión SDP4 de los conjuntos SGP. En este caso, los datos de partida serían los archivos TLE que transmiten los elementos orbitales de todas las constelaciones GNSS.

Para evaluar la eficacia de estos dos posibles propagadores se han creado dos script² de Python:

- **PROP-Receptor:** basado en los modelos utilizados en los receptores GNSS. Se ha programado íntegramente un script con el modelo de ecuaciones descrito al apartado 1.3.2.3.
- **PROP-SDP4:** basado en los modelos SGP. Se ha utilizado el script *python-sgp4* (Copyright © 2012–2016 Brandon Rhodes, con licencia MIT) [30] basado en las ecuaciones propuestas en [12]. Se ha utilizado la parte correspondiente a la propagación de órbitas lejanas, correspondiente al modelo SDP4 y se han adaptado funciones complementarias del código del script *python-skyfield* (Copyright © 2012–2016 Brandon Rhodes, con licencia MIT) [31] para transformar las

² Programa que permite conectar distintos componentes para crear otro nuevo o similar.

coordenadas inerciales (TEME), que retorna *python-sgp4*, a coordenadas fijas (ITRF)³. También ha sido necesario programar nuevas funciones para adaptar el código de esos scripts al de nuestro propagador y para implementar otros procesos y funciones, como por ejemplo, la importación de los TLE.

2.3. EVALUACIÓN DE LOS PROPAGADORES CANDIDATOS

Para evaluar cada propagador vamos a predecir, para distintas épocas, la posición de cuatro satélites de la constelación GPS (G12, G15, G19 y G24).

Tabla 8. Datos de los satélites utilizados en la evaluación de los propagadores.

| NORAD ID | Constelación | Subconstelación | Nombre | SVN | PRN | Sobrenombre |
|----------|--------------|-----------------|---------|-----|-----|-------------|
| 29601 | GPS | Block-IIR-M | BIIRM-3 | 058 | 12 | G12 |
| 32260 | GPS | Block-IIR-M | BIIRM-4 | 055 | 15 | G15 |
| 28190 | GPS | Block-IIR | BIIR-11 | 059 | 19 | G19 |
| 38833 | GPS | Block-IIF | BIIF-3 | 065 | 24 | G24 |

NOTA: Para simplificar la obtención de datos de partida para el PROP-Receptor, se han seleccionado los satélites G12, G15, G19 y G24 por ser satélites que mandaron su mensaje de navegación a la estación BELLO0ESP a las 20h 00m 00s del día 13 de enero de 2017, pudiendo utilizar así un solo fichero de navegación y un solo fichero de observables.

Para cada propagador se procederá de la siguiente manera:

- Se partirá de las efemérides transmitidas y de los TLE para el PROP-Receptor y para el PROP-SDP4, respectivamente.
- Se propagará la órbita de los satélites, hacia adelante en el tiempo, hasta las 20h00m00s de los días 13, 14 (correspondientes a los días 5 y 6 de la semana GPS 1931, respectivamente), de los días 15, 16, 17, 18, 19, 20 y 21 (correspondientes a los días 0 a 6 de la semana GPS 1932, respectivamente) y de los días 22, 23, 24 y 25 (correspondientes a los días 0 a 3 de la semana GPS 1933, respectivamente).
- Se calculará, para cada caso, su posición XYZ (geocéntricas).
- Finalmente, cada una de estas posiciones se comparará con la posición real del satélite en dichas épocas, que se obtendrán de las efemérides finales sp3.

2.3.1. PROP-RECEPTOR

Para el PROP-Receptor se partirá de los parámetros orbitales y de las perturbaciones de dichos satélites a las 20h 00m 00s, del día 13 de enero de 2017, que corresponde a las 20h del 5º día de la semana GPS número 1931. Estos parámetros los obtendremos del mensaje de navegación transmitido ese día al

³ Según la notificación [32] del IGN France, la relación entre WGS84 e ITRF están por debajo de los 10 centímetros, muy por debajo de las precisiones requeridas para este proyecto, por lo que se consideran aquí, como coincidentes.

receptor de la estación de Bellmunt de Segarra (BELL00ESP) de la red permanente EUREF.

ftp://geofons.icc.cat/rinex3/diari_30s/RefData.17/Month.Jan/Day.01/BELL00ESP_R_20170010000_01D.GN.rnx

NOTA: La fecha y la hora seleccionadas no responden a ninguna época trascendente, sólo coinciden con el momento de iniciar este estudio de comparación de propagadores.

Tabla 9. Datos de la estación de referencia BELL00ESP de la red permanente EUREF.

| Código | Longitud | | | Latitud | | | UTM-ETRS89 | | |
|-----------|----------|----|---------|---------|----|-----------|------------|------------|-------------|
| | ° | m | s | ° | m | s | Eh | X | Y |
| | | | | | | | m | m | m |
| 271116002 | 1 | 24 | 40.9361 | 41 | 35 | 58.61.505 | 853.371 | 366757.087 | 4606557.991 |

Figura 19. Efemérides transmitidas a la estación BELL00ESP.

```

3.02          N: GNSS NAV DATA      G: GPS           RINEX VERSION / TYPE
GR50 V4.02      ICGC            20170112 235942 UTC PGM / RUN BY / DATE
GPSA  7.4506D-09 -1.4901D-08 -5.9605D-08  1.1921D-07 IONOSPHERIC CORR
GPSB  8.8064D+04 -4.9152D+04 -1.9661D+05  3.2768D+05 IONOSPHERIC CORR
GPUT  3.7252902985D-09 1.509903313D-14   61440 1932 TIME SYSTEM CORR
          18     18    1929       7 LEAP SECONDS
          (...) END OF HEADER

G24 2017 01 13 20 00 00-2.898415550590D-05-5.684341886081D-13 0.000000000000D+00
      3.300000000000D+01 3.218750000000D+00 4.707338936704D-09 2.097355154255D+00
      2.197921276093D-07 5.367794306949D-03 1.196376979351D-05 5.153712722778D+03
      5.040000000000D+05-1.508742570877D-07 3.094098994740D+00-1.303851604462D-08
      9.471941497526D-01 1.417187500000D+02 4.343075235202D-01-7.991047144903D-09
      -6.343121359322D-10 1.000000000000D+00 1.931000000000D+03 0.000000000000D+00
      2.000000000000D+00 0.000000000000D+00 2.328306436539D-09 3.300000000000D+01
      4.968000000000D+05

G15 2017 01 13 20 00-3.403760492802D-04-1.136868377216D-12 0.000000000000D+00
      6.800000000000D+01-4.250000000000D+00 5.865244310968D-09-3.112250095217D+00
      -3.054738044739D-07 9.018928627484D-03 4.537403583527D-06 5.153718723297D+03
      5.040000000000D+05-1.024454832077D-07 2.008542467056D+00-1.266598701477D-07
      9.291819284770D-01 2.769375000000D+02 5.613510380211D-01-8.549284683452D-09
      -5.571660653459D-11 1.000000000000D+00 1.931000000000D+03 0.000000000000D+00
      2.000000000000D+00 0.000000000000D+00-1.071020960808D-08 6.800000000000D+01
      4.968000000000D+05

G19 2017 01 13 20 00 -5.158013664186D-04 9.094947017729D-13 0.000000000000D+00
      5.900000000000D+01-5.803125000000D+01 4.478400829085D-09 8.066222959249D-01
      -2.890825271606D-06 1.020216615871D-02 1.091510057449D-06 5.153641494751D+03
      5.040000000000D+05-2.309679985046D-07-9.706886623819D-01 2.793967723846D-08
      9.757980486026D-01 3.707812500000D+02 9.058478979609D-01-8.507140070817D-09
      -3.032269163325D-10 1.000000000000D+00 1.931000000000D+03 0.000000000000D+00
      2.000000000000D+00 0.000000000000D+00-1.490116119385D-08 5.900000000000D+01
      4.968000000000D+05

G12 2017 01 13 20 00 3.889612853527D-04-2.273736754432D-13 0.000000000000D+00
      6.200000000000D+01 1.432187500000D+02 3.821944913632D-09 6.281228440663D-01
      7.435679435730D-06 6.261964561418D-03 7.290393114090D-06 5.153606155396D+03
      5.040000000000D+05-3.911554813385D-08-2.057158077280D+00 6.146728992462D-08
      9.897694531506D-01 2.523750000000D+02 7.822014986778D-01-7.958545791091D-09
      2.478674675321D-10 1.000000000000D+00 1.931000000000D+03 0.000000000000D+00
      2.000000000000D+00 0.000000000000D+00-1.257285475731D-08 6.200000000000D+01
      4.968000000000D+05

(...)
```

Para determinar el retraso entre relojes (satélite y receptor), debemos dividir la distancia, o mejor dicho, la pseudodistancia, entre receptor y satélite por la velocidad de la luz (la velocidad a que viaja la señal). La pseudodistancia entre los satélites y BELL00ESP se obtiene de las observaciones recibidas, en esta estación, en la época de las efemérides.

Figura 20. Observables de la estación BELL00ESP.

| 2.10 | OBSERVATION DATA | | | M (MIXED) | RINEX VERSION / TYPE | |
|------------------------------|------------------|--------------|--------------|--------------------|--|--------|
| GPServer 2.74 3724 | Rinex Merge | | | 17-Jan-17 09:32:09 | PGM / RUN BY / DATE | |
| BELL | | | | | MARKER NAME | |
| 13431M001 | | | | | MARKER NUMBER | |
| ICGC | ICGC | | | | OBSERVER / AGENCY | |
| 1831708 | LEICA GR50 | | | 4.02/7.00 | REC # / TYPE / VERS | |
| 0 | | | | | RCV CLOCK OFFS APPL | |
| 726356 | LEIAR25.R4 | | | NONE | ANT # / TYPE | |
| 4775849.5920 | 116814.0900 | 4213018.6940 | | | APPROX POSITION XYZ | |
| 0.0770 | 0.0000 | 0.0000 | | | ANTENNA: DELTA H/E/N | |
| 1 | 1 | 0 | | | WAVELENGTH FACT L1/2 | |
| 6 | C1 | P2 | L1 L2 S1 S2 | | # / TYPES OF OBSERV | |
| 1.000 | | | | | INTERVAL | |
| 2017 | 1 | 13 | 19 | 0 | TIME OF FIRST OBS | |
| coordenadas ajust ICGC2015.1 | | | | | COMMENT | |
| | | | | | END OF HEADER | |
| (...) | | | | | | |
| 17 | 1 | 13 | 20 | 0 | 0.0000000 | |
| | | | | | 0 20G02G06G10G12G14G15G19G24G25G29G32R06 | |
| | | | | | R07R08R09R15R16R17R18R19 (...) | |
| (...) | | | | | | |
| PG12 | 20243002.526 | | 20243000.646 | 106377721.23909 | 82891752.70807 | 51.000 |
| | | | | | | |
| (...) | | | | | | |
| PG15 | 23810245.022 | | 23810243.422 | 125123688.07007 | 97498964.42102 | 47.000 |
| | | | | | | |
| (...) | | | | | | |
| PG19 | 25273992.512 | | 25273994.352 | 132815726.32405 | 103492790.34101 | 42.000 |
| | | | | | | |
| (...) | | | | | | |
| PG24 | 24379766.618 | | 24379762.558 | 128116560.50305 | 99831059.20602 | 41.000 |

Con estos datos y con las constantes que definen el sistema de referencia de la constelación (GPS WGS84), μ y Ω_e' , podemos ya ejecutar el modelo propagador PROP-Receptor. Las ecuaciones que definen este modelo se han descrito en el capítulo anterior. El código del script programado con este modelo se puede consultar al final del anexo 2.

2.3.2. PROP-SDP4

Para el propagador PROP-SDP4 se partirá de los TLE adquirido del web *Celestrak* [19]. Corresponden a los TLE actualizados el día 13 de enero de 2017 a las 19h 56m 30s (5º día de la semana GPS número 1931).

NOTA: Se pretendía partir de una actualización de los datos TLE a la misma hora que la de las efemérides transmitidas utilizadas para el PROP-Receptor (20:00:00h), pero no existe ninguna actualización a esa hora para el día 13 de enero de 2017. Se ha utilizado la actualización que más se aproxima a esa hora.

Figura 21. Fichero TLE.

```

GPS BIIR-11 (PRN 19)
1 28190C 04009A 17013.83090278 -.00000000 00000-0 00000-0 0 130
2 28190 55.9191 52.0310 0100034 50.9716 45.5777 2.00567014 10
(...)

GPS BIIRM-3 (PRN 12)
1 29601C 06052A 17013.83090278 .00000000 00000-0 00000-0 0 137
2 29601 56.7149 349.8172 0061155 43.2017 35.9616 2.00572805 11
(...)

GPS BIIRM-4 (PRN 15)
1 32260C 07047A 17013.83090278 .00000000 00000-0 00000-0 0 137
2 32260 53.2684 222.7322 0088974 30.8928 181.3697 2.00556529 19
(...)

GPS BIIIF-3 (PRN 24)
1 38833C 12053A 17013.83090278 .00000000 00000-0 00000-0 0 132
2 38833 54.2529 284.9384 0052686 22.6522 120.8301 2.00559299 15
(...)
```

2.3.3. POSICIONES REALES Y CALCULADAS DE LOS SATÉLITES

Las coordenadas geocéntricas reales de los satélites en cada momento, se han obtenido de las efemérides precisas distribuidas por [16] correspondientes a las fechas y a las horas propagadas.

Tabla 10. Efemérides precisas. Posición real de los satélites.

| Calendario gregoriano | Posiciones reales de G12 | | | Posiciones reales de G15 | | |
|-----------------------|--------------------------|------------|------------|--------------------------|-----------|-------------|
| | X (m) | Y (m) | Z (m) | X (m) | Y (m) | Z (m) |
| 13/01/2017 | 14632088.5 | 2760726.2 | 21830579.8 | 23534494.1 | 4649768.6 | -11948510.6 |
| 14/01/2017 | 14334248.4 | 3395965.0 | 21942587.8 | 23183722.0 | 4785421.7 | -12564365.8 |
| 15/01/2017 | 14044907.5 | 4038244.2 | 22025128.5 | 22817156.3 | 4929106.3 | -13164491.5 |
| 16/01/2017 | 13764581.1 | 4686749.5 | 22078130.8 | 22435579.9 | 5081214.6 | -13747933.4 |
| 17/01/2017 | 13493699.3 | 5340624.1 | 22101576.6 | 22039740.5 | 5242210.0 | -14313897.3 |
| 18/01/2017 | 13232628.8 | 5998949.3 | 22095481.2 | 21630341.7 | 5412598.2 | -14861736.7 |
| 19/01/2017 | 12981690.1 | 6660737.8 | 22059886.4 | 21208050.2 | 5592887.2 | -15390912.9 |
| 20/01/2017 | 12741169.7 | 7324939.9 | 21994862.6 | 20773506.9 | 5783553.9 | -15900952.6 |
| 21/01/2017 | 12511325.5 | 7990451.5 | 21900514.0 | 20327340.8 | 5985017.4 | -16391412.9 |
| 22/01/2017 | 12292389.9 | 8656125.8 | 21776986.1 | 19870183.1 | 6197618.1 | -16861855.3 |
| 23/01/2017 | 12084570.2 | 9320784.1 | 21624470.8 | 19402682.3 | 6421601.5 | -17311829.2 |
| 24/01/2017 | 11888050.1 | 9983230.8 | 21443208.9 | 18925523.6 | 6657104.9 | -17740859.7 |
| 25/01/2017 | 11702987.4 | 10642272.7 | 21233487.9 | 18439451.3 | 6904148.4 | -18148440.8 |

| Calendario gregoriano | Posiciones reales de G19 | | | Posiciones reales de G24 | | |
|-----------------------|--------------------------|------------|------------|--------------------------|------------|------------|
| | X (m) | Y (m) | Z (m) | X (m) | Y (m) | Z (m) |
| 13/01/2017 | -3762432.6 | 14695722.5 | 21574269.2 | 20550700.7 | 11732799.2 | 12219990.1 |
| 14/01/2017 | -4450716.0 | 14710859.9 | 21441361.9 | 20736697.7 | 12057016.1 | 11583071.0 |
| 15/01/2017 | -5133982.4 | 14735585.8 | 21280304.2 | 20914682.5 | 12363865.3 | 10931143.8 |
| 16/01/2017 | -5811044.9 | 14769718.3 | 21091321.5 | 21083790.0 | 12653257.4 | 10265164.2 |
| 17/01/2017 | -6480758.5 | 14813058.7 | 20874656.6 | 21243183.4 | 12925110.3 | 9586156.5 |
| 18/01/2017 | -7142040.9 | 14865376.7 | 20630583.7 | 21392065.7 | 13179390.7 | 8895169.8 |
| 19/01/2017 | -7793881.2 | 14926399.6 | 20359414.6 | 21529681.8 | 13416136.4 | 8193251.8 |
| 20/01/2017 | -8435339.9 | 14995805.5 | 20061499.9 | 21655314.0 | 13635473.5 | 7481423.4 |
| 21/01/2017 | -9065544.6 | 15073220.1 | 19737226.8 | 21768275.0 | 13837619.7 | 6760664.3 |
| 22/01/2017 | -9683685.5 | 15158214.5 | 19387014.4 | 21867901.3 | 14022882.3 | 6031901.7 |
| 23/01/2017 | -10289009.2 | 15250301.9 | 19011312.3 | 21953545.9 | 14191652.7 | 5296000.3 |
| 24/01/2017 | -10880810.5 | 15348935.7 | 18610599.7 | 22024572.3 | 14344395.7 | 4553757.9 |
| 25/01/2017 | -11458418.7 | 15453507.0 | 18185390.2 | 22080349.5 | 14481632.9 | 3805909.6 |

Tabla 11. Resultados obtenidos con el propagador PROP-Receptor y la comparación con la posición real (G12).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|------------|------------|-------------|----------|--------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 14632169.7 | 2760556.0 | 21830545.7 | 81.3 | -170.2 | -34.1 | 191.7 |
| 14/01/2017 (1) | 14334789.5 | 3395125.5 | 21942507.9 | 541.0 | -839.5 | -79.9 | 1001.9 |
| 15/01/2017 (2) | 14045725.6 | 4036748.2 | 22025238.3 | 818.1 | -1496.0 | 109.8 | 1708.6 |
| 16/01/2017 (3) | 13765515.1 | 4684470.3 | 22078638.6 | 934.0 | -2279.2 | 507.8 | 2515.0 |
| 17/01/2017 (4) | 13494651.9 | 5337314.3 | 22102650.2 | 952.6 | -3309.8 | 1073.6 | 3607.6 |
| 18/01/2017 (5) | 13233584.7 | 5994281.6 | 22097254.3 | 956.0 | -4667.8 | 1773.2 | 5083.9 |
| 19/01/2017 (6) | 12982716.6 | 6654354.9 | 22062471.6 | 1026.5 | -6382.9 | 2585.2 | 6962.6 |
| 20/01/2017 (7) | 12742403.4 | 7316501.4 | 21998362.3 | 1233.7 | -8438.5 | 3499.7 | 9218.3 |
| 21/01/2017 (8) | 12512953.1 | 7979675.2 | 21905026.1 | 1627.5 | -10776.3 | 4512.0 | 11795.6 |
| 22/01/2017 (9) | 12294625.0 | 8642820.2 | 21782601.6 | 2235.1 | -13305.6 | 5615.5 | 14614.0 |
| 23/01/2017 (10) | 12087629.4 | 9304872.9 | 21631266.7 | 3059.1 | -15911.2 | 6795.9 | 17570.1 |
| 24/01/2017 (11) | 11892126.7 | 9964765.7 | 21451237.4 | 4076.5 | -18465.2 | 8028.5 | 20543.5 |
| 25/01/2017 (12) | 11708227.5 | 10621429.0 | 21242768.1 | 5240.1 | -20843.7 | 9280.2 | 23410.3 |

Tabla 12. Resultados obtenidos con el propagador PROP-SDP4 y la comparación con la posición real (G12).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|------------|------------|-------------|----------|--------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 14632529.8 | 2760816.0 | 21830019.6 | 441.3 | 89.8 | -560.2 | 718.8 |
| 14/01/2017 (1) | 14334765.4 | 3396057.2 | 21942115.9 | 517.0 | 92.2 | -472.0 | 706.1 |
| 15/01/2017 (2) | 14045573.8 | 4038135.1 | 22024789.2 | 666.4 | -109.1 | -339.2 | 755.7 |
| 16/01/2017 (3) | 13765460.9 | 4686242.8 | 22077963.7 | 879.8 | -506.6 | -167.1 | 1028.9 |
| 17/01/2017 (4) | 13494854.6 | 5339531.4 | 22101609.1 | 1155.3 | -1092.7 | 32.4 | 1590.5 |
| 18/01/2017 (5) | 13234122.3 | 5997092.4 | 22095734.8 | 1493.5 | -1856.9 | 253.6 | 2396.5 |
| 19/01/2017 (6) | 12983585.1 | 6657953.2 | 22060387.8 | 1895.0 | -2784.6 | 501.3 | 3405.3 |
| 20/01/2017 (7) | 12743528.1 | 7321082.0 | 21995651.7 | 2358.4 | -3857.9 | 789.1 | 4590.0 |
| 21/01/2017 (8) | 12514205.7 | 7985397.6 | 21901648.5 | 2880.2 | -5053.9 | 1134.5 | 5926.6 |
| 22/01/2017 (9) | 12295843.4 | 8649782.1 | 21778540.0 | 3453.5 | -6343.6 | 1553.9 | 7388.1 |
| 23/01/2017 (10) | 12088637.8 | 9313093.5 | 21626528.9 | 4067.5 | -7690.6 | 2058.2 | 8940.1 |
| 24/01/2017 (11) | 11892755.6 | 9974178.1 | 21445858.1 | 4705.5 | -9052.8 | 2649.2 | 10541.0 |
| 25/01/2017 (12) | 11708332.6 | 10631884.8 | 21236808.0 | 5345.2 | -10388.0 | 3320.1 | 12145.1 |

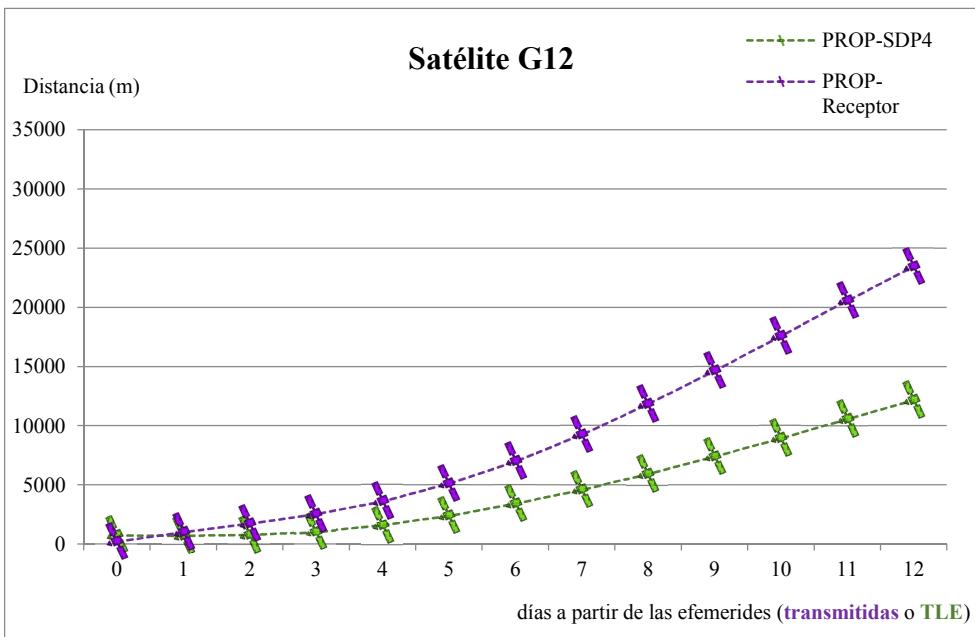


Figura 22. Variaciones entre la posición real y las predichas por los dos propagadores (G12).

Tabla 13. Resultados obtenidos con el propagador PROP-Receptor y la comparación con la posición real (G15).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|-----------|-------------|-------------|---------|----------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 23534612.7 | 4649721.7 | -11948295.7 | 118.6 | -46.9 | 214.9 | 249.9 |
| 14/01/2017 (1) | 23183516.8 | 4785856.2 | -12564142.9 | -205.2 | 434.5 | 222.9 | 529.7 |
| 15/01/2017 (2) | 22816341.7 | 4930523.3 | -13164496.6 | -814.6 | 1417.0 | -5.1 | 1634.5 |
| 16/01/2017 (3) | 22433733.8 | 5084155.9 | -13748615.9 | -1846.1 | 2941.4 | -682.5 | 3539.1 |
| 17/01/2017 (4) | 22036367.9 | 5247155.0 | -14315779.0 | -3372.6 | 4944.9 | -1881.7 | 6274.4 |
| 18/01/2017 (5) | 21624945.0 | 5419887.8 | -14865284.2 | -5396.8 | 7289.6 | -3547.5 | 9739.1 |
| 19/01/2017 (6) | 21200190.8 | 5602687.1 | -15396451.0 | -7859.4 | 9799.9 | -5538.1 | 13728.8 |
| 20/01/2017 (7) | 20762853.8 | 5795849.5 | -15908620.5 | -10653.1 | 12295.6 | -7667.9 | 17985.2 |
| 21/01/2017 (8) | 20313703.3 | 5999634.7 | -16401156.5 | -13637.5 | 14617.3 | -9743.6 | 22239.3 |
| 22/01/2017 (9) | 19853527.5 | 6214264.5 | -16873446.0 | -16655.5 | 16646.4 | -11590.7 | 26246.1 |
| 23/01/2017 (10) | 19383131.4 | 6439921.7 | -17324900.0 | -19551.0 | 18320.2 | -13070.8 | 29811.3 |
| 24/01/2017 (11) | 18903334.6 | 6676749.4 | -17754954.3 | -22189.0 | 19644.5 | -14094.6 | 32816.4 |
| 25/01/2017 (12) | 18414969.5 | 6924850.7 | -18163069.9 | -24481.8 | 20702.3 | -14629.1 | 35241.4 |

Tabla 14. Resultados obtenidos con el propagador PROP-SDP4 y la comparación con la posición real (G15).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|-----------|-------------|-------------|--------|---------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 23534993.5 | 4650013.6 | -11947819.8 | 499.5 | 245.0 | 690.8 | 886.9 |
| 14/01/2017 (1) | 23183976.5 | 4785772.2 | -12563707.4 | 254.5 | 350.6 | 658.4 | 788.1 |
| 15/01/2017 (2) | 22817160.6 | 4929548.9 | -13163846.7 | 4.4 | 442.6 | 644.8 | 782.1 |
| 16/01/2017 (3) | 22435328.6 | 5081771.7 | -13747357.0 | -251.3 | 557.1 | 576.4 | 840.1 |
| 17/01/2017 (4) | 22039223.6 | 5242929.3 | -14313482.8 | -516.9 | 719.3 | 414.5 | 977.9 |
| 18/01/2017 (5) | 21629541.8 | 5413538.5 | -14861579.3 | -800.0 | 940.3 | 157.4 | 1244.6 |
| 19/01/2017 (6) | 21206939.0 | 5594107.8 | -15391085.2 | -1111.2 | 1220.6 | -172.3 | 1659.6 |
| 20/01/2017 (7) | 20772045.3 | 5785106.4 | -15901492.4 | -1461.7 | 1552.5 | -539.8 | 2199.6 |
| 21/01/2017 (8) | 20325481.4 | 5986941.6 | -16392321.3 | -1859.4 | 1924.1 | -908.4 | 2825.8 |
| 22/01/2017 (9) | 19867874.7 | 6199943.3 | -16863102.4 | -2308.4 | 2325.2 | -1247.0 | 3505.7 |
| 23/01/2017 (10) | 19399873.9 | 6424353.4 | -17313365.5 | -2808.4 | 2751.9 | -1536.3 | 4221.4 |
| 24/01/2017 (11) | 18922164.0 | 6660318.6 | -17742632.2 | -3359.6 | 3213.7 | -1772.5 | 4975.6 |
| 25/01/2017 (12) | 18435482.7 | 6907885.0 | -18150410.6 | -3968.6 | 3736.6 | -1969.8 | 5795.9 |

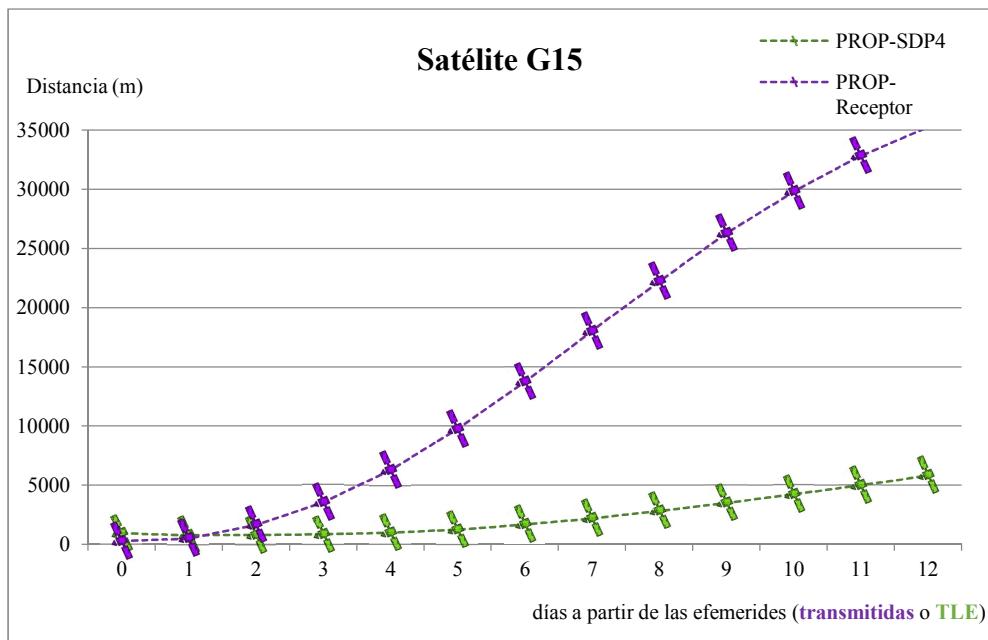


Figura 23 Variaciones entre la posición real y las predichas por los dos propagadores (G15).

Tabla 15. Resultados obtenidos con el propagador PROP-Receptor y la comparación con la posición real (G19).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|-------------|------------|------------|-------------|--------|----------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | -3762203.9 | 14695718.8 | 21574308.8 | 228.8 | -3.8 | 39.6 | 232.2 |
| 14/01/2017 (1) | -4451035.8 | 14711247.2 | 21441100.3 | -319.8 | 387.3 | -261.6 | 566.3 |
| 15/01/2017 (2) | -5134669.9 | 14736631.1 | 21279493.2 | -687.5 | 1045.3 | -811.0 | 1490.9 |
| 16/01/2017 (3) | -5812076.9 | 14771655.1 | 21089726.6 | -1032.0 | 1936.9 | -1594.8 | 2712.9 |
| 17/01/2017 (4) | -6482243.9 | 14816058.3 | 20872077.1 | -1485.4 | 2999.6 | -2579.5 | 4225.9 |
| 18/01/2017 (5) | -7144177.3 | 14869534.8 | 20626858.2 | -2136.4 | 4158.1 | -3725.5 | 5977.7 |
| 19/01/2017 (6) | -7796905.3 | 14931735.0 | 20354419.9 | -3024.1 | 5335.5 | -4994.7 | 7909.5 |
| 20/01/2017 (7) | -8439480.6 | 15002266.7 | 20055148.1 | -4140.7 | 6461.2 | -6351.9 | 9961.9 |
| 21/01/2017 (8) | -9070982.3 | 15080696.5 | 19729463.7 | -5437.7 | 7476.3 | -7763.1 | 12071.9 |
| 22/01/2017 (9) | -9690519.1 | 15166551.1 | 19377822.2 | -6833.6 | 8336.5 | -9192.2 | 14166.6 |
| 23/01/2017 (10) | -10297230.9 | 15259319.0 | 19000712.7 | -8221.6 | 9017.1 | -10599.5 | 16163.3 |
| 24/01/2017 (11) | -10890291.1 | 15358452.3 | 18598657.3 | -9480.6 | 9516.6 | -11942.4 | 17974.1 |
| 25/01/2017 (12) | -11468909.0 | 15463368.2 | 18172210.0 | -10490.3 | 9861.3 | -13180.1 | 19519.4 |

Tabla 16. Resultados obtenidos con el propagador PROP-SDP4 y la comparación con la posición real (G19).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|-------------|------------|------------|-------------|--------|--------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | -3762847.0 | 14695224.9 | 21574745.8 | -414.4 | -497.6 | 476.6 | 804.0 |
| 14/01/2017 (1) | -4451107.1 | 14710301.6 | 21441850.5 | -391.1 | -558.3 | 488.6 | 838.7 |
| 15/01/2017 (2) | -5134265.2 | 14734967.8 | 21280765.9 | -282.8 | -618.0 | 461.7 | 821.6 |
| 16/01/2017 (3) | -5811121.2 | 14769038.1 | 21091745.8 | -76.3 | -680.1 | 424.3 | 805.3 |
| 17/01/2017 (4) | -6480530.3 | 14812316.7 | 20875070.5 | 228.2 | -742.0 | 413.8 | 879.7 |
| 18/01/2017 (5) | -7141424.7 | 14864581.0 | 20631044.8 | 616.2 | -795.8 | 461.1 | 1107.1 |
| 19/01/2017 (6) | -7792820.2 | 14925566.9 | 20359997.7 | 1061.0 | -832.6 | 583.2 | 1469.4 |
| 20/01/2017 (7) | -8433810.7 | 14994958.7 | 20062281.6 | 1529.2 | -846.8 | 781.6 | 1914.8 |
| 21/01/2017 (8) | -9063558.3 | 15072382.5 | 19738272.2 | 1986.3 | -837.6 | 1045.4 | 2395.8 |
| 22/01/2017 (9) | -9681281.1 | 15157404.3 | 19388370.0 | 2404.3 | -810.2 | 1355.6 | 2876.6 |
| 23/01/2017 (10) | -10286243.8 | 15249529.0 | 19013001.5 | 2765.4 | -772.9 | 1689.2 | 3331.4 |
| 24/01/2017 (11) | -10877747.2 | 15348201.3 | 18612623.5 | 3063.3 | -734.3 | 2023.8 | 3744.2 |
| 25/01/2017 (12) | -11455118.2 | 15452807.2 | 18187729.1 | 3300.4 | -699.8 | 2338.9 | 4105.3 |

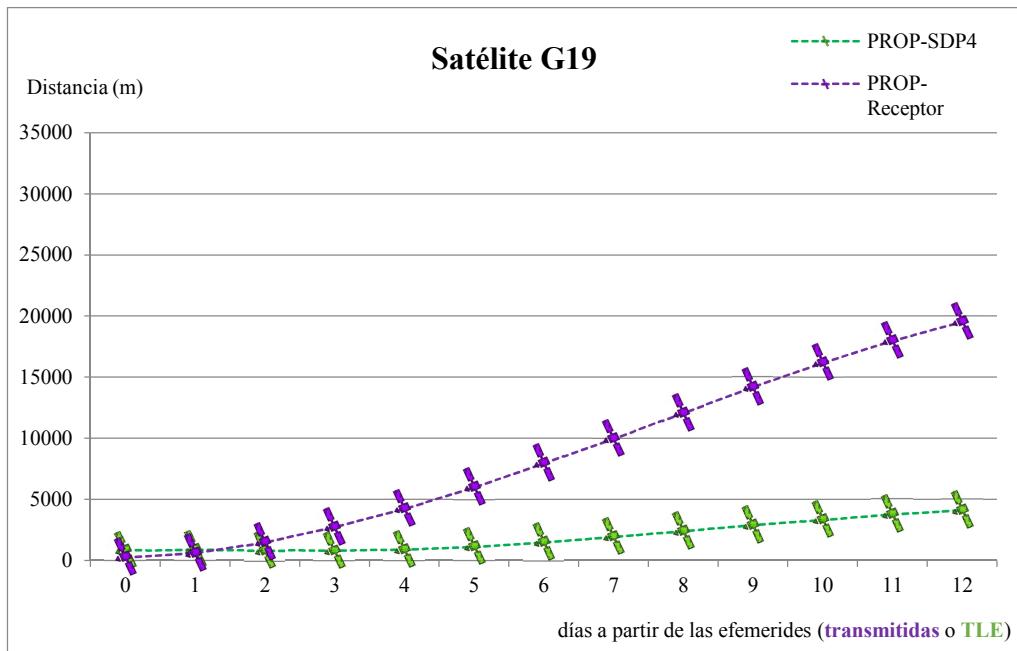


Figura 24. Variaciones entre la posición real y las predichas por los dos propagadores (G19).

Tabla 17. Resultados obtenidos con el propagador PROP-Receptor y la comparación con la posición real (G24).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|------------|------------|-------------|---------|---------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 20550646.3 | 11732701.1 | 12220170.7 | -54.4 | -98.1 | 180.7 | 212.7 |
| 14/01/2017 (1) | 20736644.6 | 12056196.0 | 11583940.1 | -53.0 | -820.1 | 869.1 | 1196.1 |
| 15/01/2017 (2) | 20914533.8 | 12361995.7 | 10933261.8 | -148.7 | -1869.5 | 2118.0 | 2829.0 |
| 16/01/2017 (3) | 21083527.3 | 12650128.9 | 10268957.8 | -262.7 | -3128.5 | 3793.6 | 4924.2 |
| 17/01/2017 (4) | 21242848.6 | 12920668.7 | 9591866.0 | -334.9 | -4441.6 | 5709.5 | 7241.4 |
| 18/01/2017 (5) | 21391733.6 | 13173732.2 | 8902839.4 | -332.0 | -5658.4 | 7669.5 | 9536.8 |
| 19/01/2017 (6) | 21529433.1 | 13409480.1 | 8202744.7 | -248.8 | -6656.3 | 9492.9 | 11596.7 |
| 20/01/2017 (7) | 21655214.3 | 13628115.3 | 7492461.9 | -99.7 | -7358.2 | 11038.5 | 13266.5 |
| 21/01/2017 (8) | 21768363.7 | 13829882.8 | 6772882.5 | 88.6 | -7737.0 | 12218.2 | 14462.1 |
| 22/01/2017 (9) | 21868188.6 | 14015068.1 | 6044908.8 | 287.4 | -7814.2 | 13007.1 | 15176.6 |
| 23/01/2017 (10) | 21954019.7 | 14183996.4 | 5309452.6 | 473.9 | -7656.3 | 13452.3 | 15485.7 |
| 24/01/2017 (11) | 22025212.6 | 14337031.3 | 4567434.3 | 640.4 | -7364.5 | 13676.4 | 15546.4 |
| 25/01/2017 (12) | 22081150.0 | 14474573.5 | 3819781.6 | 800.4 | -7059.3 | 13872.0 | 15585.5 |

Tabla 18. Resultados obtenidos con el propagador PROP-SDP4 y la comparación con la posición real (G24).

| Calendario Gregoriano (días desde el origen) | Propagadas | | | Diferencias | | | |
|---|------------|------------|------------|-------------|---------|--------|------------|
| | X (m) | Y (m) | Z (m) | ΔX (m) | ΔY (m) | ΔZ (m) | ΔDist. (m) |
| 13/01/2017 (0) | 20550023.5 | 11734231.1 | 12219978.9 | -677.3 | 1432.0 | -11.2 | 1584.1 |
| 14/01/2017 (1) | 20735921.8 | 12058216.1 | 11583269.4 | -775.9 | 1200.1 | 198.4 | 1442.7 |
| 15/01/2017 (2) | 20913797.3 | 12364700.5 | 10931658.1 | -885.2 | 835.2 | 514.3 | 1321.2 |
| 16/01/2017 (3) | 21082812.7 | 12653601.6 | 10266102.9 | -977.3 | 344.2 | 938.7 | 1398.1 |
| 17/01/2017 (4) | 21242155.3 | 12924863.6 | 9587621.2 | -1028.1 | -246.7 | 1464.6 | 1806.4 |
| 18/01/2017 (5) | 21391043.9 | 13178488.2 | 8897249.5 | -1021.8 | -902.5 | 2079.7 | 2486.7 |
| 19/01/2017 (6) | 21528729.5 | 13414554.5 | 8196012.0 | -952.4 | -1581.9 | 2760.2 | 3320.9 |
| 20/01/2017 (7) | 21654492.0 | 13633226.8 | 7484899.6 | -822.0 | -2246.7 | 3476.1 | 4219.8 |
| 21/01/2017 (8) | 21767636.0 | 13834753.5 | 6764859.5 | -639.1 | -2866.2 | 4195.2 | 5120.9 |
| 22/01/2017 (9) | 21867485.9 | 14019460.6 | 6036792.1 | -415.4 | -3421.8 | 4890.4 | 5983.0 |
| 23/01/2017 (10) | 21953382.6 | 14187742.0 | 5301549.9 | -163.3 | -3910.7 | 5549.6 | 6791.0 |
| 24/01/2017 (11) | 22024680.4 | 14340049.3 | 4559940.6 | 108.1 | -4346.5 | 6182.7 | 7558.4 |
| 25/01/2017 (12) | 22080746.2 | 14476878.5 | 3812732.0 | 396.7 | -4754.4 | 6822.5 | 8325.1 |

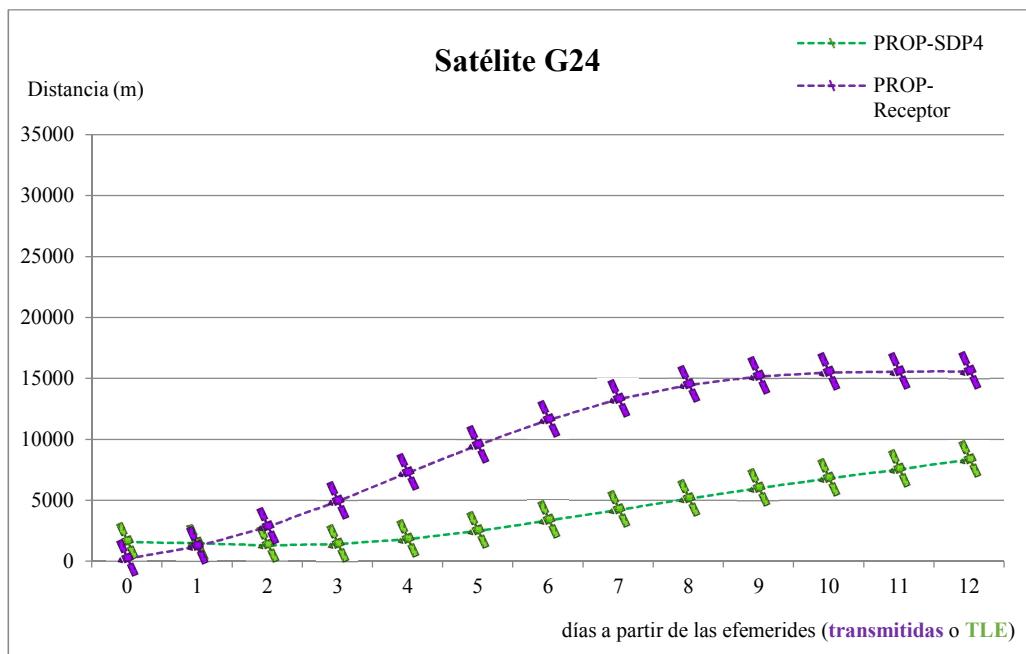


Figura 25. Variaciones entre la posición real y las predichas por los dos propagadores (G24).

2.4. VALORACIÓN DE LOS CANDIDATOS

Como era de esperar, los dos propagadores obtienen mejores resultados en los días cercanos al origen de las efemérides, ya que sus parámetros orbitales se refieren a ese instante. A medida que transcurren los días, aumentan las distancias entre las posiciones calculadas y las reales (ΔDist). Aun así, se observan diferencias entre propagadores en el origen (2017-01-13 20:00:00 h). El PROP-Receptor obtiene mejores resultados ya que parte de efemérides referidas exactamente a esa época y el PROP-SDP4 lo hace con efemérides referidas a un instante anterior, adelantadas 3 minutos y 30 segundos.

Transcurrido un día, los dos propagadores presentan resultados muy similares, no superando en ninguno de los casos los 1.5 km de error, incluso por debajo del kilómetro en la mayoría de los satélites.

A partir del segundo día, las posiciones de los satélites calculadas con el PROP-SDP4 son más cercanas a las posiciones reales que las calculadas con el PROP-Receptor. El PROP-SDP4 presenta una estabilidad en los primeros 4-5 días que incluso, conlleva una disminución de la ΔDist en los días inmediatamente posteriores a la fecha de inicio. A partir del quinto día, esta variación aumenta de forma constante, con magnitudes distintas dependiendo de cada satélite. Sin embargo, en el PROP-Receptor, la ΔDist aumenta gradualmente ya desde su primera época.

A largo plazo, las diferencias entre las ΔDist obtenidas con cada propagador se hacen más evidentes. Con el PROP-SDP4, de los 4 satélites estudiados, sólo el G12 sobrepasa los 10 km de ΔDist , y lo hace a partir del día 11, manteniendo siempre valores por debajo de los 5 km hasta el séptimo día. Por el contrario, el PROP-Receptor, en todos los casos, obtiene valores de ΔDist superiores a 10 km a partir del séptimo día y sobrepasa los 5 km a partir del quinto día. En PROP-Receptor, las diferencias con la posición real superan los 15 km al décimo día y llega a superar los 35 km en el duodécimo día en G15.

Cabe destacar, la estabilidad que presentan los resultados obtenidos con el PROP-SDP4. Sobre todo en los primeros 4-5 días, manteniendo siempre valores de ΔDist por debajo de los 2,5 km. Esta estabilidad permitiría hacer las planificaciones de los trabajos de campo con una antelación de más de 3-4 días.

Donde también se observan grandes diferencias es en la accesibilidad a los datos de partida y su adaptación al entorno de desarrollo de la aplicación. Como hemos visto, el PROP-Receptor utiliza las efemérides radiodifundidas. Estas se pueden obtener, por ejemplo, de una estación permanente que distribuya los mensajes de navegación enviados por los satélites de los que ha recibido señal. Por lo tanto, para una época concreta, no bastará con las efemérides que nos proporcione una sola estación, pues no recibirá el mensaje de navegación de toda la constelación completa. Además, el formato de los datos de las efemérides radiodifundidas, varía según sea la constelación. Esto conllevaría tener que programar un módulo de importación de datos para cada uno de los formatos y este módulo tendría que

actuar en múltiples enlaces para obtener datos de varias estaciones de referencia. Además no todas las estaciones de referencia están preparadas para recibir datos de todas las constelaciones.

El PROP-SDP4, sin embargo, utiliza los TLE como formato para los datos de partida. Este formato se mantiene uniforme para todas las constelaciones y se distribuye en un mismo enlace (sólo varía la parte final del enlace). Otra ventaja es que el formato TLE es simple de enlazar con cualquier programa capaz de leer archivos .txt. Además, sólo se distribuye un archivo para cada constelación donde se incluyen los datos de todos sus satélites.

La actualización de los archivos TLE la lleva a cabo periódicamente [5] substituyendo los datos siempre en un mismo enlace. Esto facilita enormemente la recogida automática de estos datos y asegura la adquisición de la última versión disponible. Cada vez que se ejecute la aplicación se accederá al mismo enlace que contendrá los datos lo más actualizados posible, sin tener que preocuparse, ni tan solo, de la nomenclatura de los ficheros, ya que esta se mantiene invariable.

Otra de las ventajas de escoger el PROP-SDP4 es que su implementación en código Python se encuentra disponible en [30] y ha sido ampliamente testeado por otros usuarios. Además, el inconveniente de transformar las coordenadas inerciales (TEME), que retorna [30], a coordenadas fijas (ITRF) ya ha sido solventado al programar dicha transformación para el presente estudio de comparación.

Por lo tanto, el propagador utilizado será el PROP-SDP4, basado en [30] y [12].

El código del script programado para el PROP-Receptor se puede ver en el anexo 2. El código del PROP-SDP4 no se ha publicado de forma independiente sino que se ha integrado en el propio código de GNSSplannig. En este caso, la parte recuperada del script *python-sgp4* (Brandon Rhodes, 2012-2016) se encuentra en los módulos de la librería *sgp4* dentro del directorio *GNSSplanning/scripts* (ver apartado 3.4) y la parte nueva programada para este complemento, se encuentra en los módulos *ImportTLE.py*, *Propagate.py*, *Satellite.py* y *transform.py*, en ese mismo directorio.

2.5. LIMITACIONES TEMPORALES

Como se ha comprobado en este estudio, la predicción de la posición de los satélites, sea con el propagador que sea, no consigue resultados ajustados in aeternum. A medida que trascurren los días a partir de la actualización de las efemérides las posiciones calculadas se alejan de las reales. Como el complemento GNSSplanning no deja de ser una herramienta de planificación, se tendrá que informar al usuario que las predicciones de más de n días conllevan errores importantes entre la posición calculada y la real. Vamos ahora a proponer un valor para n .

Con el propagador seleccionado (PROP-SDP4) y suponiendo que el error en distancia ΔDist . se produjera en la dirección perpendicular a la línea receptor-satélite, la variación angular que el observador percibiría sería de:

$$\tan \alpha = \frac{\Delta\text{Dist} \text{ km}}{20.000 \text{ km}} \quad (63)$$

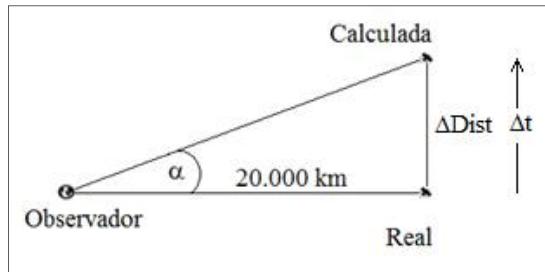


Figura 26. Variación entre las posiciones calculadas y posiciones reales.

En la tabla 19 se recogen los resultados considerando la velocidad aproximada de un satélite de 3.218 km/h y la distancia media aproximada entre un receptor y los satélites de la constelación GPS, unos 20.000 km.

Tabla 19. Variación entre las posiciones calculadas y posiciones reales (distancia, ángulo y tiempo).

| Días desde efemérides | PROP-SDP4.Distancia entre posición calculada y real. ΔDist . (m) | | | | | Angulo α (minutos)* | Tiempo transcurrido desde la posición real hasta la calculada (segundos)* |
|-----------------------|--|--------|--------|--------|--------|----------------------------|---|
| | G12 | G15 | G19 | G24 | Media | | |
| 0 días | 718.8 | 886.9 | 804.0 | 1584.1 | 998.5 | 0.2 | 1.1 |
| 1 día | 706.1 | 788.1 | 838.7 | 1442.7 | 943.9 | 0.2 | 1.1 |
| 2 días | 755.7 | 782.1 | 821.6 | 1321.2 | 920.1 | 0.2 | 1.0 |
| 3 días | 1028.9 | 840.1 | 805.3 | 1398.1 | 1018.1 | 0.2 | 1.1 |
| 4 días | 1590.5 | 977.9 | 879.7 | 1806.4 | 1313.6 | 0.2 | 1.5 |
| 5 días | 2396.5 | 1244.6 | 1107.1 | 2486.7 | 1808.7 | 0.3 | 2.0 |
| 6 días | 3405.3 | 1659.6 | 1469.4 | 3320.9 | 2463.8 | 0.4 | 2.8 |
| 7 días | 4590.0 | 2199.6 | 1914.8 | 4219.8 | 3231.1 | 0.6 | 3.6 |
| 8 días | 5926.6 | 2825.8 | 2395.8 | 5120.9 | 4067.3 | 0.7 | 4.6 |
| 9 días | 7388.1 | 3505.7 | 2876.6 | 5983.0 | 4938.4 | 0.8 | 5.5 |
| 10 días | 8940.1 | 4221.4 | 3331.4 | 6791.0 | 5821.0 | 1.0 | 6.5 |
| 11 días | 10541.0 | 4975.6 | 3744.2 | 7558.4 | 6704.8 | 1.2 | 7.5 |
| 12 días | 12145.1 | 5795.9 | 4105.3 | 8325.1 | 7592.8 | 1.3 | 8.5 |

* suponiendo que el error en distancia de produjera en la dirección perpendicular a la dirección receptor-satélite.

Por lo tanto si el usuario no sobrepasa los 7 días de predicción, como máximo tendrá un error angular de 0.6 minutos, y verá el satélite en el gráfico polar de la aplicación 3.6 segundos antes/después de que esto ocurra realmente.

Aunque puedan parecer variaciones poco importantes, su efecto, por ejemplo, en el cálculo de los valores DOP puede ser significativo, por lo que se debe recomendar no hacer predicciones más allá de una semana.

3. EL COMPLEMENTO GNSSplanning

3.1. ENTORNO DE TRABAJO

La captura de geodatos (datos con posicionamiento en el espacio) a través de receptores GNSS y su posterior representación y análisis en entornos SIG, es una constante en todo tipo de proyectos de múltiples disciplinas. La mayoría de programas SIG ya incorporan herramientas para trabajar con datos GNSS, ya sea conectándose directamente con el receptor e interpretando los observables en “crudo” o incorporando los datos una vez hayan sido procesados. Pero la fusión de estos dos procesos no está todavía del todo desarrollada.

El principal motivo para desarrollar esta aplicación en un entorno SIG es poder aprovechar parte del gran potencial en análisis geoespacial que nos ofrece este entorno. No sólo se pretende crear una herramienta útil ya por sí misma, sino que además sea capaz de interactuar con información geográfica en formato vectorial o raster.

3.1.1. QGIS: EL ENTORNO SIG

QGIS es una herramienta de análisis de información que se autodefine como “*Un Sistema de Información Geográfica libre y de Código Abierto*” (sic). Está disponible en [1] y es compatible en múltiples plataformas: GNU/Linux®, Unix®, Mac® OS, Microsoft® Windows y, según se anuncia en su portal web, próximamente estará disponible para Android.

Si bien en el mercado de herramientas SIG de pago, los productos de la compañía ESRI® [33] (ArcGIS, ArcPad, etc.) se encuentran encabezando el mercado de una forma dominante, su homólogo en el ámbito de las herramientas de código libre es, sin lugar a dudas, QGIS. Es una apuesta de OSGEO [34] que lo adoptó como software propio desde que se liberó sus primera versión en 2009.

Opera bajo la licencia GNU GPL por lo cual puede ser modificado libremente, lo que permite una gran adaptabilidad y personalización de cara al usuario. Está desarrollado en C++ y basa su interfaz gráfica de usuario (GUI) sobre la biblioteca Qt [35]. Además, permite la integración de complementos satélites al software principal desarrollados por los mismos usuarios. Estos complementos también se pueden crear en C++ o en Python con la misma GUI de Qt.

3.1.2. PYTHON: EL ENTORNO DE PROGRAMACIÓN

Python es un lenguaje de programación de código abierto que permite una integración fácil y efectiva entre diferentes sistemas. Actualmente se utiliza en múltiples disciplinas como en desarrollo de herramientas web y soluciones de Internet, análisis de datos, cálculos matemáticos y científicos, implementación de protocolos, interfaces de usuario o en gestión de bases de datos.

De hecho, es un lenguaje de programación ampliamente utilizado en los SIG que gestionan grandes volúmenes de datos (alfanuméricos y gráficos). Python goza de grandes ventajas que lo hacen idóneo para estos entornos:

- es un lenguaje de programación orientado a objetos.
- está definido por una sintaxis simple y eficiente, fácil de leer y de interpretar.
- es simple y rápido de aprender, comparable a otros lenguajes como C ++ o Visual Basic.
- es un lenguaje gratuito y de código abierto, que permite a la activa comunidad que tiene detrás enriquecerlo constantemente.
- se puede utilizar en múltiples plataformas (Windows®, Mac®, Linux®).
- no necesita de una compilación para ser ejecutado.
- es extensible, puede enriquecerse de la gran variedad de librerías de que dispone, facilitando así las operaciones más comunes y repetitivas.

Además, es un lenguaje excelente para generar scripts. Un script es un programa que permite conectar varios componentes existentes para crear uno nuevo y similar.

En este sentido, el desarrollo de la aplicación GNSSplanning se basa en el aprovechamiento de las aportaciones que ofrece la extensa comunidad de desarrolladores de código Python, con múltiples paquetes de funcionalidades liberalizados para ser reutilizados en nuevos proyectos. Esta filosofía de trabajo colaborativo facilita enormemente el desarrollo de nuevas aplicaciones ya que el desarrollador se sirve de funciones básicas ya programadas para construir su aplicación, sin tener que reescribir código ya existente. Al igual que para la mayoría de lenguajes de programación, existen paquetes de código reutilizables. Por ejemplo, el paquete de código `math`, nos permite hacer cálculos matemáticos más o menos complejos (matriciales, trigonométricos, etc.) de forma fácil sólo con importar sus módulos a nuestro script.

Pero la comunidad de Python no sólo ofrece paquetes de código de funcionalidades básicas sino que podemos encontrar infinidad de proyectos con licencia libre para ser reutilizado. Se puede encontrar, por ejemplo, varios proyectos dedicados a temas astronómicos y orbitales como PyEphem [36].

Se pretende que finalmente, una vez validada la aplicación GNSSPlanning, ésta también se publique de forma libre para ser reutilizado para quien le seas de interés.

3.1.3. HERRAMIENTAS PARA CREAR EL COMPLEMENTO

Vamos a realizar un breve inventario de las herramientas utilizadas para la creación del complemento GNSSPlanning.

- Qt: Desarrollada por la compañía Qt Company [35] es un entorno para la creación de interfaces gráficas de usuarios disponible con licencia GPL v3 y LGPL v2 [37]. Está construido sobre bibliotecas programadas en C++ pero también dispone de las librerías equivalentes en Python, pyQt4 [38], lo que facilita enormemente la integración del código de la parte GUI con el código puramente de la aplicación.
- PyScripter: Aunque QGIS incorpora una consola de Python para ejecutar procesos directamente, para mayor comodidad se ha optado por crear el código en un IDE (del inglés, *Integrated Development Environment*) de código libre con funcionalidades específicas para este lenguaje y que facilitan su escritura y depuración.
- Plugin Builder [39]: Uno de los muchos complementos que incorpora QGIS de serie es este *plugin* que sirve para, precisamente, crear otro *plugin*. Permite la implementación de la estructura de forma controlada, garantizando la funcionalidad de los archivos que se necesitan.

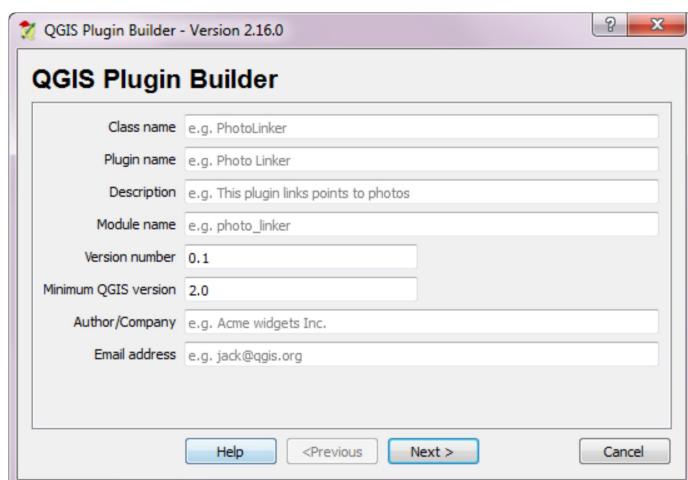


Figura 27. Plugin Builder

- Plugin Reloader [40]: Otro de los complementos que nos ofrece QGIS es el Plugin Reload muy útil cuando se está en la fase de testeo del propio *plugin*. Permite, sin tener que reiniciar QGIS, cargar las modificaciones que se hayan hecho en el de código y actualizar el complemento.

3.1.4. LIBRERÍAS DE PYTHON INTEGRADAS EN GNSSplanning

A parte del código creado específicamente para el complemento, es necesario que se carguen en los módulos distintas librerías que son imprescindibles tanto para obtener los mecanismos de interoperabilidad con el entorno de trabajo, como para ejecutar procesos complementarios.

- QGIS API [41]: cualquier complemento creado con Python, necesita de los módulos de esta librería para interactuar con las instancias de QGIS. Se han utilizado para integrar GNSSplanning dentro de QGIS y para desarrollar las funciones que interactúan con este.
- PyQt4 (QtCore, QtGui) [38]: la GUI del *plugin* está desarrollada sobre esta librería Python para implementar todas las funcionalidades que ofrece Qt.
- sgp4 1.4 [30]: esta librería desarrollada por Brandon Rhodes (Copyright © 2012–2016), está basado en las ecuaciones propuestas en [12]. Se ha utilizado para la propagación de la posición de los satélites, tal y como se ha explicado en apartados anteriores.
- Matplotlib [29]: este conjunto de librerías de código Python se centra en la representación sobre gráficas de todo tipo de datos. Se ha utilizado para representar en un gráfico polar la posición de los satélites sobre el horizonte del observador.
- grass [42]: esta no es una librería propia de Python, sino una herramienta de procesado que incorpora QGIS en su versión extendida con GRASS7 para trabajar con procesos sobre elementos vectoriales y raster. Se ha utilizado en parte del proceso para generar los perfiles de horizonte.

3.2. INTERFAZ GRÁFICA DEL USUARIO

3.2.1. DISEÑO DE LA INTERFAZ GRÁFICA

Pretendiendo que la interacción con el usuario sea lo más sencilla e intuitiva posible, se ha intentado crear una interfaz gráfica del usuario (GUI) clara, ordenada y fácil de interpretar. El usuario no debe

perder ni un instante en saber cómo manejar el complemento. Su diseño se ha pensado para que, con un solo vistazo, y sin ningún tipo de ayuda, el usuario sepa que datos debe o puede introducir y dónde se mostrarán los resultados de su consulta.

La GUI se ha desarrollado sobre Qt con librerías PyQt4 y se compone de una sola ventana que queda definida en el archivo `gnss_planning_dialog_base.ui`.

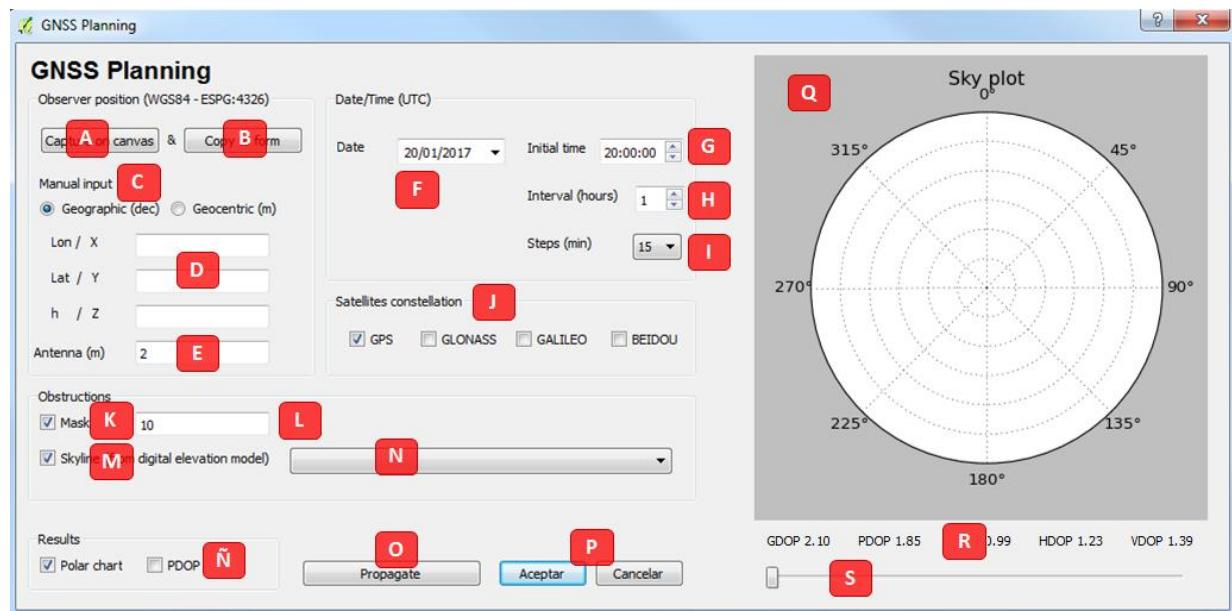


Figura 28. Interfaz gráfica del usuario.

Esta ventana principal se puede subdividir en tres partes: inserción, activación y resultados.

Siguiendo un orden lógico de escritura occidental, la primera parte con la que el usuario va a interactuar es con la inserción de datos. Esta parte se compone a su vez de:

- *Observer position (WGS84-ESPG:4326)*: en este apartado el usuario debe introducir la posición del observador y de la altura de la antena de su receptor. El usuario tiene dos opciones para introducir su posición:
 - Capturando en el espacio de trabajo de QGIS: con el botón **A**, se ejecuta la función que permite al usuario interactuar con el espacio gráfico de QGIS. Con el segundo botón **B** se copian estas coordenadas a las líneas de edición **D**.
 - Manualmente: mediante dos botones commutables **C** se le permite escoger qué tipo de coordenadas va a introducir: geográficas o geodésicas. El espacio para escribir las son las tres líneas de edición **D**. También puede introducir la altura de la antena del receptor en otra línea de edición **E**.

- *Date/Time*: el usuario puede introducir en este apartado todo lo referente a las épocas con las que quiere trabajar:
 - *Date F*: permite seleccionar la fecha de la primera época que se analizará.
 - *Time G*: permite seleccionar la hora en UTC de la primera época que se analizará.
 - *Interval H*: permite seleccionar el intervalo de tiempo (en horas) desde la primera época a analizar hasta la última época. Si su valor es 0 sólo se analiza la primera época.
 - *Steps I*: permite seleccionar el intervalo de tiempo entre una época y la siguiente. Se puede escoger entre 15, 30, 60 minutos.
- *Satellites constellation J*: el usuario puede decidir qué constelaciones va a usar para sus observaciones y seleccionarlas en este apartado activando o desactivando los botones de chequeo.
- *Obstructions*: el usuario puede decidir incorporar algunas limitaciones que limiten la visibilidad del cielo para descartar satélites.
 - *Mask*: permite definir una máscara uniforme sobre el horizonte para descartar satélites situados en una altura baja. Se activa mediante un botón de chequeo **K** y se introduce el ángulo de elevación de la máscara en grados en una línea de edición **L**.
 - *Skyline*: permite incorporar la variable terreno al análisis. Se activa con el botón de chequeo **M**. El usuario deberá informar mediante el desplegable **N** del MDT sobre el que se va a trabajar. Este MDT debe cargarse anteriormente como una capa válida raster en el entorno QGIS.
- *Results N*: el usuario puede seleccionar que resultados quiere mostrar, el gráfico polar y/o los valores de DOP. Por defecto están todas las opciones activadas.

La segunda parte de la GUI, la de activación, se compone de tres botones de acción:

- *Propagate O*: es el botón de activación que inicia el proceso de propagación.
- Aceptar/Cancelar **P**: los dos botones permiten salir del complemento sin más.

En la tercera parte se muestran los resultados y permite navegar por las distintas épocas posteriores a la inicial.

- **SkyPlot Q:** este gráfico polar es dónde se muestra la posición de los satélites.
- **DOP R:** en esta línea de etiquetas se muestran los resultados de los distintos DOP calculados para los satélites que se muestran en el *SkyPlot*.
- **EpochLine S:** esta línea de tiempo o de procesos permite al usuario ver los instantes intermedios entre la época inicial y la época final.

3.3. FUNCIONAMIENTO DEL COMPLEMENTO

Nota: GNSSPlanning se ha creado en una plataforma Windows®, para QGIS 2.18 (Las Palmas de G.C) y con las versiones de las librerías y herramientas expuestas en esta memoria. No se ha testeado en otras plataformas ni en otras versiones por lo que no se puede garantizar su correcto funcionamiento si se trabaja fuera del entorno definido.

3.3.1. INSTALACIÓN DEL COMPLEMENTO

La instalación de un complemento de QGIS se hace mediante la opción *Administrar e instalar complementos...* del menú *Complementos*, donde se ofrece un listado de los complementos existentes tanto los ya cargados en nuestro terminal como los disponibles para ser instalados.

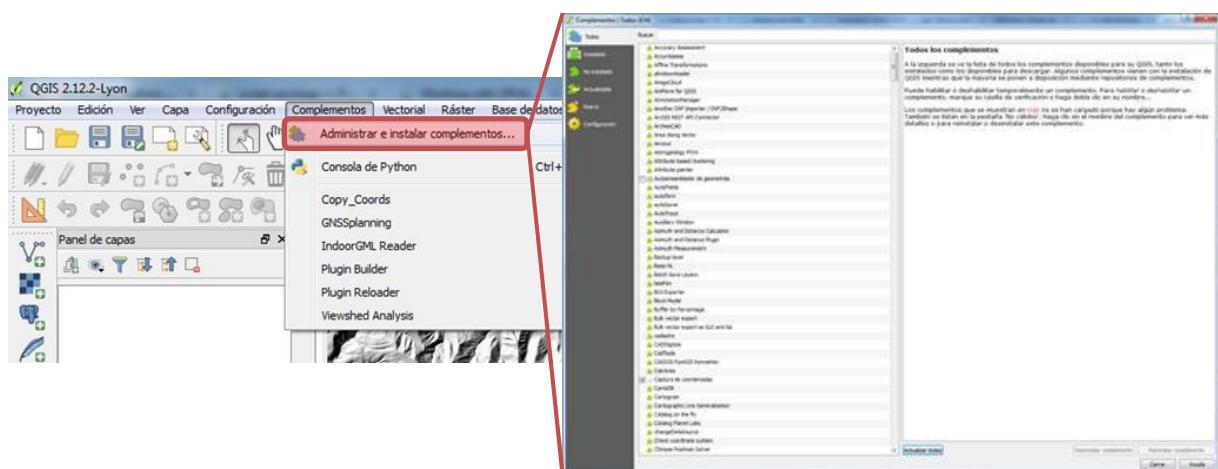


Figura 29. Administrador de complementos de QGIS

Como, por ahora, GNSSPlanning no se encuentra en el directorio oficial de los complementos de QGIS, tendremos que instalarlo manualmente. El proceso es muy sencillo, basta con copiar y descomprimir la carpeta madre del complemento en el directorio:

- UNIX/Mac⁴: ~./qgis/python/plugins
- Windows: ~./qgis/python/plugins

Dónde ~/ normalmente corresponde a C:\Users\ (user)

Al reiniciar QGIS, si volvemos a acceder al administrador de complementos, GNSSPlanning debe aparecer en el listado de instalados:

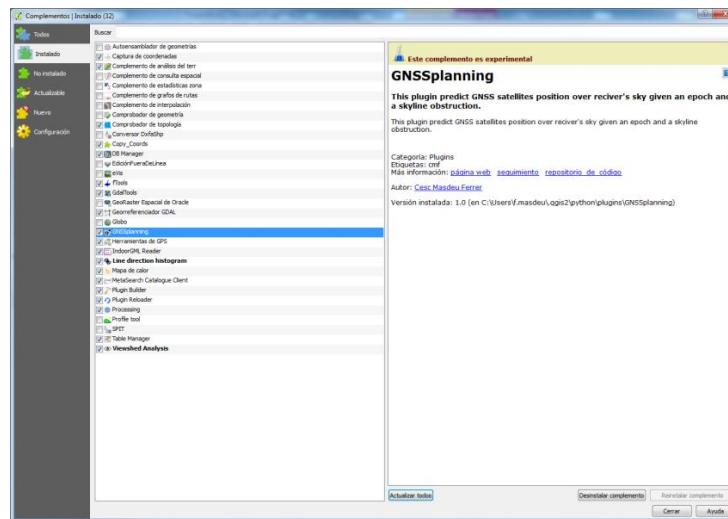


Figura 30. GNSSPlanning en el administrador de complementos de QGIS.

También aparecerá en el menú desplegable de complementos y como un nuevo botón el panel de herramientas.

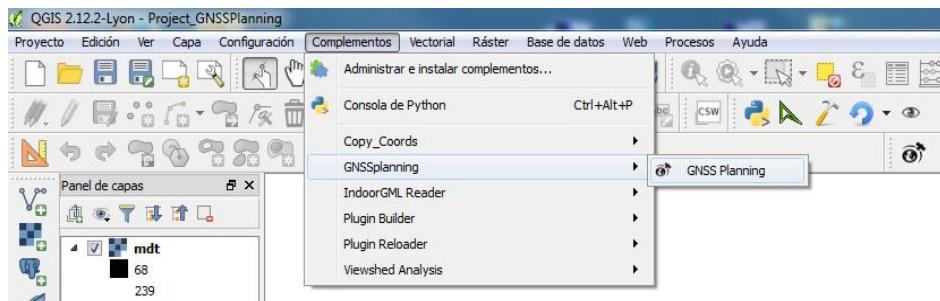


Figura 31. GNSSPlanning en el menú de tareas de QGIS.

⁴ El complemento GNSSPlanning no se ha testeado para estos dos sistemas operativos.

3.3.2. UTILIZACIÓN DEL COMPLEMENTO

Podemos iniciar GNSSPlanning accediendo por el menú desplegable de complementos o por el icono del panel de herramientas. Lo primero que veremos es que se inicia la ventana principal, pero cuando la aplicación se activa, lo primero que hace es cargar el módulo *Import_TLE*, accediendo a los enlaces web de donde recoger los TLE e incorporar los datos en los ficheros locales, tal y como se explicará en el apartado de definición de este módulo.

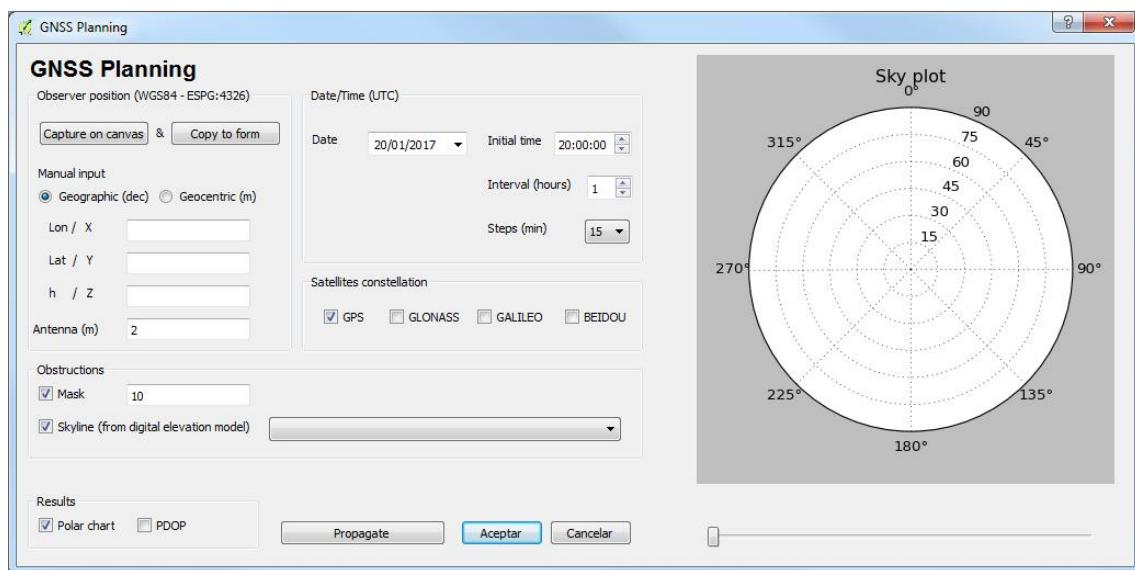


Figura 32. Ventana principal de GNSSplanning.

NOTA: situando el puntero del ratón encima de cada elemento de la ventana principal aparece una etiqueta explicativa de la función de cada uno.

El primer paso es introducir los datos de la posición del usuario. Como se explicó en el apartado de la Interfaz gráfica del usuario, para ello tenemos dos opciones:

- podemos capturar las coordenadas directamente desde el espacio gráfico de trabajo de QGIS mediante el botón *Capture on canvas* y seleccionando sobre el mapa el punto deseado: mediante un clic con el botón izquierdo del ratón sobre el lienzo de trabajo de QGIS se captura la posición con el sistema referencia de coordenadas que esté activado por defecto en ese momento. En el módulo *Capture_coords* se transforman automáticamente (si fuese necesario) esas coordenadas a geográficas WGS84 (ESPG:4326). Para pasar estas coordenadas al

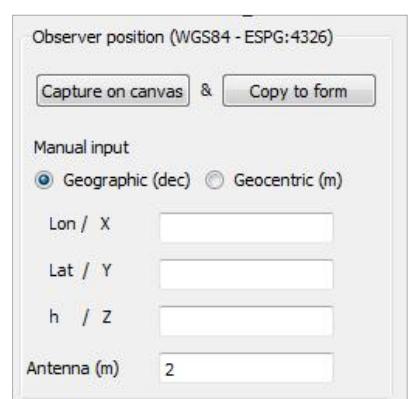


Figura 33. Insertar posición del observador.

formulario es necesario pulsar el botón *Copy to form*.

- Podemos insertar las coordenadas manualmente, sin más, en las líneas de edición. Con esta opción se nos deja escoger si queremos insertarlas como geográficas WGS84 o como coordenadas geocéntricas.

Si insertamos las coordenadas manualmente, podemos escoger el tipo de coordenadas entre geográficas o geodésicas. Si las capturamos en pantalla, automáticamente se transforman a coordenadas geográficas WGS84, sea cual sea el sistema de coordenadas que se esté utilizando. Se da la opción de introducir la altura de la antena.

El segundo paso es introducir los datos de las épocas para las que vamos a calcular la posición de los satélites, seleccionando la fecha de inicio en el calendario *Date*, la hora inicial en *Initial Time*, el intervalo de tiempo desde la primera a la última época en *Interval* y cada cuanto queremos propagar la posición, en *Steps*.

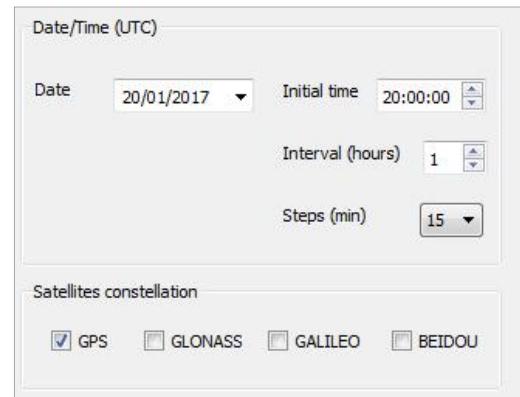


Figura 34. Insertar primera época.

Definidos los parámetros de tiempo, podemos seleccionar que constelaciones queremos utilizar.

Estos son los parámetros imprescindibles que necesitamos definir antes de poder ejecutar la aplicación. Los parámetros de obstrucción son opcionales, aunque muy recomendables e interesantes en su influencia con el resultado.

En el apartado de obstrucciones podemos seleccionar la máscara de horizonte que queremos aplicar. Esta máscara es constante para todo el horizonte. Pero si queremos saber exactamente la obstrucción del horizonte real podemos incorporar un modelo digital del terreno para determinarlo.

Seleccionamos la opción de *SkyLine* y seleccionamos en el desplegable el MDT que deberíamos haber cargado anteriormente en QGIS.

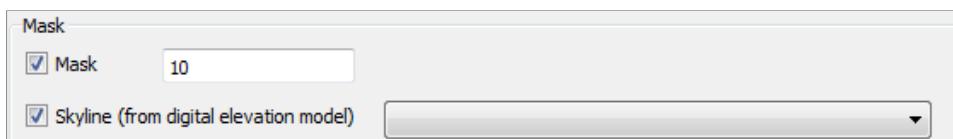


Figura 35. Insertar obstrucciones.

Seleccionados todos los parámetros necesarios, podemos ya ejecutar la aplicación pulsando el botón *Propagate*. Con la activación de este botón el programa lee todos los inputs que el usuario ha

introducido, calcula la posición de los satélites escogidos para la primera época y los muestra en el *SkyPlot* si no están ocultos por las obstrucciones seleccionadas. Los procesos que se ejecutan se puede consultar en el diagrama de flujos del módulo *gnss_planning* (módulo principal) en el anexo 1.

El resultado se ofrece en la parte derecha de la ventana principal en forma de gráfico polar, para la representación de los satélites en el cielo, y en forma de línea de texto, para los DOP calculados. Los satélites aparecen en forma de etiquetas con los nombres de cada uno y las obstrucciones en forma de líneas. Para cada época el gráfico se actualiza apareciendo el nuevo estado.

El primer resultado corresponde a la época inicial:

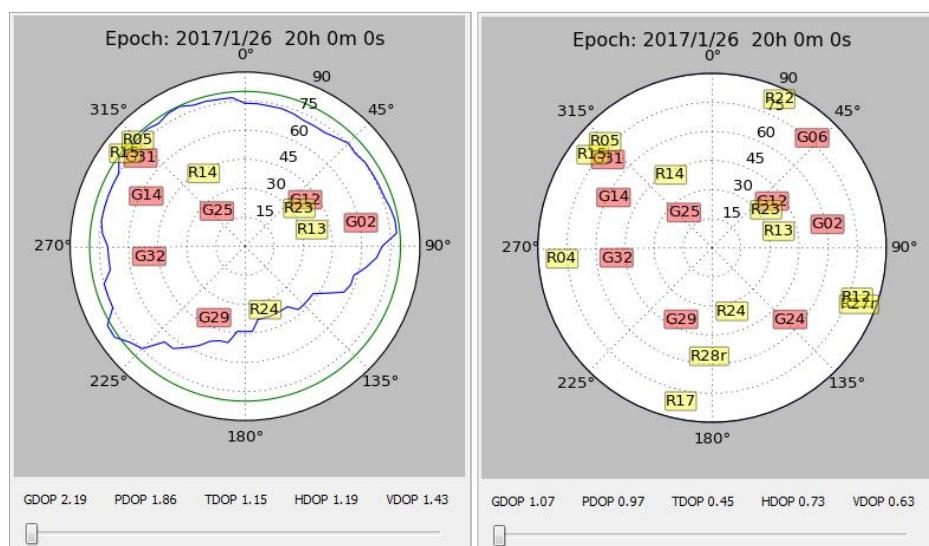


Figura 36. Resultados con y sin obstrucciones.

Nótese la diferencia de los valores del DOP en las dos figuras anteriores y la importancia de utilizar obstrucciones en nuestro estudio de planificación para reflejar lo más fiel posible el escenario que nos vamos a encontrar realmente durante los trabajos de campo.

Con la línea de tiempo situada debajo de los resultados del DOP podemos avanzar o retroceder en las épocas hasta el final del intervalo que hayamos seleccionado, incrementando un paso (*step*) cada vez. Se puede interactuar con el ratón o con las teclas de avance y retroceso del mismo teclado. Cada vez que se ejecuta un paso (hacia adelante o hacia atrás) la aplicación vuelve a calcular la posición de los satélites, los muestra en el *SkyPlot* y calcula de nuevo los DOP. La diferencia entre la línea de épocas y el botón de activación *Propagate*, radica en que, en el primer caso, no se vuelve a leer los inputs del usuario (posición, fechas, constelaciones u obstrucciones).

A continuación se muestra la secuencia las 4 siguientes épocas, habiendo seleccionado como datos de entrada: intervalo de 1h con propagaciones cada 15 minutos; las constelaciones GPS y GLONASS; una máscara de 10º (línea verde) y la línea de horizonte sobre un MDT (línea azul).

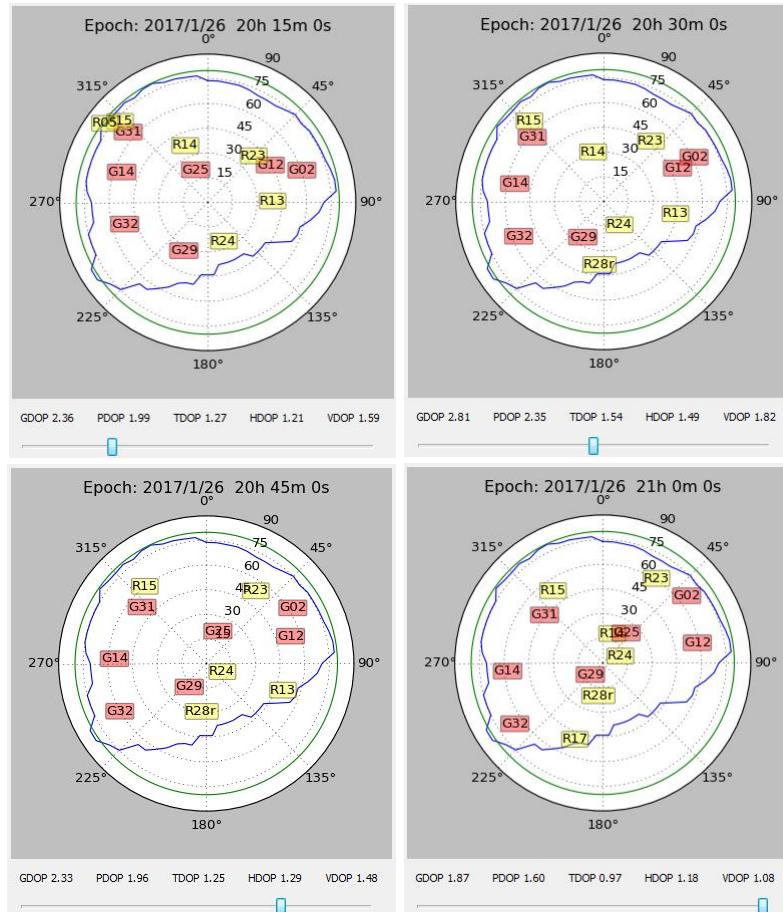


Figura 37. Secuencia de resultados.

3.4. MÓDULOS Y FUNCIONES PRINCIPALES

En este capítulo se van a describir los módulos que componen el complemento *GNSSplanning* y sus principales funciones. Se puede ver el código correspondiente en el anexo 2.

3.4.1. ESTRUCTURA DE FICHEROS

Todos los módulos se localizan dentro del directorio principal *GNSSplanning* que debe ubicarse en el directorio *.qgis2/plugins*, tal y como se muestra en la figura 38.

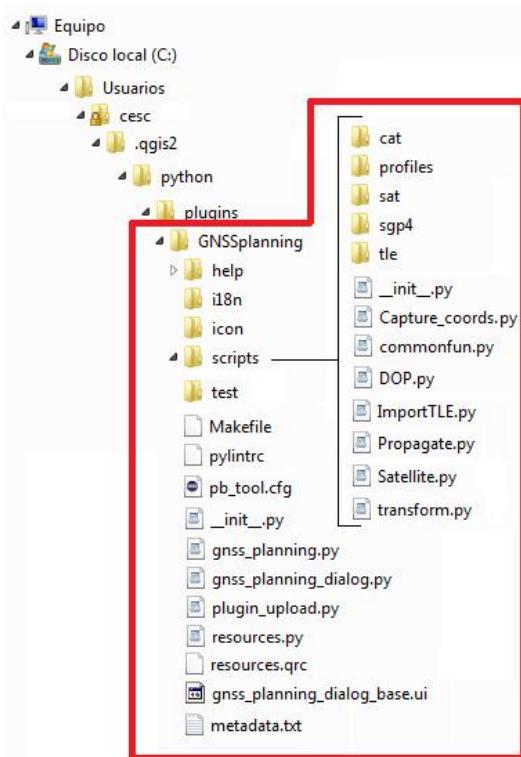


Figura 38. Estructura de directorios del complemento *GNSSplanning*.

Dentro del directorio principal se ubican varios ficheros y subdirectorios. Algunos de ellos, como el directorio *test* o el módulo *resources.py* son propios de los complementos genéricos para QGIS y no se van a describir en este capítulo. Se puede conocer más sobre ellos en [1] y [38].

En el directorio *GNSSplanning* se encuentra el módulo principal *gnss_planning.py* y el subdirectorio *scripts* donde se localizan los módulos secundarios y las carpetas donde, por ejemplo, se guardan los ficheros TLE descargados o se guarda el fichero *horizon.txt* con los azimuts y alturas del perfil de horizonte del observador.

3.4.2. DESCRIPCIÓN DE LOS MÓDULOS Y DE LAS FUNCIONES

| | |
|----------------------------------|------------------------------|
| 3.4.2.1. gnss_planning.py | GNSSplanning\gnss_plannig.py |
|----------------------------------|------------------------------|

Este es el módulo principal del complemento donde se ejecutan la mayoría de las funciones y donde se hacen las llamadas a los otros módulos para que hagan las funciones complementarias.

Funciones genéricas:

Además de las funciones propias del complemento, incluye funciones genéricas que contienen todos los complementos de QGIS, como por ejemplo, las encargadas de cargar el mismo complemento en el entorno de QGIS.

- Importaciones:

Como en la mayoría de scripts de Python, para ejecutar el código de este módulo es necesario importar las librerías que permiten la interacción con otros objetos, elementos o módulos predefinidos de Python.

En el módulo `gnss_planing.py` se importan un total de 27 librerías entre las que se encuentran los elementos para interactuar con QGIS (`PyQt4`, `qgis`), los módulos propios de Python (`math`, `numpy`, `os`, `sys`, etc.) y todos los submódulos programados para esta complemento (`Propagate`, `transform`, `DOP`, `Capture_coords`, etc.), que a su vez, también importan las librerías que ellos necesitan.

- Carga y descarga del complemento a QGIS:

Otra de las funcionalidades genéricas es la que definen cómo insertar los botones del complemento en la interfaz de QGIS así como su registro en el menú contextual. Este código se genera automáticamente en el proceso de creación desde la estructura del complemento con el `Plugin Builder`. Se ha modificado para adaptarlo a las necesidades del complemento `GNSSplaning`.

Las funciones `initGui` y `add_action` definen distintos parámetros genéricos del complemento, como dónde se encuentra la imagen del ícono, el texto que aparecerá en el menú y el comportamiento del *plugin* al ser ejecutado consecutivamente en una misma sesión de QGIS [43].

La función `unload` define las funcionalidades para desinstalar al complemento de su instancia de ejecución cuando se decide sacar el complemento del entorno de trabajo.

- Constructor y ejecución del complemento.

La implementación del complemento se hace mediante la clase `GNSSplanning` que, como todas las implementaciones en Python, dispone de un método constructor `__init__`. En él, se define la comunicación con la instancia de ejecución de QGIS (`dlg`) y con la interfaz del complemento (`iface`).

Dentro del método constructor también se define el comportamiento de los botones de ejecución del complemento, con los que el usuario interactúa en la interfaz. Por ejemplo se define qué debe hacer el programa cuando se pulsa el botón *Propagate*. En realidad, aquí el programa sólo llama a las funciones encargadas de ejecutar los procesos, para que sean ellas las que realmente actúen.

Una de estas llamadas es la que inicia el módulo `ImportTLE.py`. Este módulo se ejecuta automáticamente cada vez que el usuario inicia el complemento y es el que se encarga de descargar la información de los satélites procedente de los TLE, habilitados en el web de Celestrak. Este módulo se explicará con más detalle más adelante.

Funciones propias:

Las funciones propias del módulo `gnss_planning.py` corresponden a las programadas íntegramente para este complemento. En el anexo 1 se puede ver el diagrama de flujo por el que se rigen estas funciones.

- Funciones de activación:

Una vez ejecutado el módulo `ImportTLE.py` y abierta la GUI, el usuario puede activar las funciones `get_on_canvas` y `copy_on_form` con los botones *Select on canvas* y *Copy form*, respectivamente. Son dependientes una de otra y permiten al usuario interactuar con el lienzo de trabajo de QGIS. Mediante la utilización del ratón, se puede seleccionar sobre un mapa u ortofotografía el punto donde se ubica el observador y traspasar las coordenadas en el formulario de la GUI. La función `get_on_canvas` ejecuta una llamada al módulo `Capture_coords.py` donde realmente se captura la información. Este módulo se explicará con más detalle más adelante.

La función, `run_first_propagate` se activa con el botón *Propagate* de la GUI y con ella se ejecuta las funciones: `read_parameters`, `observer_position`, `get_epochs` y `run_propagate`. La función `run_first_propagate` sólo muestra los resultados para la primera época.

Para mostrar los resultados de las restantes épocas el usuario debe interactuar con la línea de tiempo. Al avanzar o retroceder la línea de tiempo se ejecuta directamente la función `run_propagate`.

- Leer datos introducidos por el usuario (activadas con la función `run_first_propagate`):

La función, `read_parameters`, lee todos los datos introducidos por el usuario y los guarda en variables que utilizan otras funciones o módulos. En esta función se evalúa la consistencia y la completitud de los datos, devolviendo un valor de falso o verdadero según se cumplan o no los formatos preestablecidos, o si faltan o no datos en los formularios. Si la evaluación es negativa, se muestra un mensaje en pantalla avisando al usuario del problema y no se prosigue con el proceso.

La función `observer_position` transforma las coordenadas de la posición del observador: si se han definido como cartesianas geocéntricas, se transforman a geográficas, y viceversa. Esta transformación se hace llamando a las funciones `geographic_to_ITRF` y `ITRF_to_geographic` del módulo `transform.py`.

A continuación se ejecuta la función `get_epoch`. En ella se definen todas las épocas para las que se van a calcular las posiciones de los satélites. Cada época consta de una fecha y una hora. También se define los parámetros de la barra de tiempo por donde el usuario va a poder interactuar para pasar de una época a la siguiente o a la anterior.

Por último, la función `horizon` ejecuta la función de la librería `grass` [42]. Esta aplicación extrae la línea de horizonte definida sobre el modelo digital del terreno que el usuario habrá introducido en la GUI. El resultado de esta función es un fichero de texto (`./scripts/profiles/.horizon.txt`) con los acimuts y las alturas sobre el horizonte de los puntos que lo definen visto desde la posición del observador. Se calcula un punto cada 5 grados de circunferencia desde el Norte y en sentido horario, hasta completar los 360 grados. Se ha considerado para este proyecto que la definición de puntos de horizonte cada 5 grados es suficiente para conseguir un buen equilibrio entre precisión y eficacia computacional.

- Propagar:

Una vez introducidos todos los datos y calculado las variables que se derivan de ellos, se ejecuta el proceso de cálculo de la posición de los satélites mediante la función `run_propagate`. Esta función, a su vez llama a ejecutar las funciones de los módulos `Propagate.py` y `DOP.py`, que se describen más adelante. Para los resultados devueltos por el módulo `DOP.py`, esta función crea una línea de texto en la GUI donde muestra los valores DOP.

- Presentar resultados en un gráfico polar:

Los resultados de los procesos ejecutados en la función `run_propagate`, se muestran en la GUI mediante la función `plot` donde se definen los parámetros de los objetos que se mostrarán en el gráfico polar.

Se identifica cada satélite mediante una llamada a la clase `Satellite.py` y se muestra en el gráfico (según sus coordenadas cartesianas horizontales) en forma de etiquetas con el nombre del satélite y de un color distinto según la constelación a la que pertenezca. La máscara se representa con un círculo concéntrico al cenit y la línea de horizonte con una línea continua que une los puntos calculados en la función `horizon`.

3.4.2.2. ImportTLE.py

GNSSplanning\scripts\ImportTLE.py

Este es el primer módulo que se ejecuta al iniciar el complemento. Para su correcto funcionamiento se necesita tener conexión a Internet, por lo que si no se consigue acceder a los enlaces predefinidos, se presenta una advertencia al usuario avisando de que no se han podido actualizar los datos de los ficheros TLE.

Para cada grupo de constelaciones predefinidas para este complemento (GPS, GLONASS, GALILEO y BEIDOU) se accede al enlace correspondiente de CelesTrak donde se almacenan los ficheros TLE. Cada uno de estos ficheros recoge información de todos los satélites de una misma constelación, por lo que al final se descargan cuatro ficheros que se almacenaran el subdirectorio `tle`. También se crean cuatro nuevos ficheros (con extensión .cat) que se guardan en el subdirectorio `cat`. Los ficheros .cat contienen un listado de los satélites disponibles para cada constelación.

El módulo lee cada uno de estos ficheros TLE almacenados y crea, en el subdirectorio *sat*, y para cada uno de los satélites, un nuevo archivo (con extensión *.sat*) donde guarda los parámetros orbitales procedentes de los TLE así como datos adicionales procedentes de un archivo *.csv* almacenado en el mismo subdirectorio.

La clase *Satellite_calss*, del módulo *Satellite.py*, accede a los ficheros *.sat* y *.cat* para definir los parámetros de cada objeto-satélite.

| | |
|------------------------------|--|
| 3.4.2.3. Satellite.py | GNSSplanning\scripts\Satellite.py |
|------------------------------|--|

Este módulo define la clase *Satellite_class* como un objeto de Python, con las características predefinidas de un satélite, en un instante concreto. Se puede crear un objeto *Satellite_class* desde cualquier módulo del programa sólo con llamar a esta clase adjuntando el nombre del satélite sobre el que se quiera trabajar (el nombre corresponde al número de satélite según el catálogo de NORAD).

Esta clase se compone de distintas funciones.

- Características del satélite:

La función *getSatParameters* es la encargada de leer el fichero *.sat* del satélite correspondiente y dar valor a las variables que definen sus características. Entre estas características se encuentran las propias del vehículo espacial (la constelación a la que pertenece, el bloque, su número PRN, etc.) o las que aportan las efemérides de los TLE (la época de las efemérides, la inclinación, la excentricidad de la órbita, el movimiento medio, etc.).

Se puede acceder a cualquiera de ellas mediante una instancia al objeto. Por ejemplo, si se ha definido el objeto *Satellite_class* como:

```
NORADnum = 29601          # corresponde al satélite G12 (GPS)
ve = Satellite_class(NORADnum) # se crea el objeto
ve.getSatParameters()        # se activa la función
```

si le pedimos la inclinación:

`ve.inclination`

el módulo nos devuelve el valor:

55.00

- Posición del satélite:

Se han programado cuatro funciones para obtener la posición de los satélites en distintos sistemas de coordenadas.

La función `getPosition_TEME` ejecuta el módulo `propagation.py` de la librería `sgp4` (Brandon Rhodes, 2012-2016) que determina la posición de los satélites, dando como parámetros de entrada los datos TLE y una época. Como resultado se obtienen las coordenadas en el marco TEME (marco ECI utilizado para los formatos de transferencia de datos orbitales *Two-line elements* de NORAD).

La función `getPosition_ITRF` recupera esas coordenadas en el marco TEME y ejecuta la función `TEME_to_ITRF` del módulo `transform.py`. Este, por su parte, transforma las coordenadas a cartesianas geocéntricas en un marco ITRF. Las ecuaciones para esta transformación se han adaptado de las procedentes de las librerías de `python-skyfield` (Brandon Rhodes, 2012-2016).

La función `getPosition_geo` ejecuta la función `ITRF_to_geographic` del módulo `transform.py`, que transforma las coordenadas cartesianas geocéntricas en un marco ITRF a geodésicas WGS84.

La función `getPosition_azalt` ejecuta la función `ITRF_to_horizontal` del módulo `transform.py`. Este, transforma las coordenadas cartesianas geocéntricas en un marco ITRF a coordenadas horizontales (acimut y altura), dadas las coordenadas geocéntricas (cartesianas y geográficas) de la posición del observador.

Las coordenadas de la posición del satélite son tratadas, en realidad, como características de cada satélite, pues un objeto `Satellite_class` se refiere a un solo satélite y a una época concreta. Así, se puede acceder a cualquiera de ellas instando a la clase y a la función que les da valor. Por ejemplo, dadas las coordenadas del observador y una fecha:

```
NORADnum = 29601 # corresponde al satélite G12 (GPS)
geo_observer = [1.40113711458437, 41.5996152901421, 853.37078135088]
```

```
XYZ_observer = [4775849.592, 116814.09, 4213018.694]
epoch = [2017, 01, 29, 20, 00, 0.0]

ve = Satellite_class(NORADnum) #se crea el objeto
ve.getPosition_azalt(self, geo_observer, XYZ_observer, epoch) #se activa la
función
```

si le pedimos las coordenadas horizontales:

```
ve.az, ve.alt
```

el módulo nos devuelve los valores:

```
325.0, 26.2
```

| | |
|------------------------------|--|
| 3.4.2.4. Propagate.py | GNSSplanning\scripts\Propagate.py |
|------------------------------|--|

Este módulo consta de sólo una función, `propagate`, que requiere como parámetros de entrada: las coordenadas del observador, la época (fecha y hora) sobre la que se va a propagar, el número de época (del total a calcular), el valor de la máscara, un valor booleano de si se va a usar máscara o no y las constelaciones que se quieren usar.

En primer lugar, y si se ha seleccionado un modelo digital del terreno para definir una línea de horizonte, se crear una *array* de 2×72 ($72 = 360^\circ / 5^\circ$), con los valores acimut y altura para cada columna. Estos valores se obtienen del fichero `horizon.txt`, antes citado.

Seguidamente, la función crea una lista con los nombres de todos los satélites que hay en el subdirectorio `sat` (tantos como ficheros `.sat` contenga). Para cada satélite de la lista creada se procede de la siguiente forma:

- se crea un objeto `Satellite_class`.

```
ve = Satellite_class(29601)
```

- se pide a qué constelación pertenece (tal y como hemos visto en el apartado del módulo `Satellite.py`). Si el satélite pertenece a una de las constelaciones seleccionadas, el proceso continua. En caso contrario, el satélite se descarta.
- Se obtiene la posición en coordenadas horizontales

```
ve.getPosition_azalt(geo_observer, XYZ_observer, epoch)
```

- Se compara la altura del satélite con el valor de la máscara y si está por debajo, el satélite se descarta.

```
ve.alt > mask
```

- Se determina ahora qué satélites están por debajo de la línea de horizonte. Para ello, se compara su altura con el valor correspondiente de la *array* horizonte que hemos creado antes. El valor correspondiente será la altura emparejada con el azimut con valor igual a *ve.az* (azimut del objeto satélite). Como la *array* horizonte sólo contiene valores múltiplos de 5, se debe buscar para *ve.az* el valor múltiplo de 5 más cercano. Por ejemplo si *ve.az* = 212.12 el valor más cercano es 210.00. Con este valor se extrae de la matriz la altura del horizonte y se compara con la altura del satélite. Si está por debajo, el satélite se descarta.

```
ve.alt > altura horizonte
```

- Los satélites que finalmente pasan todos los filtros, se guardan en forma de dos diccionarios de Python. El primer diccionario tiene como índices el nombre de cada satélite y como valores las coordenadas cartesianas geocéntricas en ITRF. Este diccionario se utiliza para el módulo DOP.py donde se calcularán los valores DOP según la posición de los satélites visibles.

El segundo diccionario, también tiene como índices los nombres de los satélites, pero como valores se guardan las coordenadas horizontales (azimut y altura). Este diccionario se utiliza para dibujar los satélites en el gráfico polar en la GUI.

3.4.2.5. DOP.py

GNSSplanning\scripts\DO.py

El módulo DOP.py calcula los valores DOP (GDOP, PDOP, TDOP, HDOP, VDOP) para cada época, teniendo en cuenta todos los satélites visibles por encima de las obstrucciones impuestas (máscara y horizonte). Se compone de dos funciones que interactúan conjuntamente.

La función *getDOP_values* requiere como parámetros de entrada la posición del observador y el diccionario de satélites visibles con las coordenadas cartesianas geocéntricas. Según se ha explicado en capítulos anteriores, el valor mínimo de DOP se obtiene comparando los resultados de hacer

todas las combinaciones posibles de satélites visibles, partiendo de combinaciones de grupos de 4 satélites hasta hacer un solo grupo de tantos satélites como haya visibles.

Como se vio, este proceder puede llevar a tener que calcular una gran cantidad de combinaciones. Pero como también vimos, los mejores resultados se obtienen con el mayor número de satélites posibles (el volumen de la figura geométrica generada entre satélites y observador, normalmente será mayor cuanto más satélites contenga la figura). Por lo que, para mejorar la eficiencia computacional, se calcula sólo el DOP para grupos igual a n y $n-1$, siendo n el número total de satélites visibles.

Seleccionados los grupos de satélites se calcula, para cada uno de ellos, los valores DOP mediante la segunda función, `dop`. Las ecuaciones utilizadas para dicho cálculo se han definido en el apartado 1.5.4. Todos los valores DOP calculados son devueltos a la primera función donde se comparan y se selecciona el de menor valor. Este valor, es el que se mostrará en la GUI.

| | |
|--|--|
| 3.4.2.6. Capture_coords.py | GNSSplanning\scripts\Capture_coords.py |
|--|--|

Este módulo contiene la clase `CaptureCoords` implementada como una herramienta de mapa de QGIS (`QgsMapTool`). Las herramientas de mapa de QGIS son herramientas interactivas con el usuario para manipular o actuar con el lienzo del mapa. Por ejemplo, la función `zoom` se implementan como herramientas de mapa.

La clase contiene, además de su correspondiente método constructor `__init__`, la función `canvasReleaseEvent` donde se crea un evento. Este evento corresponde a la propia acción que realiza el usuario sobre el lienzo de QGIS. Cuando sucede este evento, la función recoge las coordenadas del punto seleccionado y las transforma a coordenadas geográficas WGS84 (EPSG 4326) y a coordenadas UTM 31N (EPSG 25831). Queda pendiente para la segunda versión de `GNSSplanning` el poder seleccionar los sistemas de referencia según preferencias del usuario.

Las transformaciones se realizan con funciones propias de la librería de QGIS (`qgis`) importada al principio del módulo como `QgsCoordinateTransform`.

3.4.2.7. transforms.py

GNSSplanning\scripts\transforms.py

Las transformaciones entre sistemas de coordenadas se realizan en el módulo `transform.py`. Este módulo da cobertura a distintas funciones de otros módulos cuando necesitan transformar las coordenadas de un satélite o de la posición del observador. Se compone de varias funciones:

- Transformación de sistema inercial (TEME) a sistema fijo (ITRF):

Se han adaptado funciones complementarias del código del script *python-skyfield* (Brandon Rhodes, 2016) para transformar las coordenadas inerciales (TEME), que retorna *python-sgp4* (Brandon Rhodes, 2016), a coordenadas fijas (ITRF). El resultado de esta adaptación son las funciones `rot_x`, `rot_y`, `rot_z`, `theta_GMST1982`, `julian_day`, `julian_date` y `TEME_to_ITRF`. Su funcionamiento y estructura se puede consultar en [43].

- Transformación de coordenadas geocéntricas cartesianas a geográficas y viceversa:

La funciones `ITRF_to_geographic` y `geographic_to_ITRF` transforman las coordenadas de un sistema a otro basándose en las siguientes ecuaciones [44]:

De geográficas a geocéntricas:

Siendo a el semieje mayor de la elipse, e la excentricidad y λ, φ, h las coordenadas geográficas:

$$N = \frac{a}{\sqrt{1 - e^2 \cdot \sin^2 \varphi}} \quad (64)$$

$$X = (N + h) \cdot \cos \varphi \cdot \cos \lambda \quad (65)$$

$$Y = (N + h) \cdot \cos \varphi \cdot \sin \lambda \quad (66)$$

$$Z = (N \cdot (1 - e^2) + h) \cdot \sin \varphi \quad (67)$$

De geocéntricas a geográficas:

Siendo a el semieje mayor de la elipse, e la excentricidad y X, Y, Z las coordenadas geocéntricas:

$$D = \sqrt{X^2 + Y^2} \quad (68)$$

$$\lambda = \operatorname{atan} \frac{Y}{X} \quad (70)$$

Sustituyendo (65) y (66) en (68), usando (67), se obtiene la ecuación no lineal para φ :

$$\varphi = \operatorname{atan} \frac{Z + (e^2 \cdot N \cdot \operatorname{sen} \varphi)}{D} \quad (71)$$

Aplicando el Método de Punto Fijo (para valores pequeños de e convergerá a la solución buscada), iteramos para aproximar el valor de φ , partiendo de $N=0$

$$\varphi_0 = \operatorname{atan} \frac{Z}{D} \quad (72)$$

Obteniendo la sucesión:

$$\varphi_n = \operatorname{atan} \frac{Z + (e^2 \cdot N \cdot \operatorname{sen} \varphi_{n-1})}{D} \quad (73)$$

Y finalmente, sustituyendo el valor límite obtenido mediante la iteración en las expresiones previas se obtiene:

$$N = \frac{a}{\sqrt{1 - e^2 \cdot \operatorname{sen}^2 \varphi}} \quad (74)$$

$$h = \frac{D}{(\cos \varphi)} - N \quad (75)$$

- Transformación de coordenadas geocéntricas a horizontales:

La función ITRF_to_horizontal se encarga de proporcionar la posición del satélite en coordenadas horizontales a partir de las coordenadas geocéntricas del mismo satélite y del observador. Utiliza las siguientes ecuaciones según [45]:

P_i = posición del observador.

X_i = vector P_i .

λ_i, φ_i = longitud y latitud P_i .

n_i, e_i, u_i = direcciones norte, este y arriba (up) del sistema local de coordenadas de P_i .

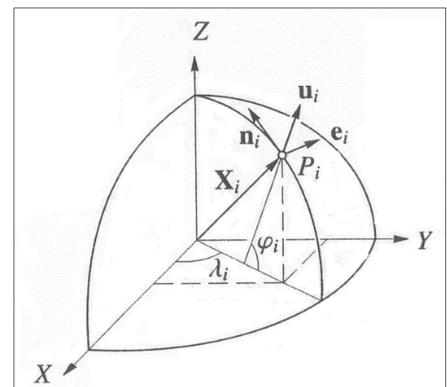


Figura 39. Coordenadas globales y locales.

Siendo P_j la posición del satélite, el vector posición observador-satélite en el sistema global corresponde a:

$$X_{ij} = \begin{pmatrix} \Delta X_{ij} \\ \Delta Y_{ij} \\ \Delta Z_{ij} \end{pmatrix} \quad (76)$$

Y en el sistema local:

$$x_{ij} = \begin{pmatrix} n_{ij} \\ e_{ij} \\ u_{ij} \end{pmatrix} = \begin{pmatrix} n_i \cdot X_{ij} \\ e_i \cdot X_{ij} \\ u_i \cdot X_{ij} \end{pmatrix} \quad (77)$$

Donde:

$$n_i = \begin{pmatrix} -\sin \varphi_i \cos \lambda_i \\ -\sin \varphi_i \sin \lambda_i \\ \cos \varphi_i \end{pmatrix}, \quad e_i = \begin{pmatrix} -\sin \lambda_i \\ -\cos \lambda_i \\ 0 \end{pmatrix}, \quad u_i = \begin{pmatrix} \cos \varphi_i \cos \lambda_i \\ \cos \varphi_i \sin \lambda_i \\ \sin \varphi_i \end{pmatrix} \quad (78) (79) (80)$$

La relación entre un sistema y otro pues, queda expresada como:

$$x_{ij} = R_i^T \cdot X_{ij} \quad (81)$$

Siendo R_i la matriz de rotación:

$$R_i = \begin{pmatrix} -\sin \varphi_i \cos \lambda_i & -\sin \lambda_i & \cos \varphi_i \cos \lambda_i \\ -\sin \varphi_i \sin \lambda_i & \cos \lambda_i & \cos \varphi_i \sin \lambda_i \\ \cos \varphi_i & 0 & \sin \varphi_i \end{pmatrix} \quad (82)$$

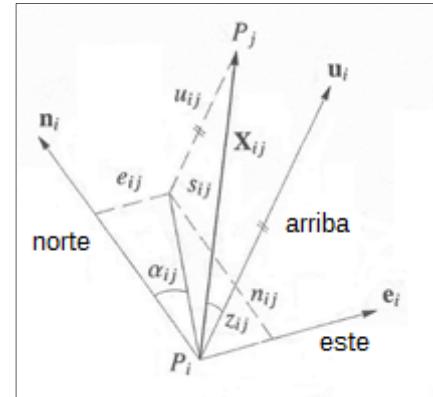


Figura 40. Sistema local.

Las coordenadas en el sistema local serán:

$$S_{ij} = \sqrt{n_{ij}^2 + e_{ij}^2 + u_{ij}^2} = \text{Distancia entre satélite y observador.} \quad (83)$$

$$\cos z_{ij} = \frac{u_{ij}}{\sqrt{n_{ij}^2 + e_{ij}^2 + u_{ij}^2}} \rightarrow \text{Altura sobre el horizonte} = 90 - Z_{ij} \quad (84)$$

$$\tan \alpha_{ij} = \frac{e_{ij}}{n_{ij}} \rightarrow \alpha = \text{Acimut} \quad (85)$$

3.5. COMPARACIÓN CON OTRAS APLICACIONES

Para comprobar los resultados que se obtienen con el complemento creado, se han comparado los valores obtenidos de DOP con los que se obtienen de otra aplicación similar existente en el mercado.

Para ello, se ha elegido la aplicación *GNSS Planning Online* de Trimble[®] [46], que ofrece funciones similares a las programadas en este proyecto para el complemento para QGIS. También permite incorporar una o varias constelaciones, incorporar una máscara o definir de forma manual (dibujando sobre el gráfico polar) una línea de horizonte. En el ejemplo, no se ha considerado esta línea de horizonte por no poder definirla exactamente igual a la que se obtiene sobre un MDT (lo que sí ocurre en nuestro complemento GNSSPlanning para QGIS).

En la aplicación de Trimble[®], los valores de DOP se muestran en un gráfico, pero no en forma de valores. Aun así, de ese gráfico se pueden extraer dichos valores fácilmente.

Se ha hecho la comparación para:

Tabla 20. Valores para el estudio de comparación.

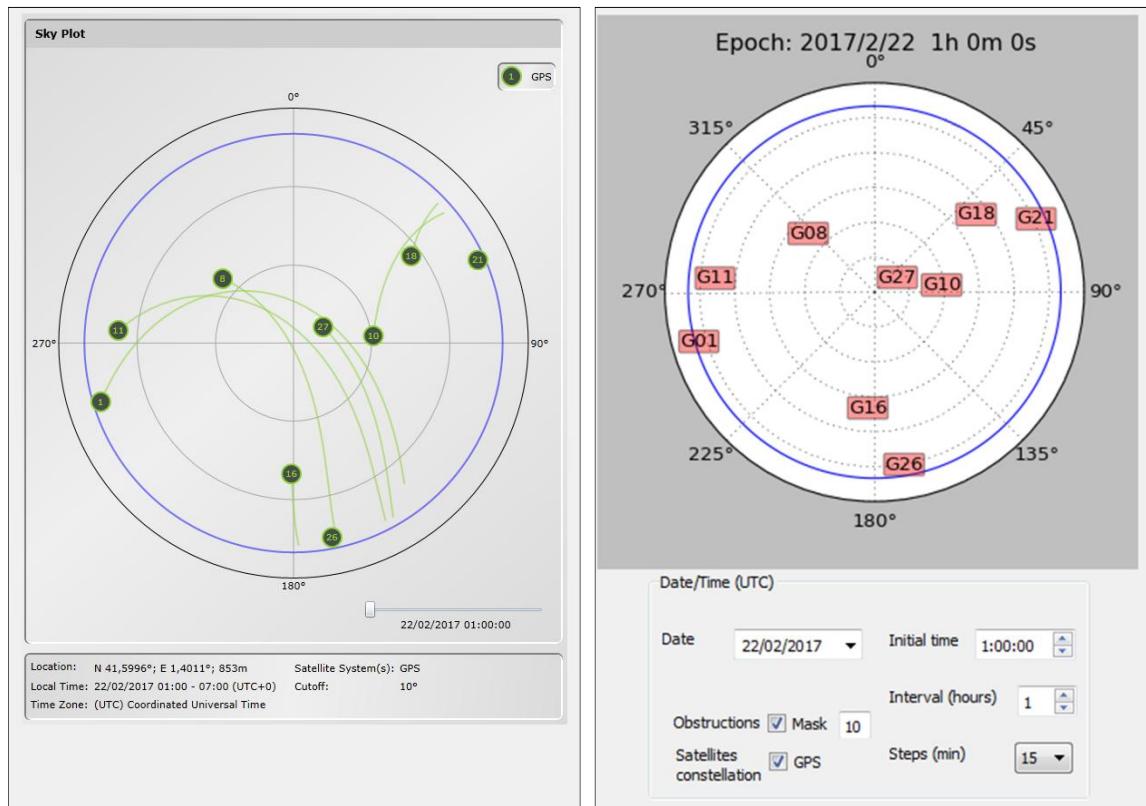
| | |
|-------------------------|---------------------------------------|
| Época | 22/02/2017 1:00:00 UTC |
| Posición del observador | lon = 1.4011° lat: 41.5996° h = 853 m |
| Máscara | 10° |
| Constelaciones | GPS |

Los resultados obtenidos son:

Tabla 21. Tabla comparativa de valores de DOP.

| | GDOP | PDOP | TIME | HDOP | VDOP |
|----------------------|------|-------|-------|------|-------|
| Trimble [®] | 1.75 | 1.55 | 0.76 | 0.92 | 1.27 |
| QGIS GNSSplanning | 1.74 | 1.56 | 0.77 | 0.87 | 1.29 |
| Diferencia | 0.01 | -0.01 | -0.01 | 0.05 | -0.02 |
| Diferencia relativa | 0.6% | -0.6% | -1.3% | 5.6% | -1.6% |

Las diferencias reales de los valores obtenidos entre estas dos aplicaciones no sobrepasan, en la mayoría de los casos, el 2%. Sólo en el HDOP esta diferencia llega al 5.6%. Esto nos demuestra que nuestro complemento calcula las posiciones de los satélites y los valores de DOP de forma satisfactoria, obteniendo resultados muy similares a los ofrecidos por una aplicación desarrollada por una empresa comercial, con gran experiencia en el mundo de los GNSS.



Trimble® GNSS Planning Online

QGIS GNSSPlanning

Figura 41. Comparación de resultados con otra aplicación existente en el mercado.

4. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

4.1. CONCLUSIONES

Este trabajo se ha centrado en desarrollar con código Python un complemento de QGIS para planificar trabajos de campo con GNSS. Con su implementación ha quedado demostrado el valor añadido que representa el hecho de integrar una herramienta como GNSSplanning en un software SIG. Así, se ha cumplido con el objetivo principal de mostrar los beneficios que esta simbiosis podía aportar en la fase de planificación de proyectos GNSS.

En la primera parte de esta memoria, a lo largo del capítulo primero, se han expuesto de forma clara los conceptos teóricos sobre el movimiento orbital, las perturbaciones, los modelos de propagación de órbitas y los errores que influyen en la precisión. Conocer estos conceptos es básico para adentrarse en la programación de un complemento como GNSSplanning. Por ejemplo, sin saber cómo actúan los modelos de propagación habría sido imposible aventurarse a programar un propagador para este trabajo.

En este sentido, uno de los retos más importantes al crear la aplicación ha sido, precisamente, la implementación del modelo de propagación. La complejidad de este tipo de modelos suponía un obstáculo al tener que crear todos los procesos y las funciones necesarias. El desarrollo de esta parte del complemento ha sido complejo, y ha necesitado de varios módulos y librerías de Python. La obtención de parte del código procedente de la librería *python-sgp4* (Brandon Rhodes, 2012-2016), ha facilitado en gran medida esta implementación.

Una de las partes más interesantes del proyecto, centrada también en los modelos de propagación, ha sido el trabajo realizado en la fase previa al desarrollo del propio complemento. Para encontrar qué propagador sería el adecuado para nuestra aplicación, se ha puesto especial énfasis en seleccionar un modelo que se adaptara lo mejor posible a nuestras expectativas, buscando que, además de ser efectivo en el cálculo, fuese fácilmente integrable en el complemento y sencillo de utilizar.

En el estudio realizado para seleccionar el propagador, se ha observado, tal y como era de esperar, que los dos propagadores candidatos obtenían los mejores resultados en los días cercanos al origen de las efemérides, pero a medida que transcurrían los días, aumentaban las distancias entre las posiciones calculadas y las reales. Pero el estudio ha revelado que el propagador basado en el modulo SDP4, ofrecía

una gran estabilidad en sus resultados a lo largo de los 4-5 días posteriores, y las posiciones aproximadas de los satélites que calculaba, no se alejaban demasiado de las reales. Esta estabilidad ha posicionado a este propagador como el candidato elegido para implementar en nuestra aplicación.

Para corroborar su correcto funcionamiento, se han comparado sus resultados con otro programa similar: Trimble GNSS Planning On-line. La comparación se ha hecho mediante los valores DOP ligados a las posiciones de los satélites en el cielo visible del observador. Los resultados han verificado el correcto comportamiento de nuestro propagador obteniendo valores similares para ambas aplicaciones. Por ejemplo, las diferencias relativas de sus valores GDOP no superan el 0.6%.

Otro de los retos importantes era averiguar cómo podría interactuar el complemento con las herramientas propias de QGIS y con las capas de información que estuvieran alojadas en su lienzo. La librería de Python del propio QGIS, ofrece las funciones necesarias para establecer esta relación. Aun así, ha sido necesario estudiar su contenido y aprender su funcionamiento, ya que era la primera vez que se usaba esta librería.

En este sentido, una de las funcionalidades más interesantes con la que se ha trabajado ha sido la encargada de mostrar sobre el gráfico polar el perfil real del horizonte según un modelo digital del terreno. Para su desarrollo se ha tenido que investigar primero, cuál sería la mejor opción. Se partió de la idea de crear un módulo nuevo que interactuara con los valores del modelo digital del terreno. Mediante funciones de la librería de Python de QGIS, se pretendía extraer valores de las cotas, calcular las máximas pendientes, etc. Finalmente se optó por aprovechar la herramienta de procesado r.horizon.height de GRASS GIS 7 (incorporada en QGIS) y que hace gran parte del proceso de cálculo que se requería para nuestro complemento. Evidentemente, ha sido necesario adaptar este proceso para que pudiese ejecutarse dentro de nuestro complemento e interpretar sus resultados para poder elaborar el perfil de horizonte final.

Se ha programado también con herramientas de la librería de Python de QGIS la función de captura de coordenadas de la posición aproximada del observador, interactuando de forma eficaz con el lienzo de QGIS y con las herramientas de transformación de coordenadas que se ofrecen.

En definitiva, la relación entre el complemento y las funcionalidades propias de QGIS ha sido todo un éxito pudiendo aprovechar una pequeña parte del gran potencial de este programa.

Por otro lado, se pretendía incorporar más modelos o mapas temáticos en los procesos de determinación de las obstrucciones que se podrían encontrar en los trabajos de campo. Se quería incorporar modelos ionosféricos o mapas de densidad arbórea, pero se ha preferido centrar los esfuerzos en la correcta utilización de los modelos digitales del terreno, dejando estas otras opciones

para futuras actualizaciones del complemento.

En general, los resultados obtenidos ofrecen una solución muy práctica y eficaz para los estudios de planificación de trabajos con GNSS, mostrando un escenario realista de la situación que se encontrará el usuario en sus trabajos de campo.

Haciendo referencia a la propia programación de GNSSplaning, hay que mencionar que no se partía de un conocimiento avanzado del lenguaje Python. En tal caso, hubiese sido relativamente sencillo programar un complemento como el que se ha creado en este proyecto. Pero al no conocer con profundidad el lenguaje ni las librerías específicas, como las propias de QGIS, esta tarea ha resultado mucho más compleja de lo cabía esperar.

4.2. FUTURAS LÍNEAS DE TRABAJO

Con la implementación de este complemento, se ha demostrado los beneficios de incorporar una herramienta de planificación, como GNSSplanning, sobre un entorno SIG. De momento, la aplicación se aprovecha de forma limitada del potencial de los SIG, pero abre la puerta a incorporar nuevas funcionalidades que actúen de forma similar con este entorno de trabajo.

A modo de ejemplo, mediante la incorporación de cartografía vectorial, con información de distintas infraestructuras, se podría analizar la cercanía de la posición del receptor a tendidos de alta tensión, a antenas de telefonía o zonas con inhibidores de señales. Con esta información se podrían descartar las posiciones dónde sería recomendable no trabajar con receptores GNSS.

También se podría añadir más elementos en el modelo de obstrucciones. En muchos trabajos de campo con receptores GNSS, la cobertura vegetal es un impedimento importante para recibir una señal del satélite nítida y directa. Incorporando mapas de cobertura arbórea dónde se indiquen valores de altura de la masa forestal y la densidad de esta, se podría definir un índice de pérdida de nitidez de la señal, dando al usuario otro factor para su toma de decisiones en la planificación.

Como se ha comentado en esta memoria, la aplicación utiliza el modelo SDP4 partiendo de los ficheros TLE publicados en CelesTrak. Recientemente, en esta web se han incorporado los *Supplemental TLE*. La información que incorporan estos nuevos ficheros es mucho más precisa. Para generar estos ficheros se utilizan más fuentes de datos que antes no se utilizaban. Según sus autores “algunos de estos datos orbitales, como para la constelación de GPS, están disponibles públicamente a través de Internet. El objetivo de este nuevo servicio es tomar esos datos y generar los TLE como parte de los procesos normales de actualización de datos en CelesTrak” (Kelso, T.S., 2017). Se puede ver un estudio de los

resultados de las primeras versiones para los satélites GPS y GLONASS en [19]. Así pues, se podría mejorar el modelo de propagación de GNSSplanning utilizando estos nuevos ficheros TLE.

Otra línea a explorar sería realizar un análisis para determinar si es posible incorporar el uso del Filtro de Kalman para obtener mejores estimaciones de la posición de los satélites.

Por otro lado, al escribir el código para la creación de este complemento se ha intentado seguir las normas de estilo que se recomiendan en la web de Python. Estas normas no son obligatorias, pero su seguimiento ayuda a la comprensión del código. Aun así, se podría mejorar el código escrito en distintos aspectos. Se podría simplificar en algunas funciones y mejorar su rendimiento computacional, o se podrían comentar más partes del código para explicar mejor qué se está ejecutando en cada línea.

En definitiva, creemos que partiendo de la aplicación creada en este trabajo se puede seguir buscando sinergias interesantes entre los sistemas de información geográfica y los proyectos con receptores GNSS, ya sea en su fase de planificación como en fases posteriores de análisis de datos observados.

REFERENCIAS BIBLIOGRÁFICAS

Por orden de aparición en la memoria:

- [1] Página oficial de Quantum GIS (QGIS): <http://qgis.org/>
- [2] Python ©2001-2017. Python Software Foundation. <https://www.python.org/>
- [3] BERNÉ VALERO, J. (2014) *GNSS. GPS: fundamentos y aplicaciones en Geomática*. Valencia. Universidad Politécnica de Valencia. Valencia.
- [4] SÁNCHEZ. J.A. (2010) *Conceptos sobre Órbitas. X Curso GPS en Geodesia y Cartografía*, Montevideo. Centro de Observaciones Geodésicas. Instituto Geográfico Nacional. España.
- [5] NORAD. North American Aerospace Defense Command. <http://www.norad.mil/>
- [6] CORBASÍ ORTÍN, A (1998) *Sistemas de Navegación: desde el compás magnético a la navegación por satélite*. McGraw-Hill Interamérica de España, S.A.U. Madrid.
- [7] *Astronautics Primer*. Jon Sellers. Analytical Graphics Incorporated.
<https://www.agi.com/resources/educational-alliance-program/astro-primer/primer1.htm>
- [8] ESA. Agencia Espacial Europea. <http://www.esa.int/ESA>
- [9] Hoots F.R., Roehrich, R. L. (1980) *SPACETRACK REPORT NO. 3 Models for Propagation of NORAD Element Sets*.
- [10] NASA. Administración Nacional de la Aeronáutica y del Espacio. <https://www.nasa.gov/>
- [11] D.A. VALLADO, P. CRAWFORD (2008) *SGP4 Orbit Determination*. Center for Space Standards and Innovation, Colorado Springs, Colorado. USA.
- [12] D.A. VALLADO, P. CRAWFORD (2006) *Revisiting Spacetrack Report #3*. Center for Space Standards and Innovation, Colorado Springs, Colorado. USA.
- [13] U.S. Coast Guard Navigation Center.
- [14] ICGC. Institut Cartogràfic i Geològic de Catalunya. <http://www.icgc.cat/>
- [15] SOPAC. Scripps Orbit and Permanent Array Center. <http://sopac.ucsd.edu/>
- [16] IGS, International GNSS Service. <http://www.igs.org/>
- [17] CODE. Analysis Center. Center of Orbit Determination for Europe
http://www.aiub.unibe.ch/research/code_analysis_center/index_eng.html
- [18] NGS. National Geodetic Survey. <https://www.ngs.noaa.gov/>
<https://www.navcen.uscg.gov/>
- [19] Celestrak. T.S. Kelso (2017) <http://celestrak.com/>

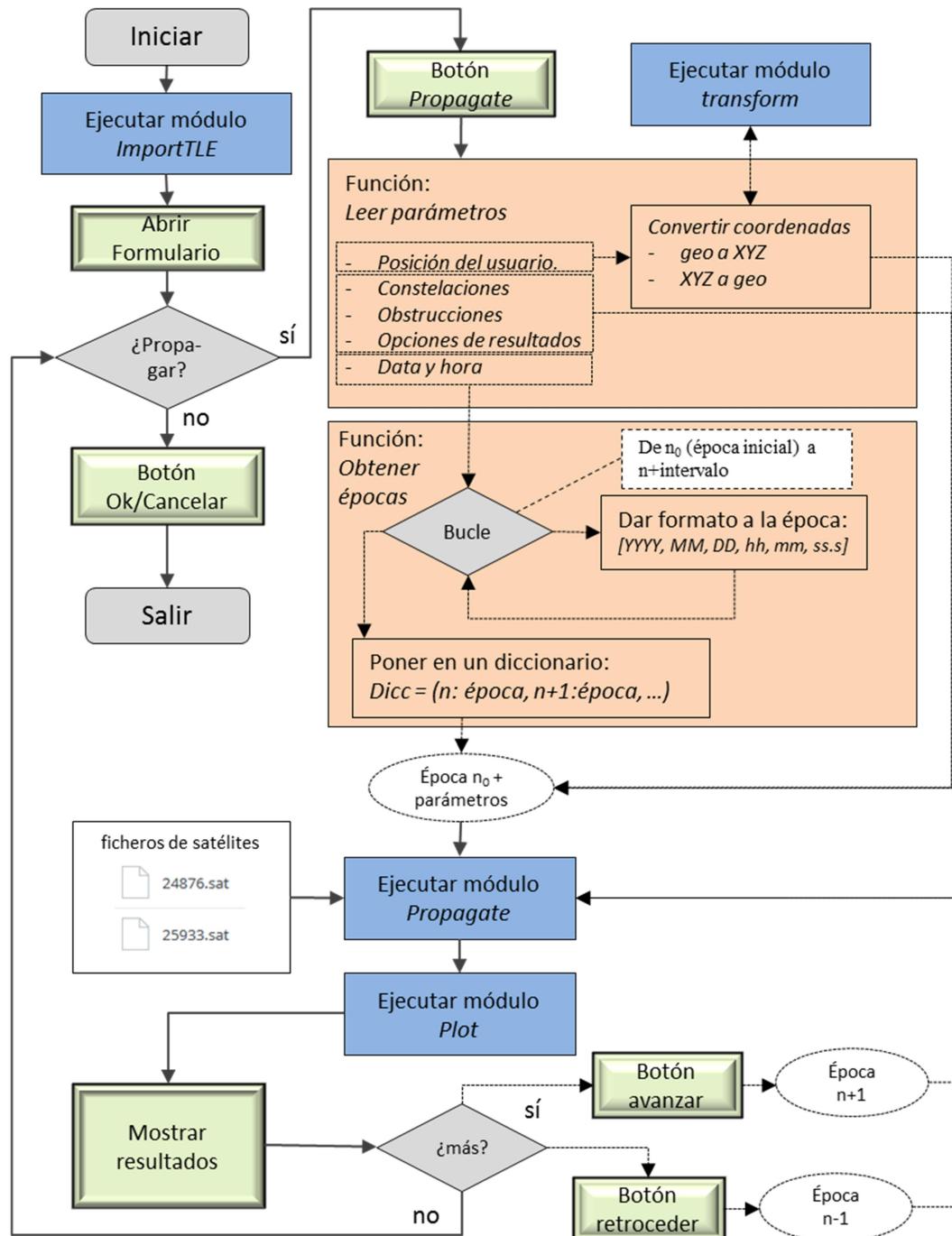
- [20] NASA. [Definition of Two-line Element Set Coordinate System.](https://spaceflight.nasa.gov/reldata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html)
https://spaceflight.nasa.gov/reldata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html
- [21] M. Hernández-Pajares, J.M. Juan, J. Sanz () *Procesado de Datos GPS: código y fase. Algoritmos, Técnicas y Recetas*. Grupo de Astronomía y Geomática. Centre de Publicacions del Campus Nord, UPC. Barcelona.
- [22] OLMEDILLAS, J.C. (2012) *Introducción a los sistemas de navegación por satélite*. Editorial UOC. Barcelona.
- [23] JAIME PÉREZ, R. (1995) *Radionavegació*. Ediciones de la Universitat Politècnica de Catalunya, S.L.
- [24] FARRELL, J. & BARTH, M. (1999) *The Global Position System & Inertial Navigation*. McGraw-Hill. New York.
- [25] IERS. International Earth Rotationa and Reference System Service.
https://www.iers.org/IERS/EN/Home/home_node.html
- [26] ИАЦ © 2005-2017 Information and Analysis Center for Positioning, Navigation and Timing, Korolyov, Russia. <https://www.glonass-iac.ru/en/>
- [27] *China Space Report. Beidou Navigation Satellite System*. © 2017.
<https://chinaspacereport.com/spacecraft/beidou/>
- [28] Navipedia. ESA. http://www.navipedia.net/index.php/Main_Page
- [29] Matplotlib © 2002 – 2012, John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team. <http://matplotlib.org>
- [30] sgp4 1.4 © 2015, Brandon Rhodes. <https://pypi.python.org/pypi/sgp4/>
- [31] Página oficial de SkyField. <http://rhodesmill.org/skyfield/>
- [32] ITRF (2013) *ITRS and WGS84*. <ftp://itrf.ensg.ign.fr/pub/itrf/WGS84.TXT>
- [33] Esri. Copyright © Esri. All rights reserved. www.esri.com.
- [34] Página oficial de la OGC: <http://www.opengeospatial.org/ogc>
- [35] Página oficial de Qt Company: <https://www.qt.io/about-us/>
- [36] PyEphem ©2008, Brandon Craig Rhodes. <http://rhodesmill.org/pyephem/>
- [37] Página oficial de licencias GPL: <https://www.gnu.org/licenses/gpl.html>
- [38] PyQt4 4.11.4: <https://pypi.python.org/pypi/PyQt4>
- [39] Plugin Builder: <https://plugins.qgis.org/plugins/pluginbuilder/>
- [40] Plugin Loader: https://plugins.qgis.org/plugins/plugin_reloader/
- [41] Documentación sobre la API de QGIS: <http://doc.qgis.org/api/>
- [42] Página official de GRASS GIS. <https://grass.osgeo.org/>

- [43] MARISCAL, A. & MUÑOZ, A. (2017) *IndoorGML Reader, un plugin de QGIS*. Trabajo Fin de Grado de Ingeniería Informática. UOC. Cataluña.
- [44] ARTANO, K. CHAVEZ M. (2012) *Ejercicios prácticos para el procesado de datos GNSS- PBGNSS*. Proyecto Final de Máster. Máster en Geotecnologías Cartográficas en Ingeniería y Arquitectura. Escuela Politécnica Superior de Ávila. Universidad de Salamanca.
- [45] Hofmann-Wellenhof, B & Lichtenegger, Herbert, Wasle, Elmar (2008). *GNSS - GPS, GLONASS, Galileo and more*. Springer, New York.
- [46] Trimble©GNSS Planning On-line. <http://www.gnssplanningonline.com/>

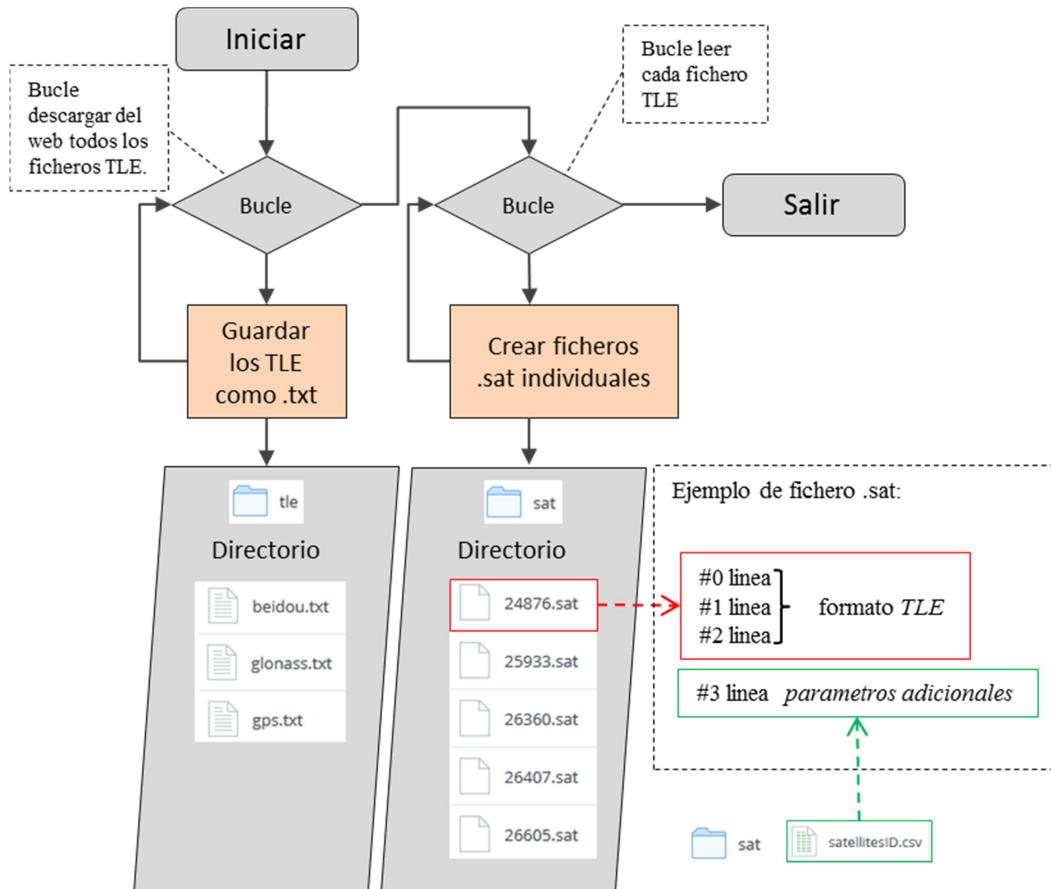
ANEXO 1. DIAGRAMAS DE FLUJO

En este apartado se muestran los diagramas de flujo de los principales módulos y clases programados íntegramente para este trabajo. Los módulos que sólo contienen funciones y no describen ningún flujo interno, no se muestra en este apartado.

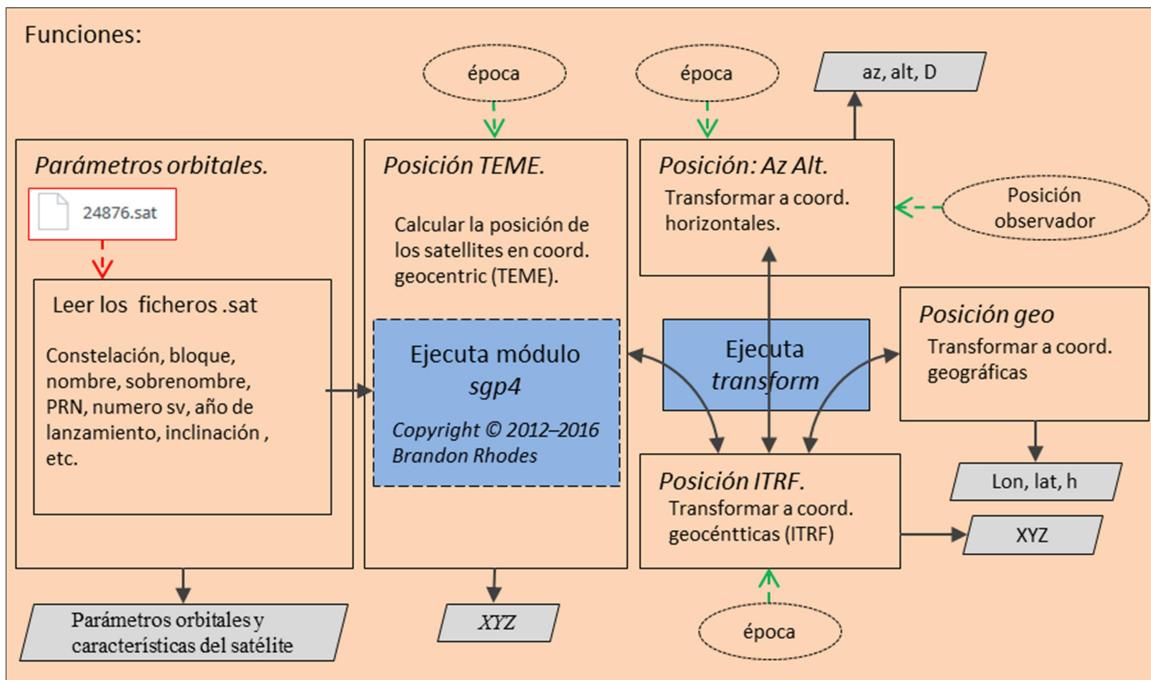
a. MÓDULO *gnss_planning.py*



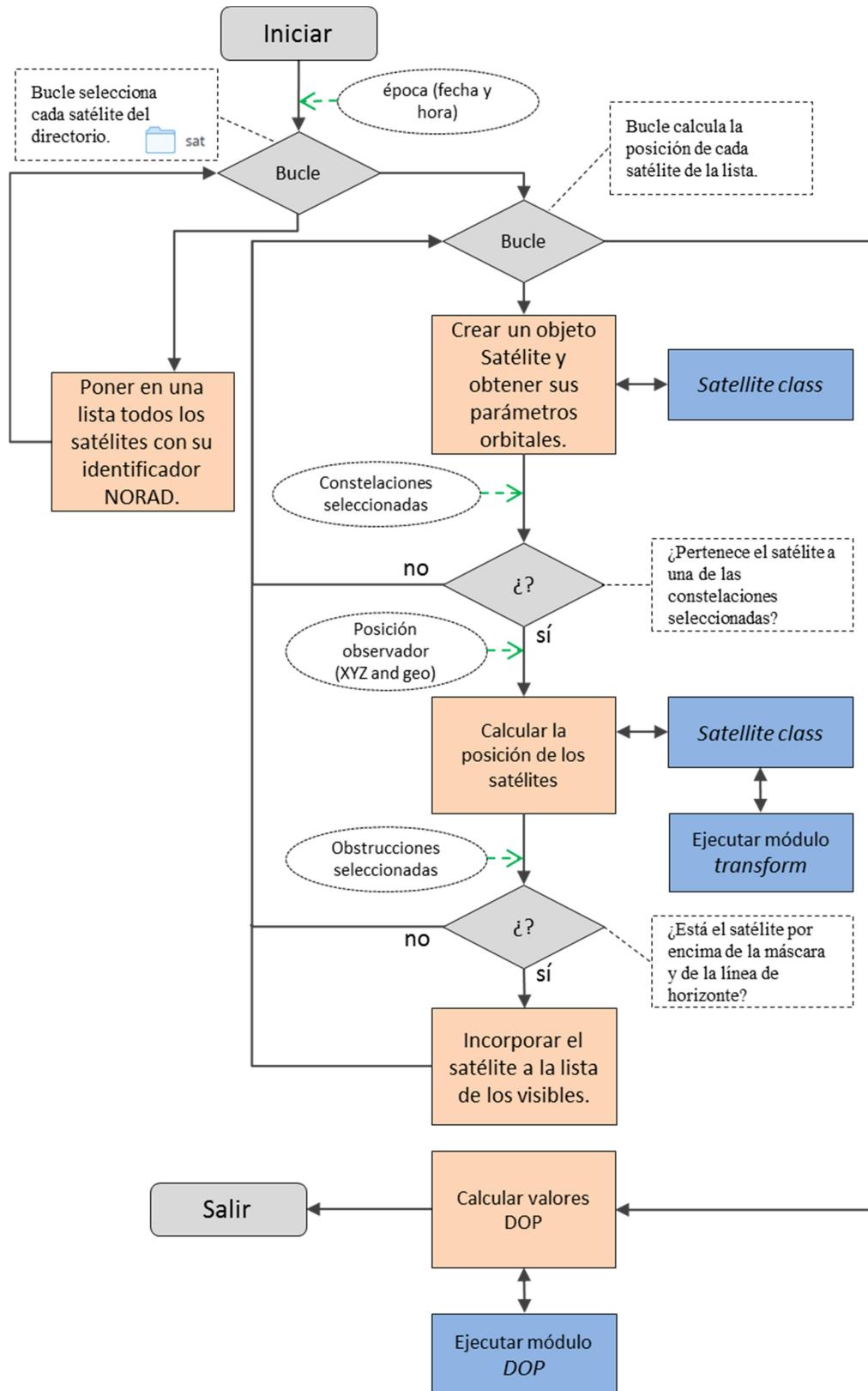
b. MÓDULO ImportTLE.py



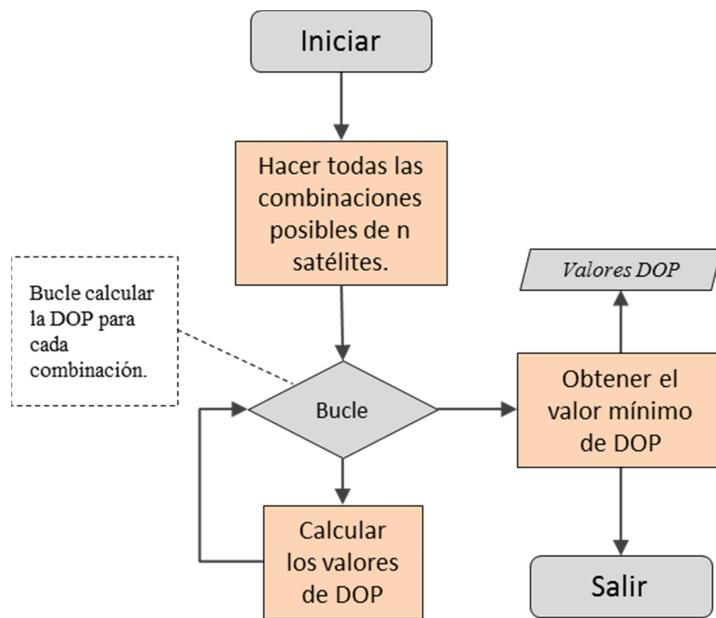
c. CLASE Satellite.py



d. MÓDULO *Propagate.py*



e. MÓDULO *DOP.py*



ANEXO 2. CÓDIGO

En este apartado se ha incorporado el código de todos los módulos y clases programados íntegramente para este trabajo. Se incluye al final de este apartado, el código del script PROP-Receptor, programado exclusivamente para el estudio detallado en el capítulo 2.

a. Código *gnss_planning.py*

```
# -*- coding: utf-8 -*-
"""
*****
GNSSPlanning
    A QGIS plugin
This plugin predict GNSS satellites position over receiver's sky given an
epoch and a skyline obstruction.
-----
begin          : 2017-02-09
git sha        : $Format:%H$
copyright      : (C) 2017 by Cesc Masdeu Ferrer
email          : cescmf@gmail.com
*****
*/
*
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*****
# Import qgis libraries in order to interoperate with QGIS elements.
import qgis
from qgis.core import *
from qgis.gui import *
from qgis.utils import iface
from osgeo import gdal
import processing
# Import PyQt4 libraries in order to satisfy functional needs. (QMessageBox,
QFileDialog)
from PyQt4 import QtCore, QtGui
from PyQt4.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt4.QtGui import QAction, QIcon, QFileDialog, QMessageBox, QApplication
# Initialize Qt resources from file resources.py
import resources
# Import the code for the dialog
from gnss_planning_dialog import GNSSplanningDialog
# Import other modules
import sys
import os.path
import os
```

```

import numpy as np
from math import pi
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt4agg import (FigureCanvasQTAgg as FigureCanvas,
NavigationToolbar2QT as NavigationToolbar)
import matplotlib.pyplot as plt
# Import plugin modules
from scripts.ImportTLE import Import_TLE
from scripts.transform import geographic_to_ITRF, ITRF_to_geographic
from scripts.Propagate import propagate
from scripts.DOP import getDOP_values
from scripts.Polarplot import PolarChart_class
from scripts.Profiles import linesProfiles
from scripts.Capture_coords import CaptureCoords
from scripts.Satellite import Satellite_class
from scripts.commonfun import getCurrentPath

class GNSSplanning():
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
            which provides the hook by which you can manipulate the QGIS
            application at run time.
        :type iface: QgsInterface
        """

        # Create the dialog (after translation) and keep reference
        self.dlg = GNSSplanningDialog()
        # Save reference to the QGIS interface
        self.iface = iface
        # initialize plugin directory
        self.plugin_dir = os.path.dirname(__file__)
        # initialize locale
        locale = QSettings().value('Locale/userLocale')[0:2]
        locale_path = os.path.join(
            self.plugin_dir,
            'i18n',
            'GNSSplanning_{}.qm'.format(locale))

        if os.path.exists(locale_path):
            self.translator = QTranslator()
            self.translator.load(locale_path)

            if qVersion() > '4.3.3':
                QCoreApplication.installTranslator(self.translator)

        # Declare instance attributes
        self.actions = []
        self.menu = self.tr(u'&GNSSplanning')
        # TODO: We are going to let the user set this up in a future iteration
        self.toolbar = self.iface.addToolBar(u'GNSSplanning')
        self.toolbar.setObjectName(u'GNSSplanning')

        # Import TLE files:

```

```

try:
    Import_TLE()
except:
    msg = QMessageBox()
    msg.setText("Internet connection error")
    msg.setDetailedText('Please check your Internet connection')
    msg.exec_()

# Capture coordinates from QGIS canvas.
self.dlg.pushButton_Select.clicked.connect(self.select_on_canvas)
# Capture coordinates from QGIS canvas.
self.dlg.pushButton_Copy.clicked.connect(self.copy_on_form)
# Initialize propagate button.
self.dlg.pushButton_Propagate.clicked.connect(self.run_first_propagate)
# propagate next epochs
self.dlg.horizontalSlider_Plot.valueChanged.connect(self.run_propagate)

self.layers = self iface.legendInterface().layers()
layer_list = []
for layer in self.layers:
    layer_list.append(layer.name())
    self.dlg.comboBox_Skyline.addItem(layer_list)

self.skyline = "off"
self.mask = 0
self.plt = plt
self.plot(plotSat_List=[])

# noinspection PyMethodMayBeStatic
def tr(self, message):
    """Get the translation for a string using Qt translation API.

    We implement this ourselves since we do not inherit QObject.

    :param message: String for translation.
    :type message: str, QString

    :returns: Translated version of message.
    :rtype: QString
    """
    # noinspection PyTypeChecker,PyArgumentList,PyCallByClass
    return QCoreApplication.translate('GNSSPlanning', message)

def add_action(
    self,
    icon_path,
    text,
    callback,
    enabled_flag=True,
    add_to_menu=True,
    add_to_toolbar=True,
    status_tip=None,
    whats_this=None,
    parent=None):
    """Add a toolbar icon to the toolbar.

```

```

:param icon_path: Path to the icon for this action. Can be a resource
    path (e.g. ':/plugins/foo/bar.png') or a normal file system path.
:type icon_path: str

:param text: Text that should be shown in menu items for this action.
:type text: str

:param callback: Function to be called when the action is triggered.
:type callback: function

:param enabled_flag: A flag indicating if the action should be enabled
    by default. Defaults to True.
:type enabled_flag: bool

:param add_to_menu: Flag indicating whether the action should also
    be added to the menu. Defaults to True.
:type add_to_menu: bool

:param add_to_toolbar: Flag indicating whether the action should also
    be added to the toolbar. Defaults to True.
:type add_to_toolbar: bool

:param status_tip: Optional text to show in a popup when mouse pointer
    hovers over the action.
:type status_tip: str

:param parent: Parent widget for the new action. Defaults None.
:type parent: QWidget

:param whats_this: Optional text to show in the status bar when the
    mouse pointer hovers over the action.

:returns: The action that was created. Note that the action is also
    added to self.actions list.
:rtype: QAction
"""

icon = QIcon(icon_path)
action = QAction(icon, text, parent)
action.triggered.connect(callback)
action.setEnabled(enabled_flag)

if status_tip is not None:
    action.setStatusTip(status_tip)

if whats_this is not None:
    action.setWhatsThis(whats_this)

if add_to_toolbar:
    self.toolbar.addAction(action)

if add_to_menu:
    self iface.addPluginToMenu(
        self.menu,
        action)

self.actions.append(action)

```

```
    return action

def initGui(self):
    """Create the menu entries and toolbar icons inside the QGIS GUI."""
    icon_path = ':/plugins/GNSSPlanning/icon.png'
    self.add_action(
        icon_path,
        text=self.tr(u'GNSS Planning'),
        callback=self.run,
        parent=self.iface mainWindow())

def unload(self):
    """Removes the plugin menu item and icon from QGIS GUI."""
    for action in self.actions():
        self.iface.removePluginMenu(
            self.tr(u'&GNSSPlanning'),
            action)
        self.iface.removeToolBarIcon(action)
    del self.toolbar

def select_on_canvas(self):
    """ capture coordinates from QGIS in one click in EPSG:4326 (WGS 1984)
    independently from what your current CRS is."""
    self.mapTool = CaptureCoords(self.iface)
    self.iface.mapCanvas().setMapTool(self.mapTool)
def copy_on_form(self):
    try:
        self.dlg.lineEdit_LonX.setText(self.mapTool.x_WGS)
        self.dlg.lineEdit_LatY.setText(self.mapTool.y_WGS)
    except:
        pass

def run_first_propagate(self):
    self.horizon()
    """Propagate satellites position at first epoch."""
    if self.dlg.checkBox_Skyline.isChecked():
        self.skyline = "on"

    else:
        self.skyline = "off"

    if self.read_parameters():
        self.observer_position()
        self.get_epochs()
        self.run_propagate()
    else:
        pass

def horizon(self):
    try:
```

```

selectedLayerIndex = self.dlg.comboBox_Skyline.currentIndex()

rasterlayer = self.layers[selectedLayerIndex]
horizonresults = self.plugin_dir + r'\scripts\profiles\horizon.txt'
x = self.mapTool.x_UTM
y = self.mapTool.y_UTM
xy = str(x)+","+str(y)

processing.runalg('grass7:r.horizon.height',rasterlayer.name(),xy,0.0,5.0,0.0,360.0,1
0000.0,'1.0',True,'456817.5,484552.5,4549752.5,4668457.5',None,horizonresults)

except:
    pass

def run_propagate(self):
    try:
        self.epochNum = self.dlg.horizontalSlider_Plot.value()
        self.epoch = self.dicc_epochs[self.epochNum]
        results = propagate(self.geo_observer, self.XYZ_observer, self.epoch,
self.epochNum, self.mask, self.skyline, self.constellations)
        plotSat_List = results[0]
        XYZSat_List = results[1]
        meus = results[2]

        self.plot(plotSat_List)

        DOP = getDOP_values(self.XYZ_observer, XYZSat_List[1])
        message = "    GDOP %.2f      PDOP %.2f      TDOP %.2f
HDOP %.2f      VDOP %.2f" % (DOP[1][0], DOP[1][1], DOP[1][2], DOP[1][3], DOP[1][4])
        self.dlg.label_DOP.setText(str(message))

    except:
        pass

def read_parameters(self):
    check = True

    try:
        self.lonX_obs = float(self.dlg.lineEdit_LonX.text())
        self.latY_obs = float(self.dlg.lineEdit_LatY.text())
        self.hZ_obs = float(self.dlg.lineEdit_hZ.text())
        self.antenna = float(self.dlg.lineEdit_Antenna.text())
    except:
        check = False
        msg = QMessageBox()
        msg.setText("Invalid parameters")
        msg.setDetailedText("Coordinates and mask values must be integer or
float. Empty forms are not allowed.")
        msg.exec_()

    self.inputCoord_observer = [self.lonX_obs, self.latY_obs, self.hZ_obs +
self.antenna]

    # input and check mask value:
    if self.dlg.checkBox_Mask.isChecked():

```

```

self.mask = float(self.dlg.lineEdit_Mask.text())
if self.mask < 0:
    check = False
    msg = QMessageBox()
    msg.setText("Invalid mask parameter")
    msg.setDetailedText("Mask values must be >= 0")
    msg.exec_()
else:
    self.mask = 0.0

# add constellations:
self.constellations = []
if self.dlg.checkBox_GPS.isChecked():
    self.constellations.append("GPS")
if self.dlg.checkBox_GLONASS.isChecked():
    self.constellations.append("Glonass")
if self.dlg.checkBox_GALILEO.isChecked():
    self.constellations.append("Galileo")
if self.dlg.checkBox_BEIDOU.isChecked():
    self.constellations.append("BeiDou")

# input epochs values:
self.dateEpoch = str(self.dlg.dateEdit_Date.date().toPyDate()).split("-")
self.timeEpoch = str(self.dlg.timeEdit_Time.time().toPyTime()).split(":")
self.interval = int(self.dlg.spinBox_Interval.text())
self.step_min = int(self.dlg.comboBox_Steps.currentText())

return check

def observer_position(self):
    if self.dlg.radioButton_Geogr.isChecked() or
self.dlg.radioButton_Select.isChecked():
        self.geo_observer = self.inputCood_observer
        self.XYZ_observer = geographic_to_ITRF(self.geo_observer)
    else:
        self.XYZ_observer = self.inputCood_observer
        self.geo_observer = ITRF_to_geographic(self.XYZ_observer)

def get_epochs(self):
# calculate dates and times for each epoch:
    Y = int(self.dateEpoch[0])
    M = int(self.dateEpoch[1])
    D = int(self.dateEpoch[2])
    h = int(self.timeEpoch[0])
    m = int(self.timeEpoch[1])
    s = int(self.timeEpoch[2])

    # get number of epochs:
    interval = self.interval
    step = self.step_min
    steps = interval * 60/step

    # put all epochs into diccionary:
    obs = 1
    self.dicc_epochs = {}

```

```

while (obs-1) <= steps:
    epoch = [Y, M, D, h, m, s]
    self.dicc_epochs[obs] = epoch
    m = m + step

    if m >= 60:
        m = m - 60
        h = h + 1
    if h == 24:
        h = 0
        D = D + 1
    dicc_dayM = {1:31, 2:28, 3:31, 4:30, 5:31, 6:30,
                 7:31, 8:31, 9:30, 10:31, 11:30, 12:31}
    if D > dicc_dayM[M]:
        D = 1
        M = M + 1
    if M > 12:
        M = 1
        Y = Y + 1

    obs = obs + 1 # loop

# Define slider parameters
self.dlg.horizontalSlider_Plot.setMinimum(1)
self.dlg.horizontalSlider_Plot.setMaximum(obs-1)
self.dlg.horizontalSlider_Plot.setSingleStep(1)
self.dlg.horizontalSlider_Plot.setValue(1)

def plot(self, plotSat_List):
    fig = Figure()
    ax = fig.add_subplot(111, projection='polar')
    ax.set_theta_direction(-1)
    ax.set_theta_zero_location('N')
    ax.set_title("Sky plot", va='bottom')
    ax.set_rticks([90, 75, 60, 45, 30, 15])

    if len(plotSat_List) == 0:
        pass
    else:
        self.dlg.mplvl.removeWidget(self.canvas)

        theta = [np.pi * 2]
        radii = [90]
        width = [np.pi * 2]
        bars = ax.bar(theta, radii, width=width, bottom=0.0)

        # Use custom colors and opacity
        for r, bar in zip(radii, bars):
            bar.set_alpha(0)

    # get satellites features calling Satellite_class. Get NORAD number back
    from txt files.
    labels = []
    positions = []
    obsNum, sat_to_plot_list = plotSat_List[0], plotSat_List[1]
    for sat_to_plot in sat_to_plot_list:

```

```

# get satellites NORAD number and get parameters back
NORADnum = sat_to_plot[2]
ve = Satellite_class(NORADnum)
ve.getSatParameters()
labels.append(ve.nickname)
# get satellites position
azSat = sat_to_plot[0] / 180 *pi # radians
altSat = 90 - sat_to_plot[1]
positions.append([azSat, altSat])

# put into array
poitionsArray = np.array(positions)
azP = poitionsArray[:, 0]
altP = poitionsArray[:, 1]

# plot satellites labels
for label, x, y in zip(labels, azP, altP):
    if label[0] == "R": #Glonass
        color = 'yellow'
    elif label[0] == "G": # GPS
        color = 'red'
    elif label[0] == "C": # BeiDu
        color = "blue"
    elif label[0] == "E":
        color = "green"

    ax.annotate(label, xy=(x, y), xytext=(10, -5),
    textcoords='offset points', ha='right', va='bottom',
    bbox=dict(boxstyle='round,pad=0.1', fc=color, alpha=0.4))

# get skyline from txt files
pluginPath = str(getCurrentPath())
if self.skyline == "on":
    horizon = np.genfromtxt(str(pluginPath) + "profiles\\\" +
'horizon.txt', delimiter=',')
    horizon = horizon[1:, :]
# plot skyline
ax.plot((horizon[:, 0]) / 180 * pi , (90-horizon[:, 1]))

# make maskline
maskList = []
for angle in range(0, 361):
    maskList.append([float(angle), float(self.mask)])
maskArray = np.array(maskList)
# plot maskline
ax.plot((360 - maskArray[:, 0]) / 180 * np.pi , (90 - maskArray[:, 1]))

ax.set_title("Epoch: %s/%s/%s %sh %sm %ss" % (str(self.epoch[0]),
str(self.epoch[1]), str(self.epoch[2]),str(self.epoch[3]),str(self.epoch[4]),
str(self.epoch[5])), va='bottom')

self.canvas = FigureCanvas(fig)
self.dlg.mplvl.addWidget(self.canvas)
self.canvas.draw()

def run(self):
    """Run method that performs all the real work"""

```

```
# show the dialog
self.dlg.show()
# Run the dialog event loop
result = self.dlg.exec_()
# See if OK was pressed
if result:
    # Do something useful here - delete the line containing pass and
    # substitute with your code.
    Pass
```

b. Código ImportTLE.py

```
"""
-----
Name:      ImportTLE
Purpose:   This script was used to create the initial satellite data
           repository by converting Celestrak TLE files into a
           satellites.dat and .cat files.
           You should have a .cat file for each category as well as a
           satellites.dat file in the ./sat/ folder
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
-----
"""

import os
import string
import urllib

def Import_TLE():

    # Satellite groups
    groups = {
        "gps-ops" : "GPS",
        "glo-ops" : "Glonass",
        "galileo" : "Galileo",
        "beidou" : "BeiDou"
    }

    # URL TLE files
    urlprefix = "http://celestrak.com/NORAD/elements/"
    plugin_dir = os.path.dirname(__file__)

    for group, name in groups.iteritems():
        webfile = urlprefix + group + ".txt"
        localfile = plugin_dir + '/tle/' + group + ".txt"

        urllib.urlretrieve (webfile, localfile)

    # For each input file
    for group, name in groups.iteritems():

        # open TLE file for reading
        tlefile = open(plugin_dir + '/tle/' + group + '.txt', 'r')

        # create category file
        category = plugin_dir + "/cat/" + group + ".cat"
        catfile = open(category, 'w')

        # first line is the group name
        catfile.write(name+'\n')

        while 1:
            # read three lines at a time; strip trailing whitespaces
            line0 = tlefile.readline().strip()

            if not line0:
```

```

        break
line1 = tlefile.readline().strip()
line2 = tlefile.readline().strip()

# catalog number; strip leading zeroes
catnum = line1[2:7].lstrip('0')

# add satellite to category
catfile.write(catnum+'\n')

# read satellite Names and IDs from satellitesIDs.sat
satIDs = open(plugin_dir + '/sat/' + 'satellitesID.csv', 'r')
for line in satIDs:

    if line[0:5] != catnum:
        line3 = catnum + " NULL NULL NULL NULL NULL NULL"
    else:
        line3 = line
        break
satIDs.close()

# add TLE and satellites names and IDs to satellite file
satfile = open(plugin_dir + '/sat/' + catnum + '.sat', 'w')

satfile.write(line0+'\n')
satfile.write(line1+'\n')
satfile.write(line2+'\n')
satfile.write(line3+'\n')

satfile.close()

# close TLE and CAT files
tlefile.close()
catfile.close()

```

c. Código *Satellite.py*

```
"""
-----
Name:      Satellite
Purpose:   This script define satellites properties.
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
-----

"""

import math
import os
from numpy import array, cross, einsum, zeros_like
from sgp4.earth_gravity import wgs84      # python-sgp4 (Copyright ? 2012?2016 Brandon Rhodes)
from sgp4.io import twoline2rv           # python-sgp4 (Copyright ? 2012?2016 Brandon Rhodes)
from transform import julian_date, TEME_to_ITRF, ITRF_to_geographic,
ITRF_to_horizontal
#import ephem
from commonfun import convert, getCurrentPath

#-----
# SATELLITE OBJECT:
#
class Satellite_class():
    def __init__(self, satNum):
        self.satNum = satNum # NORAD NUM

    def getSatParameters(self):
        '''(satNum)-> Satellite parameters
        Get satellites parameters from .sat (files with TLE format).
        >>>Satellite.getSatParameters('41586')
        BsiDou      # constellation
        G7          # nickname
        41586      # NORAD number
        U           # unclassified
        etc...
        '''
        # get the plugin path from this file path
        pluginPath = str(getCurrentPath())

        # open TLE file for reading
        satfile = open(pluginPath + 'sat\\' + self.satNum + '.sat', 'r')

        # read 3 lines at a time; strip trailing whitespaces
        self.line0 = satfile.readline().strip()
        self.line1 = satfile.readline().strip()
        self.line2 = satfile.readline().strip()
        self.line3 = satfile.readline().strip()
        self.listIDsat = self.line3.split(" ")
        a = math.pow(3,3)

        # create a SDP4 object
        self.satSDP4 = twoline2rv(self.line1, self.line2, wgs84)
```

```

# create a ephem object
#self.satEphem = ephem.readtle(self.line0, self.line1, self.line2)

# define satellites parameters
self.constellation = self.listIDsat[5]
self.block = self.listIDsat[6]
self.name = self.listIDsat[4]
self.nickname = self.listIDsat[1]
self.prn = self.listIDsat[2]
self.sv = self.listIDsat[3]
self.norad = self.line1[2:7]
self.clasUnclas = self.line1[7]
self.launchYear = self.line1[9:11]
self.launchNumber = self.line1[11:14]
self.launchPice = self.line1[14:17]
self.epochYear = self.line1[18:20]
self.epochDay = self.line1[20:32]
self.firstDMM = self.line1[33:43]
self.secondDMM = self.line1[44:52]
self.dragBSTAR = self.line1[53:61]
self.ephemType = self.line1[62]
self.elementSet = self.line1[64:68]
self.checksum1 = self.line1[68]
self.inclination = self.line2[8:16]
self.raan = self.line2[17:25]
self.eccentricity = "0." + self.line2[26:33]
self.augPerigee = self.line2[34:42]
self.meanAnomaly = self.line2[43:51]
self.meanMotion = self.line2[52:63]
self.revolution = self.line2[63:68]
self.checksum2 = self.line2[68]

def getPosition_TEME(self, dateTime):
    '''(dateTime)-> position and velocity
    Get position and velocity from satellite object into TEME frame
    >>>Satellite("29601").getPosition_TEME([2017, 01, 03, 0, 0, 0.0])
    (-26237.73044512979, 4795.304682136226, 413.75989340053593)      # position
    (-0.44695695555351667, -2.0707587248243144, -3.2234138176852225)  # velocity
    '''
    # propagate position and velocity (TEME)
    self.position_TEME, self.velocity_TEME = self.satSDP4.propagate(*dateTime)
    self.X_TEME = self.position_TEME[0]
    self.Y_TEME = self.position_TEME[1]
    self.Z_TEME = self.position_TEME[2]

    return self.position_TEME, self.velocity_TEME

def getPosition_ITRF(self, dateTime):
    '''(dateTime)-> position and velocity
    Get position and velocity from satellite object into ITRF frame
    >>>Satellite("29601").getPosition_ITRF([2017, 01, 03, 0, 0, 0.0])
    (-26237.73044512979, 4795.304682136226, 413.75989340053593)      # position
    (-0.44695695555351667, -2.0707587248243144, -3.2234138176852225)  # velocity
    '''
    # first, get poition into TEME frame

```

```

position_from_TEME, velocity_from_TEME = self.getPosition_TEME(dateTime)

# transform TEME into ITRF
year, month, day      = dateTime[0], dateTime[1], dateTime[2]
hour, minute, second = dateTime[3], dateTime[4], dateTime[5]
jday = julian_date(year, month, day, hour, minute, second)

position_ITRF_KM, velocity_ITRF_KM = TEME_to_ITRF(jday, position_from_TEME,
velocity_from_TEME)

self.X_ITRF = position_ITRF_KM[0] * 1000
self.Y_ITRF = position_ITRF_KM[1] * 1000
self.Z_ITRF = position_ITRF_KM[2] * 1000
self.i_ITRF = velocity_ITRF_KM[0]
self.j_ITRF = velocity_ITRF_KM[1]
self.k_ITRF = velocity_ITRF_KM[2]
self.position_ITRF = [self.X_ITRF, self.Y_ITRF, self.Z_ITRF]
self.velocity_ITRF = [self.i_ITRF, self.j_ITRF, self.k_ITRF]

return self.position_ITRF, self.velocity_ITRF


def getPosition_geo(self, dateTime):
    '''(dateTime)-> position
    Get position and velocity from satellite object in geographic sr
    >>>Satellite("29601").getPosition_geo([2017, 01, 03, 0, 0, 0])
    (47.869896160692306, -8.005048373017335, 20034419.68723937) # Lat, Lon,
alt(elip)
    '''
    # first, get poition into ITRF frame
    position_from_ITRF = self.getPosition_ITRF(dateTime)[0]
    # transform ITRF(XYZ) into geographic
    self.position_geo = ITRF_to_geographic(position_from_ITRF)
    self.lon = self.position_geo[0]
    self.lat = self.position_geo[1]
    return self.position_geo


def getPosition_azalt_op2(self, geo_observer, dateTime):
    '''(observer_geo, dateTime)-> Satellite position
    Get satellite position (altitude and azimut) from observer
    situation (geographic coord: Longitude, Latitude and elevation)
    at a specific date and time, by using TLE files.
    >>>Satellite.getPosition([2.2399, 41.4451, 5.14], [2017, 01, 01, 20, 00,
00.0])
    40.6438611111, 16.1937222222 # decimal degrees
    '''
    observer = ephem.Observer()
    observer.lon = geo_observer[0]
    observer.lat = geo_observer[1]
    observer.elevation = geo_observer[2]
    date_time = str(dateTime[0]) + "/" + str(dateTime[1]) + "/" +
str(dateTime[2]) + " " + str(dateTime[3]) + ":" + str(dateTime[4])
    observer.date = date_time

    # compute observer

```

```
self.satEphem.compute(observer)

self.altitude = convert(self.satEphem.alt)
self.azimuth = convert(self.satEphem.az)
self.nameEphem = self.satEphem.name

return self.azimuth, self.altitude


def getPosition_azalt(self, geo_observer, XYZ_observer, dateTime):
    '''(geo_observer, XYZ_observer, dateTime)-> az, alt, D
    Get satellite position from observer situation at a specific date and time
    by using TLE files.
    Observer position:
        - geographic coord (longitude, latitude and elevation),
        - UTM projection (x, y, H) #TODO
        - Geocentric ITRF
    satellite position:
        - horizontal coord (azimuth, altitude)

>>>Satellite.getPosition_azalt([41.5996, 1.40113, 853.3707],
                               [4775849.592, 116814.09, 4213018.69],
                               [2017, 01, 01, 20, 00, 00.0])
287.678319329999, 76.5702066161654      # decimal degrees
'''

# get satellite position into ITRF frame
XYZ_sat = self.getPosition_ITRF(dateTime)[0]

# transform sat position into horizontal coordenates
self.position_azalt = ITRF_to_horizontal(geo_observer, XYZ_observer, XYZ_sat)
self.az = self.position_azalt[0]
self.alt = self.position_azalt[1]
self.D = self.position_azalt[2]
return self.az, self.alt, self.D
```

d. Código *Propagate.py*

```
"""
-----
Name: Propagate
Purpose: This script is used by get satellites positions.
Created: 2017 by Cesc Masdeu Ferrer
Licence: Free
-----

"""

import os, shutil
import numpy as np
from Satellite import Satellite_class
from Polarplot import PolarChart_class
from DOP import getDOP_values
from commonfun import truncate, getCurrentPath

def propagate(geo_observer, XYZ_observer, epoch, epochNum, mask, skyline,
constellations):

    # -----
    # INICIALIZATION PROCES
    # get the module path from getCurrentPath()
    modulePath = str(getCurrentPath())
    s = -18.13
    resultats = []

    # get skyline from txt files
    if skyline == "on":
        skyL = np.genfromtxt(modulePath + "profiles\\\" + 'horizon.txt',
delimiter=',')

    # -----
    # READ SATELLITES EFEMERIDES
    listSat = []
    for file in os.listdir(modulePath + 'sat\\\'):
        try:
            if file[-1] == "t":
                fileName = file.split(".")[0]
                listSat.append(fileName)
        except Exception as e:
            pass

    #-----
    # PROPAGATE SATELLITES POSITION FOR THE EPOCH
    overMask_sats_AzAlt = []
    overMask_sats_XYZ= {}

    plotSat_List = [] # list with 2 items: #1 number of epoch; #2 list of all
satellite's coordinates (Az, Alt, NORAD num).
    XYZSat_List = [] # list with 2 items: #1 number of epoch; #2 dicctionary with:
key = nickname, value = coordinates (X, Y, Z); for all satellites.
```

```

# search which satellites are in constellation list and over the masks:
for satellite in listSat:
    # define satellite like a object:
    ve = Satellite_class(satellite)
    ve.getSatParameters()

    # is the satellite constellation in the constellations list that user has
selected?:
    if ve.constellation in constellations:
        # if yes, let's to propagate its position on the epoch
        # and get the azimute and altitude from user position:
        ve.getPosition_azalt(geo_observer, XYZ_observer, epoch )           # with
sgp4
    # now, is this sat at this epoch overs the mask that user has marked?:
    if ve.alt > mask:
        # then, search if it is over the skyline:
        if skyline == "on":
            azSat_Trunc = truncate(ve.az) # truncate azimute value to
multip-5 to comapre withn skylines points
            for skyL_point in skyL:
                if skyL_point[0] == azSat_Trunc: # compare only the
skyline-point with the same azimute as sat's one
                    altDif = ve.alt - skyL_point[1] # altDif = (skyline
altitude) - (sat altitude)
                    # if sat is over the skyline, then put it into a list
with sats that will be plotted:
                    if altDif >= 0:
                        overMask_sats_AzAlt.append([ve.az, ve.alt, ve.norad])
                        overMask_sats_XYZ[ve.nickname] = [ve.X_ITRF,
ve.Y_ITRF, ve.Z_ITRF]
                    else:
                        overMask_sats_AzAlt.append([ve.az, ve.alt, ve.norad])
                        overMask_sats_XYZ[ve.nickname] = [ve.X_ITRF, ve.Y_ITRF,
ve.Z_ITRF]
                #
# packed everything for plotting:
plotSat_List.append(epochNum)
plotSat_List.append(overMask_sats_AzAlt)
# packed everything for DOP:
XYZSat_List.append(epochNum)
XYZSat_List.append(overMask_sats_XYZ)

return plotSat_List, XYZSat_List, resultats

```

e. Código *DOP.py*

```
"""
-----
Name:      DOP
Purpose:   This script calculate DOP values.
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
-----
"""

import numpy as np
from math import sqrt, pow
import itertools
from random import shuffle

def dop(XYZ_satGroup, XYZ_obs):
    x, y, z = XYZ_obs[0], XYZ_obs[1], XYZ_obs[2]

    # measurement residual equations
    vector_list = []
    for sat in XYZ_satGroup:
        xsat, ysat, zsat = sat[0], sat[1], sat[2]
        R = sqrt(pow(xsat-x,2) + pow(ysat-y,2) + pow(zsat-z,2))
        i = -((xsat-x)/R)
        j = -((ysat-y)/R)
        k = -((zsat-z)/R)
        l = 1

        vector = [i,j,k,l]
        vector_list.append(vector)

    A = np.array(vector_list)

    # AT = transpose of A
    AT = A.transpose()

    # AT*A (AT is 4x4 matrix, A is 4x4 matrix, result is 4x4)
    ATA = [[0,0,0,0],
            [0,0,0,0],
            [0,0,0,0],
            [0,0,0,0]]

    # iterate through rows of AT
    for i in range(len(AT)):
        # iterate through columns of A
        for j in range(len(A[0])):
            # iterate through rows of A
            for k in range(len(A)):
                ATA[i][j] += AT[i][k] * A[k][j]

    # Q = (AT * A)^-1
    Q = np.linalg.inv(ATA)

    PDOP = sqrt(Q[0][0] + Q[1][1] + Q[2][2])
```

```

TDOP = sqrt(Q[3][3])
GDOP = sqrt(pow(PDOP,2) + pow(TDOP,2))
HDOP = sqrt(Q[0][0] + Q[1][1])
VDOP = sqrt(Q[2][2])

return GDOP, PDOP, TDOP, HDOP, VDOP

def getDOP_values(XYS_obs, sat_dicc):

    pack_minGDOP_list = []
    all_minGDOP_list = []

    # list satellite's nicknames:
    sat_list = sat_dicc.keys()

    # uncomment next line if you want explore all possible combination --> minium
    # pack:4 satellites, maximum pack: all satellites.
    #num_of_Sats_packs = range(4, len(sat_list)+1)
    num_of_Sats_packs = [len(sat_list)-1, len(sat_list)] # calclate DOP ussing pack
    of n and n-1 satellites (n = all visible satellites).

    for num_of_Sats in num_of_Sats_packs:

        # get all group of n satellites combinations
        combinations = list(itertools.combinations(sat_list, num_of_Sats))

        # calculate DOP values for each grup-of-satellites combination:
        DOP_dicc = {}

        for group in combinations:
            # get satellite coordinates from nickname
            XYZ_group = []
            position_in_group = 0
            for sat in group:
                name_Sat = group[position_in_group]
                coord_Sat = sat_dicc[name_Sat]
                XYZ_group.append(coord_Sat)
                position_in_group = position_in_group +1
            # calculate DOPs values and put into a dicctionary (key = list with
            # nickname's sats of group).
            DOP_dicc[group] = dop(XYZ_group, XYS_obs)

        # find the minium value of GDOP for this pack combination.
        pack_GDOP_list = []
        for key, value in DOP_dicc.items():
            #print key, value
            pack_GDOP_list.append(value[0])
        pack_GDOP_list.sort() # sort from lowest to highest

        # get the winner group combination
        for key, value in DOP_dicc.items():
            if value[0] == pack_GDOP_list[0]:
                pack_min_GDOP = [key, value]
            else:
                pass
        pack_minGDOP_list.append(pack_min_GDOP[1])

```

```
# find the minium value of GDOP of all packs combinations.
for pack_minGDOP_value in pack_minGDOP_list:
    all_minGDOP_list.append(pack_minGDOP_value[0])
all_minGDOP_list.sort()

# get the absolutte winner combination
for key, value in DOP_dicc.items():
    if value[0] == all_minGDOP_list[0]:
        winner_group = [key, value]
    else:
        pass

return winner_group
```

f. Código *Capture_coords.py*

```
"""
-----
Name:      Capture_coords
Purpose:   This script capture coordinates from canvas
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
-----

"""

from PyQt4.QtCore import *
from PyQt4.QtGui import *

from qgis.core import *
from qgis.gui import *

from ..gnss_planning_dialog import GNSSplanningDialog

class CaptureCoords(QgsMapTool):
    def __init__(self, iface):
        QgsMapTool.__init__(self, iface.mapCanvas())
        self.canvas = iface.mapCanvas()

        self.dlg = GNSSplanningDialog()

    def canvasReleaseEvent(self, event):
        #Get the click
        crsSrc = self.canvas.mapRenderer().destinationCrs()

        QApplication.setOverrideCursor(Qt.WaitCursor)
        self.x_crs = event.pos().x()
        self.y_crs = event.pos().y()
        point = self.canvas.getCoordinateTransform().toMapCoordinates(self.x_crs,
        self.y_crs)

        #convert coords to EPSG:4326
        crsWGS = QgsCoordinateReferenceSystem(4326)
        xform_WGS84 = QgsCoordinateTransform(crsSrc, crsWGS)
        point_WGS84 = xform_WGS84.transform(QgsPoint(point.x(),point.y()))
        QApplication.restoreOverrideCursor()
        self.x_WGS = str(point_WGS84.x())
        self.y_WGS = str(point_WGS84.y())

        #convert coords to EPSG:25831
        crsETRS89 = QgsCoordinateReferenceSystem(25831)
        xform_ETRS89 = QgsCoordinateTransform(crsSrc, crsETRS89)
        point_ETRS89 = xform_ETRS89.transform(QgsPoint(point.x(),point.y()))
        QApplication.restoreOverrideCursor()
        self.x_UTM = str(point_ETRS89.x())
        self.y_UTM = str(point_ETRS89.y())
```

g. Código transform.py

```
"""
-----
Name:      transform
Purpose:   This script transform coordinates into diferents systems.
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
"""

import numpy as np
from math import pi, pow, acos, asin, atan, sqrt, sin, cos, radians, atan2
import math
import os
from numpy import array, cross, einsum, zeros_like

# CONSTANTS:
# -----
# Time:
DAY_S = 86400.0
T0 = 2451545.0
_second = 1.0 / (24.0 * 60.0 * 60.0)

# Angles.
tau = 6.283185307179586476925287 # lower case, for symmetry with math.pi

# WGS84
e2 = 0.0066943800
a = 6378137

# -----
# Rotation
def rot_x(theta):
    c = math.cos(theta)
    s = math.sin(theta)
    return array([(1.0, 0.0, 0.0), (0.0, c, -s), (0.0, s, c)])

def rot_y(theta):
    c = math.cos(theta)
    s = math.sin(theta)
    return array([(c, 0.0, s), (0.0, 1.0, 0.0), (-s, 0.0, c)])

def rot_z(theta):
    c = math.cos(theta)
    s = math.sin(theta)
    zero = theta * 0.0
    one = zero + 1.0
    return array((c, -s, zero), (s, c, zero), (zero, zero, one)))

#
# def theta_GMST1982(jd_ut1):
#     '''Return the angle of Greenwich Mean Standard Time 1982 given the JD.
#     This angle defines the difference between the idiosyncratic True
```

*Equator Mean Equinox (TEME) frame of reference used by SGP4 and the more standard Pseudo Earth Fixed (PEF) frame of reference.
From AIAA 2006-6753 Appendix C.*

```

'''
```

```

t = (jd_ut1 - T0) / 36525 # One sidereal second is approximately 365.25/366.25

g = 67310.54841 + 8640184.812866 * t + 0.093104 * math.pow(t,2) -(0.0000062) *
math.pow(t,3) #GMST (in seconds at UT1=0)
dg = 8640184.812866 * t + 0.093104 * 2 * math.pow(t,2) -(0.0000062) *
math.pow(t,3)

theta = (jd_ut1 % 1.0 + g * _second % 1.0) * tau
theta_dot = (1.0 + dg * _second / 36525.0) * tau
return theta, theta_dot
```

```

# -----
```

```

def julian_day(year, month, day):
    """Given a proleptic Gregorian calendar date, return a Julian day int."""
janfeb = month
return (day
        + 1461 * (year + 4800 - janfeb) // 4
        + 367 * (month - 2 + janfeb * 12) // 12
        - 3 * ((year + 4900 - janfeb) // 100) // 4
        - 32075)
```

```

# -----
```

```

def julian_date(year, month, day, hour, minute, second):
    """Given a proleptic Gregorian calendar date, return a Julian date float."""
    return julian_day(year, month, day) - 0.5 + (
        second + minute * 60.0 + hour * 3600.0) / DAY_S
```

```

# -----
```

```

def TEME_to_ITRF(jd_ut1, rTEME, vTEME, xp=0.0, yp=0.0):
    '''Convert TEME position and velocity into standard ITRS coordinates.
    This converts a position and velocity vector in the idiosyncratic
    True Equator Mean Equinox (TEME) frame of reference used by the SGP4
    theory into vectors into the more standard ITRS frame of reference.
    The velocity should be provided in units per day, not per second.
    From AIAA 2006-6753 Appendix C.'''
    theta, theta_dot = theta_GMST1982(jd_ut1)
    zero = theta_dot * 0.0
    angular_velocity = array([zero, zero, -theta_dot])
    R = rot_z(-theta)

    if len(rTEME) == 1:
        rPEF = (R).dot(rTEME)
        vPEF = (R).dot(vTEME) + cross(angular_velocity, rPEF)
    else:
        rPEF = einsum('ij...,j....>i...', R, rTEME)
        vPEF = einsum('ij...,j....>i...', R, vTEME) + cross(
            angular_velocity, rPEF, 0, 0).T

    if xp == 0.0 and yp == 0.0:
        rITRF = rPEF
```

```
vITRF = vPEF
else:
    W = (rot_x(yp)).dot(rot_y(xp))
    rITRF = (W).dot(rPEF)
    vITRF = (W).dot(vPEF)

return rITRF, vITRF

# -----
def ITRF_to_geographic(XYZ_coordinates):      # WGS84
    '''Convert geocentric coordinates into geographics coordinates.
    ...
    X = XYZ_coordinates[0]
    Y = XYZ_coordinates[1]
    Z = XYZ_coordinates[2]

    D = sqrt((pow(X,2) + pow(Y,2)))
    lat0 = atan(Z/(D*(1-e2)))

    # iteration
    #1
    N = a/sqrt(1-e2*sin(pow(lat0,2)))
    lat0 = atan((Z+e2*N*sin(lat0))/D)
    #2
    N = a/sqrt(1-e2*pow(sin(lat0),2))
    lat0 = atan((Z+e2*N*sin(lat0))/D)
    #3
    N = a/sqrt(1-e2*pow(sin(lat0),2))
    lat0 = atan((Z+e2*N*sin(lat0))/D)

    lat = lat0*180/pi
    lon = atan(Y/X)*180/pi
    h = D/cos(radians(lat))-N

    return lon, lat, h

# -----
def geographic_to_ITRF (geo_coordinates):      # WGS84
    '''Convert geographic coordenates into geocentric coordinates.
    ...
    lon = geo_coordinates[0]
    lat = geo_coordinates[1]
    h = geo_coordinates[2]

    N = a/sqrt(1-e2*pow(sin(radians(lat)),2))

    X = (N+h)*cos(radians(lat))*cos(radians(lon))
    Y = (N+h)*cos(radians(lat))*sin(radians(lon))
    Z = (N*(1-e2)+h)*sin(radians(lat))

    return X, Y, Z

# -----
def ITRF_to_horizontal(geo_observer, XYZ_observer, XYZ_sat):
```

```

# XYZ observer geocentric coordenates
#print observer_XYZ
Xobs = XYZ_observer[0]
Yobs = XYZ_observer[1]
Zobs = XYZ_observer[2]

# lon,lat,h observer geographic coordinates
lon = radians(geo_observer[0])
lat = radians(geo_observer[1])
h = geo_observer[2]

# XYZ satellite geocentric coordenates
Xsat = XYZ_sat[0]
Ysat = XYZ_sat[1]
Zsat = XYZ_sat[2]

# Calculate increments
dX = Xsat - Xobs
dY = Ysat - Yobs
dZ = Zsat - Zobs
incr = [[dX],[dY],[dZ]]
m_incr = np.matrix(incr) # convert to matrix notation

# rotation matrix transposed (R)T
rota = [[-sin(lat)*cos(lon) , -sin(lat)*sin(lon) , cos(lat)],
        [-sin(lon)           , cos(lon)           , 0],
        [cos(lat)*cos(lon)   , cos(lat)*sin(lon) , sin(lat)]] 

m_rota = np.matrix(rota) # convert to matrix notation

# transform to ENU coordenates (north, est, up)
matrix = m_rota * m_incr

north = matrix[0]
est = matrix[1]
up = matrix[2]

# calculate horizontal coordenates
D = sqrt(pow(north,2) + pow(est,2) + pow(up,2))
alpha = atan(est/north)*180/pi
beta = acos(up/D)*180/pi

if north < 0:                      # 2nd and 3rd quadrant
    az = alpha + 180
elif est < 0 and north >= 0: # 4th quadrant
    az = alpha + 360
else:                                # 1st quadrant
    az = alpha

alt = 90-beta

return az, alt, D

```

h. Código *PROP-Receptor.py*

```

"""
-----
Name:      PROP-Receptor
Purpose:   Propagate satellites orbit from broadcast ephemerides
Created:   2017 by Cesc Masdeu Ferrer
Licence:   Free
-----

"""

import math

# -----
# CONSTANTS WGS84:
vLight = 299792458.0          # light velocity
mu = 3986004418000000.0       # gravitational constant (WGS84)
W_earth = 0.000072921151467 # earth rotation velocity (WGS84)

# -----
# SATELLITE-EPEMERIDES OBJECT
class SatelliteEphem():

    def __init__(self, satParameters, pseudoDist):
        self.satP = satParameters
        self.pseudoDist = pseudoDist

        # Navegation parameters:
        self.name = self.satP[0][0:3]
        self.date = self.satP[0][4:14]
        self.time = self.satP[0][15:23]
        self.a0 = float(self.satP[0][23:42].replace("D", "e"))
        self.a1 = float(self.satP[0][42:61].replace("D", "e"))
        self.a2 = float(self.satP[0][61:80].replace("D", "e"))

        self.IODE = float(self.satP[1][4:23].replace("D", "e"))
        self.Crs = float(self.satP[1][23:42].replace("D", "e"))
        self.Dn = float(self.satP[1][42:61].replace("D", "e"))
        self.M0 = float(self.satP[1][61:80].replace("D", "e"))

        self.Cuc = float(self.satP[2][4:23].replace("D", "e"))
        self.e = float(self.satP[2][23:42].replace("D", "e"))
        self.Cus = float(self.satP[2][42:61].replace("D", "e"))
        self.SQR_a = float(self.satP[2][61:80].replace("D", "e"))

        self.toe = float(self.satP[3][4:23].replace("D", "e"))
        self.Cic = float(self.satP[3][23:42].replace("D", "e"))
        self.W0 = float(self.satP[3][42:61].replace("D", "e"))
        self.Cis = float(self.satP[3][61:80].replace("D", "e"))

        self.i0 = float(self.satP[4][4:23].replace("D", "e"))
        self.Crc = float(self.satP[4][23:42].replace("D", "e"))
        self.w = float(self.satP[4][42:61].replace("D", "e"))
        self.W_dot = float(self.satP[4][61:80].replace("D", "e"))

        self.i_dot = float(self.satP[5][4:23].replace("D", "e"))

```

```

self.L2_C = float(self.satP[5][23:42].replace("D", "e"))
self.GPSweek = float(self.satP[5][42:61].replace("D", "e"))
self.L2_P = float(self.satP[5][61:80].replace("D", "e"))

self.precision = float(self.satP[6][4:23].replace("D", "e"))
self.helth = float(self.satP[6][23:42].replace("D", "e"))
self.TGD = float(self.satP[6][42:61].replace("D", "e"))
self.IODC = float(self.satP[6][61:80].replace("D", "e"))

self.toc = float(self.satP[7][4:23].replace("D", "e"))

# PROPAGATION MODEL:
# -----
def propagationModel(self, data_time):
    gpsWeek_pedict, gpsDay_pedict, time = data_time[0], data_time[1],
data_time[2]
    h, m, s = time[0], time[1], time[2]

    t = (60*60*24*gpsDay_pedict) + (60*60*h) + (60*m) + s
    tk = t - self.toe -(self.pseudoDist/vLight) + 86400*7 * (gpsWeek_pedict -
self.GPSweek)

    a = math.pow(self.SQR_a,2)
    n0 = math.sqrt(mu/(math.pow(a,3)))
    n = n0 + self.Dn
    M = self.M0 + n * tk

    E = self.eAnomaly(M)

    v = math.atan((math.sqrt(1-math.pow(self.e,2))*math.sin(E))/(math.cos(E)-
self.e))
    u0 = v + self.w
    u = u0 + self.Cuc * math.cos(2*u0) + self.Cus * (math.sin(2*u0))
    W = self.W0 + ((self.W_dot-W_earth) * tk - (W_earth * self.toe))
    i = self.i0 + self.i_dot*tk + self.Cic*math.cos(2*u0) +
self.Cis*math.sin(2*u0)
    r = a*(1-self.e*math.cos(E)) + self.Crc*math.cos(2*u0) +
self.Crs*math.sin(2*u0)

    self.x = r * math.cos(u)
    self.y = r * math.sin(u)

    self.X = self.x * math.cos(W) - self.y * math.cos(i) * math.sin(W)
    self.Y = self.x * math.sin(W) + self.y * math.cos(i) * math.cos(W)
    self.Z = self.y * math.sin(i)

def eAnomaly(self, M):
    E = M
    cont = 1
    while cont < 7:
        E = M + self.e * math.sin(E)
        cont = cont +1
    return E

```

EXAMPLE:

- Satellites: G10, G12, G15, G19, G24
- Date & time ephemerides: 2017 01 13 20 00 00
- Propagate its position for next 12 days at 20:00:00h

```

# -----
# DATE AND TIME TO PROPAGATE SATELLITE POSITION:
# date_time_n = (GPSweek, GPSday, UTC(hh,mm,ss), "info string: Julian date & UTC
# time")
time_UTC = (20,00,00)
hh, mm, ss = time_UTC[0], time_UTC[1], time_UTC[2]

data_time_1 = (1931, 5, time_UTC, "13/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_2 = (1931, 6, time_UTC, "14/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_3 = (1932, 0, time_UTC, "15/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_4 = (1932, 1, time_UTC, "16/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_5 = (1932, 2, time_UTC, "17/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_6 = (1932, 3, time_UTC, "18/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_7 = (1932, 4, time_UTC, "19/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_8 = (1932, 5, time_UTC, "20/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_9 = (1932, 6, time_UTC, "21/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_10 = (1933, 0, time_UTC, "22/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_11 = (1933, 1, time_UTC, "23/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_12 = (1933, 2, time_UTC, "24/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")
data_time_13 = (1933, 3, time_UTC, "25/01/2017 "+str(hh)+":h"+str(mm)+"m"+str(ss)+"s")

data_time_List = (data_time_1, data_time_2, data_time_3, data_time_4,
                  data_time_5, data_time_6, data_time_7, data_time_8, data_time_9,
                  data_time_10, data_time_11, data_time_12, data_time_13)

# BROADCAST EPHEMERIDES:
# Ref station: BELL00SPN
# Read Navegation file for reading
navFile = open('BELL00ESP_R_20170130000_01D_GN.rnx', 'r')

satDic = {}
first = "yes"
satParam = []

for line in navFile:
    if line[0] == "G":
        if first == "no":
            satDic[satNameDateTime] = satParam
        satNameDateTime = line[0:23]
        satParam = []
        satParam.append(line)
        first = "no"
    else:
        satParam.append(line)

# Propagate each satellite:
listOfSatellites = ['G10', 'G12', 'G15', 'G19', 'G24']
# Take pseudodistance from observable file BELL013T.17o
pseudoDistList = [25533310.170, 20243000.646, 25273994.352, 24379762.558,
                  21172041.640]

```

```

for satellite, pseudoDist in zip(listOfSatellites, pseudoDistList):
    stringInput = str(satellite) + " 2017 01 13 20 00 00" # write the ephem date-time
    sat = SatelliteEphem(satDic[stringInput], pseudoDist)

    # Satellite positions XYZ geocentric coordinates.
    print
    print "Satellite positions XYZ geocentric coordinates:"
    print "-----"
    for data_time in data_time_List:
        sat.propagationModel(data_time)
        print data_time[3], "X = {0:.2f}".format(sat.X), "Y = {0:.2f}".format(sat.Y),
    "Z = {0:.2f}".format(sat.Z)

    # -----
    # PRINT RESULTS:

##    # Uncomment if you want to print ephemerides parameters
##    # Navegation parameters:
##    print "Navegation parameters:"
##    print "-----"
##    print "sat.name = ", sat.name
##    print "sat.date = ", sat.date
##    print "sat.time = ", sat.time
##    print "sat.a0 = ", sat.a0
##    print "sat.a1 = ", sat.a1
##    print "sat.a2 = ", sat.a2
##
##    print "sat.IODE = ", sat.IODE
##    print "sat.Crs = ", sat.Crs
##    print "sat.Dn = ", sat.Dn
##    print "sat.M0 = ", sat.M0
##
##    print "sat.Cuc = ", sat.Cuc
##    print "sat.e = ", sat.e
##    print "sat.Cus = ", sat.Cus
##    print "sat.SQR_a = ", sat.SQR_a
##
##    print "sat.toe = ", sat.toe
##    print "sat.Cic = ", sat.Cic
##    print "sat.W0 = ", sat.W0
##    print "sat.Cis = ", sat.Cis
##
##    print "sat.i0 = ", sat.i0
##    print "sat.Crc = ", sat.Crc
##    print "sat.w = ", sat.w
##    print "sat.W_dot = ", sat.W_dot
##
##    print "sat.i_dot = ", sat.i_dot
##    print "sat.L2_C = ", sat.L2_C
##    print "sat.GPSw = ", sat.GPSweek
##    print "sat.L2_P = ", sat.L2_P
##
##    print "sat.prec = ", sat.precision
##    print "sat.helth = ", sat.helth
##    print "sat.TGD = ", sat.TGD
##    print "sat.IODC = ", sat.IODC
##
##    print "sat.toc = ", sat.toc

```