UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Evaluación de la calidad de la estimación de la posición y orientación de un vehículo aéreo a partir de fusión sensorial

---

GRADO EN INGENIERÍA INFORMÁTICA
TRABAJO DE FIN DE GRADO
JULIO DE 2017
LAS PALMAS DE GRAN CANARIA

AUTOR: Indra González Cabrera

TUTORES: José Daniel Hernández Sosa
Dpto. de Informática y Sistemas
Francisco Suárez González
Aerolaser System S.L.

Agradecimientos

A Elvis y a mi familia, por apoyarme y apostar por mí cuando ni yo lo hacía. Esto también lo han conseguido ustedes.

A mis tutores, especialmente a José Daniel Hernández Sosa, porque sin tu esfuerzo, dedicación e implicación, este proyecto no habría sido posible. Gracias, de corazón.

A todos los pequeños informáticos que han hecho este largo y arduo trayecto mucho más fácil y divertido.

Índice

Índice de ilustraciones

Introducción

Aerolaser Systems SL es una empresa topográfica dedicada a la producción de información digital a partir de sensores embarcados en aeronaves, además de las correspondientes aplicaciones para cartografía, ortofotos, fotografía térmica u obtención de modelos digitales.

Uno de los aspectos fundamentales del proceso es la toma de datos para su posterior posprocesado y su uso en aplicaciones para clientes. Esta recolección de medidas cartográficas se lleva a cabo durante el vuelo y, por tanto, los errores generados, tanto por ruido como por vibraciones e interferencias de campos magnéticos, son considerables.

A fin de mejorar la precisión de la toma y recolección de medidas, la empresa ha desarrollado hardware específico. Este nuevo dispositivo permitirá que los instrumentos de medida, antes fijos, puedan instalarse sobre una estructura móvil. Esto se lleva a cabo para compensar el movimiento del vehículo que transporta estos instrumentos, de tal forma que se evita que queden zonas que hayan escapado al barrido a la hora de tomar las mediciones.

Uno de los principales problemas que desea resolver la empresa es que, para poder utilizar de forma efectiva este nuevo hardware desarrollado, es preciso encontrar una estimación adecuada de la posición y la orientación del vehículo móvil en el que se encuentran los dispositivos de medición. Dicho vehículo incorpora una IMU (unidad de medición inercial) y un GPS (sistema de posicionamiento global) y, tal y como se mencionó anteriormente, esta instrumentación presenta ciertas limitaciones. Los acelerómetros y los giróscopos integrados en la IMU presentan un error que crece en el tiempo. Así, a pesar de los errores del GPS son asumibles, por la integración de los sistemas INS y GPS las medidas se ven contaminadas y no pueden ser tomadas directamente, sino que deben estimarse.

De esta forma, el trabajo propuesto tiene por objeto el estudio de diferentes métodos de estimación a partir de datos multisensoriales. Para la alternativa seleccionada, se realizarán análisis con el propósito de determinar la precisión del resultado y la robustez del mismo ante variaciones en los parámetros de los modelos y algoritmos empleados. Se generarán resultados comparativos, tanto a nivel numérico como gráfico, que ofrezcan soporte al personal técnico de la empresa.

Se plantea el desarrollo de un prototipo basado en la simulación de modelos de vehículos y trayectorias, a fin de contrastar los resultados esperados con los obtenidos. Finalmente, se estudiará la viabilidad del traslado de estos resultados a un caso real, como validación de las hipótesis formuladas.

Estructura del documento

En el primer capítulo se tratará la motivación general que ocasiona la elección del proyecto, así como el estado actual de los sistemas de navegación y del filtro Kalman. Se tratarán también los objetivos a cumplir, así como las competencias específicas relacionadas con el desarrollo del trabajo y las aportaciones del trabajo a la sociedad y a la ciencia.

En el segundo capítulo se explicará el proceso de estudio previo y análisis del problema, así como su planteamiento inicial y la documentación necesaria para tratarla. Esta incluirá el estudio de conceptos básicos de navegación y el estudio de las opciones de resolución. Además, se desarrolla el estudio del filtro Kalman y su base matemática.

En el tercer capítulo se tratará el estudio de las alternativas de implementación, tanto desde el punto de vista de tecnologías y plataformas como desde el punto de vista de búsqueda e investigación de librerías y código abierto. Finalmente, será necesario d ocumentar el uso de la librería y de las funcionalidades relativas a la aplicación.

En el cuarto capítulo, a partir del ejemplo seleccionado como alternativa de resolución de la librería descrita anteriormente, se desarrollará un prototipo basado en la simulación de modelos de trayectorias generadas sintéticamente, que llevará a cabo pruebas sobre diversos parámetros del sistema para analizar su impacto sobre la bondad de la estimación.

Así, se diseñará una aplicación de análisis de datos sintéticos mediante el diseño de los escenarios y los casos de prueba, el diseño de los experimentos a realizar y la evaluación de los resultados.

En el quinto capítulo, cuando la fase de dantos artificiales proporciona las conclusiones necesarias y a fin de contrastar los resultados de datos sintéticos, se estudiará la viabilidad del traslado de estos resultados en un caso real, como validación o refutación de las hipótesis formuladas y como un acercamiento a la resolución final del problema descrito que forma la base de este trabajo.

Así, se diseñará una aplicación de análisis de datos reales, explicando la colaboración con la empresa y los programas de lectura utilizados. Finalmente, se tratará la

adaptación del ejemplo de LargeHeading con datos reales y se explicarán las otras opciones de implementación de estimadores.

En el último capítulo se tratarán las conclusiones obtenidas del proyecto y las conclusiones del proceso de realización del trabajo. Además, se tratarán los temas de posibles trabajos futuros.

CAPÍTULO 1: Introducción

En el primer capítulo se tratará la motivación general que ocasiona la elección del proyecto, así como el estado actual de los sistemas de navegación y del filtro Kalman. Se tratarán también los objetivos a cumplir, así como las competencias específicas relacionadas con el desarrollo del trabajo y las aportaciones del trabajo a la sociedad y a la ciencia.

Motivación general

La idea y la motivación del presente trabajo surge a partir de la realización de las prácticas de empresa en Aerolaser Systems SL. Esta es una empresa que se ha ido consolidando con los años gracias a la innovación y al desarrollo en un campo que ha ido pereciendo, como es la topografía. Por tanto, el peso que esta institución otorga al trabajo de investigación hace de ella un entorno ideal para realizar un proyecto de investigación de computación.

Más específicamente, la motivación se basa, en cierta medida, en el carácter matemático del problema a tratar, íntimamente relacionado con mi intensificación y un aspecto fundamental en la informática. Adicionalmente, la oportunidad de llevar a cabo el proyecto con el apoyo de una empresa aporta la posibilidad de ver materializado el trabajo realizado, además del interés que proporciona contribuir a la mejora del sector.

Estado actual

"La determinación de la posición, velocidad y orientación de un vehículo resulta de gran importancia en diversas aplicaciones y desde los inicios de la navegación se han propuesto distintas técnicas para lograrlo." (Grondona, 2013)

SISTEMAS DE NAVEGACIÓN

Desde los inicios de la navegación se conocen los problemas que los sistemas tradicionales de navegación astronómica presentan. Estos requieren el exacto conocimiento del tiempo, dependen de los factores meteorológicos, presentan falta de continuidad, son precisos muchos cálculos para obtenerlos, etc. En un intento de resolución o disminución de estas limitaciones surge la radionavegación, uno de los primeros avances electrónicos en la navegación. Así, forma parte de la llamada navegación electrónica, que viene a completar y no a sustituir conocimientos, colaborando en un manejo más racional y seguro de los sistemas inerciales.

Con la radionavegación, un receptor radio instalado a bordo era capaz de recibir información sobre la situación meteorológica, tráfico portuario, etc. Con la instalación en tierra de estaciones emisoras de forma continua o programa comienzan los avances, pudiendo obtener información de señales horarias para los cálculos, partes meteorológicos, avisos, controles, etc. Estos equipos radioemisores y radioreceptores se fueron mejorando, disminuyendo su coste y tamaño y haciéndolos más potentes.

Durante la 1ª Guerra Mundial surgen los equipos radiobuscadores de dirección y radiogoniómetros, que incorporan una antena direccional que permite disponer de una línea de posición y demora trazada una estación de emisión conocida.

Mediante los estudios de propagación de ondas que ocasionan los avances nombrados anteriormente, se consigue, antes de la 2ª Guerra Mundial, el desarrollo de los equipos RADAR, capaces de obtener una distancia de un objeto y, con ello, una línea de posición circular.

Ya la 2ª Guerra Mundial fue el gran banco de pruebas de nuevos sistemas de radionavegación. Así, Gran Bretaña desarrolló el sistema GEE -precursor del DECCA- y, casi simultáneamente, Estados Unidos desarrolló un equipo similar, el LORAN A.

Con el final de la guerra, todos estos avances, antes secretos, pasan al uso público con la aprobación de la Conferencia Internacional de La Haya, potenciándose la investigación en ese terreno. Con esto, van surgiendo avances, como las radiobalizas RAMARK y RACON. También surgen DECCA y el LORAN C, sucesores de los creados en la guerra.

Como resultado, surge OMEGA, con la colaboración de británicos y norteamericanos, el primer sistema en ofrecer cobertura mundial con ocho estaciones y constituyendo uno de los sistemas más importantes hasta que, en 1997, cesó su servicio.

Finalmente, en 1957 y a raíz del lanzamiento del primer satélite artificial, el SPUTNIK, comienzan los sistemas de radionavegación por satélite, que utilizaban las medidas Doppler para la determinación de un punto terrestre. El primero en desarrollarse fue el TRANSIT, creado por la Armada de los Estados Unidos y siendo el precursor del sistema que se usa en la actualidad, el GPS.

El GPS (Global Positioning System) es un sistema de posicionamiento global que nace de los resultados de la investigación del TRANSIT, mejorando sustancialmente a su predecesor. Con 24 satélites, ofrece cobertura mundial continua en tres dimensiones con cualquier condición meteorológica y con niveles de exactitud nunca antes conseguidos en los años 80. (Armada, 2010)

En la actualidad, los sistemas de navegación más utilizados serían los GNSS (Global Navigation Satellite Systems), como el GPS, nombrado anteriormente y el GLONASS (Global'naya Navigatsionnaya Sputnikovaya Sistema), que sería el homólogo ruso del GPS. Estos sistemas de navegación permiten medir la posición y la velocidad absoluta del vehículo. (Grondona, 2013) En comparación, las prestaciones que ofrecen son muy similares. Sin embargo, la popularidad del GPS es mucho mayor que la del GLONASS. Esto ha ido cambiando en los últimos años por las condiciones políticas de Rusia, lo que ha impulsado a este sistema y le ha otorgado mucha más popularidad de la que gozaba en el pasado. Usar uno frente al otro no presenta mayor diferencia o ventaja, sin embargo, el uso de dos sistemas de posicionamiento en lugar de uno ofrecen una mejora de la precisión en condiciones estándar de un 60-70% a unas cifras cercanas al 100%. Por ello, puede decirse que el futuro de estos sistemas de posicionamiento estará en la combinación de estos dos nombrados anteriormente y de otros sistemas, como el sistema chino BeiDou o el sistema Galileo de la Unión Europea. (Novosti, 2014)

Una de las principales desventajas de estos sistemas -los sistemas GNNS y los GLONASS- es que poseen una baja tasa de muestreo (entre diez milisegundos y un segundo) dependiendo del receptor y la técnica utilizada. Esta característica no los hace aptos para vehículos con altas dinámicas ya que, además, presentan una alta vulnerabilidad a interferencias externas. Esta falta de robustez es el principal motivo por el cual este tipo de sistemas no se utiliza en aplicaciones de aviación comercial o lanzadores satelitales.

Sin embargo, cuentan con la gran ventaja de poseer un error acotado dentro de un margen que se puede determinar a priori.

El otro tipo que se utiliza son los INS (Inertial Navigation Systems), que basan su funcionamiento en la integración de las mediciones de giróscopos y acelerómetros. Estos, a diferencia de los anteriores, presentan la ventaja de la alta velocidad de muestreo y alta inmunidad a interferencias externas. Sin embargo, cuentan con la desventaja de que el error en la determinación de la posición, velocidad y orientación del vehículo crece de manera no acotada en el tiempo. Este error depende, en gran medida, de la calidad de los giróscopos y acelerómetros utilizados, lo que hace que los sistemas de INS de precisión, utilizados en aviación comercial o en sistemas aeroespaciales, tengan un coste económico muy elevado.

En la última década se han desarrollado los llamados instrumento inerciales basados en MEMS (Microelectromechanical Systems) que presentan la ventaja de poseer bajo coste y tamaño reducido. Sin embargo, aún no han logado el desarrollo de giróscopos y acelerómetros MEMS que satisfagan los requerimientos de un sistema inercial de navegación de alta precisión. (Grondona, 2013)

Como puede verse, las desventajas de los sistemas INS se compensan por los sistemas GNSS y viceversa. Debido a la complementariedad de sus atributos, integrar ambos sistemas de forma débilmente integrada (*loosely coupled*) o fuertemente integrada (*tightly coupled*) es beneficioso en términos de mejorar la precisión, la continuidad, la fiabilidad, etc. (Liu & Chang, 2016)

Este método basado en la explotación de las ventajas y la compensación de las desventajas de ambos sistemas de navegación se denomina navegación integrada y es una técnica que permite reducir los costos y el tamaño de los sistemas de navegación sin sacrificar la precisión. Esta técnica, por otra parte, requiere mayor capacidad de cómputo dado que es necesario ejecutar algoritmos complejos para estimar los errores de cada uno de los instrumentos del sistema. Gracias al avance del desarrollo de microprocesadores es posible contar con un computador de navegación capaz de resolver estos algoritmos sin incrementar significativamente los costes del sistema. (Grondona, 2013)

De esta forma, en la actualidad, estos son los sistemas que se utilizan en la navegación aeroespacial y en aviación comercial. Estos sistemas integrados, como hemos nombrado

anteriormente, permiten obtener posición y orientación del vehículo en vuelo con una precisión notablemente mayor, con lo que se han mejorado las condiciones de los vuelos, decrementando los fallos de posición y la intervención humana en el proceso. Con esto se han reducido los riesgos en la navegación comercial y se ha mejorado de forma significativa la calidad de vuelo, pues con más precisión sobre los datos reales y los errores, es posible compensar dichos errores para obtener una navegación más precisa, correcta e independiente de las condiciones externas.

Además, es digno de mención el auge de los vehículos no tripulados como los drones, en los que los sistemas inerciales son esenciales para el funcionamiento, como se comentará en mayor extensión en el apartado de Aportaciones.

Todos estos avances nos han permitido avanzar de forma exponencial en este campo, mejorando en el desarrollo de vehículos aeroespaciales que nos permitirán en el futuro, explorar y expandir los límites del conocimiento que, en última instancia, es el objetivo de la ciencia.

FILTRO KALMAN

Como se comentó anteriormente, para la obtención de las medidas reales de los sistemas integrados de navegación, es necesario utilizar algoritmos para estimar el error y obtener la medición correcta.

Los estudios de estimación dinámica han tenido especial fuerza en los últimos años y gracias al avance de la computación como herramienta de soporte para guardar información y realizar cálculos y a la necesidad creciente de métodos de estimación para realizar predicciones en tiempo real. Entre las herramientas matemáticas para este propósito, Kalman destaca entre los demás ya que constituye el principal procedimiento para estimar sistemas dinámicos representados en la forma de estado-espacio.

El filtro Kalman fue desarrollado en 1960 por Rudolf E. Kalman basándose en el Observador de Luenberger, con la peculiaridad de que funciona cuando se ha sometido el sistema a ruido blanco aditivo. Además, el filtro Kalman es más efectivo y autónomo, al elegir de forma autónoma la retroalimentación del error óptima, mientras que el algoritmo del observador requiere que esa ganancia se proporcione de forma manual.

Se basa en un algoritmo recursivo que estima el estado no observable de un sistema dinámico dado un conjunto de observaciones que proporcionan información y

caracterización del sistema en cada momento. Opera mediante un mecanismo de predicción y corrección, pronosticando un nuevo estado a partir de su estimación previa y la corrección que añade, de tal forma que el error de predicción se minimiza estadísticamente. Es un filtro muy poderoso, en el sentido de que es capaz de dar soporte a estimaciones pasadas, presentes y de futuros estados, aún cuando la naturaleza del sistema de modelado es desconocida.

A pesar de que se desarrolló hace más de cincuenta años, su uso se ha generalizado recientemente en una variedad de aplicaciones que van desde la realidad virtual hasta la estimación de la posición y orientación en sistemas de navegación integrados, siendo una aplicación común la guía, navegación y control de vehículos, especialmente naves espaciales. Además, el filtro es ampliamente usado en campos como procesamiento de señales y econometría. Por tanto, es el mejor estimador posible o un estimador muy efectivo para una amplitud de problemas creciente, por lo que es que se utiliza con más frecuencia, en especial en los entornos de posicionamiento y en las nuevas tecnologías. Además, presenta la ventaja de ser fácil de comprender y de la adaptabilidad, pues es posible usarlo con pocas herramientas conceptuales. (Prado Obando, 2005)

En cuanto a las primeras aplicaciones del filtro, este está íntimamente relacionado con los sistemas de navegación, siendo la primera aplicación completa del filtro como el sistema de guía del sistema astronáutico en el Programa Apolo de la NASA. El éxito de este filtro en dichas investigaciones impulsó el uso del mismo en otros proyectos de navegación de transporte aéreo.

En estos proyectos, el filtro Kalman fue capaz de resolver el problema de la fusión de datos asociado con la combinación de los datos del sistema de navegación integrado para lograr una aproximación global de la trayectoria de la aeronave con éxito. Desde entonces, el filtro es una parte integral de la estimación de trayectorias a bordo de aeronaves y en el diseño de sistemas de control.

De esta forma, el filtro de Kalman se considera el gran logro en la teoría de estimación del siglo XX ya que ha sido esencial en muchos logros tecnológicos y científicos desde su introducción. Es seguro decir que este filtro ha sido una de las tecnologías que hizo posible, junto con los sistemas integrados, la era espacial, puesto que la precisión y eficiencia en la navegación en astronavegación no hubiese sido posible sin la aportación del filtro.

Actualmente, además de los campos anteriormente nombrados, el principal uso de Kalman ha sido en sistemas de control modernos, en el seguimiento y navegación de todo tipo de vehículos y en el diseño predictivo de estimación de los mismos.

Objetivos

El trabajo propuesto tiene como objeto el estudio de los diferentes métodos de estimación a partir de datos multisensoriales. Para ello, en un proceso previo de documentación e investigación, se estudiarán las posibles alternativas en el plano teórico, teniendo en cuenta, también, las limitaciones que presenta la aplicación práctica del algoritmo elegido. Además, influirá en la decisión de la alternativa las posibilidades de aplicación del algoritmo en los diferentes lenguajes y plataformas software y hardware de las que la empresa dispone y tiene acceso.

Para la alternativa seleccionada, en primer lugar, se estudiará en profundidad el algoritmo usado y, posteriormente, se diseñarán las pruebas necesarias para realizar análisis con el propósito de determinar la precisión del resultado y la robustez del mismo ante variaciones en los parámetros de los modelos y algoritmos empleados.

Cuando se lleven a cabo las pruebas diseñadas a partir del estudio del modelo, se generarán resultados comparativos, tanto a nivel numérico como gráfico. Estos resultados nos ofrecerán un análisis previo que servirá como soporte al personal técnico de la empresa encargado de llevar a cabo el desarrollo completo y la implementación del filtro en el software y hardware ya existente.

Se plantea el desarrollo de un prototipo basado en la simulación de modelos de vehículos y trayectorias, lo más parecidos al caso real posible, a fin de ser capaces de contrastar los resultados esperados con los obtenidos. Además, las simulaciones dentro del marco teórico se llevan a cabo dándole un peso mayor en la estimación a los parámetros que la empresa considera más importantes a la hora de realizar la integración del hardware.

En este caso, el parámetro más determinante a la hora de integrar el código en el hardware ya desarrollado sería el *heading* o rumbo del vehículo. Por tanto, el algoritmo desarrollado hará énfasis en la estimación de este parámetro.

Finalmente, se estudiará la viabilidad del traslado de estos resultados a un caso real, como validación de las hipótesis formuladas. Para ello, se deberá simular una lectura de los datos en vuelo, es decir, en tiempo real, y sin que se haya llevado a cabo el posprocesado de la información. Con esta simulación del caso real con los datos leídos de la misma forma que se obtendrían de la IMU en vuelo conseguiremos caracterizar mejor el problema de acuerdo al objetivo final que sería el del desarrollo de software para integrar el nuevo hardware con los instrumentos de medida en el vehículo.

Aportaciones

Como se ha nombrado anteriormente, la optimización del filtro Kalman en el marco de los sistemas de navegación integrados aporta precisión a otras tecnologías sin la cuál no sería posible su funcionamiento.

Con este trabajo se pretende establecer los parámetros que influyen en la calidad y en la precisión del filtro basándose en la función sensorial. Con esto, en la práctica, seríamos capaces de mejorar la precisión que proporciona Kalman y reducir el tiempo de convergencia, tras el cuál el filtro comienza a ser realmente efectivo. Esta calibración debería ser lo más reducida en el tiempo posible, ya que, el objetivo final es un sistema preciso que sea capaz de responder a tiempo real con la mayor precisión posible desde el momento inicial.

En un principio y de forma directa, los resultados de este trabajo se utilizarían para la implementación del filtro en el futuro en la IMU integrada en el vehículo de medición. Ello permitiría utilizar la información obtenida en tiempo real en el dispositivo que la empresa ha diseñado para contrarrestar el error proporcionado en la medición.

Más adelante y de forma más amplia, esta mejora en la determinación de los parámetros iniciales del filtro influye en muchos campos del mundo tecnológico que nos atañe. Dado que estos parámetros son la única intervención manual del filtro y son especialmente determinantes en la calidad de la predicción que obtendremos, una aproximación a un proceso previo de determinación de los parámetros óptimos para Kalman podrían constituir una mejora importante en el sector.

Actualmente, además de los campos ya nombrados y explicados en extensión anteriormente, uno de los campos de la aeronavegación más prometedores y que precisan de Kalamn y de los sistemas integrados de navegación para su funcionamiento son los drones o UAV (unmanned aerial vehicule). Durante años se han utilizado los drones, principalmente en el sector militar, pero en la actualidad, la popularización de estos vehículos ha resultado en un afloramiento de aplicaciones para los mismos, como pueden ser en labores de seguridad y rescate, vigilancia, investigación, etc. Con esta demanda y posibles vías y campos de utilización, los avances tecnológicos surgen.

Como explicamos anteriormente, los UAV precisan de sistemas de navegación inercial y de filtros Kalman para su funcionamiento debido a que, a pesar de su pequeño tamaño y el pequeño coste capital que es producirlos con las nuevas tecnologías, es necesario que operen con eficiencia para poder aprovechar las ventajas de estas plataformas baratas en que pueden desarrollarse. Además, el factor que presentan los RPAS (remotely-piloted aircraft systems) de no intervención humana hacen necesario el uso de estas tecnologías para reducir la intervención manual en el proceso.

Por tanto, tanto en los sectores ya establecidos que han usado el Filtro Kalman desde sus inicios, como en los sectores en auge como son los UAVs, precisan de la utilización de un Filtro Kalman cada vez más eficiente, especialmente estos últimos.

Si, como indican las tendencias actuales, el futuro de la aeronavegación va a asentarse sobre la base de los drones, el trabajo de mejorar las condiciones en que se lleva a cabo la predicción de Kalman es esencial para el avance tecnológico futuro.

En este aspecto, el trabajo proporciona una aproximación inicial a lo que sería un proceso previo de determinación de los parámetros óptimos para Kalman, siendo las posibilidades de mejora y las aplicaciones potenciales del trabajo muy amplias.

Justificación de competencias específicas cubiertas

- **CP03: Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.**

CII06: Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.

Una fase esencial del proyecto es la investigación exhaustiva de algoritmos y métodos de resolución al problema de la estimación de las medidas reales sin error para su uso posterior en la integración del nuevo hardware.

Para cada una de las posibles soluciones algorítmicas o aproximaciones, es necesario llevar a cabo un análisis sobre la posibilidad de aplicarlas al problema en cuestión. Por tanto, en el análisis de soluciones y algoritmos en este caso, es preciso decidir y elegir aquella opción que más se ajuste.

En este caso en concreto, entre todas las posibles opciones es necesario elegir aquella que realice la estimación con el menor grado de error y que estime, en la mejor medida posible, el parámetro que más nos atañe que, como se expuso anteriormente, sería el *heading*.

En el apartado de Desarrollo se explica con más detalle esta fase de documentación y elección de la mejor plataforma posible entre las alternativas y la justificación de la elección que se lleva a cabo.

- **CII08: Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.**

  Como se explica anteriormente, es necesario elegir el algoritmo necesario para la resolución del problema.

  Una vez se ha llevado a cabo esta fase previa, es necesario elegir el entorno y el lenguaje de programación óptimo. Para ello, tras la investigación y documentación de las librerías existentes que apoyan el uso del método de resolución elegido, se realiza el proceso de construcción de la aplicación.

  Para el desarrollo de la aplicación, según las librerías encontradas, además de las necesidades del cliente -en nuestro caso, la empresa- y de las condiciones en que

se usará la aplicación, decidimos el lenguaje de programación y el enfoque que se usará.

En este caso, se ha encontrado una librería bastante completa en Matlab que, además, cumple con los requisitos de aproximación del parámetro más sensible, el *heading*. Teniendo en cuenta las peticiones de la empresa y el entorno de uso de la aplicación, esto constituye una buena opción porque Matlab permite una migración del código a lenguajes de programación que utiliza la empresa, como serían C++, con lo que supone una base óptima sobre la que construir la aplicación.

Además, en la fase final de simulación del caso real, es necesario crear software de lectura de datos y de conversión de los mismos, para los cuales fue necesario el uso de librerías de transformaciones geográficas, además del uso de QT. Este último, apoyándose en Visual Studio permitió la generación de programas más flexibles y adaptables puesto que facilita la adaptación del código a múltiples y diversos dispositivos. Con esto conseguimos una aplicación más eficiente, robusta y más fácilmente adaptable y acondicionado para modificaciones en el futuro.

- **CP01: Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática.**

  Con este trabajo se pretende realizar un análisis de los parámetros que influyen en la calidad y precisión del filtro. Con esto, seríamos capaces de establecer resultados comparativos valorando la aportación y la influencia de cada uno de los parámetros en los resultados finales, tanto en simulaciones como en las pruebas de casos con datos reales. Una vez realizada esta fase, seremos capaces de seleccionar los valores óptimos de los parámetros para casos simulados y reales.

  En un principio y de forma directa, esto nos permite mejorar la precisión que proporciona Kalman y reducir el tiempo de convergencia del filtro, lo que hace que mejore el sistema, siendo más efectivo desde momentos más cercanos al momento inicial.

En la práctica, esto se utilizaría en la implementación del filtro en la IMU integrada en el vehículo de medición para usar la información obtenida en tiempo real en el hardware diseñado por la empresa.

De forma más general y como aportación al desarrollo del campo de la informática, esta mejora en la determinación de los parámetros iniciales del filtro y la influencia de los mismos en los resultados finales influye en muchos campos tecnológicos. Como se ha comentado con anterioridad, los parámetros que son caso de estudio constituyen una de las pocas intervenciones manuales y son especialmente determinantes en la calidad de la predicción que obtendremos. Por tanto, una acercamiento a un proceso previo de determinación de los parámetros óptimos para Kalman podrían constituir una mejora importante en el sector.

CAPÍTULO 2: Documentación

En el segundo capítulo se explicará el proceso de estudio previo y análisis del problema, así como su planteamiento inicial y la documentación necesaria para tratarla. Esta incluirá el estudio de conceptos básicos de navegación y el estudio de las opciones de resolución. Además, se desarrolla el estudio del filtro Kalman y su base matemática.

Estudio previo y análisis

Dada la complejidad del problema a resolver y al carácter exploratorio del proyecto, fue necesaria una fase inicial de planteamiento del trabajo y familiarización con el mismo. A fin de comprender completamente la cuestión a resolver y todas las especificidades del proyecto, además de la fase actual en que se encontraba y de todos los avances y problemas de las fases previas, fueron necesarias varias entrevistas con el personal de la empresa en varios departamentos.

PLANTEAMIENTO DEL PROBLEMA

En primer lugar, se llevó a cabo el planteamiento inicial del problema durante varias reuniones con el encargado de la parte informática del proyecto, tutor de las prácticas de empresa y co-tutor del trabajo de fin de grado, Francisco Suárez González y con el dueño Tomás Herrera Azorin, ya que, a pesar de su cargo, está muy implicado en las actividades y proyectos de la empresa. En estas reuniones se trataron las necesidades y problemas de la empresa por los que surge el desarrollo y la necesidad de este proyecto.

Con esto, se comienza a establecer en contexto en que surge el trabajo, que serían años de mediciones en helicóptero con equipamiento fijo que solo permite la toma de imágenes y mediciones en un solo ángulo, el del cuerpo del helicóptero. Esto ocasiona que, actualmente, para llevar a cabo la medición de un área concreta, sean necesarios varios barridos de la zona en el vehículo aéreo para poder disponer de distintas muestras que se superpongan y eliminen los errores de vibración y de cambios de ángulos del

helicóptero. Esto ocasiona un gasto innecesario desde el punto de vista económico y temporal ya que es necesario mucho más tiempo de vuelo en vehículos especializados que precisan, para muy poco tiempo, de un gran aporte económico en mantenimiento, combustible, alquiler, etc. Este coste podría minimizarse si fuese posible compensar los ángulos de giro del helicóptero para que las mediciones fuesen continuas.

Posteriormente y, con ayuda del equipo de técnicos del departamento electrónico de la empresa, se expusieron las fases previas del proyecto, que incluyen el desarrollo de un dispositivo móvil para integrar los aparatos de medida y realizar giros en los tres ejes de coordenadas que permitan solucionar el problema anteriormente descrito. Este hardware, con el software y las medidas apropiadas, sería capaz de orientar los aparatos de medida a pesar de los giros del helicóptero en los tres ejes para que las mediciones se tomasen en un barrido único del área a examinar.

En las Ilustraciones 1-3 se ilustra la diferencia entre las mediciones que se toman actualmente y la potencial mejora que supondría la implantación de este nuevo método en que se basa el proyecto. Estas no son más que un método didáctico de mostrar de forma sencilla la mejora que supondría la implementación del trabajo y lo que se pretende y no una representación exacta de la realidad.

Así, puede verse en la Ilustración 1 una prueba de cómo quedaría un barrido de una zona mediante el método tradicional. Como puede observarse, no toda la zona estaría cubierta, por lo que haría falta un segundo barrido, quedando como en la Ilustración 2. Aquí quedarían cubiertas la mayoría de la superficie pero seguiría siendo necesario un tercer recorrido.

Sin embargo, como puede verse en la Ilustración 3, si el hardware de rotación gira con el vehículo aéreo y suponiendo que exista algún tipo de error y retardo, una aproximación de la toma de medidas requiere un solo barrido con una pérdida mínima o nula de información.

Ilustración 1: Medición con el método clásico (barrido 1)

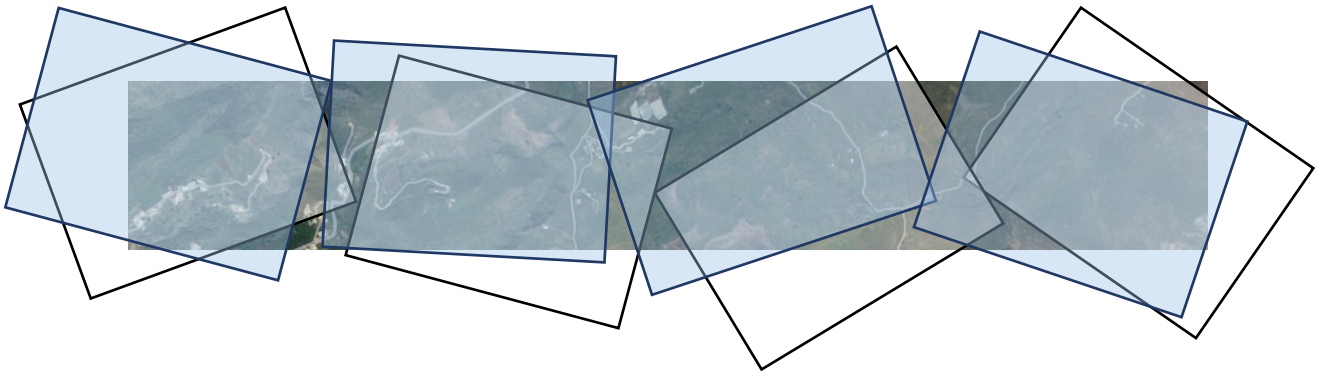


Ilustración 2: Medición con el método clásico (barrido 2)

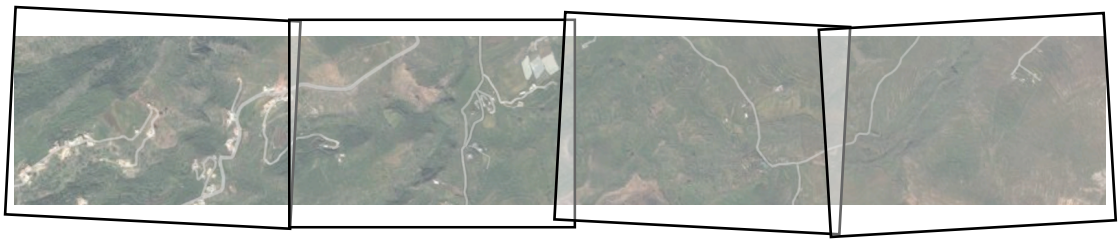


Ilustración 3: Medición con el método expuesto en el proyecto

Para conseguir el resultado que se muestra anteriormente, es necesario desarrollar software para que el dispositivo de rotación gire de forma apropiada. Para esto, sería necesario conocer la posición y orientación el vehículo móvil con el menor error posible y es en esta fase en la que el equipo se había encontrado con problemas, ocasionando una interrupción en el proyecto.

DOCUMENTACIÓN Y FAMILIARIZACIÓN CON EL PROBLEMA

Para una completa comprensión del problema de la fase actual del proyecto fue imprescindible un proceso de documentación e investigación de los sistemas de navegación -haciendo hincapié en los sistemas inerciales- y su funcionamiento e integración. Como resultado de este proceso de contextualización, conjuntamente con las reuniones continuadas con el equipo de la empresa, fue posible una visión amplia del problema.

El vehículo utilizado por la empresa sería un sistema de navegación integrado de INS (sistemas de navegación inerciales) y GNSS (particularmente, el GPS), disponiendo de una IMU (unidad de medición inercial) que integra giróscopos y acelerómetros y un GPS (sistema de posicionamiento global), tal y como se mencionó anteriormente.

Como también se explicó en el apartado de estado actual, esta integración, si bien es altamente recomendable para sistemas de alta dinámica y es beneficioso en términos de mejorar la precisión, la continuidad, la fiablidad, entre otros, también es cierto que requiere mayor capacidad de cómputo dado que es necesario ejecutar algoritmos complejos para estimar los errores de cada uno de los instrumentos del sistema.

Es en la implementación e integración de algoritmos dónde se centra el trabajo a realizar. Debido a que, en los sistemas inerciales, el error en la determinación de la posición, velocidad y orientación del vehículo crece de manera no acotada en el tiempo, el coste de IMUs que lleven a cabo la obtención de esas medias es muy elevado en el mercado. Por tanto, la empresa pretende obtener estas medidas implementando los algoritmos en las IMUs ya adquiridas de forma interna en la empresa. Para esto, el equipo de producción dispone de programas de posprocesado con los que son capaces, una vez de vuelta a la central y obtenidos todos los datos, obtener las mediciones sin error. Sin embargo, es imposible incorporar estos algoritmos a la estimación en vuelo puesto que los cálculos son demasiado pesados para el procesador de la IMU y demasiado lentos para realizarse a tiempo real.

De esta forma, se define la fase actual y, por tanto, el trabajo propuesto, como un estudio de los mejores métodos de estimación de estas medidas a partir de los datos multisensoriales de la IMU y el GPS. Se centrará, además, en obtener la mejor alternativa y el mejor resultado de análisis del algoritmo para el *heading* o rumbo, puesto que el parámetro más determinante.

En la Ilustración 4 pueden verse los diferentes ángulos que intervienen en la orientación de los vehículos aéreos con la terminología utilizada en el sector de la navegación. Teniendo en cuenta lo expuesto anteriormente sobre la toma de medidas y capturas de imagen que se llevan a cabo con el método tradicional, estos serían los factores a estimar para poder implementar el nuevo método. Así, el *roll*, *pitch* y el *heading* o *yaw* intervienen de forma conjunta en la determinación de la orientación. El motivo de elegir el *heading* se basa en que, mediante un análisis continuado de los datos obtenidos a lo largo de diferentes y extensas mediciones, el equipo de producción ha observado que, con diferencia, este es parámetro en que se acumula más error y que, tras las correcciones, sufre mayores cambios. De esta forma, siendo la posición y el *pitch* y el *roll* factores con un error bastante pequeño en la escala en la que trabajamos, el *heading* resulta ser el parámetro más determinante en la obtención de la posición y orientación del vehículo.

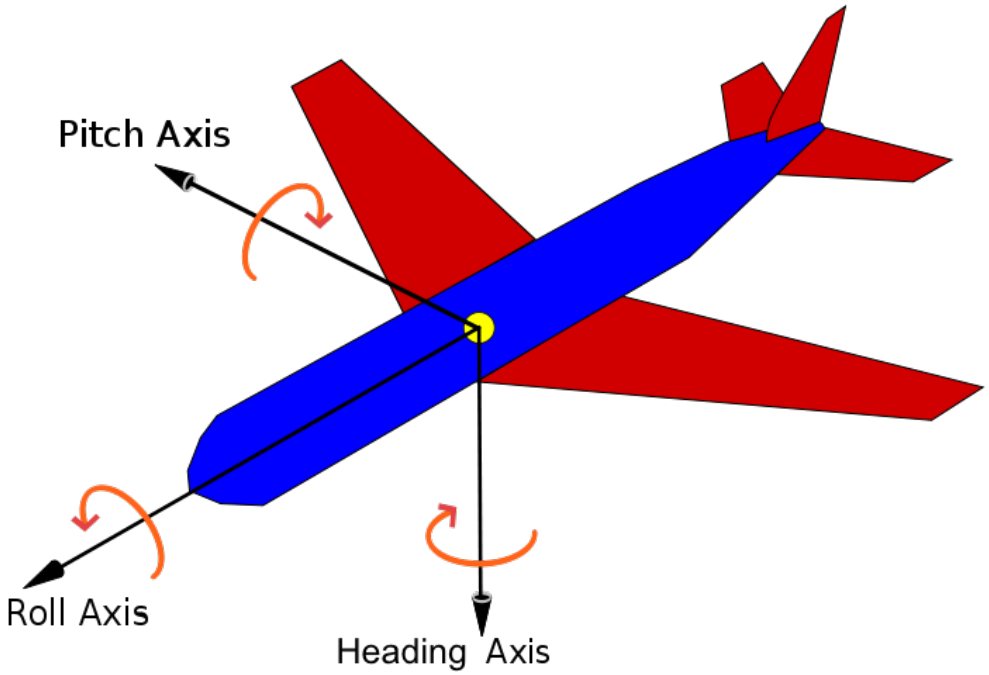


Ilustración 4: Ángulos de navegación de los sistemas inerciales

ESTUDIO DE CONCEPTOS BÁSICOS DE NAVEGACIÓN

Con el objetivo de comprender completamente conceptos y términos que son esenciales en este contexto, el de la navegación aeroespacial, es preciso, también, adquirir una base teórica de los sistemas de navegación que incluirá a los sistemas de referencia implicados en el procedimiento, los ángulos implicados en la orientación y una introducción a la navegación inercial -estos dos últimos no se tratarán en este apartado pues ya han sido explicados en apartados anteriores-.

Las variables de navegación (posición, velocidad y orientación), dada su característica vectorial, deben expresarse en algún sistema o terna de referencia. Este sería un sistema cartesiano de tres ejes ortogonales orientados de forma positiva con origen en algún punto de interés para la navegación del vehículo.

Existen, por tanto, diversas ternas de referencia que resultan útiles para la navegación, cada una con características particulares que las hacen más apropiadas para ciertas aplicaciones.

Tendríamos, entonces, los siguientes sistemas de referencia que se usarán en este problema y, en general, en la mayoría de los problemas de navegación:

- Terna terrestre o ECEF: Es el sistema de referencia estándar para la representación de la posición y velocidad de los sistemas GNSS y, por tanto, suele predominar en los algoritmos de navegación que se integran con mediciones GPS. Las ecuaciones, además, resultan más sencillas de expresar en esta terna, lo que permite mejorar la precisión y reducir la complejidad del cálculo.

  Esta terna está centrada en la Tierra y gira a la par que esta. La coordenada z^e es paralela y está alineada con la dirección del vector de rotación terrestre. Sobre el plano ecuatorial, la coordenada x^e señala al Meridiano de Greenwich mientras que la coordenada y^e completa la terna derecha.

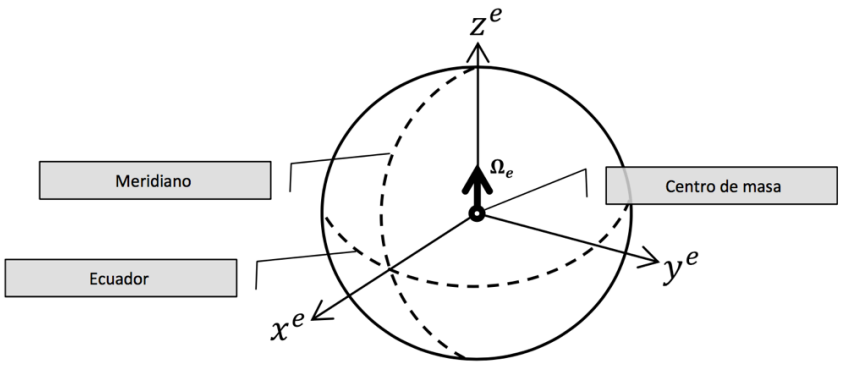


Ilustración 5: Terna de Referencia Terrestre

- Terna Inercial (ECI): El sistema de referencia inercial representa un comportamiento fundamental en aplicaciones de sincronización precisa para estimaciones basadas en mediciones GPS.

  Esta terna es el coincidente con la terna terrestre en el inicio de la navegación aunque conserva su orientación invariante en el espacio inercial durante el transcurso de la misma, es decir, no gira a la velocidad angular de la Tierra. De esta forma, la relación entre ambas ternas luego de transcurrido un tiempo *t* de la navegación es un ángulo proporcional a la velocidad angular terrestre rotado alrededor del eje z^e, coincidente con z^i.

- Terna geodésica y geocéntrica: En las ternas geodésica y geocéntrica se asignan coordenadas a puntos sobre la superficie terrestre y se completa la posición informando una medida de la altitud del cuerpo.

  Ambas ternas difieren respecto del punto sobre el cual nacen los vectores que dan lugar a los ángulos de las coordenadas y a la altitud: en la terna geocéntrica, latitud, longitud y altitud se calculan del vector que va desde la superficie equipotencial de la tierra hasta la posición del usuario y, en la terna geodésica, se computan a partir del vector normal a la superficie equipotencial de la posición del usuario.

  La terna geodésica es conocida en cartografía y resulta útil, en otras funciones, para la representación de la posición de cuerpos en mapas terrestres.

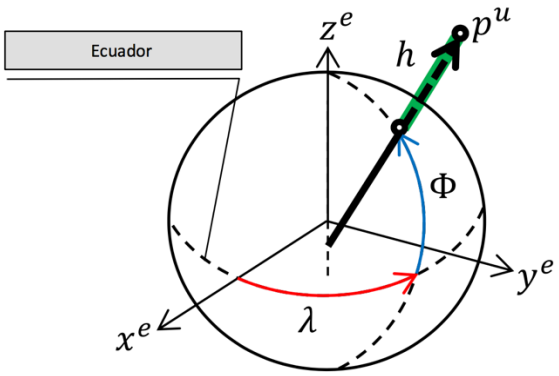


Ilustración 6: Terna de referencia geodésica

- Terna del cuerpo: Los sensores inerciales, giróscopo y acelerómetro, miden la velocidad angular y aceleración o fuerza específica referenciados en una terna de coordenadas fija al sensor, conocida como terna del cuerpo. Esta terna es solidaria al vehículo, cuya dirección *x* apunta hacia el sentido de avance, el eje *z* apuntará hacia abajo (o hacia arriba, dependiendo de la convención utilizada) y el eje *y* será perpendicular al eje de avance.

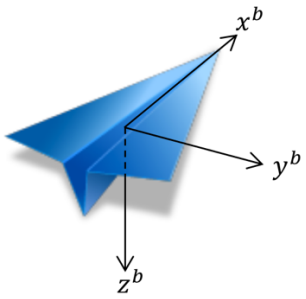


Ilustración 7: Terna de referencia del cuerpo

La combinación de mediciones de diferentes sensores que expresan sus medidas en diversas ternas de referencia conduce a la necesidad de realizar cambios entre sistemas de referencia. Así, dadas las ternas *{a}* y *{b}*, la matriz que representa la transformación lineal que toma un elemento del sistema de referencia *{b}* y lo convierte en un elemento del sistema de referencia *{a}* viene dado de la forma $\boldsymbol{C_b^a}$. (Grondona, 2013)

La derivación de las ecuaciones que conducen a la obtención esas matrices se puede encontrar en el material teórico que proporcionó la empresa de forma extensa. (Robert M. Rogers, 2007)

ESTUDIO DE LAS OPCIONES DE RESOLUCIÓN

Una vez ya hemos definido el trabajo a realizar y las condiciones que deben cumplirse, es necesario realizar un estudio de los diferentes algoritmos y metodologías que permitirían estimar la posición y orientación del vehículo, eliminando el error acumulativo que generan los sistemas inerciales.

Este estudio está basado en diferentes trabajos previos de expertos, siendo estos tesis doctorales, otros trabajos de fin de grado y trabajos de investigación. Estos trabajos son, en su mayoría, trabajos comparativos de diferentes métodos, expresando sus ventajas y desventajas y casos de uso recomendados.

Los principales artículos y documentos encontrados se basan en realizar comparativas entre el filtro Kalman, que se ha explicado anteriormente y que es uno de los más usados como algoritmos de predicción, corrección y fusión en los sistemas de navegación y otros algoritmos de fusión, como pueden ser algoritmos de filtros de partículas y redes neuronales.

En trabajos comparativos entre el filtro Kalman y el filtro de partículas para la estimación de trayectorias se obtiene que una combinación Kalman-partículas sería el más efectivo en cuanto a precisión pero se recomienda que, en caso que necesitar resultados en tiempo real, se recomienda Kalman, pues la diferencia de precisión no es tan significativa pero la diferencia temporal sí lo es y supondría un retardo considerable. (Candelas & Ramón, 2008)

En trabajos comparativos entre el filtro Kalman y las redes neuronales en la estimación de la velocidad, sale victorioso el filtro Kalman. La red neuronal, en contrapartida, presenta menos gasto computacional pero no es capaz de presentar una respuesta satisfactoria a la estimación de la velocidad cuando se producen variaciones en la misma, mientras que Kalman sí es capaz de estimarla. (González, Marcos, & Pacheco, 2004)

Aunque estos artículos y trabajos no son exactamente estimaciones de vehículos aéreos, los consideramos como válidos para la comparativa teórica puesto que tratan de estimar parámetros similares a los que vamos a estimar nosotros y, en el primer caso, se trata de estimar trayectorias que, aunque no son aéreas, es el objetivo final de nuestro trabajo.

Existen otras muchas comparativas en otros campos no tan relacionados con los sistemas de navegación que confirman la teoría de que Kalman es uno de los filtros más adaptables y precisos sin sacrificar la rapidez de respuesta, destacando la gran capacidad de cómputo paralelo en comparación con su bajo costo en desarrollo. Esto, sumado con la tendencia general, tanto en numerosos trabajos y tesis como en redes sociales de investigación como ResearchGate a usar Kalman para los sistemas de navegación inerciales y, en concreto, algoritmos de fusión de sistemas integrados y estimación de posición y velocidad, confirman la elección de Kalman como el algoritmo que constituirá la base de la resolución del problema.

ESTUDIO DEL FILTRO KALMAN Y SU BASE MATEMÁTICA

Antes de poder realizar ninguna implementación, fue necesaria una documentación exhaustiva acerca del filtro Kalman. Esto implica el estudio de la base matemática que subyace en el filtro y la adquisición de conceptos matemáticos previos necesarios para la comprensión de Kalman, para lo que fue necesario la experimentación y simulación de pequeños ejemplos en entornos matemáticos como Octave y Matlab.

Para este propósito, la empresa proporcionó material teórico adquirido para la documentación de investigaciones previas sobre el filtro Kalman y su aplicación a los sistemas de navegación. Concretamente, se trata la Navegación Inercial con secciones que explican el filtro Kalman, los diferentes tipos y algunas mejoras e implementaciones concretas. Gran parte de la base teórica de esta sección y secciones futuras está basada, a no ser que se especifique lo contrario, está basada en este material, que se correspondería con las entradas [1] y [2] de la Bibliografía: Robert M. Rogers (2007) y Mohinder S. Grewal, Lawrence R. Weill, Angus P. Andrews (2007).

Además, se ha realizado un trabajo de búsqueda e investigación de documentación académica sobre Kalman básico para comprenderlo mejor y adquirir el vocabulario específico que se maneja, así como nomenclaturas, versiones y tipos de implementación del mismo. Esta documentación incluye principalmente investigaciones, tesis doctorales y trabajos de fin de grado o de master relacionados con aplicaciones de Kalman, entre otros.

A la hora de comprender el funcionamiento del filtro Kalman es necesario saber que se basa en el procesamiento de mediciones ruidosas de un sistema lineal perturbado por ruido aleatorio Gaussiano a fin de estimar su estado interno. Así, Kalman es capaz de

resolver de forma óptima el problema de estimar el estado interno de un sistema estocástico mediante la utilización únicamente de las salidas del sistema, que serán medidas relacionadas con los estados internos. (Dematties & Iglesias, 2012)

Para que el filtro pueda llevarse a cabo, deben cumplirse las siguientes condiciones en el sistema:

- El sistema deberá ser LTV (Lineal Variante en el Tiempo).
- Los ruidos de proceso y medición deberán ser ruidos blancos gaussianos de media nula y sin correlación entre sí.

Para este tipo de sistemas con esas condiciones, puede demostrarse que existe un estimador óptimo en relación a la minimización del MSE (error cuadrático medio) que, además, es lineal. Este constituiría el Filtro Kalman. (Dematties & Iglesias, 2012)

Una primera aproximación al funcionamiento del FK (Filtro Kalman) sería el mostrado en la Ilustración 8.

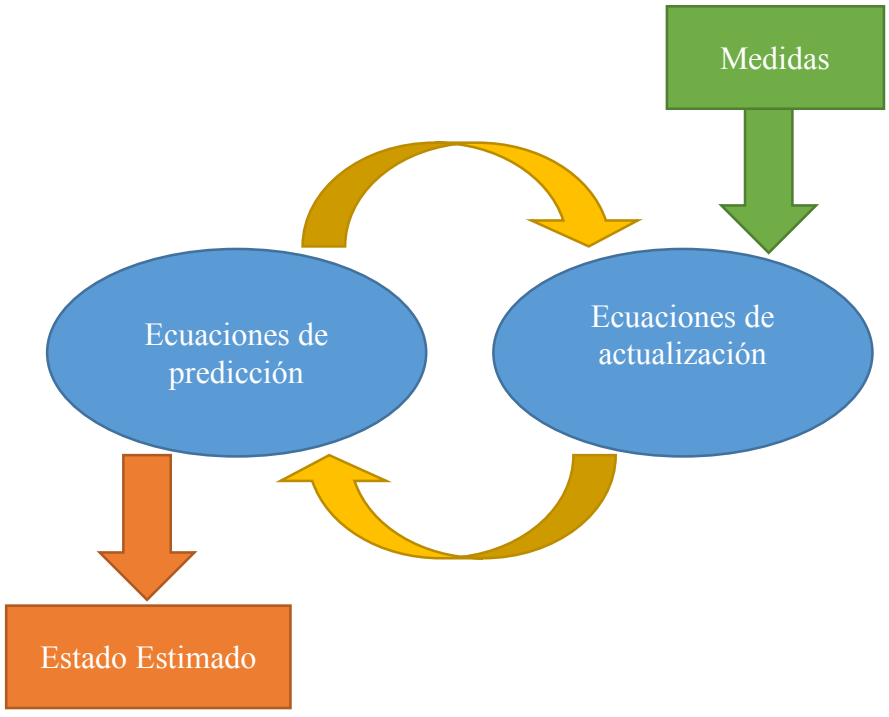


Ilustración 8: Esquema de funcionamiento del Filtro Kalman

Como puede observarse, el FK funciona mediante dos procesos: actualización y predicción. De forma continua, Kalman está llevando a cabo un proceso de predicción a partir de los datos de un vector de variables que permiten que, a partir de las condiciones iniciales, se pueda propagar la solución del estado estimado en el tiempo. En este paso se va actualizando la estimación y la incertidumbre estimada teniendo el cuenta los efectos inciertos del sistema entre mediciones.

Cada vez que se obtiene una nueva observación o medición se lleva a cabo el proceso de actualización de modifica las variables implicadas en proceso combinando el conocimiento anterior contenido en las variables y el proceso de medición. Así, en este paso, al corregir la estimación en base a la medición, se actualiza y se reduce la incertidumbre y se va mejorando el estado estimado obtenido hasta ese momento.

Con esto, Kalman consigue ir mejorando su estado estimado o predicción con la llegada de nuevas medidas que actualizan una serie de variables, vectores y matrices que almacenan el conocimiento de los estados anteriores. De esta forma, la predicción va mejorando con el tiempo, cada vez con menor incertidumbre y más precisión.

Ahondando en el funcionamiento específico del filtro, es necesario explicar previamente conceptos íntimamente relacionados con las ecuaciones de actualización y predicción que caracterizan el FK:

- **Vector de estado (x)**: El vector de estado contendrá todas las variables relevantes a la hora de aplicar el FK (relativas a cada problema específico) que se corresponderán con aquellas que se deseen estimar, eliminar el error, etc., además de aquellas variables que estén relacionadas con estas e influyan en su obtención y determinación.

  Este vector puede comenzar como un vector nulo y es el que se usará para obtener el **vector de estado estimado o estimación ($\widehat{x}$)** que será el que se actualice en cada etapa y que se corresponderá con la estimación de x de máxima probabilidad.

- **Ganancia Kalman (K)**: Manteniendo una estimación de la incertidumbre y una estimación relativa de la incertidumbre de los sensores, Kalman combina esa información óptimamente para minimizar el error de la estimación. Así, la ganancia de Kalman es una matriz óptima de pesos que, al combinarla con los

datos nuevos de los sensores obtenidos en la medición, se actualice el estado estimado a partir de datos estimados.

Esta matriz se obtiene como una solución parcial de la ecuación de Ricotti, que se encargan de propagar la matriz de covarianza y tratan la incertidumbre.

- **Matriz de Covarianza (P)**: La matriz de covarianza se corresponde con la estimación de la incertidumbre que, utilizando el vector de estado estimado, se logra a través de un estimador gaussiano de máxima probabilidad. Este estimador se basará en la obtención del *argmax* de una función de densidad de probabilidad, a partir de la cual se obtienen las ecuaciones de actualización y predicción de la ganancia de Kalman.

- **Vector de medición (z)**: Vector que contiene las medidas reales de los sensores recibidas en el intervalo de tiempo actual.

- **Matriz de sensibilidad en la medición (H)**: Es una matriz de sensibilidad de la medición que define la relación de un sistema y las mediciones que pueden ser realizadas. Así, es la que realiza la conversión del vector de estado a un vector de medición: $z = H * x$.

- **Matriz de transición de estado (A)**: Es la matriz que describe cómo evoluciona el sistema desde el estado en el momento inicial $x(t_0)$ hasta el estado en el momento actual $x(t)$ en el intervalo $[t_0,t]$. Esta matriz nos sirve para obtener el estado del sistema actual a partir del estado anterior, por lo que es esencial para la fase de predicción del filtro.

- **Matriz de covarianza del ruido del proceso (Q)**: Matriz de covarianza del ruido o error estimado del proceso. Esta intervendrá en el proceso de predicción de la matriz de covarianza *P*.

- **Matriz de covarianza del error de la medición (R)**: Matriz de covarianza del error estimado de la medición a partir de los sensores. Esta intervendrá en la determinación de la ganancia de Kalman.

Una vez se han definido estos conceptos básicos, se puede comenzar a explicar el algoritmo con sus correspondientes fórmulas, paso a paso, como puede verse en la Ilustración 9.

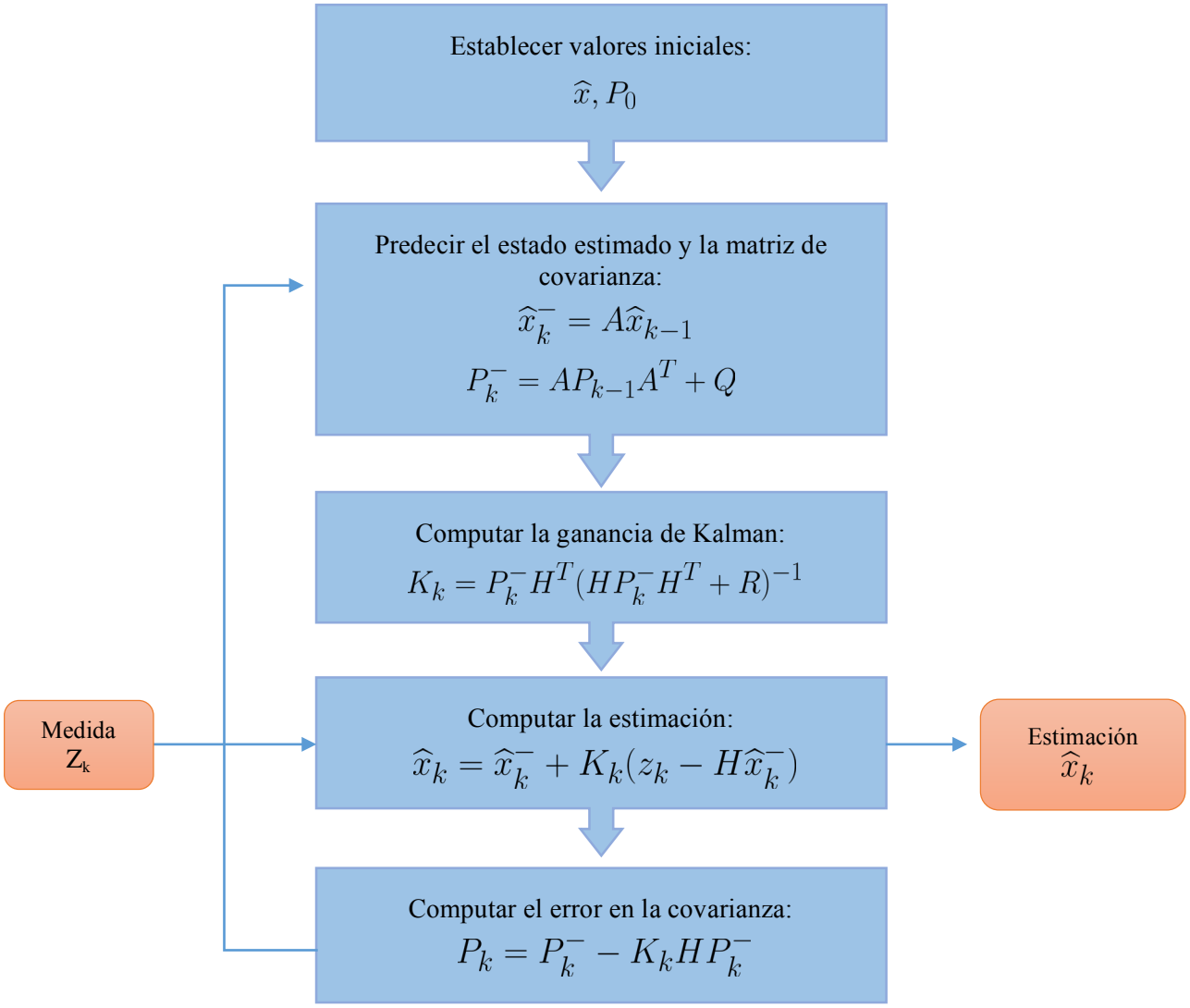


Ilustración 9: Funcionamiento del Algoritmo del Filtro Kalman

1. El primer paso sería establecer los valores iniciales del vector de estado estimado y la matriz de covarianza. En ocasiones, no tenemos datos ni información sobre la covarianza. En estos casos, los filtros Kalman se inicializan realizando una estimación sin base del estado inicial. Esto no es lo más deseable, puesto que, de forma natural, la estimación se precisa con la mayor prontitud posible, así que, lo deseable es que ni siquiera las primeras estimaciones se vean afectadas por esta "suposición" sobre el primer estado.

2. El siguiente paso sería llevar a cabo la predicción del estado estimado (primera ecuación), como se explicó anteriormente, a través de la matriz de transición de estado. De esta forma, obtenemos la estimación a priori (antes de actualizar con la medición) del estado k-ésimo a partir de la matriz de transición y una estimación a priori del estado anterior.

   También en este paso, se lleva a cabo la predicción de la matriz de covarianza (segunda ecuación). Esto se realiza de forma similar a lo descrito anteriormente para la predicción del estado estimado, pero de forma más compleja. Así, como anteriormente, usamos los parámetros de la matriz de transición (comportamiento dinámico conocido del sistema) pero, en este caso, también utilizamos en la predicción la información de la matriz de covarianza del ruido de perturbación dinámica *Q* para calcular como varía en el tiempo la matriz de covarianza *P*.

3. A continuación, se calcula la ganancia de Kalman que, como se explicó anteriormente, parte del estimador anteriormente descrito y, por tanto, su fórmula se obtiene de variaciones del mismo, junto al requisito de fijar que las funciones de densidad de probabilidad de los ruidos blancos sean normales y de media nula. (Dematties & Iglesias, 2012)

   Para este cálculo de la ganancia intervienen la matriz de ruido en la medición y la matriz de sensibilidad de la medición. Esto se debe a que estas son las matrices que intervendrán directamente en la medición y, por tanto, deberán utilizarse de forma directa o indirecta (a través de la ganancia) en el cálculo de la estimación en el paso siguiente de actualización.

4. Cuando obtenemos una medición (vector de medición *z*), calculamos la estimación del estado para ese instante de tiempo a partir de la estimación

a priori del estado calculada en pasos previos y de la ganancia de Kalman y, por último, con la intervención de la diferencia entre la medición obtenida y la estimación hasta el momento.

5. El último paso en la iteración sería la actualización y corrección de la matriz de covarianza. Esto se hace actualizando la matriz de covarianza a priori con la información de la ganancia de Kalman y la matriz de sensiblidad.

Como se puede observar en la Ilustración 9, el algoritmo del FK se trata de uno iterativo. Por tanto, cuando se alcanza el último paso, se vuelve a repetir el proceso a partir del segundo punto hasta que se alcance una condición de parada, que podría ser un número de iteraciones, que no se encuentren más medidas, que se llegue a un tiempo máximo, etc.

CAPÍTULO 3: Diseño

En el tercer capítulo se tratará el estudio de las alternativas de implementación, tanto desde el punto de vista de tecnologías y plataformas como desde el punto de vista de búsqueda e investigación de librerías y código abierto. Finalmente, será necesario documentar el uso de la librería y de las funcionalidades relativas a la aplicación.

Estudio de las alternativas de implementación

Una vez hemos adquirido los conocimientos teóricos previos fruto de la investigación y documentación, debemos proceder a investigar las diferentes alternativas de implementación del filtro y diseñar la solución deseada, con sus respectivos escenarios de prueba.

TECNOLOGÍAS Y PLATAFORMAS

Teniendo en cuenta las dimensiones y características del problema, así como los recursos y necesidades de la empresa, se lleva a cabo una investigación de las diferentes implementaciones posibles de la alternativa teórica seleccionada como resolución del problema, el Filtro Kalman.

En primer lugar, fue necesario establecer el lenguaje de programación en que iba a desarrollarse el trabajo. La empresa necesitaba integrar el código en software ya existente y, para favorecer la compatibilidad y la migración del mismo, el programa final debería estar desarrollado en C++ o tener la posibilidad de generar código en ese lenguaje a partir del lenguaje en que esté desarrollado.

Esto se debe a que todo el código de la empresa en los diferentes dispositivos y hardware se integra y migra gracias al uso de QT, del que hacen uso mediante un add-in en Visual Studio, siendo este el entorno de desarrollo establecido como estándar en la empresa. Por tanto, este nuevo código, indiferentemente de las tecnologías usadas y las plataformas empleadas para ello, debe ser portable a C++ para su posterior uso en

Visual Studio. Este IDE con QT y las librerías inerciales necesarias, fueron usados en la parte de datos reales para el desarrollo de programas de lectura de IMU y GPS que permitieron la obtención de los datos reales con los que trabajar y hacer simulaciones en la parte final del trabajo.

BÚSQUEDA E INVESTIGACIÓN DE LIBRERÍAS Y CÓDIGO ABIERTO

Debido a los requisitos de lenguaje, se buscó código abierto y librerías en C++ que implementaran el filtro Kalman. En esta búsqueda se encontraron numerosas implementaciones del Filtro Kalman y el Filtro de Kalman Extendido (se usa para sistemas no lineales). Sin embargo, ninguna de estas implementaciones permitía la integración del INS y el GPS.

Una búsqueda más exhaustiva y concreta de código abierto para la implementación de la integración del sistema de navegación dio lugar a resultados para su uso en C++, pero estos requerían numerosas librerías de cálculo, sistemas de referencia, transformaciones terrestres, etc. que hacían el código confuso, lento y pesado. Además, muchas de las aplicaciones encontradas actuaban con modelos y tecnologías concretas y no eran aplicables a un caso más general, ni daban pie a la generalización, por estar tan ligadas al problema concreto.

Todas las librerías y aplicaciones encontradas hasta el momento tampoco ofrecían solución al problema de estimación del *heading*, por lo que se amplió la búsqueda para incluir otros lenguajes de programación y otras tecnologías.

Como resultado de esta nueva búsqueda, encontramos el proyecto "*Open source aided inertial navigation*" para Matlab. Este proyecto incluye un conjunto de ejemplos de uso de Kalman para la navegación inercial de código abierto colaborativo, además de numerosas funciones auxiliares de las que se hace uso en los ejemplos y que, además, individualmente, proporcionan resolución a problemas de la navegación inercial como pueden ser: cambios de marcos de referencia, funciones de transformación, generadores de ruta, funciones de inicialización y calibración, modelado de error de la IMU, etc.

Dentro de este repositorio de código abierto cuyo enlace y acceso se correspondería con la entrada [3] de la bibliografía, se encuentran varios ejemplos diferentes:

- Implementación de un INS sin inicialización de *heading*: Una implementación de integración INS-GPS que puede ser usada sin inicialización de heading. Se usan dos estados para representar un alto nivel de incertidumbre para ángulo de *heading*.

- Implementación de un sistema fuertemente acoplado GPS-INS: Un Sistema fuertemente acoplado de GPS-INS que se mecaniza con un ECEF. El filtro Kalman solo se usa para procesar los pseudorangos.

- Demostración de sensores redundantes: Ejemplos de sistemas INS que muestran el uso de sensores inerciales redundantes.

- Implementación del *wander azimuth*: Una implementación no singular de un INS mecanizado en un marco de *wander azimuth*.

De estos ejemplos que se encuentran en el *toolkit*, los únicos que podrían constituir soluciones al problema serían los dos primeros, pues son los usados para trabajar con integración INS-GPS.

Para comenzar, se analiza ejemplo de la integración fuertemente acoplada, pues, desde el punto de vista teórico, este tipo de integración produce mejores resultados. Sin embargo, posteriormente a la exploración del código, se determina que no puede usarse. Esto se debe a que el equipamiento de la empresa no está preparado para tomar medidas de pseudorango de forma natural en vuelo. Sin embargo, para ser capaces de descartar con certeza esta vía de desarrollo, la empresa, que en esta fase del proyecto tenía a parte de su equipo en un viaje con motivos de toma de mediciones para un cliente, tomó un set de datos con la medición de pseudorango activa. Esta opción es posible con el instrumental pero, por limitaciones del hardware, el resto de las mediciones se ven comprometidas cuando se activa la medida del pseudorango.

De cualquier forma, un análisis de estos datos revela que las medidas que proporciona el GPS y la forma en que las proporciona no son adecuadas para usarlas en una integración fuertemente acoplada. Tras realizar varias pruebas y consultas se decide abandonar esta línea de investigación por infructuosa y porque, en una implementación en tiempo real, que es el objetivo, no constituiría una buena solución, al no poder llevarse a cabo de forma concurrente con la toma de las demás mediciones.

Como esta opción no da los resultados esperados, se probó el ejemplo de la integración débilmente acoplada sin el *heading* inicial. Este, originalmente, trabaja con datos sintéticos que obtiene a partir de una simulación de ruta que lleva a cabo la aplicación. Para comprobar que esta opción es viable, se trabaja con la ruta simulada, modificándola con el objetivo de encontrar una ruta que se asemeje en tiempo de vuelo y en distancia recorrida a las rutas que se llevarían a cabo en un escenario real. Es decir, del mismo orden en magnitudes de tiempo, distancia recorrida y giros realizados. Tras ajustar estos parámetros, obtenemos resultados más prometedores.

Con todo esto, la integración débilmente acoplada realiza una buena estimación de la ruta que sigue el vehículo, con un error visual casi imperceptible, como puede verse en la Ilustración 11. En cuanto al *heading*, concretamente, a pesar de no tener inicialización y de ser el parámetro con más dificultades para estimarse, obtenemos una aproximación aceptable. En esta fase de selección de alternativas de implementación no es necesario mayor grado de precisión en la medición del error, por lo que, visualmente, en la Ilustración 10 se aprecia que la estimación del *heading* está entre los rangos aceptables de error para esa medida.

Teniendo en cuenta lo anterior, de forma teórica y general, tendríamos una aproximación para la resolución del problema.

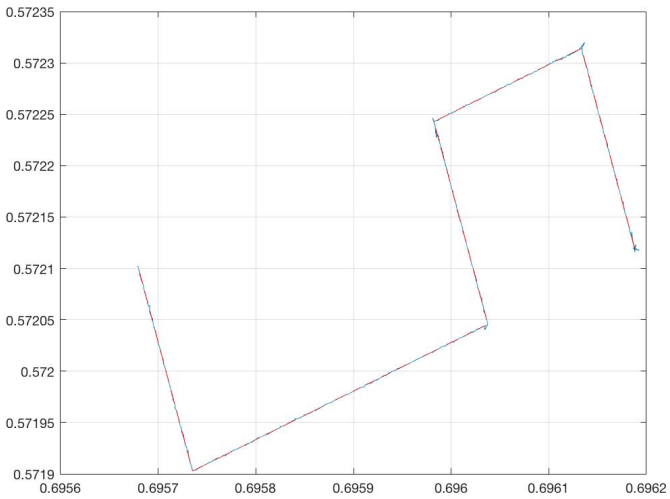


Ilustración 10: Gráfica ilustrativa de la trayectoria de la librería

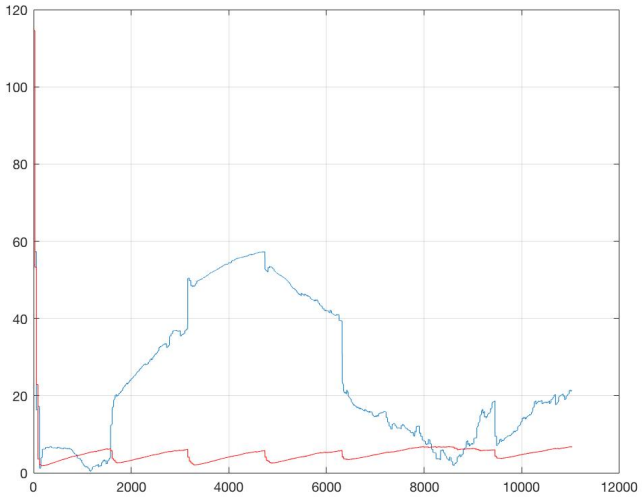


Ilustración 11: Gráfica ilustrativa del heading de la librería

Documentación y uso de la librería

Como se explicó anteriormente, la librería de código abierto usada para la realización de este trabajo sería la librería "*Open source aided inertial navigation*". Concretamente, utilizaremos el ejemplo de Implementación de un INS sin inicialización de *heading*.

Como también se expuso en el apartado anterior, esta librería dispone de un extenso número de funciones de ayuda en los cálculos de navegación inercial. Por tanto, en el proceso de documentación de la librería fue necesario estudiar y analizar el comportamiento y funcionalidad de las funciones auxiliares que se proporcionan y del propio código del ejemplo a usar. El fruto de ese extenso análisis y experimentación con el código de una librería muy poco documentada se refleja en el conocimiento del funcionamiento del mismo. Así, se explicarán de forma breve y didáctica las partes más relevantes y significativas del código del ejemplo que tienen que ver con el funcionamiento de la librería y se explicará la aportación de las funciones auxiliares implicadas en el proceso.

FUNCIONES AUXILIARES

- **AddIMUErr_v000**: Función utilizada en la generación de trayectorias sintéticas para añadir error a la IMU en función de los parámetros que definen el modelo de error para hacerlas más realistas y cercanas a una trayectoria real. Además, es necesario añadir este error para que el FK tenga algún tipo de efecto.

- **AddObsErr_v000**: Función utilizada en la generación de trayectorias sintéticas para añadir error aleatorio a la observación del GPS en función de una semilla definida como parámetro inicial del sistema.

- **alingc_pl_v000**: Función que, dada una medida del acelerómetro y un estado estimado de *azimut*, computa la rotación del sistema de referencia inercial al del cuerpo, generando y devolviendo la matriz de transformación C_{nb}, que es equivalente a C_b^n.

- **dc2dc_v000**: Función de transformación que, dado un DSR (Data Signaling Rate) continuo, lo convierte en discreto y viceversa. Esto se debe a que algunos cálculos se realizan de forma más sencilla en un tipo de sistema u en otro.

- **dcm2euler_v000**: Función de transformación que convierte las DCM (Directional Coisine Matrixes) en ángulos eulerianos. Las matrices de cosenos directores sirven para realizar transformaciones de sistemas de referencia. Así, cuando una rotación medida en un sistema de referencia se quiere pasar a otro, puede usarse y, además, transformar esa rotación en grados eulerianos.

- **diagmat_v000**: Función auxiliar que realiza la suma de submatrices diagonalmente.

- **euler2dcm_v000**: Función inversa a dcm2euler_v00.

- **geoparam_v000**: Función auxiliar que, dada la posición en latitud, longitud y altura, computa el radio y la gravedad terrestres. Estos datos se usan en los cálculos y medidas del GPS y en las matrices de cambio de sistema de referencia.

- **imu_modTI_v000**: Dadas las definiciones de los parámetros de error, genera una parte invariante del modelo de error.

- **imu_modTV_v000**: Dadas las definiciones de los parámetros de error, genera una parte variante del modelo de error.

- **readbin_v000**: Función de lectura de archivos binarios.

- **skew**: Función que, dado un vector lo transforma en una matriz anti-simétrica. Estas funciones serán necesarias en los cálculos matemáticos del filtro.

- **strapdown_ecef_dcm_v000**: Función que realiza una mecanización del sistema de referencia terrestre (ECEF) y obtención de matrices de transformación DCM en un sistema sin plataforma.

- **strapdown_wander_dcm_v000**: Función que realiza una implementación del *wander-azimut* que modela un sistema para una incertidumbre inicial elevada. El ángulo *wander* desconocido se modela con coordenadas polares [sin(α), cos(α)].

- **sys_wander_dcm_v000**: Función que define un modelo de sistema para la implementación de la función anterior (mecanización *wander-azimut*).

- **sys_wander_largeheading_v000**: Función que define un modelo de sistema con gran incertidumbre inicial en el parámetro *heading* dentro de una mecanización *wander-azimut.*

- **deg2utm**: Función externa a la librería usada hasta el momento, pero que explicamos por haberla añadido con posterioridad y constituir una más en el conjunto de funciones de navegación. Esta convierte los vectores de latitud y longitud que tiene como entrada en las tres coordenadas UTM en el Sistema Geodésico Mundial 1984 (WGS84).

- **PathGen_v002**: Esta es la función que lleva a cabo la generación de la trayectoria simulada con datos sintéticos. Para poder realizar esta generación, será necesaria la ruta donde se guardarán los archivos generados como observaciones de la IMU y el GPS y los datos sin error. También será necesario el vector con la inicialización del sistema, la matriz de definición del movimiento, la frecuencia de muestreo escalada, el modo de simulación, la configuración de los sensores y la IMU, los parámetros de vibración y, por último, la semilla para las generaciones aleatorias de ruido.

Con esta información, la función genera seis archivos binarios: “obs_pos.bin”, “imu.bin”, “imuerr.bin”, “gps.bin”, “mimu.bin” y “mnav.bin”. Los dos primeros serán, respectivamente, el archivo de medidas del GPS y de la IMU usados como datos sintéticos en la estimación como si fuesen datos reales en la estimación. El tercero será el error de la IMU y, por último, los archivos “gps.bin”, “mimu.bin” y “mnav.bin” son los propios de los datos sintéticos sin error de la trayectoria, correspondiendo con los datos ideales en que se generaría la trayectoria en un entorno sin fallos. Los dos primeros se utilizan para generar los archivos con error “obs_pos.bin” e “imu.bin” mediante la adición de errores posteriormente y el archivo “mnav.bin” se utiliza como validación de las estimaciones, siendo la trayectoria ideal que es el objetivo.

Esta función es muy compleja ya que simula todo un sistema de navegación integrado con sus correspondientes errores de medición por modelo y los erros propios de la vibración y la temperatura, generando variables con ruido aleatorio y mediciones propias de un sistema real. Por motivos didácticos y de extensión de la memoria, esta función auxiliar de la librería no se explicará más en detalle a pesar de la importancia que tiene en el desarrollo. Tan solo se ha documentado su funcionamiento y no el código con precisión puesto que la documentación de la misma se encuentra en la documentación de la librería y esta función no se modifica en la realización del trabajo, tan solo se utiliza como una función más del conjunto de las funciones disponibles de la librería.

FUNCIONALIDADES RELATIVAS A LA APLICACIÓN

- **example_largeheading**: Ejemplo de navegación integrada que comienza con una gran incertidumbre en el *heading* y, por tanto, con una estimación inicial del ángulo desconocida.

  La aplicación comienza estableciendo la definición del sistema para el problema a tratar, para luego leer los archivos de GPS e IMU que se utilizarán para los cálculos. En esta sección inicial se definen las variables iniciales, como los modelos de error y el tiempo de propagación. Este último está establecido en 33, de tal forma que, tras 33 iteraciones del bucle principal, se volverá a llevar a cabo del proceso de actualización del filtro.

Una vez se tienen estas variables iniciales del sistema, comenzamos a inicializar los parámetros del filtro. Así, inicializamos el vector de error de la IMU, que Kalman utilizará para estimar más tarde. Seguidamente, inicializamos el INS, con sus parámetros de posición, velocidad y orientación, llevando a cabo las correcciones correspondientes, así como las transformaciones necesarias, a través de las DCM nombradas anteriormente.

Antes de comenzar a aplicar el FK, es necesario puntualizar que el algoritmo utilizado lleva a cabo un escalado de los resultados y, por tanto, se definen las variables constantes de escala haciendo uso de las funciones auxiliares antes llevar a cabo el bucle principal del filtro.

Antes de comenzar las iteraciones también se define la matriz de covarianza del filtro basada en los errores definidos y los parámetros iniciales de definición del INS, además de las constantes de escala.

Una vez tenemos la matriz de covarianza, iniciamos los acumuladores y las *flags* que se usan dentro el bucle iterativo y de Kalman y, tras leer los primeros datos de los archivos, comienza el bucle.

Dentro del bucle, que continuará iterando mientras existan medidas de IMU que leer, se actualizan las medidas eliminando el error estimado, se actualiza el contador de propagación en cada iteración y se obtienen las matrices de transformaciones.

Si ocurre que las medidas de GPS e IMU están sincronizadas, es decir, que esa medida en concreto tiene el mismo *timestamp* o marca temporal o si ocurre que el contador de propagación llega al valor 33 definido con anterioridad, se llevará a cabo todo el proceso de actualización y estimación del filtro Kalman. De lo contrario, se continuarán leyendo valores hasta que esto ocurra.

Cuando llegamos al caso descrito anteriormente, procedemos a propagar la covarianza. Esto es necesario porque, desde un enfoque Bayesiano, el filtro debe propagar la densidad de probabilidad condicional de acuerdo a los conocimientos que provienen de los dispositivos de medición. Además, se forman los modelos discretos de tiempo del modelo de navegación con las funciones auxiliares y se obtienen matrices, como la matriz *Q* que se utilizarán

para la fase de predicción del filtro. Esta fase se realiza en función del modo de funcionamiento que, como se nombró anteriormente, puede ser *coarse*, es decir, menos preciso o puede ser *fine*, que tiene más refinamiento. Además, se actualiza el vector de estados de error, que es parte de la fase del proceso predictivo.

```
%Propagate
P=STM*P*STM'+Q;
prp_coun=0;
imu_avg=zeros(6,1);

%update IMU error states
ximu=STM(nst_nav+1:nst,nst_nav+1:nst)*ximu;
```

Ilustración 12: Código - Predicción de estado estimado y covarianza

Cuando el filtro se encuentra con una nueva medición de GPS que procesar, dependiendo del modo de funcionamiento en que se encuentre, establece las matrices de actualización con mayor o menor precisión para el paso posterior.

A continuación, se computa la ganancia de Kalman, para lo cual, primero, deben definirse los valores de las matrices implicadas en la ecuación en función de la posición medida con el GPS. Una vez obtenidas esas matrices, se computa la ganancia de Kalman *K*.

```
H=zeros(3,nst); %position update
H(:,1:3)=eye(3);
R=S*ObsErrDef.sR*ObsErrDef.sR'*S';
inno=S*(pos_g-obs_pos(2:4));  %position obs in m

Rinno_inv=inv(H*P*H'+R);
K=P*H'*Rinno_inv;
```

Ilustración 13: Código - Cómputo de la ganancia de Kalman

Finalmente, se lleva a cabo la fase de actualización en que se computa la estimación y la covarianza con su error. De nuevo, la precisión con la que se trata el *heading* depende del modo de funcionamiento y se utilizan las funciones auxiliares para obtener las matrices de transformación, necesarias en parte del proceso.

```
%correct current states
dx=K*inno;
%K(10:11,:)=0;
%K(12:17,:)=0;
pos_g=pos_g-S\dx(1:3);
vel_n=vel_n-dx(4:6);
mx_a=euler2dcm_v000(dx(7:9));
Cbn=mx_a*Cbn;
if (mode==0)    %coarse alignment
    wander=wander-dx(10:11);
    %%normalize wander
    wander=wander/norm(wander);
elseif (mode==1) %fine alignment
    wander=[-wander(2) wander(1);wander(1) wander(2)]*[sin(dx(10));cos(dx(10))];
end
ximu=ximu+dx(nst_nav+1:nst);
%update cov
P=(I-K*H)*P*(I-K*H)'+K*R*K';
```

Ilustración 14: Código - Actualizacion de la estimación y la covarianza

Una vez se ha completado el bucle de nueva medición de GPS, se procede a leer un nuevo dato para la siguiente iteración y a cambiar de modo de funcionamiento si fuese necesario.

Cuando volvemos al bucle principal, se realizan conversiones de sistemas de referencia con las matrices de transformación obtenidas a través de las funciones principales y de las ecuaciones de Robert M. Rogers (2007). Además, se transforman los datos y se obtiene el error en comparación con la ruta absoluta para almacenarlo en los archivos de resultados. Finalmente, se lee un nuevo dato y continúa el bucle hasta completar el registro de datos de la IMU.

Para visualizar los datos, la librería incluye una figura en Matlab que muestra la trayectoria sin errores y la trayectoria estimada, para comprobar visualmente la bondad de la estimación. También incluye una gráfica de comparación de la velocidad ideal y la estimada y una gráfica comparativa que muestra el *heading* real con el *heading* estimado para ver el error con un golpe de vista.

- **sys_path_largeheading**: Esta función es la definición de los parámetros del sistema para la función ***example_largeheading*** por lo que incluirá todas las inicializaciones necesarias para el sistema.

  Así, se definen las variables temporales de incremento de tiempo y algunas variables para conversión de unidades y se comienzan las definiciones. Es necesario, para esto, inicializar los parámetros de los modelos de error de la IMU: los modelos de error del acelerómetro y del giróscopo.

Estos modelos de error se basan en un modelo de error de espacio de estados definidos por los parámetros A, B, C, D y sP que rigen las ecuaciones:

(Salida del sensor) = u + Cx + Dv donde el error del sensor sería: ***x_dot = Ax + Bw*** y donde *u* sería la variable cinemática que se espera medir con el sensor y dónde *Dv* sería el ruido blanco aditivo.

El funcionamiento en el sistema de este modelo de error en el sistema se ve reflejado en el siguiente esquema realizado con la herramienta Simulink de Matlab.

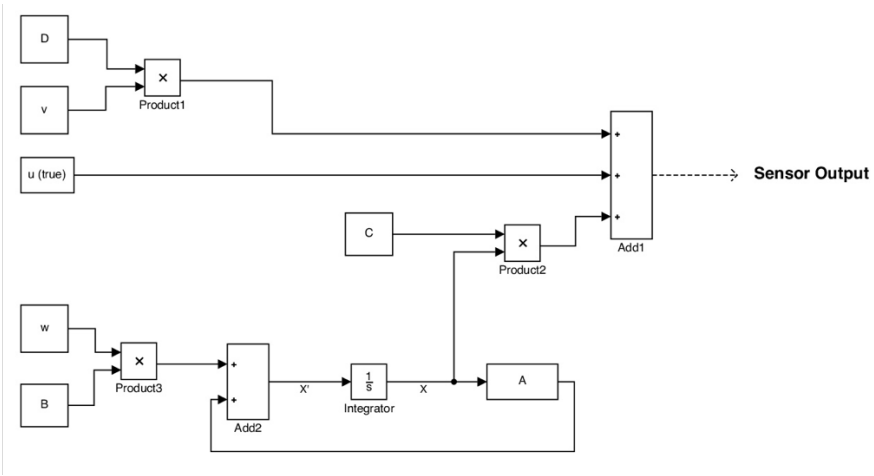


Ilustración 15: Esquema del modelo de error

Una vez se han definido los parámetros de ambos modelos de error, se inicializan los valores y las definiciones de error de posición, velocidad y orientación que definirán el estado inicial del sistema, tanto para la IMU como para el GPS

Una vez se han llevado a cabo estas definiciones, es necesario generar la trayectoria. Los datos, archivos y demás atributos de la trayectoria se generarán mediante la función auxiliar ***PathGen_v002*** pero esta, como parámetro de entrada, precisa de una matriz de definición de la trayectoria (*mot_def*) que definirá la dirección, duración y ángulos de los segmentos que componen la trayectoria. Es decir, primero es necesario definir la forma de la trayectoria antes de poder generarla.

La definición de esta matriz presentó ciertos problemas a la hora de llevar a cabo la investigación de la librería y, posteriormente, en el desarrollo, pues está muy mal documentada y el significado físico de los parámetros tuvo que ser deducido

a base de experimentación y cálculos de aproximación a casos reales. Como resultado de todo este proceso experimental, se obtienen las unidades y significado de los parámetros que componen la matriz, como se documentará a continuación.

El relleno de la matriz de definición de la trayectoria se hace, por facilidad, fila a fila, de tal forma que cada fila representa un segmento de la trayectoria final. Cada fila vendrá definida por seis valores, de tal forma que la matriz final tendría esta estructura:

- ***mot_def(:,1)*** → Tipo de movimiento o modo de funcionamiento.
- ***mot_def(:,2:5)*** → Parámetros del movimiento.
- ***mot_def(:,6)*** → Tiempo máximo para el segmento.

En primer lugar, la matriz *mot_def* presenta cinco modos de funcionamiento, que se seleccionarán con el primer elemento del vector de cada fila:

- **Modo 1 :** Modo que permite determinar las entradas directamente, siendo estas las medidas iniciales en el momento t_0.
- **Modo 2 :** Modo que permite determinar los valores absolutos (no relativos) de orientación y velocidad a alcanzar.
- **Modo 3 :** Modo que permite determinar los cambios relativos de orientación y velocidad.
- **Modo 4 :** Modo que permite determinar la orientación absoluta y la velocidad relativa.
- **Modo 5 :** Modo que permite determinar la orientación relativa y la velocidad absoluta.

En cuanto a los parámetros del movimiento, estos varían según el modo de funcionamiento y siguen la siguiente estructura:

- ***mot_def(:,2:4)*** **[Modo 1]** → Parámetros [x, y, z] de la velocidad angular del *heading* en rad/s.

- ***mot_def(:,2:4)*** **[Modos 2:5]**→ Parámetros [*roll, pitch, heading*] del ángulo final a obtener tras el tiempo definido en radianes.
- ***mot_def(:,5)*** **[Modo 1]** → Parámetros [x, y, z] de la aceleración en m^2/s.
- ***mot_def(:,5)*** **[Modos 2 y 5]** → Parámetros [x, y, z] de la velocidad en m/s.
- ***mot_def(:,5)*** **[Modos 3 y 4]** → Parámetros [x, y, z] del incremento en la velocidad en m/s.

En cuanto al tiempo máximo para la obtención de las medidas definidas anteriormente, este se expresará en segundos y es necesario tener en cuenta que este no es un tiempo constante. Es decir, este tiempo se define como un periodo temporal para alcanzar las medidas pero, en caso de obtenerse antes, el tiempo real del segmento puede ser menor y, en caso de que el tiempo sea insuficiente para obtener las medidas, estas no alcanzarán el valor deseado. Por tanto, es preciso comprobar posteriormente mediante cálculos a partir de los datos de trayectoria, que esta posee las características temporales, de distancia y de movimiento deseadas.

Con esta guía de referencia del diseño de trayectorias obtenida a través de la práctica se puede comprender el ejemplo de la librería, que define, mediante cinco segmentos, una trayectoria en forma de "S" girada con ángulos pronunciados. También, este conocimiento nos permitirá, posteriormente, diseñar otras trayectorias diferentes en la fase de desarrollo.

Tras generar la trayectoria, se añaden los errores a la IMU y al GPS para la generación de datos más parecidos a los que se obtendrían en la vida real, mediante funciones auxiliares. Seguidamente, se calculan los errores en las medidas de posición, velocidad y orientación a partir de las definiciones iniciales de error, para lo cual es necesario utilizar matrices de transformaciones en algunos casos.

Finalmente, se unifican estas medidas iniciales -con el error añadido- en un vector que será el resultado obtenido de esta función y, por tanto, lo que se obtendrá en el programa principal como inicialización del sistema.

Es necesario apuntar que el *heading*, por su carácter especial en esta aplicación se trata, también, de forma individual como *wander* para las inicializaciones y para todo el proceso del Filtro Kalman.

Fruto de un proceso largo y exhaustivo de investigación, documentación y experimentación con el código, obtenemos nociones avanzadas sobre funcionamiento de la librería que vamos a usar, así como nociones básicas de los principios físicos y matemáticos que subyacen en su funcionamiento.

CAPÍTULO 4: Datos Sintéticos

En el cuarto capítulo, a partir del ejemplo seleccionado como alternativa de resolución de la librería descrita anteriormente, se desarrollará un prototipo basado en la simulación de modelos de trayectorias generadas sintéticamente, que llevará a cabo pruebas sobre diversos parámetros del sistema para analizar su impacto sobre la bondad de la estimación. Así, se diseñará una aplicación de análisis de datos sintéticos mediante el diseño de los escenarios y los casos de prueba, el diseño de los experimentos a realizar y la evaluación de los resultados.

Diseño e implementación: Aplicación de análisis de datos sintéticos

Basándonos en el ejemplo *LargeHeading* de la librería, creamos un entorno de pruebas basadas en trayectorias sintéticas para analizar el efecto de los elementos del sistema y los parámetros que influyen en la estimación de un *heading* inicial desconocido o con un nivel de corrección en la estimación muy bajo.

Diseño de los escenarios y casos de prueba

En primer lugar, será necesario definir los entornos y los casos de prueba que se analizarán. Sin embargo, antes de definir las pruebas a realizar, es necesario establecer una serie de condiciones que se mantendrán fijas durante el desarrollo de las mismas. Como se observó al comenzar a realizar pruebas con la librería para comprobar la viabilidad de la librería, el número de muestras es un elemento determinante en la calidad de la estimación. Como es de esperar por la tendencia a converger de Kalman y por la estabilidad general de los sistemas predictivos, cuanto mayor sea el número de muestras a computar, mejor será la estimación.

Como no es un factor controlable en un caso práctico, pues lo deseable sería que el programa funcionase en tiempo real, este factor no influirá en el estudio de la influencia los parámetros. Para ello, se definen todos los escenarios de prueba con un número de muestras del mismo orden y atendiendo a las condiciones del escenario real.

Teniendo en cuenta la naturaleza de la actualización y la predicción que lleva a cabo Kalman y a la integración de los sistemas de navegación inerciales, se llega a la conclusión de que un conjunto de factores susceptibles de ser considerados como casos de prueba serían los errores del sistema. Es decir, es indispensable comprobar la estabilidad de la estimación ante un aumento en los errores desde diferentes frentes. Estos errores que influyen en el sistema serían, por un lado, los errores del GPS y, por otro, los errores de la IMU, incluyendo los errores de giróscopo y acelerómetro. Además, en la generación de estos errores en la trayectoria sintética se utiliza un generador aleatorio, por lo que la semilla también debería cambiarse a lo largo de las pruebas para que sea completamente efectivo el aumento en los errores del sistema.

Estos errores del sistema se estudiarán, de forma separada, para cada una de las tres trayectorias desarrolladas como escenarios de prueba. Todas estas trayectorias, como se indicó al inicio, presentarán unas condiciones iniciales, que serán condiciones temporales, de distancia y velocidad y, por supuesto, de frecuencia de muestreo.

Así, todas las trayectorias presentan las siguientes condiciones:

- Tiempo total de recorrido del orden de 5 minutos.
- Distancia total recorrida del orden de 10 kilómetros.
- Velocidad media durante la trayectoria del orden de 30 m/seg.
- Incremento de tiempo entre muestras reales de 0.01 segundos.
- Incremento de tiempo entre estimaciones de 0.17 segundos.

Estas condiciones se han seleccionado por ser condiciones aceptables para un vuelo de calibración de un helicóptero. Temporalmente, la calibración debe durar el menor tiempo posible y, puesto que los archivos de vuelo pueden tener una duración de horas, cinco minutos es una medida lo suficientemente pequeña para esas extensiones de archivo y, a la vez, es tiempo más que suficiente para la calibración del vehículo.

En cuanto al resto de condiciones, estas se asemejan a las de un vuelo cualquiera en el vehículo aéreo cuyas velocidad no son demasiado altas por el carácter de medición que presenta el vuelo.

La uniformidad en las condiciones de las trayectorias puede observarse de la información que proporciona el programa de cada una de las trayectorias. Así, tendríamos la información de cada una de las trayectorias a continuación:

- **Trayectoria 1**

```
INFORMACIÓN TRAYECTORIA
Tiempo total = 313.88 seg
Distancia total recorrida = 9028.41 m
Velocidad media = 28.76 m/seg
Inc. tiempo entre muestras reales = 0.01 seg
Inc. tiempo entre estimaciones = 0.17 seg
```

- **Trayectoria 2**

```
INFORMACIÓN TRAYECTORIA
Tiempo total = 300.00 seg
Distancia total recorrida = 12721.71 m
Velocidad media = 42.41 m/seg
Inc. tiempo entre muestras reales = 0.01 seg
Inc. tiempo entre estimaciones = 0.17 seg
```

- **Trayectoria 3**

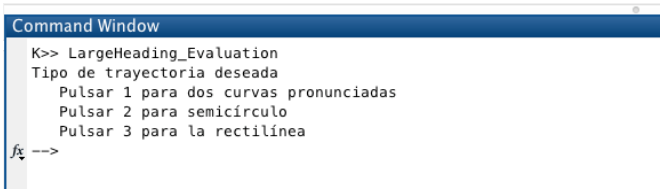
```
INFORMACIÓN TRAYECTORIA
Tiempo total = 288.00 seg
Distancia total recorrida = 8025.70 m
Velocidad media = 27.87 m/seg
Inc. tiempo entre muestras reales = 0.01 seg
Inc. tiempo entre estimaciones = 0.17 seg
```

En cuanto a la naturaleza de las trayectorias, se han creado tres escenarios diversos a la vez que cercanos a un ejemplo práctico de vuelo. Como puede verse en la Ilustración 16, las trayectorias creadas serían las siguientes:

- **Trayectoria 1**: Trayectoria parecida a la original del ejemplo, pero normalizada para que sus tramos sean de la misma distancia y reorientada. Esta trayectoria se comprondría de dos curvas pronunciadas que forman una "S". Con esta trayectoria tendríamos la prueba para recorridos con tramos rectos y curvas pronunciadas.

- **Trayectoria 2**: Trayectoria que refleja un semicírculo orientado. Esta se crea para evaluar los escenarios donde el *heading* permanece constante en el giro continuado.

- **Trayectoria 3**: Trayectoria rectilínea con algunos escalones pequeños. Esta se crea para obtener una trayectoria rectilínea no perfecta que se asemeje, en lo posible a un recorrido recto con las imperfecciones que presenta un vuelo real.



Diseño del conjunto de experimentos a realizar

Atendiendo a lo explicado en el apartado anterior, para cada una de las trayectorias se desarrolla un conjunto de experimentos similares para analizar la influencia del error en la estimación.

Para analizar el error del GPS (sistemas GNSS), bastará con modificar la variable *obs_error* que se inicializa en la función *init_conf* de la forma en que se muestra en la Ilustración 17.

```
24    %Error en la observación del GPS
25 -  sysConfig.obs_error=diag([0.5/Sn, 0.5/Se, 1.5]); %position error std (m)
```

Ilustración 17: Datos sintéticos - Inicialización de la variable obs_error

De esta forma, para todas las trayectorias, solo será necesario modificar esta variable, para aumentarla según los valores definidos en la Ilustración 18, que serían:

- El doble de su valor original → 1 metro en [x,y] y 3 metros en la z.
- Diez veces su valor original → 5 metros en [x,y] y 15 metros en la z.
- Doscientas veces su valor original → 100 metros en [x,y] y 300 metros en la z. Este último es un valor exageradamente elevado, para ver si realmente es un parámetro influyente.

```
2   function tests_obsError(numTrayect,trayect,path,sysConfig,sysConfigOriginal,paramConfigOriginal)
3
4 -     testObsError = [1,2,10,200];
5
```

Ilustración 18: Datos sintéticos – Definición de los valores de prueba de la variable obs_error

En cuanto a los parámetros de error de la IMU (sistemas INS), estos dependen de los modelos de error de los acelerómetros y giróscopos. Estos modelos de error se explicaron en el apartado de documentación de la librería y están definidos por los parámetros A, B, C, D y sP, en ambos casos.

Por tanto, para cada uno de ellos en el giróscopo y en el acelerómetro, tomamos cinco valores de prueba partiendo del valor central, que será el valor definido como valor predeterminado en la configuración del modelo de error de sensores de la librería. Así, de forma general, en base a este valor central, los demás valores pivotarán, generando cinco escenarios de prueba.

Cada una de las pruebas de cada valor de parámetro, a su vez, por motivos de fiabilidad estadística, deberá llevarse a cabo 31 veces con diferentes semillas de error (*sysConfig.rstat*).

Por este motivo, las pruebas de cada parámetro se realizarán dejando los demás parámetros en su valor central y modificando solo ese parámetro en ese escenario. Podría probarse la influencia de todas las combinaciones posibles de configuración del modelo de error con los valores de los parámetros, pero eso implicaría un tiempo de computación excesivo para los medios de los que disponemos:

5 (nº de valores) * 10 (nº de parámetros) * 31 (nº de iteraciones de cada valor) * 10 (10 seg. es el tiempo estimado de cada prueba) / 3600 = 4.3056

Esto implica que se tardaría una media de 4 horas y media para probar cada uno de los valores de los parámetros, lo que supondría un tiempo total no admisible para las pruebas.

En cuanto a los valores elegidos para cada uno de los parámetros, se ha seguido un criterio de obtención de límites de un 10% de error para la orientación, que es el sector más crítico en la estimación. Estos límites se consiguen a través de la experimentación, pivotando desde el valor central disminuyéndolo, en un caso, y aumentándolo, en el otro, en función del rango de crecimiento. Una vez obtenidos los límites, se calcula el elemento medio y así se obtienen los cinco valores basados en el valor central, que se supone ideal, con los que realizar las pruebas.

Estos valores se basan en la suposición de que los valores centrales son los que mejores resultados de error producen. Sin embargo, esta hipótesis no se cumple para todas las trayectorias.

Experimentalmente, se observa que las trayectorias 1 y 3 sí cumplen la hipótesis. Los resultados obtenidos, como se verá en el apartado de evaluación, se corresponden a la distribución esperada que es de error creciente o constante conforme nos alejamos del valor central. Así, obviando las oscilaciones, los valores coloreados en naranja presentarán el peor error, mientras que los amarillos presentarán un error medio y el valor verde será el valor central de mejor medida de error. Es necesario tener en cuenta que las diferencias pueden ser mínimas o no apreciarse y el error puede permanecer constante, por lo que los colores son solo un indicativo del comportamiento general y esperado.

Con la trayectoria 2, nos encontramos con que no cumple con lo esperado. Los valores centrales no eran los que mejor error proporcionaban, sino, en muchos casos, se daba el caso contrario, como puede verse en la Ilustración 19. Aquí se ve que para los valores centrales (3 en el eje x) de los parámetros el error en el *heading* es significativamente mayor que en los valores intermedios y en los extremos.

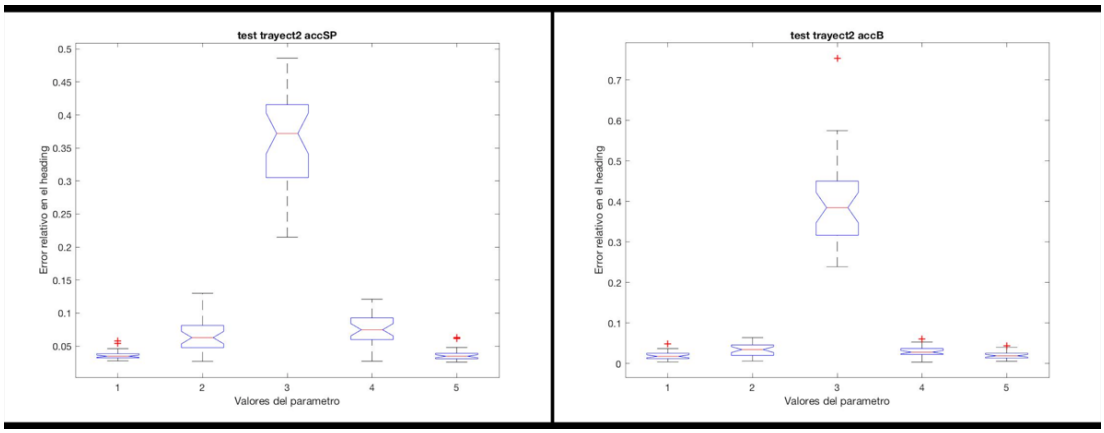


Ilustración 19: Datos sintéticos - Ejemplos mal rango en la trayectoria 2

Esto implica que es necesario redefinir los valores centrales y los límites de los parámetros. Así, los parámetros que aparecen con el sombreado rojo en la tabla son aquellos que se han redefinido y, aquellos con el sombreado azul, son aquellos que han permanecido inmutables con respecto a los de las otras dos trayectorias.

Con el acelerómetro, tan solo fue necesario modificar el parámetro A, ajustando sus valores hasta encontrar un valle, es decir, un valor en el que, al alejarse el error aumente en ambos extremos. A partir de ahí, se busca alcanzar el error del 10% en orientación y establecemos como límite el valor inmediatamente anterior a la obtención de ese valor de error.

Este procedimiento se repite con los parámetros del giróscopo, los cuales fue necesario modificar casi en su totalidad, con la excepción del parámetro A, que permanece constante. En este procedimiento, cabe mencionar la peculiaridad del parámetro C del giróscopo en la trayectoria 2, cuyo límite no se calcula con la obtención del 10% de error, sino por límites de funcionamiento. Es decir, el error en la orientación permanece constante y menor al 10%, indiferentemente del aumento o decremento del parámetro, por lo que el límite escogido es aquel en el que se permite el funcionamiento de la aplicación.

En las siguientes gráficas se pueden observar todos los valores con los que se llevarán a cabo los experimentos para cada trayectoria y para cada sensor.

TRAYECTORIAS 1 Y 3 - ACELERÓMETRO

Parámetros	Valores				
A	0.02	0.11	0.2	0.6	1
B	-115	-57.5	3e-006	57.5	115
C	-1.5e+7	-7.5e+6	1	7.5e+6	1.5e+7
D	-47	-23.5	1e - 3/sqrt(0.01)	23.5	47
sP	-104	-52	sqrt(5.1e-007)	52	104

TRAYECTORIAS 1 Y 3 - GIRÓSCOPO

Parámetros	Valores				
A	0.02	0.51	0.999980000199999	0.99999	1
B	-2.4e-4	-1.2e-4	1.9999800001658e-006	1.2e-4	2.4e-4
C	-74	-38	1	38	74
D	-1.4	-0.7	0.0006/sqrt(0.01)	0.7	1.4
sP	-5e-2	-2.5e-2	sqrt(1e-007)	2.5e-2	6e-2

TRAYECTORIA 2 - ACELERÓMETRO

Parámetros	Valores				
A	1e-18	5e-4	1e-3	5.45e-3	9.9e-3
B	-115	-57.5	3e-6	57.5	115
C	-1.5e+7	-7.5e+6	1	7.5e+6	1.5e+7
D	-47	-23.5	1e -3/sqrt(0.01)	23.5	47
sP	-104	-52	sqrt(5.1e-007)	52	104

TRAYECTORIA 2 - GIRÓSCOPO

Parámetros	Valores				
A	0.02	0.51	0.999980000199999	0.99999	1
B	-6e4	-3e4	0	3e4	6e4
C *(Estable: límites de funcionamiento)*	-2e4	-1e4	0	1e4	2e4
D	-1e8	-5e7	0	5e7	1e8
sP	-6e5	-3e5	0	3e5	6e5

Diseño e implementación del prototipo

Para el desarrollo del prototipo que lleva a cabo las pruebas automatizadas se usó como base el código de ejemplo de la librería, pero el resultado final difiere en gran medida del original.

La función encargada de llevar a cabo y dirigir el desarrollo de las pruebas sería *LargeHeading_Evaluation*, cuyo diagrama básico de funcionamiento sería el mostrado en la Ilustración 20. Este programa será el encargado de mostrar el menú de selección de trayectoria y de definir los parámetros iniciales a partir de esta trayectoria con la función *init_config*.

Esta función de inicialización, además de definir los parámetros iniciales del sistema de forma análoga a la función de la librería explicada con anterioridad, agrupa los parámetros que intervienen en dos estructuras diferenciadas: *sysConfig* y *paramConfig*. Así, *sysConfig* tendrá los parámetros del sistema usados en la inicialización y para las observaciones y definiciones del sistema de navegación y *paramConfig* contendrá los parámetros de los modelos de error que, en este caso, solo incluyen los de los acelerómetros y giróscopos de la IMU. Con estas dos estructuras se tiene la configuración del sistema en cada momento.

Tras la definición de los parámetros iniciales, se define la trayectoria en función de la opción elegida a través de la función *define_trayect* que, de acuerdo a como se explica en el apartado de documentación de la librería, define la matriz de definición del movimiento, ahora llamada *trayect*, para obtener las figuras deseadas en el recorrido. Además, esta función, dependiendo de la trayectoria seleccionada, inicializa las matrices que contienen los valores de los parámetros como se muestran en las tablas del apartado anterior para su posterior uso en la realización de pruebas.

Antes de realizar las pruebas, se guarda la configuración inicial de *sysConfig* y *paramConfig* puesto que en las pruebas se modificarán y, entre pruebas y en la generación de trayectorias, será necesario el acceso a la configuración original.

A continuación, llevamos a cabo la batería de tests con las funciones *tests_obsError* para las pruebas del GPS y con *tests_errorParams* para las pruebas del acelerómetro y el giróscopo de la IMU explicadas en el apartado previo.

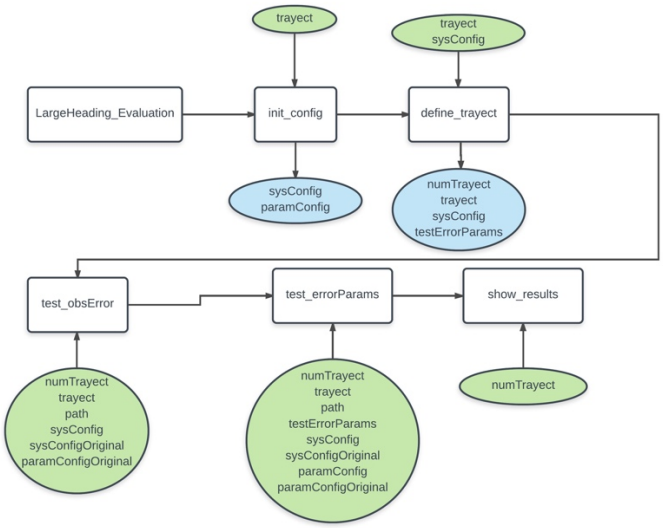


Ilustración 20: Datos sintéticos - Diagrama funcionamiento LargeHeading_Evaluation

Las pruebas basadas en la adición de error en el GPS mediante la variable *obs_error* basan su funcionamiento en el esquema de la Ilustración 21. Tras definir el vector de valores multiplicativos de aumento del valor de la variable, se lleva a cabo el bucle principal. Este lleva a cabo, para cada uno de los valores aumentados de la variable *obs_error* a probar, 31 iteraciones variando la semilla (*sysConfig.rstat*) del random y generando el nombre apropiado para el fichero y para el salvado del *workspace* de Matlab. Para que eso sea efectivo, será necesario, al cambiar los parámetros y variables del sistema, generar una trayectoria nueva con la modificación del *sysConfig* y realizar una estimación con el nuevo *sysConfig* modificado también.

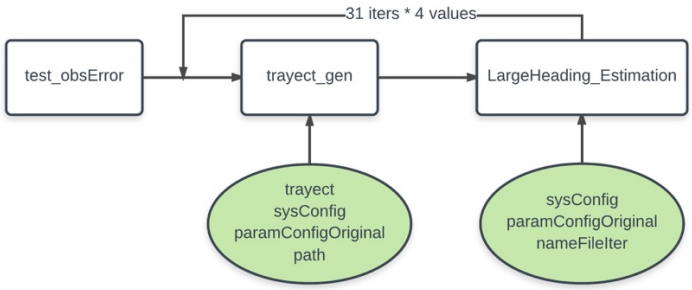


Ilustración 21: Datos sintéticos - Diagrama funcionamiento test_obsError

Tras las pruebas del GPS, tienen lugar los casos de prueba de los parámetros del modelo de error de los sensores de la IMU. Anteriormente se ha definido la matriz de pruebas *testErrorParams* y esta se va recorriendo fila a fila, leyendo los valores a probar de cada

parámetro y modificándolos con la ayuda de funciones auxiliares como *modify_params* y creando los nombre de archivos necesarios con la función auxiliar *get_param_name*. Así, como en las pruebas del GPS, se llevan a cabo 31 iteraciones con diferentes semillas del random para cada uno de los valores de los parámetros y, con estos nuevos estados del sistema (*sysConfig*) se crean las trayectorias con *trayect_gen*, que llama a la función auxiliar de generación de trayectorias y añade los errores a la IMU y al GPS. Finalmente, se lleva a cabo la estimación para cada valor.

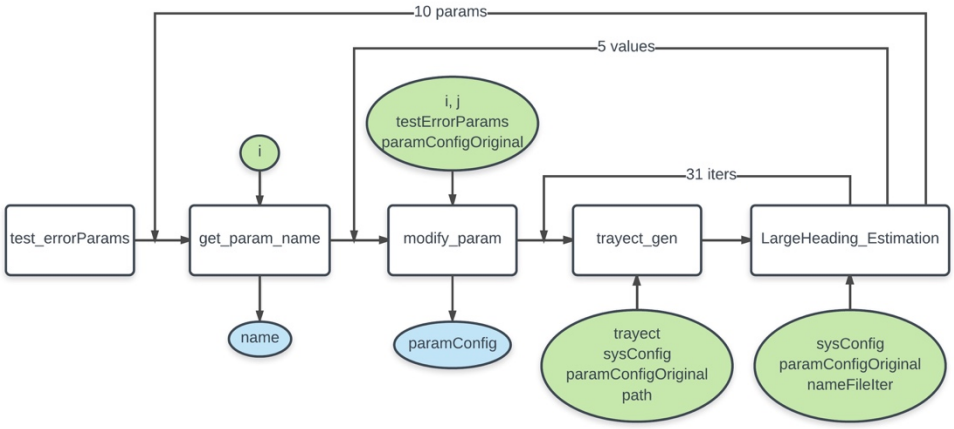


Ilustración 22: Datos sintéticos - Diagrama funcionamiento test_errorParams

Tras realizar las pruebas, se unifican los resultados para mostrar una comparativa de los diferentes valores de los parámetros. Así, se llama a una función *show_results* que, como su nombre indica, se encargará de mostrar esas ayudas visuales para las conclusiones.

Como puede verse en la Ilustración 23, se llama a *results_errorParams* con cada uno de los ficheros que unifican los resultados de los errores de cada parámetro. Así, cuando se llama a la función, esta muestra una figura de distribución de datos que unifica las 31 iteraciones de estabilidad estadística y muestra una gráfica con todos los valores del parámetro enfrentados a su error de *heading*, como se verá más extensamente en la sección de resultados.

Análogamente, se muestra un histograma que muestra la evolución de la pendiente del *heading* a lo largo del tiempo. Esto nos permitirá analizar la convergencia del ángulo.

Una vez se ha mostrado todo el soporte gráfico de los parámetros de error, se muestra la gráfica relativa a las pruebas del GPS. Esta será análoga a una función individual de un parámetro del modelo de error y la obtención de la misma es similar. Se verá y se comentará más extensamente en la sección de resultados.

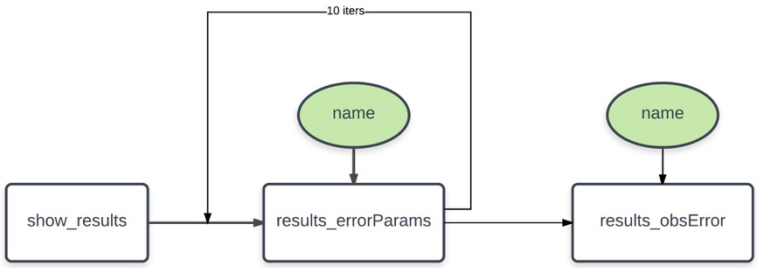


Ilustración 23: Datos sintéticos - Diagrama funcionamiento show_results

En cuanto a la función que lleva a cabo la implementación del filtro Kalman, esta es análoga a la definida en la librería, con la diferencia de que se redefine el tiempo de propagación para poder acomodarlo al nuevo intervalo de tiempo, como puede verse en la Ilustración 25. Además, al final, se lleva a cabo un proceso de obtención de errores estadísticos para obtener resultados numéricos y un proceso de visualización de una serie de gráficas que constituirán un apoyo visual de los datos.

Esto se lleva a cabo a través de las funciones *show_save_results,* que se encarga del cálculo y el almacenamiento del error y de mostrarlo en consola, y *show_graphs*, que se encarga de mostrar figuras de comparación de trayectorias, velocidades y ángulos.

La función *show_save_results* calcula, por una parte, la información relativa a la trayectoria, mostrada y caracterizada previamente. Además, para definir estadísticamente el problema, es necesario el cálculo de la media del error y la desviación típica del mismo. Para ello, es necesario interpolar los valores de la trayectoria sin errores con la trayectoria estimada, pues las diferencias de mediciones son notables. Además, es preciso ajustar los ángulos tras cada cálculo para que estos se encuentren dentro del rango de 180º, para evitar discordancias en las gráficas y la obtención de resultados negativos en los errores de los ángulos.

A la hora de computar los errores, se lleva a cabo a partir del error relativo ya que es necesario hacer uso de la escala para interpretar los errores en el contexto en que se generan. Para el caso de la posición, por ejemplo, un error de unos pocos metros en una

trayectoria de, aproximadamente, 10 kilómetros no es relevante y, por tanto, debe ser bajo. El error absoluto no tiene cabida en un sistema con tantas variables diversas.

En el caso de las variables de posición, la relativización se lleva a cabo a través del mayor elemento del vector de medidas pero, en el caso de las orientaciones, no puede llevarse a cabo de esta forma por ser demasiado pequeños los valores en algunos casos. Por ello, se estima el rango de variación de los ángulos en un vehículo aéreo y se utilizan como constantes para llevar a cabo la relativización del error. Así, para *pitch* y *roll* se utiliza 90º como rango y, para el *heading*, se utiliza 180º.

Una vez hemos calculado el error relativo, obtenemos la media y la desviación típica a través de él y la almacenamos en un archivo que unifica todos los valores de un mismo parámetro. Junto a los errores guardamos el valor de la pendiente del *heading* para analizar la convergencia del mismo. Este valor se obtiene a partir de los dos últimos tercios de la gráfica del *heading* puesto que, por motivos de inicialización, el *heading* sufría unas fluctuaciones muy elevadas al inicio y esto ocasionaba que los datos de la pendiente no fuesen del todo precisos y estuvieran falseados por esta alta variación inicial.

La función *show_graphs* muestra diversas figuras que caracterizan el resultado de la estimación desde diferentes frentes. De esta manera, tendríamos las siguientes gráficas:

- Gráfica comparativa de trayectoria verdadera (*true*) sin error (azul), trayectoria observada por el GPS con error (verde) y trayectoria estimada (rojo). Todo esto en metros y, tan solo en las coordenadas [*x,y*], sin tener en cuenta la altura, por no complicar el resultado visual con gráficas en 3D.

- Gráfica comparativa del *heading* verdadero (*true*) sin error (azul) y la estimación del mismo (rojo) en el tiempo. El ángulo de cabeceo estará en grados y el tiempo en segundos. En la misma figura también se mostrará el error absoluto del *heading* en el tiempo, por lo que se podrá visualizar el fenómeno de la convergencia.

- Gráfica explicativa que muestra la evolución de la velocidad (en m/s) en el tiempo (en segundos).

- Gráfica explicativa que muestra la evolución de la velocidad angular (en rad/s) en el tiempo (en segundos).

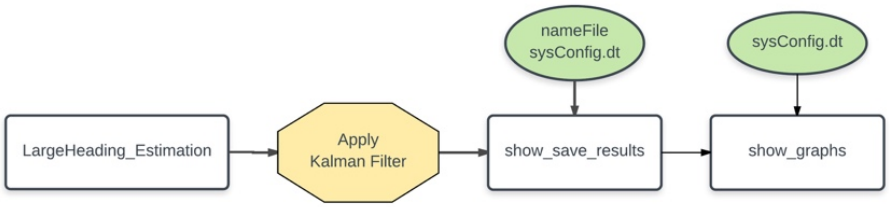


Ilustración 24: Datos sintéticos - Diagrama funcionamiento LargeHeading_Estimation

```
19
20      %propagation time for the discretized system
21 -    prp_per=round(0.33/2/sysConfig.dt);
22
```

Ilustración 25: Datos sintéticos - Redefinición del tiempo de propagación

Con todo esto, tendríamos una aplicación que, para la alternativa seleccionada, realiza un análisis con el propósito de determinar la precisión del resultado y la robustez del mismo ante variaciones en los parámetros de los modelos y algoritmos empleados.

Evaluación y validación: Resultados del análisis de datos sintéticos

Los resultados obtenidos de la batería de tests descrita anteriormente se encuentran en el ***Anexo I*** agrupados según trayectoria y sensor (IMU acelerómetro, IMU giróscopo o GPS).

ANÁLISIS DE LOS RESULTADOS

Para la primera trayectoria en el acelerómetro, nos encontramos con valores del parámetro A casi constantes. Tanto la mediana como los cuartiles parecen permanecer constantes en unos valores de error entre el 0.015 y el 0.030. Se pueden ver las oscilaciones al hacer zoom en la figura, como puede observarse en la Ilustración 26 y, además, se ve el error crecer en el límite inferior que es el que está más alejado del valor central. Estas oscilaciones son tan pequeñas que se desprecian y, a pesar de que existe una elevada dispersión en algunos, estos valores no se tienen en cuenta al realizar los cálculos por su naturaleza esporádica.

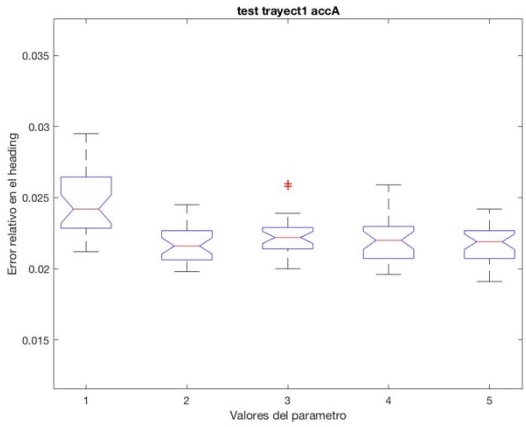


Ilustración 26: Zoom del boxplot de accA trayect1

En cuanto al parámetro B, dado que los rangos que se obtienen para el parámetro son tan elevados, es de esperar ese comportamiento tan drástico que se observa en su gráfica. Así, en los extremos, a pesar de que el error en la orientación es menor al 10%, el error en el *heading* individualmente ascienda hasta el 60%. Obviando los valores tan altos que se obtienen, este tipo de comportamiento es el esperado, que al alejarse del valor central, la estimación del *heading* empeore.

De forma similar ocurre con demás parámetros del acelerómetro, con la diferencia de que algunos presentan resultados más dispersos que otros, siendo el error más variable en esos casos, y que el rango de error varía en cada caso.

Se puede decir, en forma general, que, en la trayectoria 1, el acelerómetro presenta el comportamiento esperado, confirmando la hipótesis de que los valores centrales son los más adecuados.

En el giróscopo en la trayectoria 1 se obtiene también el comportamiento esperado, obteniendo valores de error en la mediana bastante similares para la gran mayoría de valores de los parámetros. Cabe destacar de este sensor que el error parece disminuir y permanecer así de bajo en todas las pruebas de los parámetros. Sin embargo, también presentan muchas más oscilaciones en el comportamiento en forma de “U” o “V” que se espera obtener y se obtienen unos cuartiles mucho más anchos y unos percentiles más altos, lo que índice un error más variable y menos concentrado.

En cuanto a la trayectoria 3, con ella obtenemos unos valores muy similares a los de la trayectoria 1, siendo la gráfica de la A y B del acelerómetro casi iguales. Cabe destacar que se consiguen niveles de error incluso menores. Se repite el patrón de comportamiento de los parámetros del giróscopo de la trayectoria de curvas pronunciadas en esta tercera trayectoria, la rectilínea.

Con esto, si miramos las gráficas de los valores centrales de estas dos trayectorias, tendremos una aproximación muy buena de la trayectoria (como puede verse en las Ilustraciones 27 y 28) del vehículo y una muy buena aproximación del *heading* también (como puede verse en las Ilustraciones 29 y 30).

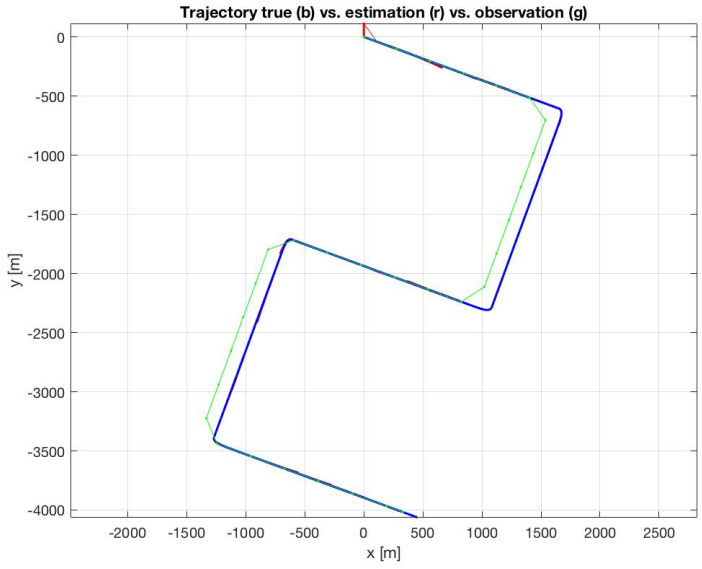


Ilustración 27: Datos sintéticos - Trayectoria 1

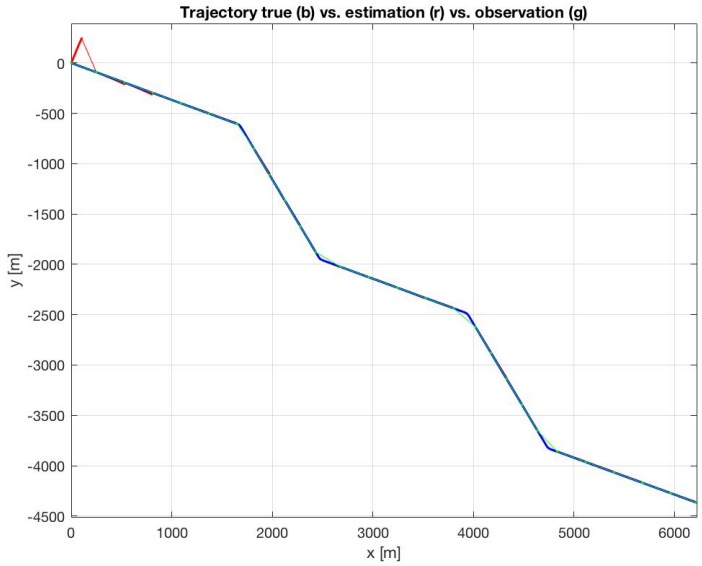


Ilustración 28: Datos sintéticos - Trayectoria 2

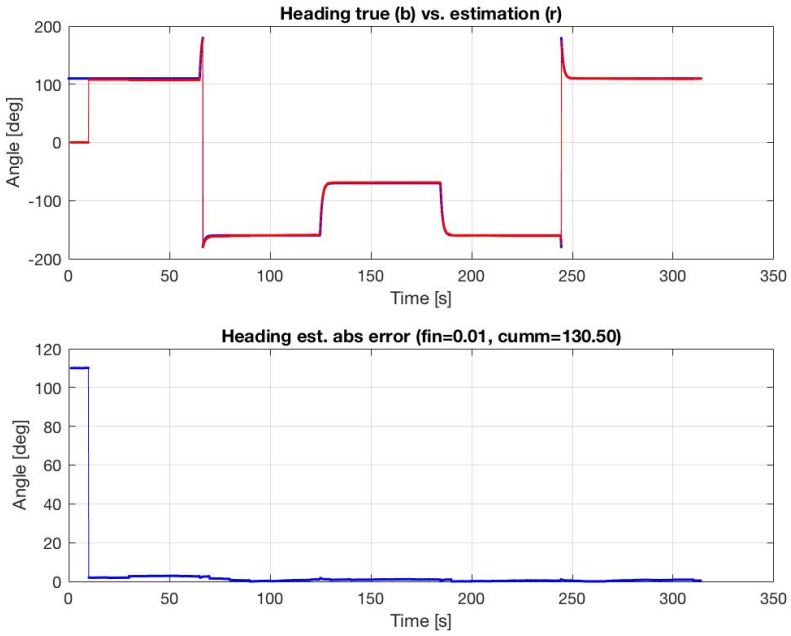


Ilustración 29: Datos sintéticos - Heading Trayectoria 1

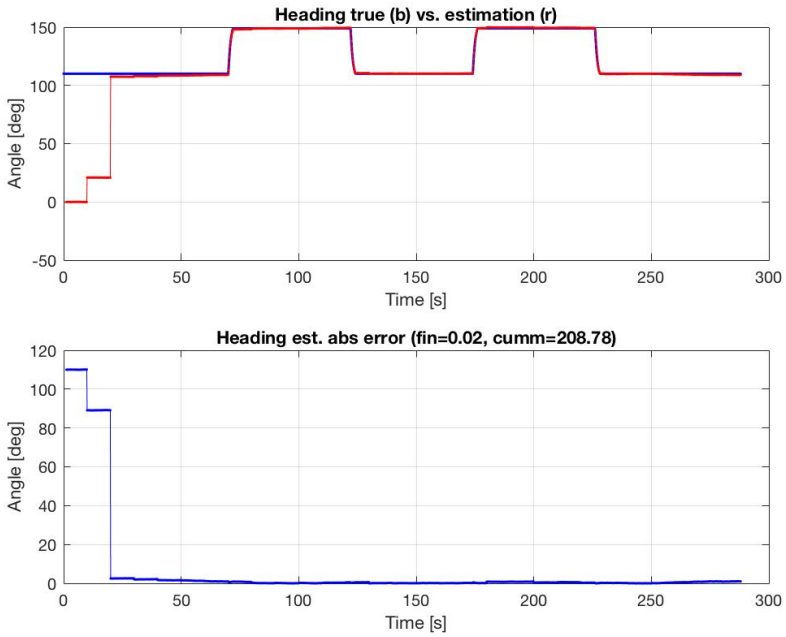


Ilustración 30: Datos sintéticos - Heading Trayectoria 2

Con la trayectoria 2 los resultados son menos concluyentes. Si utilizamos los valores centrales, como se explicó anteriormente, los errores no solo ascienden demasiado, sino que no siguen el comportamiento esperado (forman una campana en muchos casos, en lugar de formar una "V"). Sin embargo, en el acelerómetro, al modificar los rangos del parámetro A y dejando los demás fijos, obtenemos resultados casi constantes y del mismo valor para todos los valores de prueba de los demás parámetros. Este comportamiento es, cuanto menos, anómalo, pero nos ofrece resultados bastante bajos de error con estos parámetros en el modelo de error.

Además, si mantenemos este valor central de la A del acelerómetro, las pruebas en el giróscopo parecen seguir, en cierta medida, el comportamiento esperado, aunque son todos bastantes constantes, a excepción del parámetro D. Es necesario puntualizar, también, que estos resultados se obtienen tras redefinir los límites y los valores centrales del giróscopo, acción que no es necesaria en las demás trayectorias.

Para reflejar esta diferencia, se mostrarán las gráficas de trayectoria y *heading* para la trayectoria del semicírculo con la inicialización de parámetros de las trayectorias anteriores (Ilustraciones 31 y 33) y, en comparación, las mismas gráficas con los

parámetros que, mediante experimentación, se obtiene que son los idóneos para la trayectoria (Ilustraciones 32 y 34).

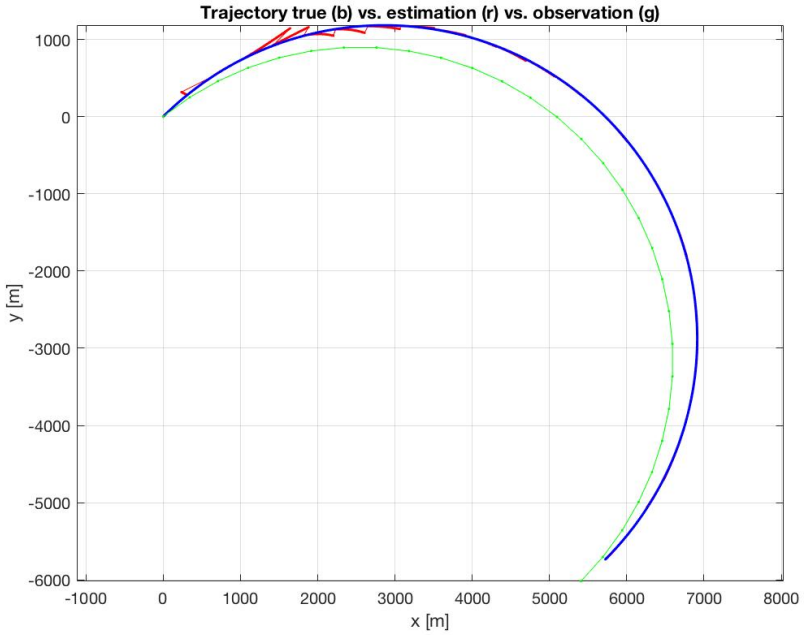


Ilustración 31: Datos sintéticos - Trayectoria 2 con mala estimación

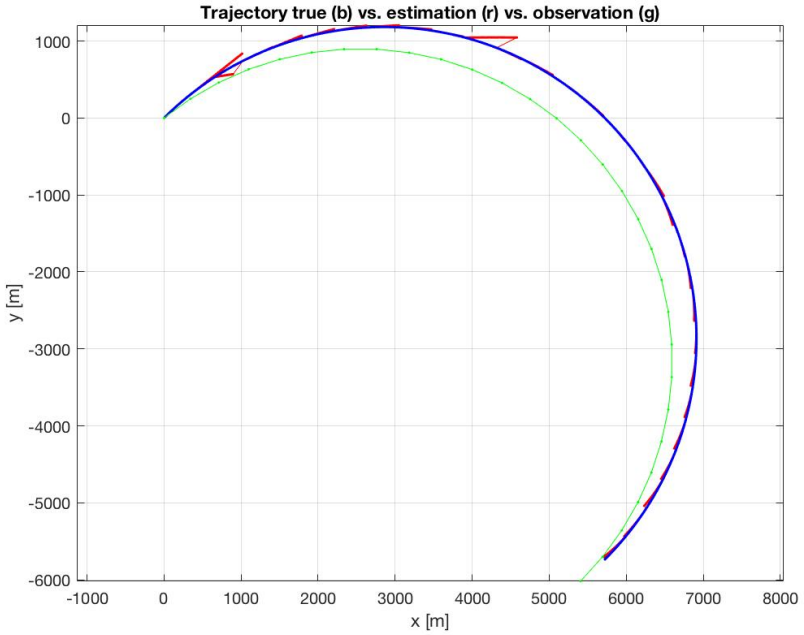


Ilustración 32: Datos sintéticos - Trayectoria 2 con buena estimación

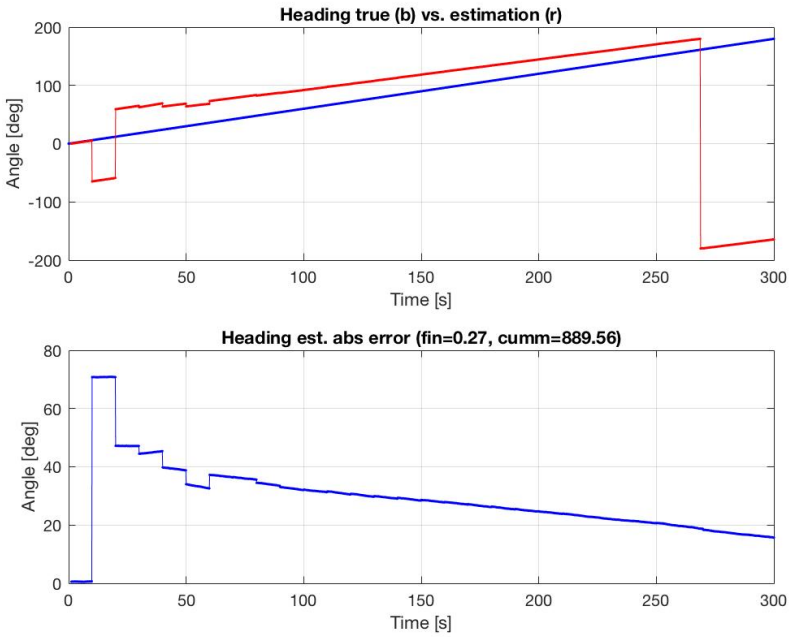


Ilustración 33: Datos sintéticos - Heading de la Trayectoria 2 con mala estimación

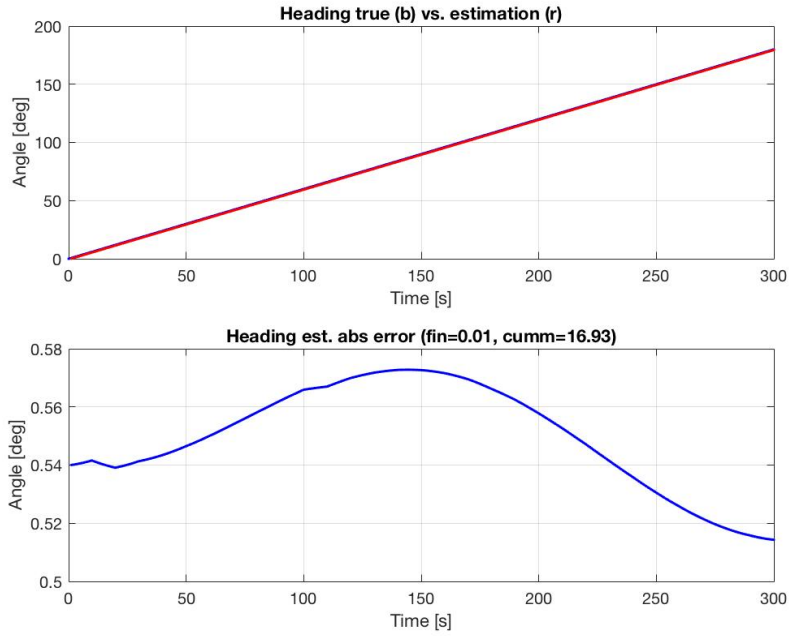


Ilustración 34: Datos sintéticos - Heading de la Trayectoria 2 con mala estimación

Como puede verse en las gráficas, la diferencia es más bien perceptible en el *heading*, que es el que más diferencia de error presenta entre una y otra. La orientación no sufre mayores cambios, pero si se intuyen algunos errores de heading en algunas zonas concretas.

En cuanto al error del GPS, para las trayectorias de curvas pronunciadas y rectilíena se obtiene el mismo patrón de resultado: cuando crece el error en el GPS, también aumenta el error en el *heading*, aunque no significativamente. Sin embargo, en la trayectoria circular, el efecto del ruido del GPS es el mismo en el error del *heading*, a pesar de aumentar considerablemente.

En cuanto a la convergencia del *heading*, esta es más fácil de ilustrar en la siguiente tabla. En ella se muestra qué parámetros de qué modelo de error (del acelerómetro o del giróscopo) tienen a la convergencia del error, es decir, presentan una pendiente mayormente negativa. Este cálculo de la pendiente se hace teniendo en cuenta únicamente los dos últimos tercios de la gráfica de error del *heading*, pues, de forma general, este presenta una tendencia a comenzar muy alejado y acercarse rápidamente en los primeros segundos. Esta clase de comportamiento ocasiona que se falseen los resultados, pareciendo que todos convergen. Por tanto, se obvia ese tramo y se tiene en cuenta el resto de la tendencia del *heading*.

Trayectoria	Acelerómetro		Giróscopo	
	Converge	No Converge	Converge	No Converge
Trayect 1	B, D	A, C, sP	-	A, B, C, D, sP
Trayect 2	A, B, C, D, sP	-	A, B, C, D, sP	-
Trayect 3	A, B, C, D	sP	B, C	A, D, sP

CONCLUSIONES Y CALIDAD DE LA ESTIMACIÓN

Analizando los resultados obtenidos de la batería de test, podemos concluir que la estimación es sensible a diversos factores.

Por un lado, es sensible a los parámetros de error del sistema, aunque no en la misma medida. Los parámetros del modelo de error de la IMU parecen influir considerablemente en la estimación, pero el error del GPS, teniendo en cuenta que el límite es el valor aumentado notablemente, no sufre gran cambio. Quizá este valor combinado con la inserción de error en otras medidas ofrecería resultados diferentes, pero, de forma individual, no aporta gran interés al análisis de influencia de parámetros.

A su vez, no todos los parámetros del modelo de error tienen la misma relevancia en la estimación global del *heading*. Parámetros como la A del acelerómetro que, dentro de un rango relativamente pequeño son capaces de producir grandes cambios en la estimación del *heading* y de la trayectoria global son los que realmente son relevantes. Sin embargo, parámetros como la B o la C del acelerómetro, que tienen un rango de variación tan desorbitadamente grande, pueden estar generando efectos de inestabilidad numérica y, por tanto, no es deseable obtener conclusiones de ellos.

Por otro lado, el sistema es sensible al tipo de trayectoria, pero esta influencia parece venir restringida, a su vez, por los parámetros del modelo de error de la IMU. La configuración inicial de parámetros parecía ser un punto de referencia de error mínimo para todas las trayectorias. Sin embargo, nos encontramos con que esta combinación de parámetros parece estar ajustada para funcionar correctamente con trayectorias rectilíneas de curvas momentáneas, como el tipo de movimiento descrito en la librería por defecto, pero no ofrece el menor error para la orientación en el caso de la trayectoria circular. Esto puede deberse a que la naturaleza de la trayectoria es distintas desde el punto de vista del parámetro.

La trayectoria circular con altos valores en el parámetro A del acelerómetro constituye un sistema inestable que comienza a oscilar y es incapaz de revertir la oscilación. Al bajar la ganancia del sistema (disminuir la A que realimenta al sistema) el sistema se estabiliza y vuelve a engancharse en la estimación.

En cuanto a la convergencia del *heading*, si bien es verdad que una mayor convergencia define un sistema más estable y más tendente a eliminar el error o a mantenerlo muy

bajo -lo deseable-, también es cierto que muchas veces, las fluctuaciones en las gráficas generan resultados engañosos de tal forma que, aunque se clasifiquen como "no-convergentes" no tiene, necesariamente, que implicar que no se vaya a lograr un sistema estable o que no se vaya a corregir el error. Únicamente es indicativo de un *heading* que fluctúa más y, para sacar conclusiones concretas, sería necesario un análisis exhaustivo de cada una de las gráficas por separado.

CAPÍTULO 5: Datos Reales

En el quinto capítulo, cuando la fase de dantos artificiales proporciona las conclusiones necesarias y a fin de contrastar los resultados de datos sintéticos, se estudiará la viabilidad del traslado de estos resultados en un caso real, como validación o refutación de las hipótesis formuladas y como un acercamiento a la resolución final del problema descrito que forma la base de este trabajo. Así, se diseñará una aplicación de análisis de datos reales, explicando la colaboración con la empresa y los programas de lectura utilizados. Finalmente, se tratará la adaptación del ejemplo de LargeHeading con datos reales y se explicarán las otras opciones de implementación de estimadores.

Diseño e implementación: Aplicaciones de análisis de datos reales

Una vez se han llevado a cabo las pruebas con datos sintéticos, es deseable trasladar los conocimientos adquiridos al ámbito real, para acercarnos un poco más a la resolución del problema raíz del que surge el desarrollo de este trabajo.

Colaboración con la empresa

Es en el apartado de datos reales donde realmente cobra importancia la relación con empresa, además de por proporcionar el planteamiento del trabajo como parte de la solución a un problema y de facilitar todo el material y recursos, tanto físicos como humanos, necesarios para la realización de este trabajo.

Para el testeo de los diferentes algoritmos en entornos lo más cercanos a los escenarios reales posible, la empresa Aerolaser Systems ha cedido diferentes conjuntos de datos a partir de los cuales se han intentado aplicar distintos estimadores con el propósito de evaluar, principalmente, el error cometido en los valores de la orientación del vehículo.

Estos conjuntos de datos proporcionados serían datos que se obtienen directamente de las unidades de medida inercial y del GPS, te tal forma que, para el uso de estos datos en los estimadores en los que lleven a cabo las pruebas, estos deberán procesarse para obtener las mediciones en unidades de uso general. Para esto, será preciso el desarrollo de programa de datos de lectura y transformación de datos a partir de archivos binarios de datos de instrumental específico de medición.

Para esta fase, la empresa proporcionó todo el material en forma de manuales del fabricante y ejemplos, además de la ayuda didáctica precisa para el posible desarrollo de estos programas. También, de este instrumental de medición específico, se nos proporciona una serie de manuales de fabricante, así como capturas de programas de obtención de medidas y modelos que incluyen un modelo de error del sistema con sus valores iniciales con el objetivo de integrarlo en el código para mejorar el acercamiento del problema teórico a los escenario reales y comerciales.

Con el fin de obtener la mejor estimación posible también la empresa proporciona la versión de posprocesado de los archivos en bruto de la IMU y el GPS a partir de los cuales se lleva a cabo la estimación. Estos archivos que en la empresa obtienen a partir de largos cálculos nos servirán para corroborar la bondad de la estimación realizada y nos servirán de guía para descartar rutas de resolución que no conduzcan a soluciones cercanas a las esperadas.

Con todo esto, la colaboración con la empresa ha sido vital para el desarrollo de este proyecto, pues permite un apunte final de acercamiento a la solución real.

Programas de lectura de datos

Como se ha explicado con anterioridad, para poder obtener un set de datos con los que poder usar los estimadores de forma estándar, es necesario el desarrollo previo de programas que lleven a cabo la transformación a partir datos en bruto en un formato propio del fabricante del instrumental.

Para esta parte fue necesaria una documentación exhaustiva a partir de manuales y ejemplos del formato específico de archivo binario para el GPS (extensión .jps) y para la IMU (extensión .std). Desarrollados en Visual Studio con el soporte de QT estos

proyectos serían: *CppBinaryFileReader* (lectura de archivos .std de la IMU) y *CppJSPFileReader* (lectura de archivos .jsp del GPS).

El primero de los proyectos (*CppBinaryFileReader*), además de la estructura general de un programa QT estándar de la empresa, contiene un lector de archivos que toma los datos binarios del archivo .std que nos son necesarios y los guarda en dos archivos diferentes que contendrán la misma información con diferentes niveles de precisión.

El segundo proyecto (*CppJSPFileReader*) es más complejo pues, además de ser binario, presenta una serio de códigos para determinar cuándo se toma una medición de GPS que deberán identificarse inequívocamente y determinar su tamaño de forma correcta para poder obtener la información del GPS sin errores. La realización de este lector de archivos, por tanto, conllevó una cantidad elevada de tiempo material que servirá, además de para obtener los datos reales ahora mismo, para una futura integración en tiempo real, de tal forma que se lea medida a medida y se vaya incorporando al filtro estimador cada vez.

Por la complejidad del programa, no entraremos en detalles sobre su funcionamiento, pero cabe destacar que es necesaria la transformación de las medidas de posición mediante una librería geográfica. Las capturas del código principal del lector de ambos proyectos se encuentra en el Anexo II.

Una vez desarrollados estos lectores y una vez se han ejecutado con los sets de datos en bruto proporcionados por la empresa, ya disponemos de sets de datos reales que funcionan como mediciones directas de la IMU y el GPS y sus análogos de posprocesado que nos permiten verificar los resultados obtenidos.

Adaptación del ejemplo LargeHeading

Dado que el ejemplo seleccionado para la realización de la simulación con datos sintéticos era el más adecuado para el tipo de estimación que deseábamos realizar, adaptamos este código para que, en lugar de tomar datos leídos de los binarios generados en la trayectoria sintética, lea datos de los archivos de datos generados con los lectores que se explican en el apartado previo.

Para realizar la adaptación de este código, se parte del código de ejemplo de la librería y tan solo es necesario modificar las dos funciones principales: *example_largeheading* y *sys_path_largeheading*.

MODIFICACIONES DEL ARCHIVO SYS_PATH_LARGEHEADING

El archivo de definición del estado del sistema debe modificarse pues el estado inicial del sistema es diferente. Para comenzar, el tiempo de muestreo cambia de 1/100 a 1/500, que es el tiempo de muestreo del GPS.

Los parámetros de los modelos de errores permanecen intactos. La empresa proporcionó datos obtenidos mediante software del fabricante sobre el modelo de error de la IMU, como se puede ver en la Ilustración 35.

Sin embargo, tras investigar con diferentes métodos de obtención de un modelo de error de estados a partir de un modelo de densidad espectral, se llegó a la conclusión de que esta no era una tarea trivial. Se encontró documentación académica de métodos de conversión del modelo de error pero debido a que esta última parte del trabajo era una fase final de acercamiento al problema real, la planificación temporal no permitía esa línea de investigación. Adicionalmente, se carece de la base física y matemática necesaria para aplicar ese tipo de transformaciones, por lo que esa línea se deja abierta como trabajo futuro de documentación, investigación y posible implementación.

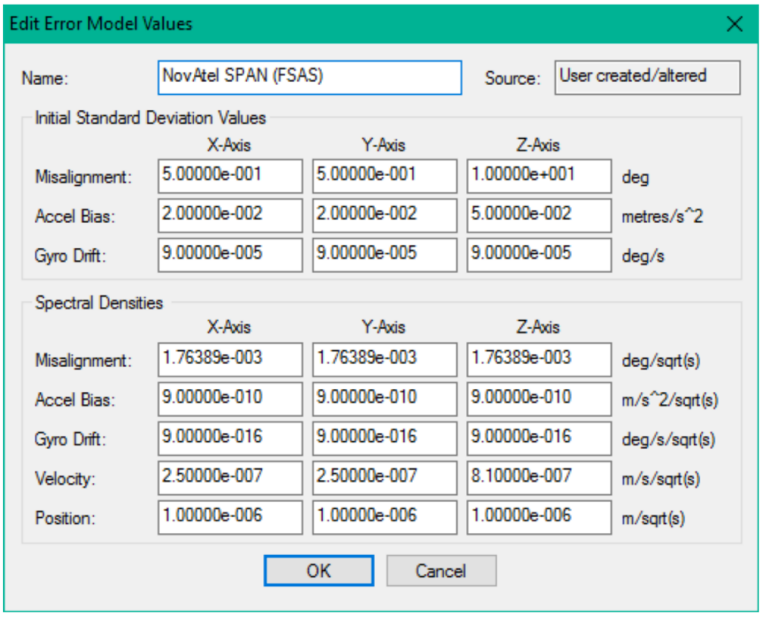


Ilustración 35: Modelo de error de la IMU real

Se dejan, entonces, los parámetros de error predefinidos por la librería y que, tras experimentación, sabemos que no se ajustan de forma satisfactoria a todas las trayectorias. Esto se lleva a cabo puesto que esta última fase, como ya se reflejó con anterioridad, es un acercamiento a la implementación real y tiene carácter didáctico y de experimentación.

Para lo que sí usamos los datos proporcionados por el software del fabricante es para la inicialización de la definición de errores de las variables de navegación, como se ve en la Ilustación 36, con los datos que se ofrecen como valores iniciales de la desviación estándar.

```
35
36      %%initial err defs
37 -    IniErrDef.pos_sP=diag([5e-1*D2R,5e-1*D2R,1e1]); %[DATOS FABRICANTE]
38 -    IniErrDef.vel_sP=diag([2e-2,2e-2,5e-2]);        %[DATOS FABRICANTE]
39 -    IniErrDef.att_sP=diag([9e-5,9e-5,9e-5]*D2R);    %[DATOS FABRICANTE]
40
```

Ilustración 36: Datos reales - Inicialización de variables del INS

Además, en los valores iniciales de navegación no se utilizarán valores o coordenadas predefinidas por la librería, sino que se utilizarán los valores de la primera medida obtenida de la lectura de datos, como se muestra en la Ilustración 37.

```
42
43      %3D simulation in NED
44      %initial nav values
45 -    vel_b=var_ini(:,1);                                    %[PRIMERA MEDIDA]
46 -    att_n=var_ini(:,2)*D2R;                                %[PRIMERA MEDIDA]
47 -    pos_n=[var_ini(1,3)*D2R,var_ini(2,3)*D2R,var_ini(3,3)]'; %[PRIMERA MEDIDA]
48
```

Ilustración 37: Datos reales - Valores iniciales de navegación

Además, se elimina toda la sección de definición de la trayectoria y de generación de la misma, pues esos datos ahora se proporcionarán por archivos de datos reales ya existentes.

Otro aspecto a modificar es la inicialización de los errores de las variables de navegación que, en este caso, no precisan de un factor aleatorio pues, por definición de los parámetros del fabricante, son esos valores y las medidas ya contienen el error aleatorio generado por la propia medición y el ruido blanco de forma natural. Con estos errores, en el vector de variables iniciales de navegación, en lugar de sumarle el error para añadírselo, se lo restamos y, así, obtenemos una primera medida más limpia y más cercana al valor a estimar.

```
54
55      %initialization errors
56 -    pos_err=diag(IniErrDef.pos_sP); %[SIN RANDOM]
57 -    vel_err=diag(IniErrDef.vel_sP); %[SIN RANDOM]
58 -    att_err=diag(IniErrDef.att_sP); %[SIN RANDOM]
59
60 -    Cerr=euler2dcm_v000(att_err);
61 -    Cbg=euler2dcm_v000(att_n);
62 -    Cbg_err=Cerr'*Cbg;
63 -    att_ini=dcm2euler_v000(Cbg_err);
64
65      %INITIAL PVA
66 -    ini_pva=[pos_n-pos_err vel_b-vel_err att_ini-att_err];  %%Attitude will be initialized using acc data
67
```

Ilustración 38: Datos reales - Vector de medidas de navegación iniciales

MODIFICACIONES DEL ARCHIVO EXAMPLE_LARGEHEADING

En este archivo también se realizan modificaciones importantes. En primer lugar, la inicialización del sistema con la función *sys_path_largeheading* se realiza después de la apertura de archivos y después de la lectura de los primero valores.

De esta forma, se leen los archivos que contienen los datos reales, en lugar de leer los binarios de la trayectoria sintética y se crea un archivo de salida para almacenar los resultados, como puede verse en la Ilustración 39. Así, como se explicó previamente, se obtiene primero la lectura de los primeros valores y estos se usan para inicializar el sistema. Además, se cambia el tiempo de propagación para acomodarlo al nuevo tiempo de muestreo.

```
5
6       %Archivos de Entrada/Salida
7 -     F_IMU = fopen([PATH 'datosIMU.txt'],'rb');       %Entrada: Datos de la IMU
8 -     F_GPS=fopen([PATH 'datosGPS.txt'],'rb');         %Entrada: Datos del GPS
9 -     F_OUT = fopen([PATH 'outputKalman.txt'],'wt');  %Salida: Resultado de aplicar Kalman
10
11      %Leemos el primer registro de entrada de la IMU y el GPS
12      %Se usará para la estimación inicial
13 -    imu_data=fscanf(F_IMU, '%f', 7);
14 -    gps_data=fscanf(F_GPS, '%f', 4);
15 -    vel_ini = imu_data(2:4);
16 -    att_ini = imu_data(5:7);
17 -    pos_ini = gps_data(2:4);
18 -    var_ini = horzcat(vel_ini,att_ini,pos_ini);
19
20      %Especificamos el modelo de error de la IMU
21      %Aclaración --> Wander representa el heading desconocido
22 -    [SenErrDef, IniErrDef, ObsErrDef, ini_pv, wander, dt]=sys_path_largeheading_EDITED(PATH, 1,var_ini);
23
```

Ilustración 39: Datos reales - Apertura de archivos, lectura e inicialización del sistema

De forma general, todas las zonas de lectura y escritura se adecuarán al nuevo método de lectura. Sin embargo, se añade una sincronización de las medidas del GPS con la IMU, pues no comienzan a la vez y esto es necesario para poder llevar a cabo el proceso del filtro Kalman.

```
 99      %Antes de leer, vamos a sincronizar las medidas
100 -    while(gps_data(1) < imu_data(1))
101 -        gps_data=fscanf(F_GPS, '%f', 4);
102 -    end
103
```

Ilustración 40: Datos reales - Sincronización de GPS e IMU

El grueso del proceso del filtro Kalman permanece constante, con la diferencia de que, a la hora de guardar los resultados, estos se hacen de una forma diferentes y más simple, eliminando las referencias y restas de la trayectoria sin error, como se ve en la Ilustración 41.

```
232          %-------------------------------------------------------------------
233 -        Cgn=[wander(2) wander(1) 0;-wander(1) wander(2) 0;0 0 1];
234 -        eul = dcm2euler_v000(Cgn'*Cbn);
235 -        wand = atan2(wander(1),wander(2));
236
237 -        if (mode==0)
238 -            mx_a=[Cgn' [0 0;0 0;-wander(2) wander(1)]];
239 -        elseif (mode==1)
240 -            mx_a=[Cgn' -[0;0;1]];
241 -        elseif (mode==2)
242 -            mx_a=Cgn';
243 -        end
244 -        mx_b=P(7:nst_nav,7:nst_nav);
245 -        mx_c=mx_a*mx_b*mx_a';
246
247 -        fprintf(F_OUT, '%8.3f ', [time_imu;pos_g;eul;wand]);
248 -        fprintf(F_OUT, '\n');
249          %-------------------------------------------------------------------
```

Ilustración 41: Datos reales - Almacenamiento de datos Kalman

Finalmente, a la hora de mostrar las gráficas, tras sincronizar las medidas de la predicción con las medidas de la solución de posprocesado, se llevan a cabo las gráficas comparativas de la trayectoria y del *heading*.

Cuando obtenemos ejecutamos el código con un set cualquiera de datos reales, obtenemos lo siguiente como gráfica comparativa de estimación de *heading* y *heading* real:

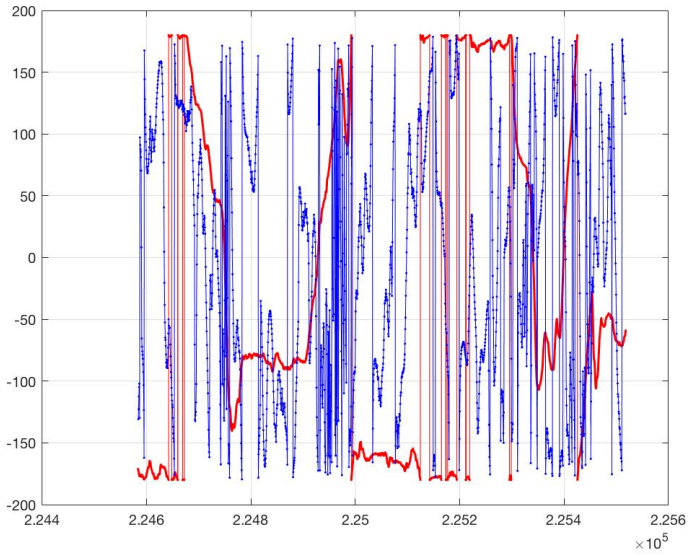


Ilustración 42: Datos reales - Gráfica comparativa del heading y su estimación

Además, el funcionamiento del programa se ve comprometido por la aparición de valores NaN a lo largo de diversas matrices de transformación y variables del sistema de navegación. Tras investigar la causa de estos valores se llega a la conclusión de que el programa no está preparado para una estimación del *heading* tan poco precisa. De forma general, es de esperar que un ángulo de cabeceo se mueva dentro de unos rangos y, al obtenerse una estimación tan irregular y oscilatoria, las matrices de transformación no son capaces de soportar el cambio de ángulo. Normalmente, para evitar estos fallos del sistema, suele hacerse uso de los cuaterniones pero, en este caso, fruto del ajuste para una trayectoria que ofrece resultados buenos y de una confianza en los rangos del ángulo, no se usan.

Esto compromete el sistema e impide la generación correcta de la trayectoria. De esta forma, por un lado no presenta una buena solución por el gran error que presenta en la estimación del *heading* y, por otro, no presenta un buen sistema que implementar en un caso real. La robustez del programa se ve en entredicho si un *heading* muy variable es suficiente para que colapse.

Otras aproximaciones para la estimación

Como la adaptación del ejemplo de la librería *LargeHeading* no ha aportado resultados aceptables para una aproximación al *heading*, se prueban otras alternativas de forma práctica y concisa con el propósito de evaluar, principalmente, el error cometido en los valores de la orientación del vehículo.

Las pruebas realizadas incluyen:

- Integrador en bucle abierto
- Filtro de Kalman Extendido
- Fusión sensorial

Estos son otro tipo de estimadores con los que nos encontramos en el proceso de investigación de alternativas de resolución del problema. Todas ellas presentan ventajas y desventajas para este tipo de implementación pero, al tratarlas desde un enfoque meramente práctico y exploratorio, nos basaremos en los resultados obtenidos para calificar la bondad de la estimación.

INTEGRACIÓN EN BUCLE ABIERTO

Se ha partido de un código disponible en el repositorio de Mathworks para integrar las medidas inerciales de la IMU. El objetivo era intentar obtener un valor de base para el error que se pudiera emplear como referencia.

Los resultados mostraron un sorprendente buen comportamiento en intervalos cortos medio de tiempo (decenas de minutos). Lógicamente, a partir de un cierto momento la deriva del sensor y la acumulación de los errores hace que las estimaciones diverjan.

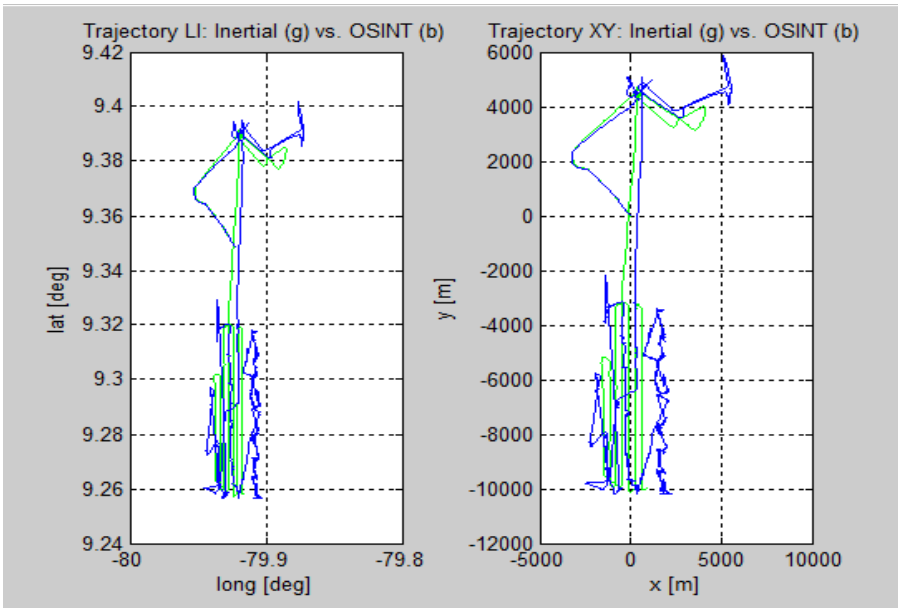


Ilustración 43: Integración en bucle abierto

Esta figura muestra un ejemplo de datos reales de vuelo en la que se puede apreciar como la trayectoria real (verde) y la estimada (azul) sólo coinciden en la parte inicial del recorrido, representada por el punto x=0, y=0.

FILTRO DE KALMAN EXTENDIDO

Se ha partido de un código de libre acceso, creado por Tim Bailey (University of Sidney). El objetivo era tratar de cerrar el lazo de control a partir de un esquema de filtro de Kalman extendido.

Sin embargo, los resultados obtenidos no fueron satisfactorios debido a que el código está preparado especialmente para robótica móvil terrestre y su adaptación al caso del helicóptero requería un esfuerzo notable.

FUSIÓN SENSORIAL

En este último caso se ha partido de un código de libre acceso disponible en el *Open Source Aided Inertial Navigation Project*, la librería utilizada para el ejemplo de *LargeHeading* utilizado en ambos enfoques del trabajo: con datos sintéticos y con datos reales.

Siguiendo la línea de la librería, se ha llevado a cabo una exploración comparativa con el otro ejemplo de los existentes que se ajustaba adecuadamente a este trabajo, *Tightly Coupled,* y el ejemplo usado con anterioridad, *LargeHeading*.

En el caso del *Tightly Coupled*, se parte de un esquema acoplado de fusión sensorial con filtro de Kalman. Las medidas de la IMU son integradas en bucle abierto mediante ciclos de predicción hasta que se recibe una medida de posicionamiento GPS, momento en el que se estima el error y se actualiza el estado.

Este ejemplo de integración fuertemente acoplada ya había sido descartado en la fase teórica y tras realizar pruebas, puesto que son necesarias medidas de pseudorango. Sin embargo, teniendo en cuenta que el ejemplo anterior no nos da resultados prometedores, rescatamos este método, pues se podrían buscar otros enfoques para suplir la opción del pseudorango o encontrar un forma de obtenerlos a partir de otras medidas, si fuera necesario.

Los resultados mostraron un comportamiento razonable sobre intervalos prolongados de tiempo. Las figuras muestran los errores que se obtienen en la estimación de los ángulos de *pitch*, *roll* y *heading* para un recorrido real de aproximadamente una hora de duración.

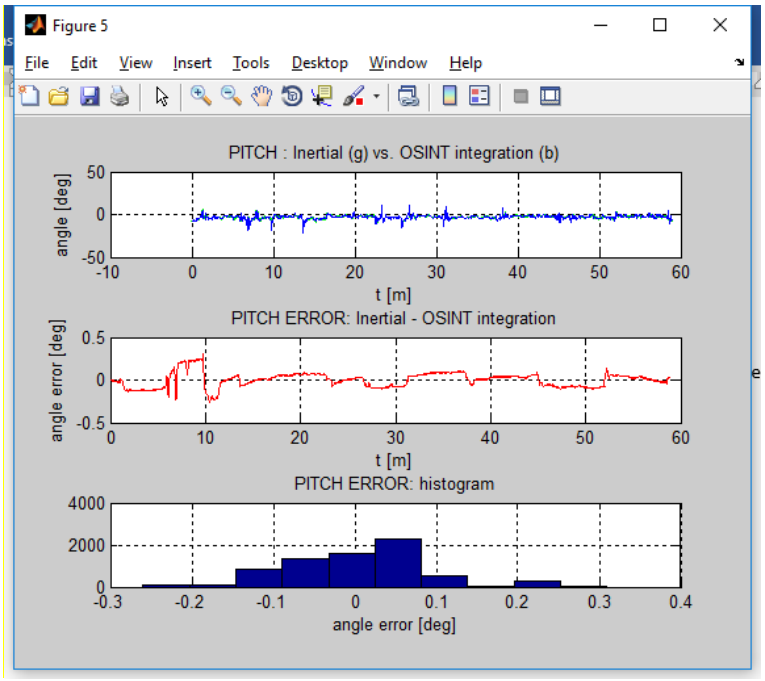


Ilustración 44: Tightly Coupled - Estimación del pitch

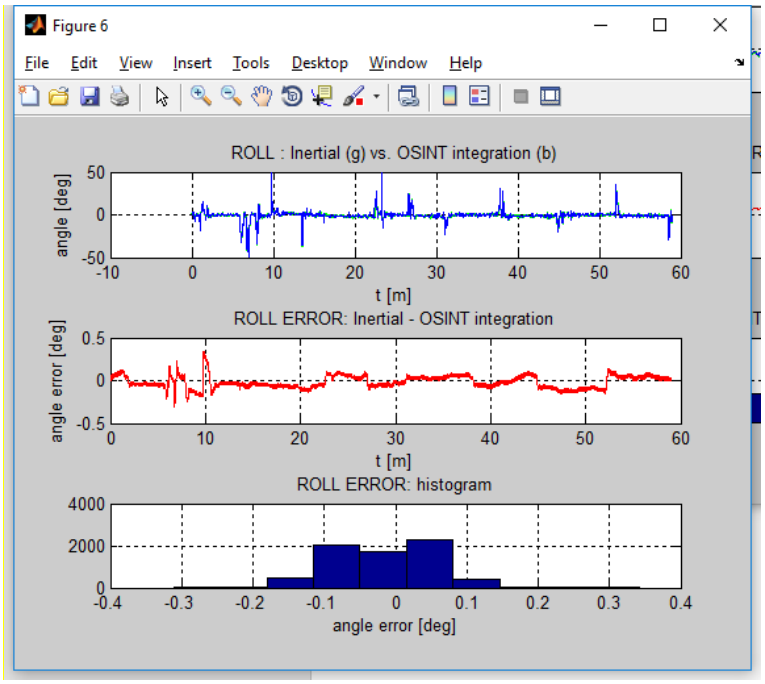


Ilustración 45: Tightly Coupled - Estimación del roll

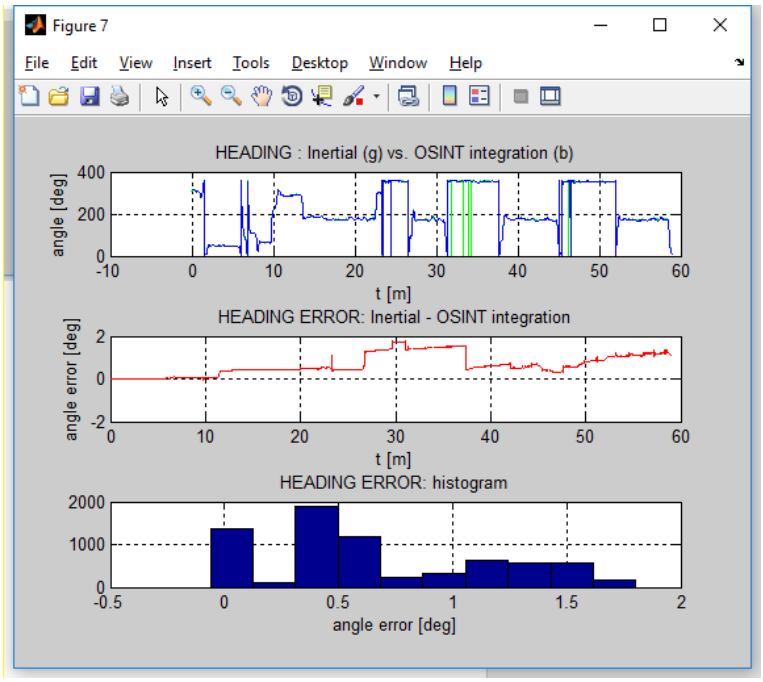


Ilustración 46: Tightly Coupled - Estimación del heading

El problema está en que el sistema necesita una estimación precisa de los valores iniciales para poder operar adecuadamente.

El ejemplo *LargeHeading* está pensado para ilustrar la capacidad que tiene el estimador para recuperarse de un error significativo en la inicialización, tal y como se ha demostrado en las pruebas realizadas sobre trayectorias sintéticas. Con datos reales, sin embargo, ese comportamiento no se produce generándose un comportamiento inestable en la estimación.

CAPÍTULO 6: Conclusiones y Trabajo Futuro

Conclusiones

De acuerdo con la aproximación inicial del problema, este trabajo se plantea como un estudio previo de los parámetros que influyen en la estimación del *heading* en un vehículo aéreo con fusión sensorial. Esto iba a llevarse a cabo para que el equipo de la empresa que plantea la necesidad de resolución de este problema dispusiese de una base de la que partir a la hora de diseñar la implementación práctica del algoritmo de fusión de sistemas inerciales.

Inicialmente, partíamos con la creencia y la hipótesis de que los parámetros de definición del modelo de error de la unidad de medida inercial serían determinantes, independientemente de la trayectoria elegida, para la estimación del *heading*, y que, el estudio estaría basado, en gran medida, en el estudio de la importancia de cada uno de ellos comparativamente.

Sin embargo, durante la realización del estudio de investigación, no solo se ven refutadas nuestras hipótesis, sino que nos encontramos con un sistema mucho más complejo de lo esperado, intrínsecamente relacionado con su estructura interna y de definición, de tal forma que cualquier perturbación o error en el mismo puede resultar en un deterioro grave de la estimación.

Uno de los motivos por los que se consigue un contraste tan drástico en la estimación del *heading* entre los datos sintéticos y los datos reales es el modelo de error. Las hipótesis iniciales no estaban desencaminadas, pero presentan el matiz de que no es tan importante la variación de un parámetro dentro de sus rangos de confianza. Lo determinante es la elección correcta del modelo de error.

En las trayectorias sintéticas, el modelo de error utilizado para generar la trayectoria era el mismo modelo utilizado para estimarla. Por tanto, a pesar del cambio en los

parámetros, la integridad estructural del programa permanece intacta y la estimación, dentro de unos límites de tolerancia, es aceptable y, en algunos casos, muy precisa.

Sin embargo, en los casos reales, el modelo de error que presentaban las medidas (basado en densidades espectrales) era completamente diferente al modelado en el software de la librería, por lo que, por muy precisa que fuese la estimación, no se acerca a la estructura del sistema y, por tanto, los resultados obtenidos están alejados de la realidad.

De forma general y a nivel personal, el trabajo ha sido una experiencia agotadora a la vez que gratificante. Durante el desarrollo de la vida académica como estudiante, las oportunidades de llevar un trabajo de investigación son más bien escasas, por el carácter didáctico del aprendizaje y por las limitaciones temporales.

Así, este trabajo que se entrelaza con la intensificación de computación ha supuesto una oportunidad de experimentar, a pequeña escala, el funcionamiento de un proyecto de investigación y la riqueza académica que puede lograr en un equipo de investigación colaborativo. Por una parte, con los miembros de la empresa y mis tutores y, por otra parte, por la gran comunidad de investigadores colaborativos y dispuestos a compartir conocimiento solidariamente en las plataformas web diseñadas para tal propósito.

Trabajo Futuro

Siguiendo la línea de las conclusiones, el trabajo futuro estaría basado en la obtención de una integración que permita estimar el modelo de error de estos instrumentos de medida. En caso de no poder conseguirse, otra línea de investigación sería la documentación, investigación e implantación de algoritmos de conversión del modelo de error de densidades espectrales a un modelo de estados. Para ello, la investigación de Karvonen & Sarkka (2016) podría constituir un buen comienzo.

Bibliografía

[1] Robert M. Rogers. (2007). Applied Mathematics in Integrated Navigation Systems.: American Institute of Aeronautics and Astronautics.

[2] Mohinder S. Grewal, Lawrence R. Weill, Angus P. Andrews. (2007). Global Positioning Systems, Inertial Navigation, and Integration.: Wiley.

[3] Open source aided inertial navigation project. (2013). Recuperado de http://www.instk.org/

[4] Why is an Inertial Navigation System (INS) important for unmanned aerial vehicle (UAV) survey and mapping applications?. (2016). Recuperado de http://www.oxts.com/technical-notes/why-is-an-inertial-navigation-system-ins-important-for-unmanned-aerial-vehicle-uav-survey-and-mapping-applications/

[5] Armada. (2010). Sistemas electrónicos embebidos. Retrieved from www.armada.mde.es/

[6] Candelas, F. A., & Ramón, J. A. C. (2008). Comparativa Algoritmos Fusión.

[7] Dematties, D. J., & Iglesias, Y. F. A. (2012). Implementación de filtro de Kalman en FPGA.

[8] González, J. A., Marcos, A., & Pacheco, E. J. (2004). Comparación de la Red Neuronal y del Filtro de Kalman en la Estimación de Velocidad del Motor de Inducción.

[9] Grondona, E. (2013). Diseño y comparativa de algoritmos de navegación fuerte y débilmente integrados con GPS. Universidad de Buenos Aires.

[10] Karvonen, T., & Sarkka, S. (2016). Approximate state-space Gaussian processes via spectral transformation. IEEE International Workshop on Machine Learning for Signal Processing, MLSP, 2016–Novem. https://doi.org/10.1109/MLSP.2016.7738812

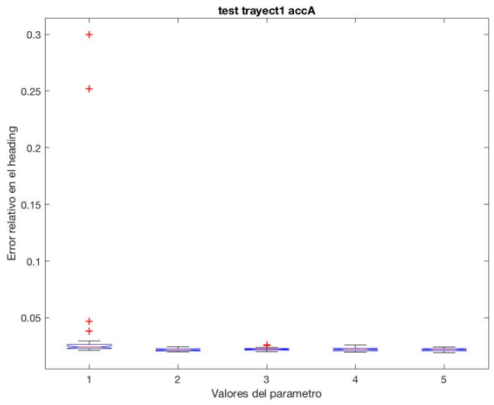
[11] Liu, M., & Chang, G. (2016). Numerically and statistically stable Kalman filter for INS/GNSS integration. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 230(2), 321–332. https://doi.org/10.1177/0954410015591614

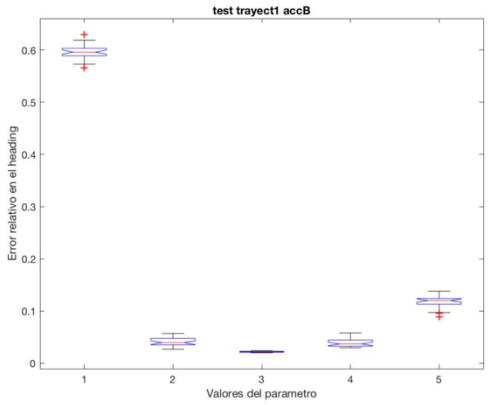
[12] Novosti, R. (2014). Dos sistemas de navegación frente a frente: GLONASS vs. GPS. Russia Beyond The Headlines.

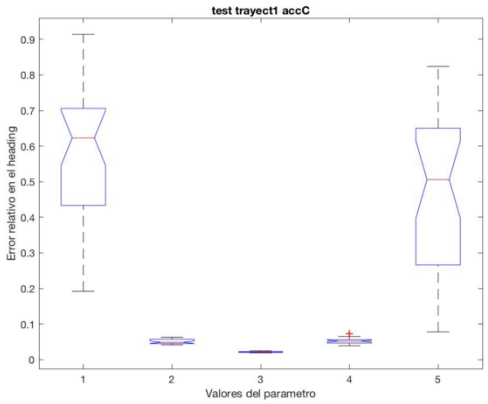
[13] Prado Obando, G. (2005). Técnicas Recursivas Para Estimación Dinamica: Una Introducción Matemática al Filtro Kalman. Fundacion Universitaria Konrad Lorenz, 14–17 (FK–1).

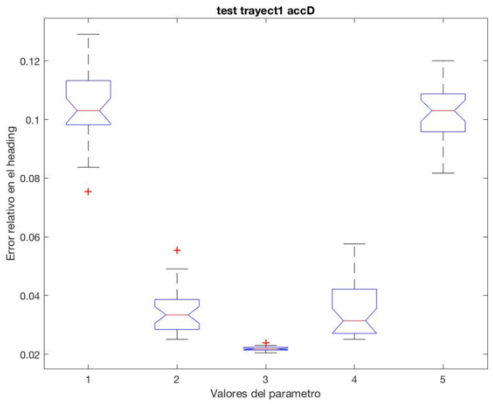
ANEXO I

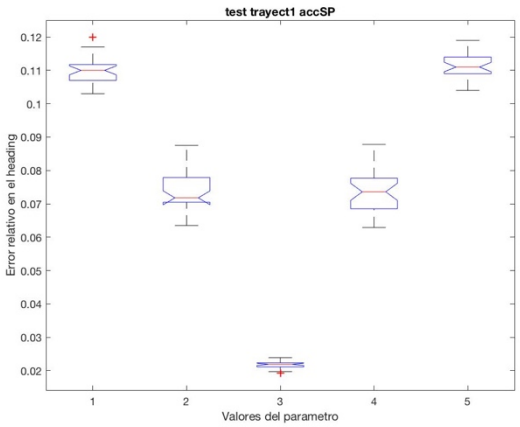
TRAYECTORIA 1: ACELERÓMETRO



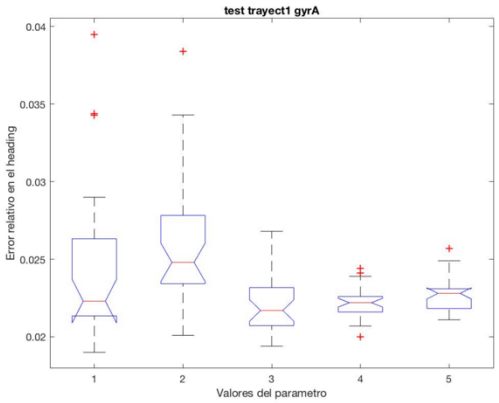


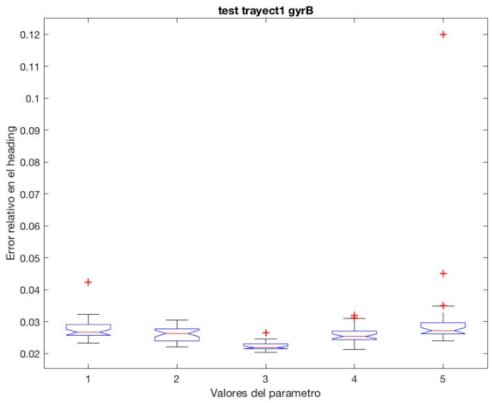


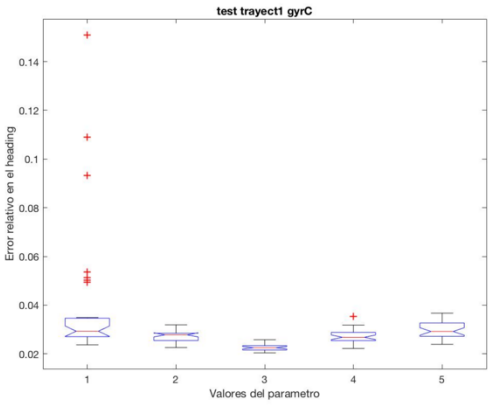


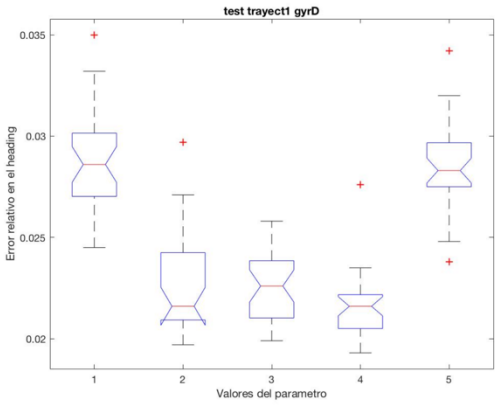


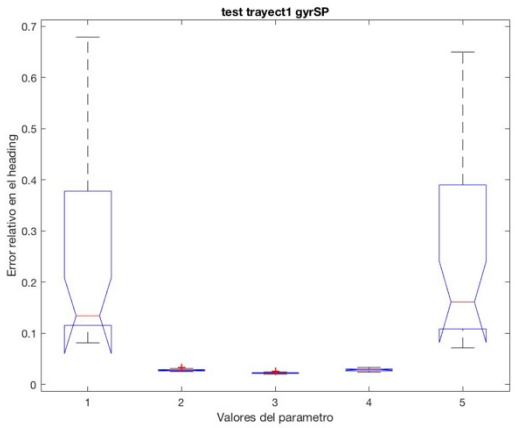
TRAYECTORIA 1: GIRÓSCOPO



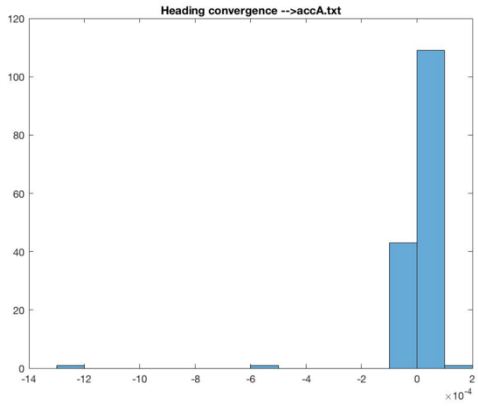


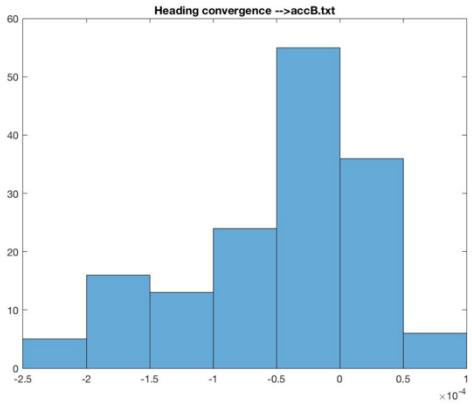


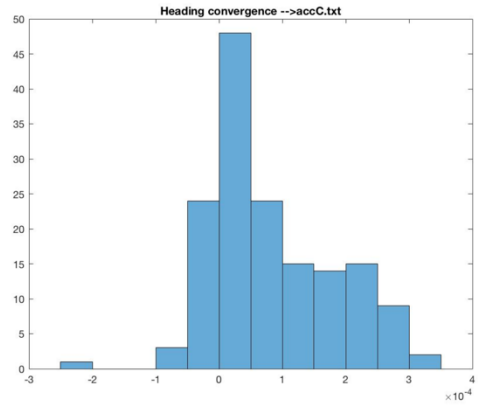


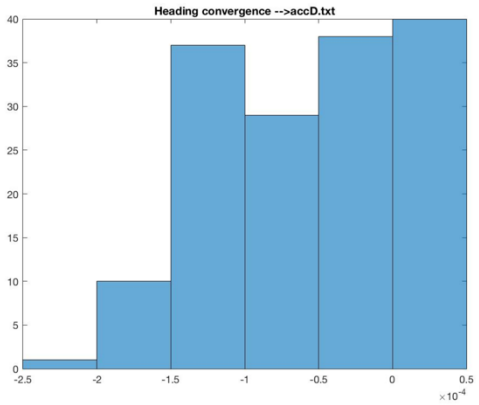


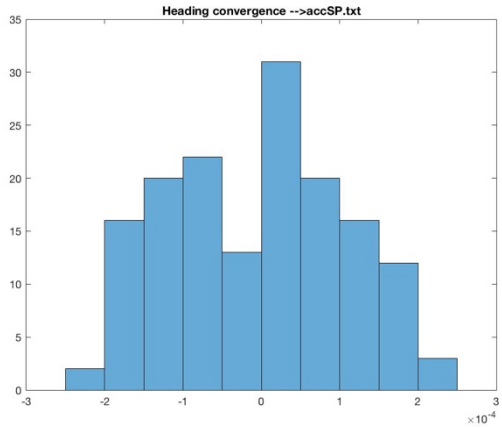
TRAYECTORIA 1: CONVERGENCIA ACELERÓMETRO



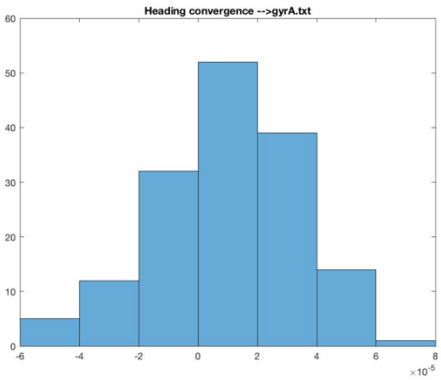


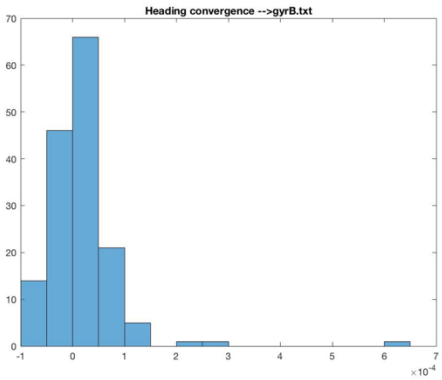


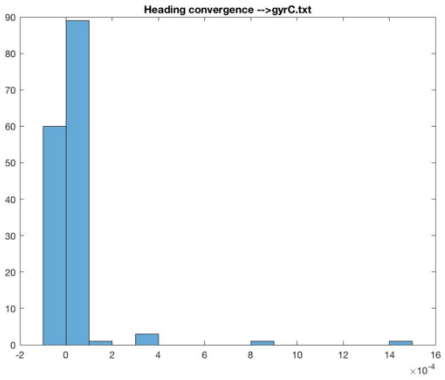


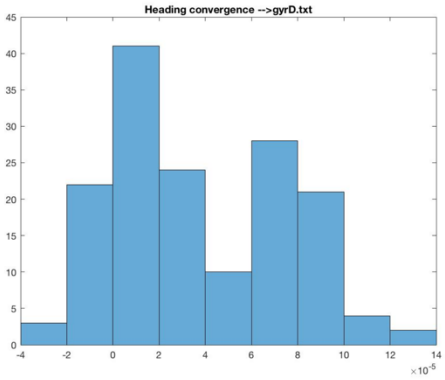


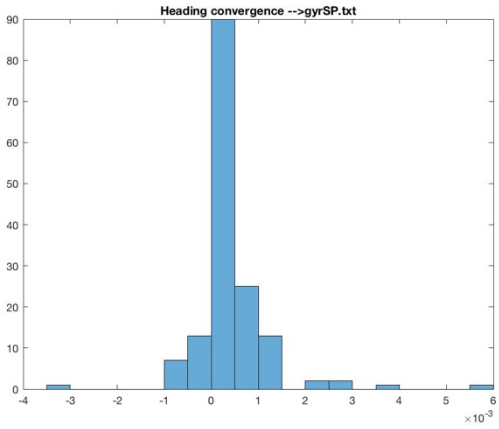
TRAYECTORIA 1: CONVERGENCIA GIRÓSCOPO



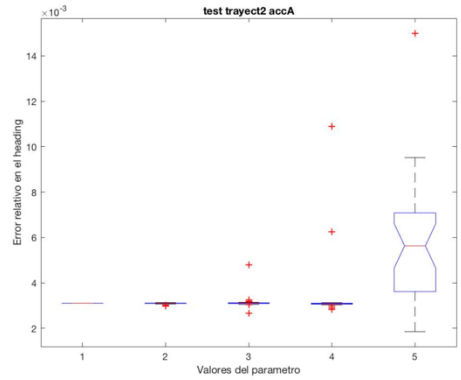


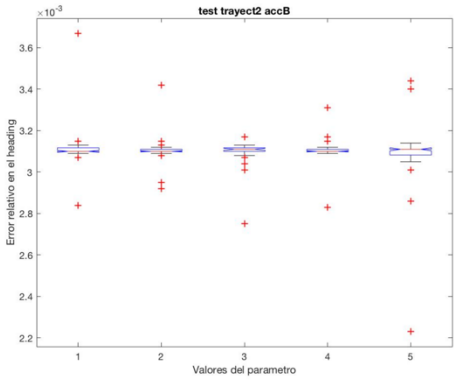


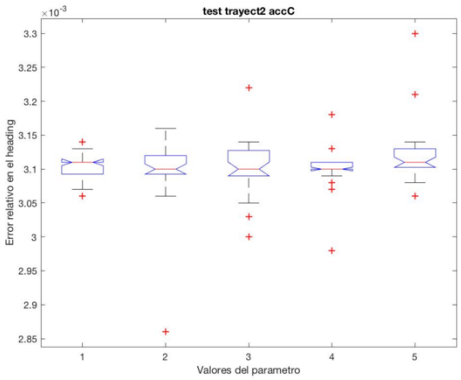


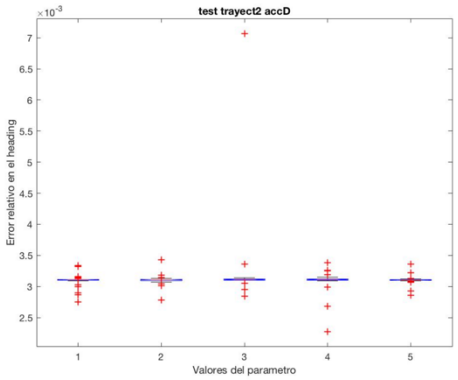


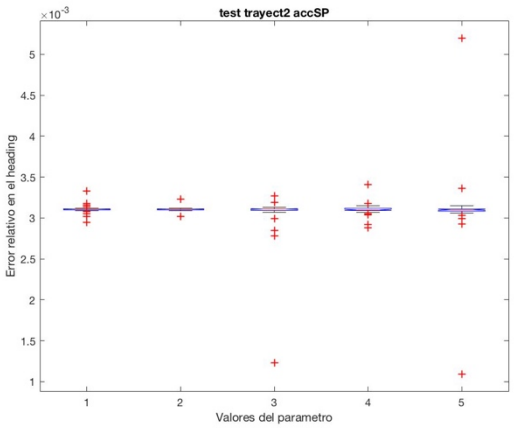
TRAYECTORIA 2: ACELERÓMETRO



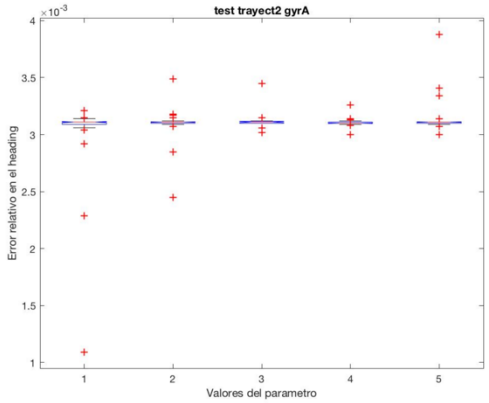


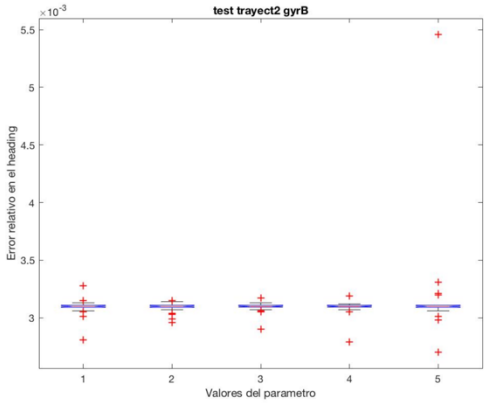


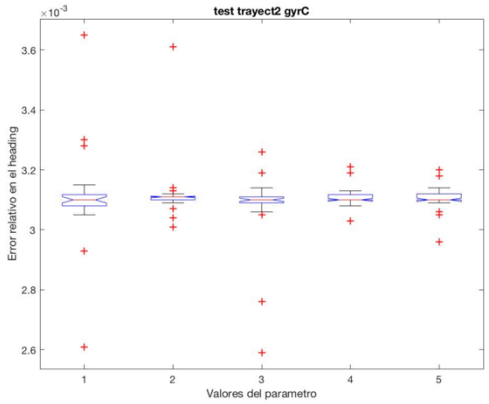


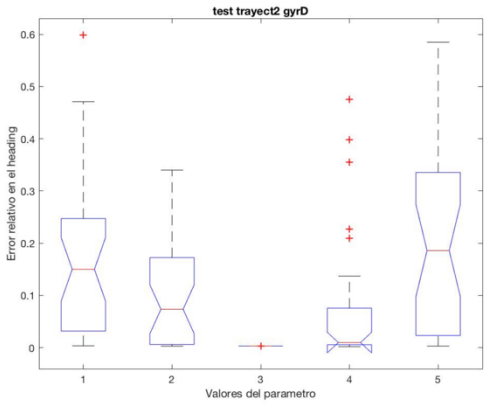


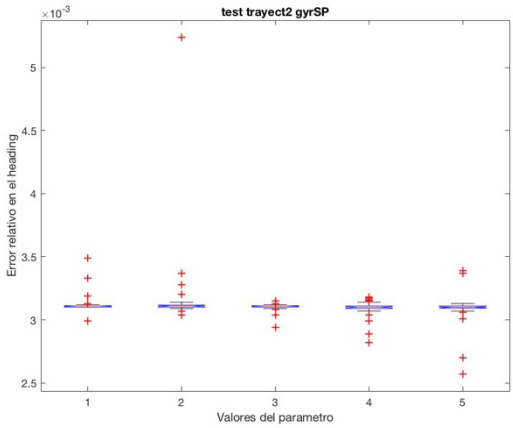
TRAYECTORIA 2: GIRÓSCOPO



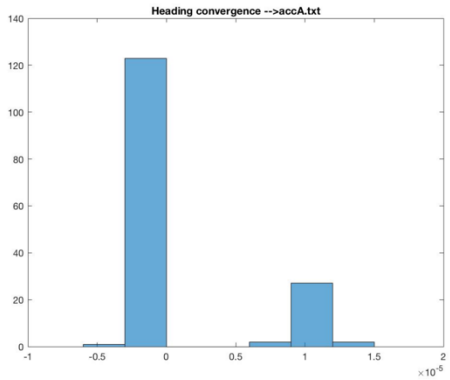


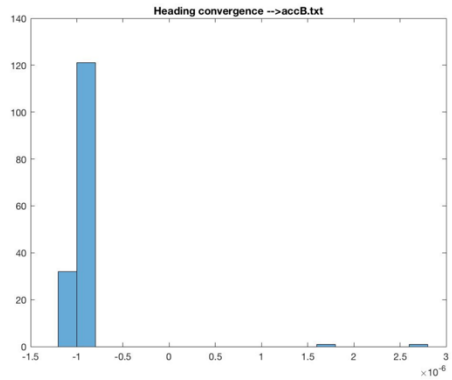


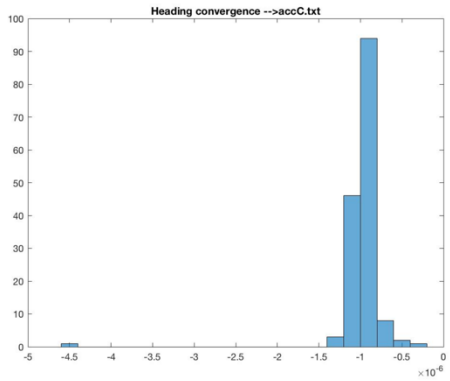


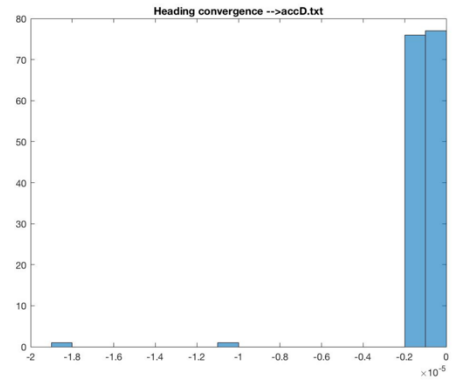


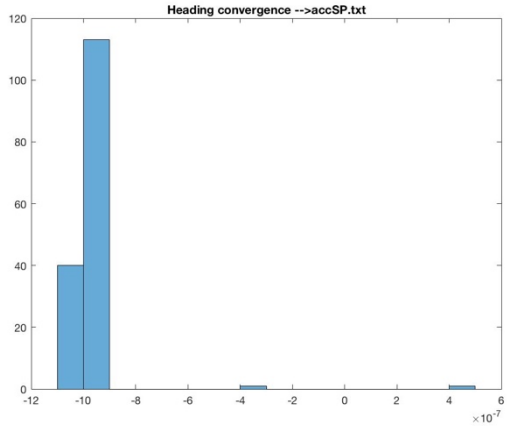
TRAYECTORIA 2: CONVERGENCIA ACELERÓMETRO



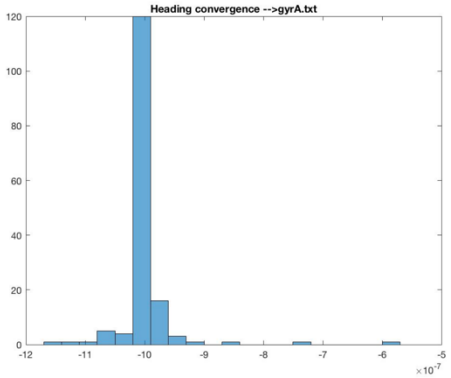


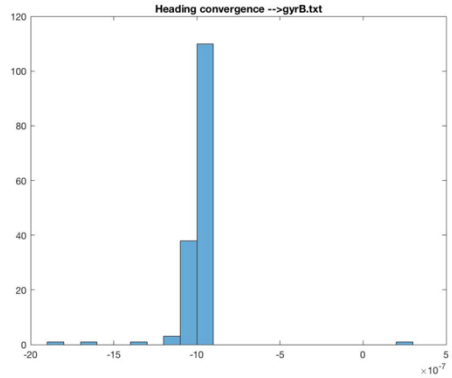


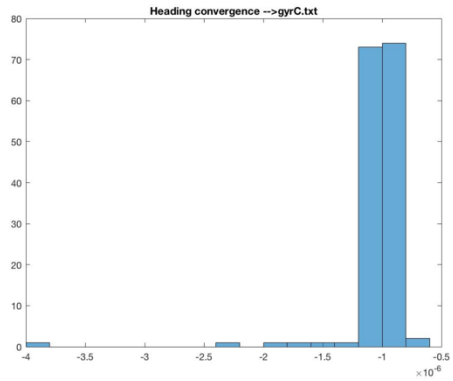


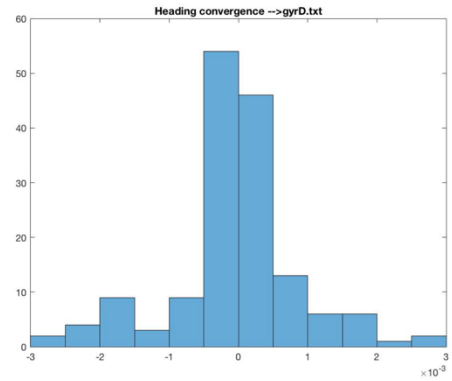


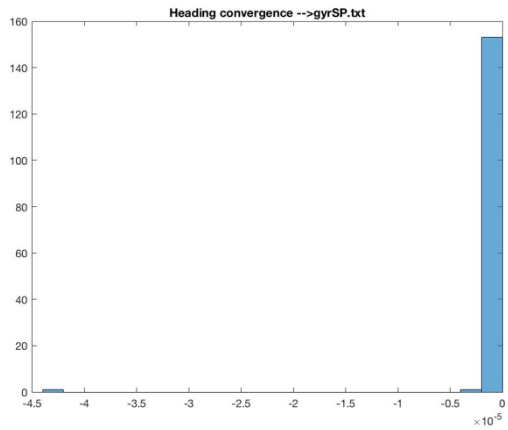
TRAYECTORIA 2: CONVERGENCIA GIRÓSCOPO



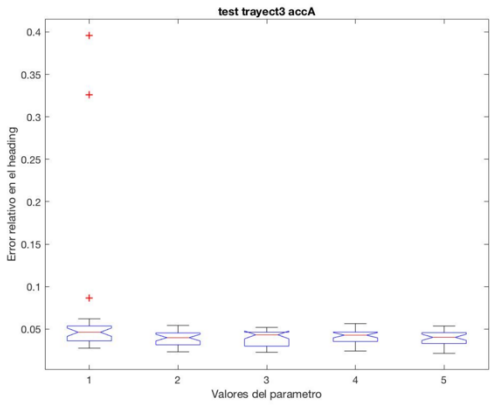


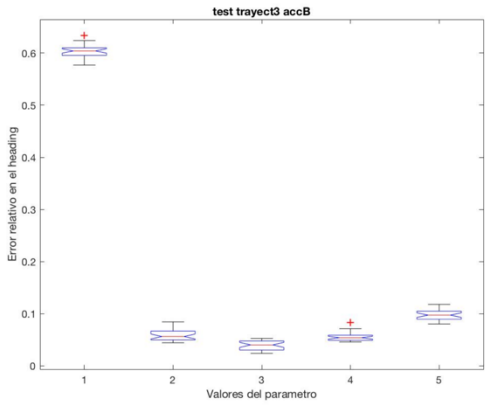


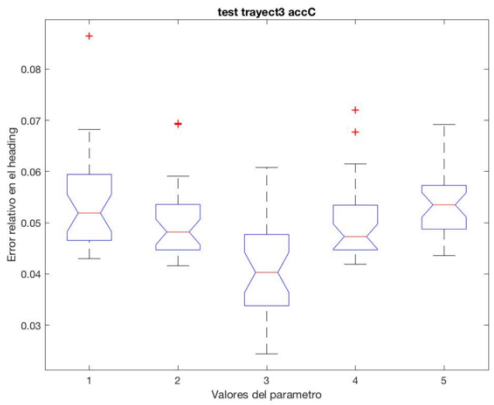


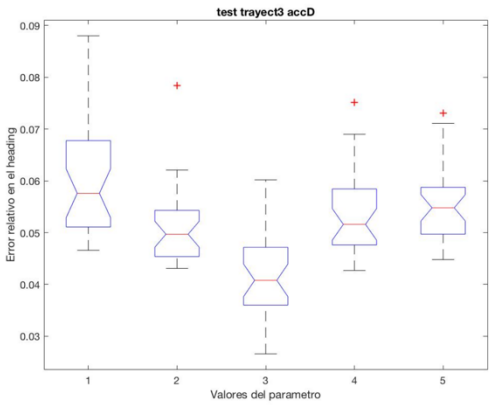


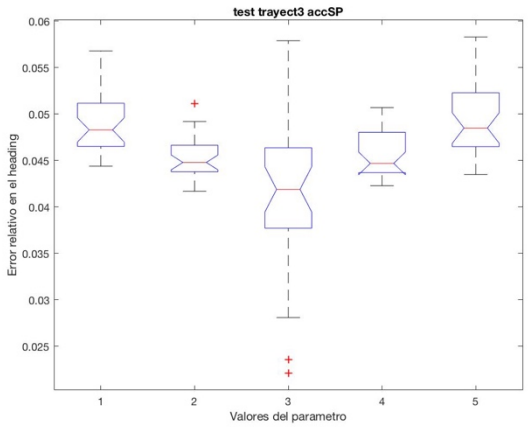
TRAYECTORIA 3: ACELERÓMETRO



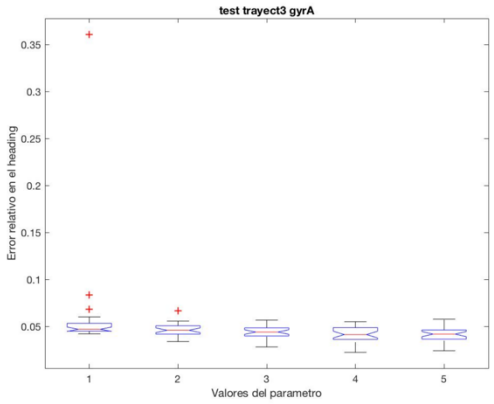


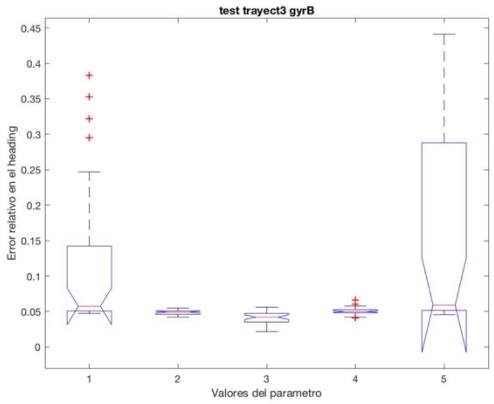


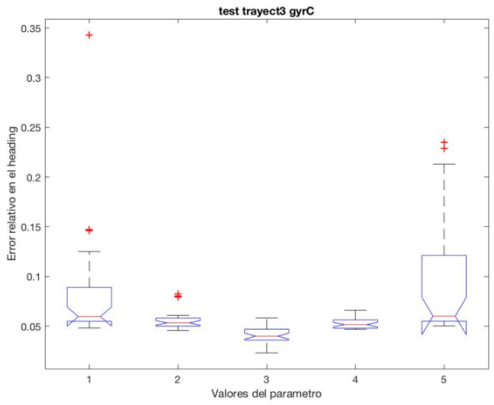


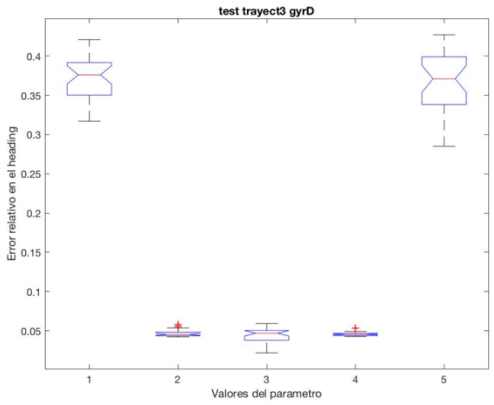


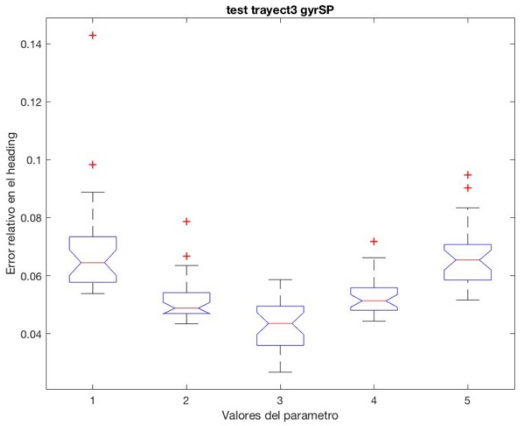
TRAYECTORIA 3: GIRÓSCOPO



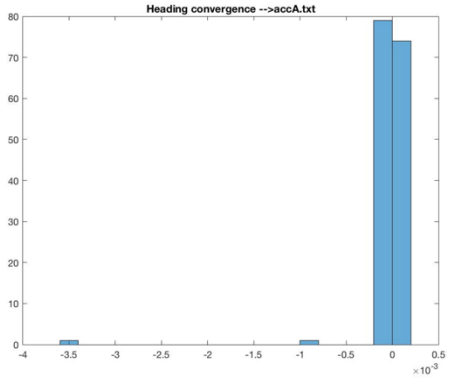


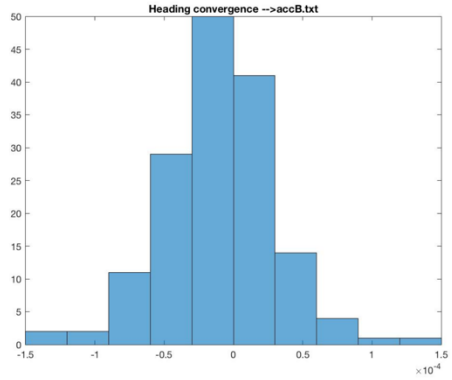


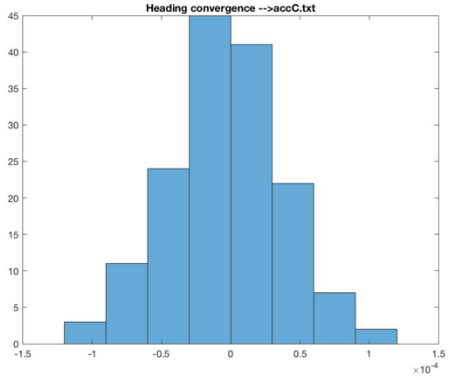


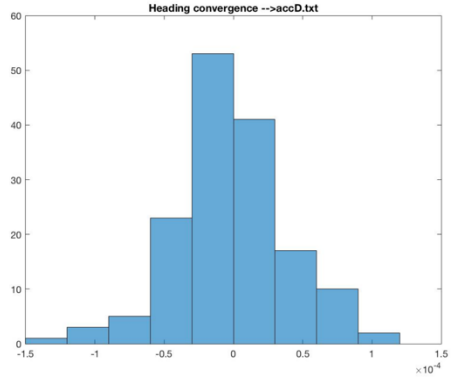


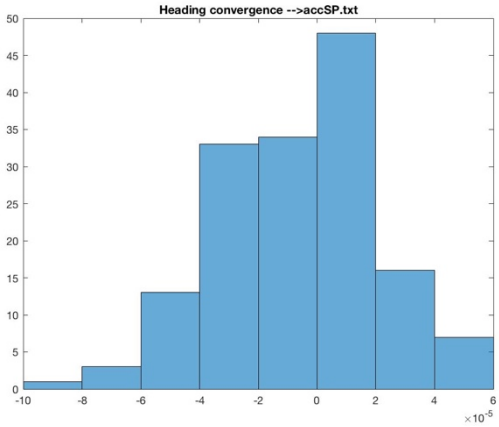
TRAYECTORIA 3: CONVERGENCIA ACELERÓMETRO



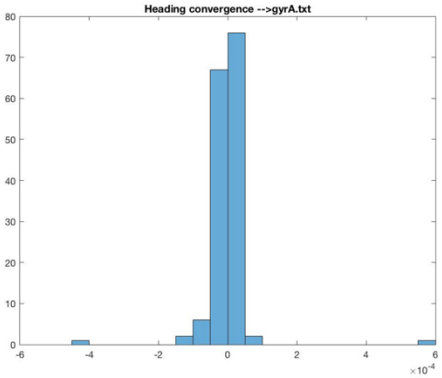


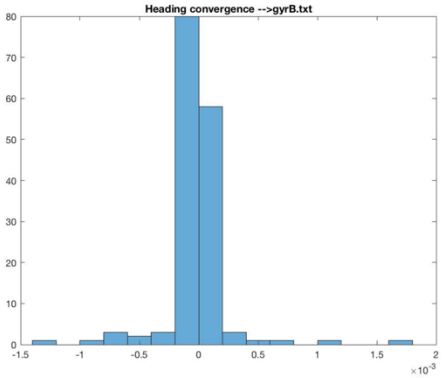


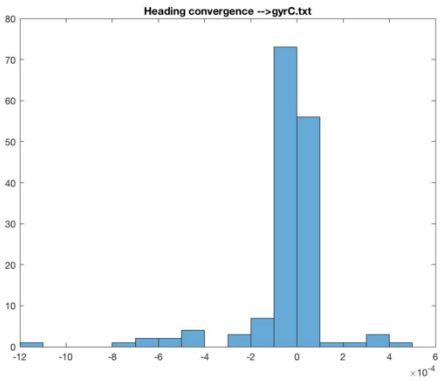


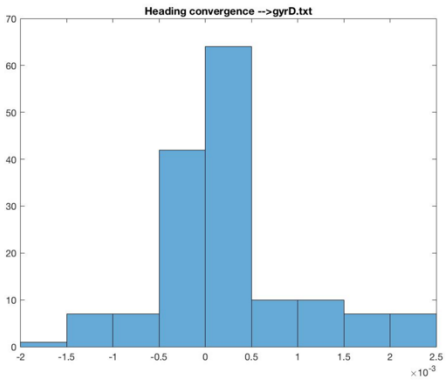


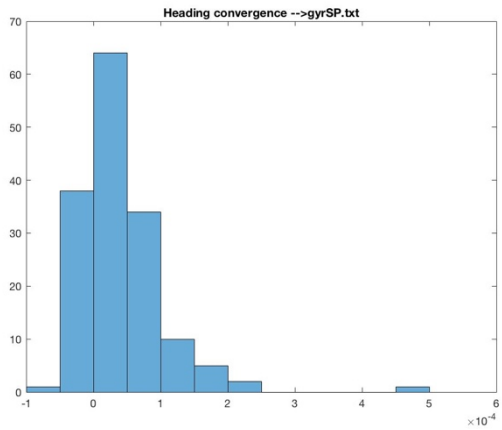
TRAYECTORIA 3: CONVERGENCIA GIRÓSCOPO





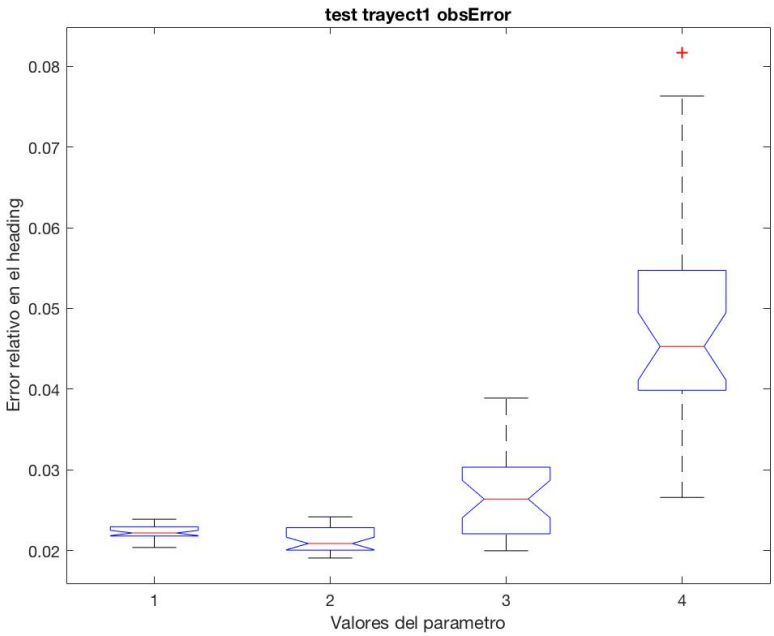




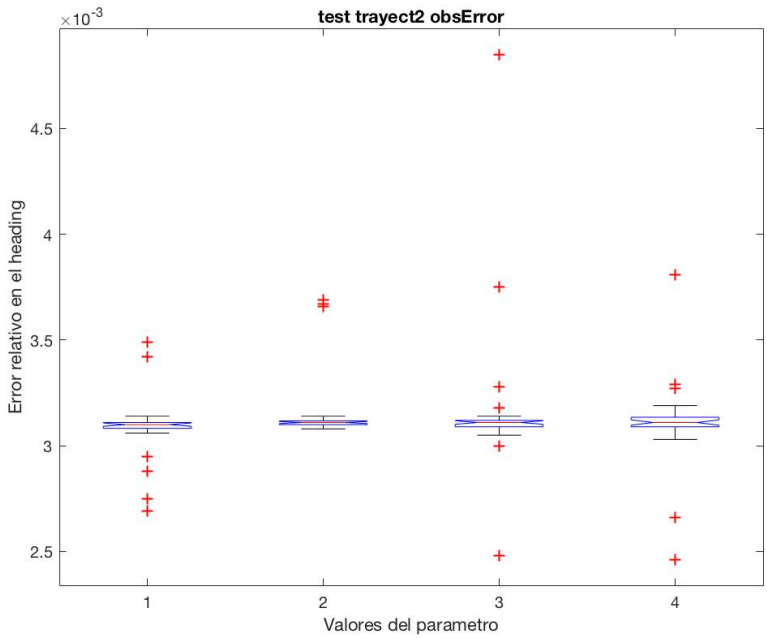


ERROR EN EL GPS (OBS_ERROR)

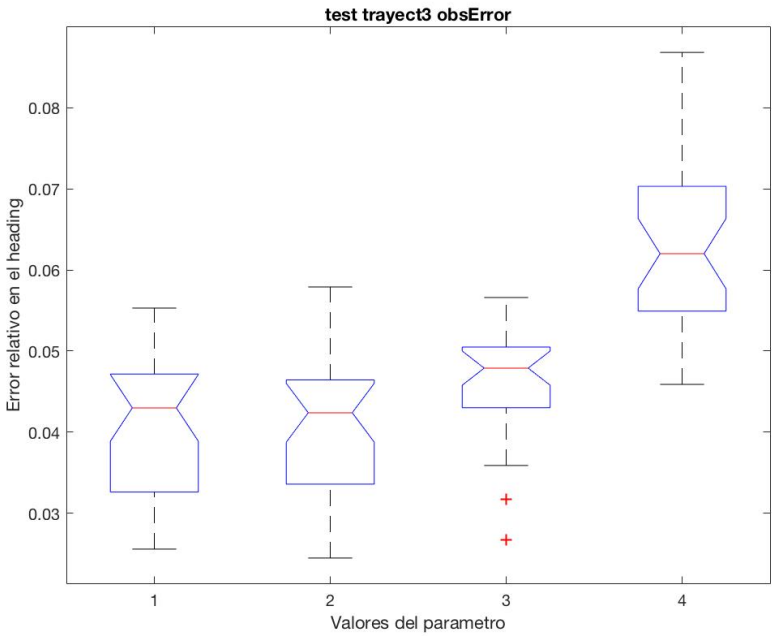
TRAYECTORIA 1



TRAYECTORIA 2



TRAYECTORIA 3



ANEXO II

CÓDIGO DEL LECTOR DE DATOS DE LA IMU

```
void Reader::run()
{
	QFile
file("//srv01/PracticasProgramacion/Datos/Kalman/SetPseudorangos/20161113_123
350.std");

	if (file.open(QIODevice::ReadOnly)) {
		QDataStream in(&file);
		in.setByteOrder(QDataStream::LittleEndian);
		while ((!in.atEnd()) && (!stopReading)) {

			Value value;
			int temp;
			in >> value.time;

			in >> temp;
			value.spinX = temp * 0.1;

			in >> temp;
			value.spinY = temp * 0.1;

			in >> temp;
			value.spinZ = temp * 0.1;

			in >> temp;
			value.speedX = ((double)temp * 0.05) / pow(2.0, 15.0);
			//value.speedX = ((double)temp * pow(2.0, 15.0))/ 0.05 ;

			in >> temp;
			value.speedY = ((double)temp * 0.05) / pow(2.0, 15.0);
			//value.speedY = ((double)temp * pow(2.0, 15.0)) / 0.05;

			in >> temp;
			value.speedZ = ((double)temp * 0.05) / pow(2.0, 15.0);
			//value.speedZ = ((double)temp * pow(2.0, 15.0)) / 0.05;

			values.append(value);
		}
	}
	writeDataFile();
}


void Reader::writeDataFile() {
```

```
	QFile
file("C:/Users/igonzalez/Desktop/DatosKalman/LargeHeading/datosIMU.txt");
	if (file.open(QIODevice::WriteOnly | QIODevice::Text))
	{
		QTextStream out(&file);

		for (int i = 0; i < values.size(); i++)
		{

	out.setRealNumberNotation(QTextStream::FixedNotation);
				out.setRealNumberPrecision(3);
				out << values[i].time << "\t";
				out.setRealNumberPrecision(8);
				out << values[i].speedX << "\t";
				out << values[i].speedY << "\t";
				out << values[i].speedZ << "\t";
				out.setRealNumberPrecision(2);
				out << values[i].spinX << "\t";
				out << values[i].spinY << "\t";
				out << values[i].spinZ << endl;
		}
	}
	file.close();

	QFile
file2("C:/Users/igonzalez/Desktop/DatosKalman/KALMAN/datosIMU.txt");
	if (file2.open(QIODevice::WriteOnly | QIODevice::Text))
	{
		QTextStream out(&file2);

		for (int i = 0; i < values.size(); i++)
		{
			out.setRealNumberNotation(QTextStream::FixedNotation);
			out.setRealNumberPrecision(3);
			out << values[i].time << "\t";
			out.setRealNumberPrecision(2);
			out << values[i].spinX << "\t";
			out << values[i].spinY << "\t";
			out << values[i].spinZ << "\t";
			out.setRealNumberPrecision(8);
			out << values[i].speedX << "\t";
			out << values[i].speedY << "\t";
			out << values[i].speedZ << endl;
		}
	}
	file2.close();

	exit();
}
```

CÓDIGO DEL LECTOR DE DATOS DEL GPS

```
void Reader::run()
{
	/*
	No funciona el checksum (el método)
	*/
	QString path =
"//srv01/PracticasProgramacion/Datos/Kalman/SetPseudorangos/20161113_123349.j
ps";
	file = new QFile(path);

	if (file->open(QIODevice::ReadOnly))
	{
		in = new QDataStream(file);
		in->setByteOrder(QDataStream::LittleEndian);

		while (!in->atEnd())
		{
			char *type1 = new char[1];
			in->readRawData(type1, 1);

			if (*type1 == 'N')
			{
				char *type2 = new char[1];
				in->readRawData(type2, 1);
				if (*type2 == 'P') {
					char *type2 = new char[1];
					int start = file->pos();
					int goBack = file->pos() + 4;

					QByteArray strSize(3, '0');
					in->readRawData(strSize.data(), 3);
					bool ok;
					int size = strSize.toInt(&ok, 16);

					int goForward = file->pos() + size - 3;
					file->seek(goForward);

					char *at = new char[1];
					in->readRawData(at, 1);
					if (*at == '@')
					{
						QByteArray strCS(2, '0');
						in->readRawData(strCS.data(), 2);
						bool ok;
						np.cs = strCS.toInt(&ok, 16);

						file->seek(goBack);
						readPackage(file, in, goForward);
					}
					delete at;
```

```
                        }
                        delete type2;
                }
                delete type1;
        }
        //qDebug() << file->pos() << endl;
        file->close();
        delete in;
    }
    delete file;
    writeDataFile();
}


void Reader::readPackage(QFile *file, QDataStream *in, int goForward)
{
    //qDebug() << "Estoy leyendo el paquete
    // Saltar hasta el marcador de válido (tamaño fijo)
    file->seek(file->pos() + 7);

    char *valid = new char[1];
    in->readRawData(valid, 1);
    file->seek(file->pos() + 1);

    if (*valid == 'V')
        readTime(file, in);
    else
        goToNextField(file,in);

    for (int i = 0; i < 5; i++) goToNextField(file,in);

    readCoordinates(file, in);

    nps.append(np);

    file->seek(goForward + 3);
    delete valid;
}

void Reader::readTime(QFile *file, QDataStream *in) {
    bool ok;
    QByteArray time(2, '0');

    int year = 2016;
    int month = 11;
    int day = 13;
    int daysOfWeek = getDayOfWeek(year,month,day);

    in->readRawData(time.data(), 2);
    np.timeHour = time.toInt(&ok, 10);

    in->readRawData(time.data(), 2);
    np.timeMin = time.toInt(&ok, 10);

    QByteArray timeSec(5, '0');
```

```
	in->readRawData(timeSec.data(), 5);
	np.timeSec = timeSec.toDouble();
	file->seek(file->pos() + 1);

	np.time = (daysOfWeek*24*60*60) + (np.timeHour * 60 * 60) +
(np.timeMin * 60) + np.timeSec;
}

void Reader::goToNextField(QFile * file, QDataStream * in)
{
	bool stop = false;
	char *next = new char[1];
	while (!stop) {
		int pos = file->pos();
		in->readRawData(next, 1);
		if (*next == ',')
			stop = true;
	}
	delete next;
}

void Reader::readCoordinates(QFile * file, QDataStream * in)
{
	char *charNS = new char[1];
	in->readRawData(charNS, 1);
	QChar ns(*charNS);
	np.ns = ns;

	bool ok;
	QByteArray latitude(2, '0');
	in->readRawData(latitude.data(), 2);
	np.latitudeD = latitude.toInt(&ok, 10);

	file->seek(file->pos() + 1);

	in->readRawData(latitude.data(), 2);
	np.latitudeM = latitude.toInt(&ok, 10);

	file->seek(file->pos() + 1);

	QByteArray latitudeSec(9, '0');
	in->readRawData(latitudeSec.data(), 9);
	np.latitudeS = latitudeSec.toDouble();

	np.latitude = np.latitudeD + ((double)np.latitudeM/60) +
(np.latitudeS/3600);
	if (np.ns == 'S') {
		np.latitude *= -1;
	}

	file->seek(file->pos() + 2);

	char *charEW = new char[1];
	in->readRawData(charEW, 1);
	QChar ew(*charEW);
```

```
	np.ew = ew;

	QByteArray longitudeDeg(3, '0');
	in->readRawData(longitudeDeg.data(), 3);
	np.longitudeD = longitudeDeg.toInt(&ok, 10);

	file->seek(file->pos() + 1);

	QByteArray longitudeMin(2, '0');
	in->readRawData(longitudeMin.data(), 2);
	np.longitudeM = longitudeMin.toInt(&ok, 10);

	file->seek(file->pos() + 1);

	QByteArray longitudeSec(9, '0');
	in->readRawData(longitudeSec.data(), 9);
	np.longitudeS = longitudeSec.toDouble();

	np.longitude = np.longitudeD + ((double)np.longitudeM / 60) +
(np.longitudeS / 3600);
	if (np.ew == 'W') {
		np.longitude *= -1;
	}

	file->seek(file->pos() + 3);

	QByteArray altitude(10, '0');
	in->readRawData(altitude.data(), 10);
	np.altitude = altitude.toDouble();

	/*EpsgConversor conversor(NULL);
	double xOut, yOut;
	conversor.transformCoordinates(np.longitude, np.latitude, xOut, yOut);
	np.latitude = xOut;
	np.longitude = yOut;*/

	delete charEW, charNS;
}

void Reader::writeDataFile() {

	QFile
file("C:/Users/igonzalez/Desktop/DatosKalman/KALMAN/datosGPS.txt");
	QFile
file2("C:/Users/igonzalez/Desktop/DatosKalman/LargeHeading/datosGPS.txt");
	if ((file.open(QIODevice::WriteOnly | QIODevice::Text)) &&
(file2.open(QIODevice::WriteOnly | QIODevice::Text)))
	{
		QTextStream out(&file);
		QTextStream out2(&file2);

		for (int i = 0; i < nps.length(); i++)
		{
			// FILE 1
			out.setRealNumberNotation(QTextStream::FixedNotation);
```

```
                    out.setRealNumberPrecision(2);
                    out << nps[i].time << "\t";
                    out.setRealNumberPrecision(8);
                    out << nps[i].longitude << "\t";
                    out.setRealNumberPrecision(4);
                    out << nps[i].latitude << "\t";
                    out.setRealNumberPrecision(4);
                    out << nps[i].altitude << endl;

                    // FILE 2
                    out2.setRealNumberNotation(QTextStream::FixedNotation);
                    out2.setRealNumberPrecision(2);
                    out2 << nps[i].time << "\t";
                    out2.setRealNumberPrecision(8);
                    out2 << nps[i].longitude << "\t";
                    out2.setRealNumberPrecision(4);
                    out2 << nps[i].latitude << "\t";
                    out2.setRealNumberPrecision(4);
                    out2 << nps[i].altitude << endl;
            }
        }
        //qDebug() << nps.length() << endl;
        file.close();
        file2.close();
}

int Reader::getDayOfWeek(int year, int month, int day) {
        tm timeStruct = {};
        timeStruct.tm_year = year - 1900;
        timeStruct.tm_mon = month - 1;
        timeStruct.tm_mday = day;
        timeStruct.tm_hour = 12;
        mktime(&timeStruct);
        return timeStruct.tm_wday;
}
```