Un método de calibración de sensores inerciales

Paula Pérez López

Tutor: Andrei Martínez Finkelshtein

Titulación: Máster en Matemáticas
Fecha de defensa: 23 de julio de 2015

Índice general

Capítulo 1

Introducción

Este trabajo surge de una colaboración con la empresa *Realtrack Systems*, creadora del dispositivo *WIMU*, con el objetivo de mejorar la precisión en los datos proporcionados por los sensores que lo constituyen.

WIMU es un dispositivo para deportistas, cuyas características veremos en la sección 2.1, que integra una unidad inercial de medida (IMU). Esta unidad de medida consiste en una combinación de diferentes tipos de sensores inerciales, generalmente y en el caso de *WIMU* en concreto, acelerómetros y giroscopios que se utilizan en navegación inercial para determinar la posición de un objeto.
Los sensores inerciales que componen estas unidades de medida inerciales están basados en tecnología microelectromecánica o MEMs. Esta tecnología se caracteriza por su pequeño tamaño y la disminución de su precio respecto al de sensores anteriores.

Esta miniaturización y la disminución en el coste de los sensores han permitido que tengan una gran aplicabilidad en muchos sectores de la ingeniería. Sin embargo, también va acompañado de una pérdida en la precisión de los datos proporcionados por los sensores y es ahí, donde surge la necesidad de realizar algoritmos de calibración que traten de conseguir mejores resultados. Los errores que aparecen en los datos, tanto en el acelerómetro como en el giroscopio, son fruto de condiciones externas e internas al propio sensor. La descripción de estos errores se detallará en la sección 2.3, pero fundamentalmente los errores que se tratarán de corregir en este trabajo son errores de ortogonalidad de los ejes de coordenadas, así como de sesgo y de cambio de escala.

La calibración de los sensores se basan en dos ideas fundamentales.

En primer lugar, para la calibración del acelerómetro nos basamos en que el módulo del vector de las mediciones que proporcione el sensor durante los periodos de tiempo en los que se encuentre estático ha de ser igual a la fuerza de la gravedad, [11]. Esta afirmación es clara, puesto que cuando un cuerpo está en reposo, la única aceleración que actúa sobre el mismo es la gravedad.
En segundo lugar, para la calibración del giroscopio se utilizará que el vector de gravedad proporcionado por el acelerómetro calibrado ha de ser el mismo que el vector de gravedad calculado al integrar las velocidades angulares proporcionadas por el giroscopio.
Estos principios los podemos encontrar en trabajos anteriores como [12],[9], [13].

En el algoritmo que se propone en este trabajo para calibrar el acelerómetro se utilizará un detector de periodos estáticos, [9]. Este detector nos permitirá conocer los periodos en los que los sensores están en reposo, de forma que podamos aplicar la idea fundamental planteada anteriormente. Así, durante estos periodos será cuando se tomen los datos necesarios para realizar la calibración.
Además, con el objetivo de adecuar la calibración de la mejor forma posible a los datos proporcionados se introduce un bloque para detectar similitudes entre distintos bloques estáticos contiguos. Esto nos va a permitir realizar una calibración conjunta en los tramos en los que los bloques estáticos contiguos sean similares y una calibración individual en aquellos en los que los errores a los que están sometidos los datos difieran más de los errores habituales.

Otra propuesta que se expone en este trabajo consiste en añadir un factor de regularización a la función objetivo que hemos de minimizar para obtener los errores de ortogonalidad, de cambio de escala y de sesgo presentes en los datos.
El motivo por el que se ha decidido introducir este factor de regularización es que se ha de afrontar un problema de mínimos cuadrados no lineales. Estos problemas se caracterizan por tener numerosos mínimos locales que no son mínimos globales, lo que supone un gran inconveniente a la hora de obtener el mínimo adecuado. Y este hecho, con los datos experimentados utilizados, es especialmente agudo.
Así, el factor de regularización permitirá que el algoritmo de minimización utilizado encuentre con más facilidad el mínimo adecuado, especialmente cuando los datos sean más homogéneos y cercanos a los datos ideales. Entendiendo por datos ideales aquellos que están libres de errores.

Estas propuestas las encontraremos en la sección 3 donde se explicará con detalle el problema a solventar y se planteará un esquema del algoritmo

implementado en *Matlab* con el que se han realizado las pruebas experimentales.
A continuación y con una estructura similar se desarrolla el problema de calibración del giroscopio y se muestra un esquema del algoritmo implementado para la calibración del mismo.
Previamente, en la primera sección se recogerán los conocimientos necesarios para el plantemiento del problema de calibración en ambos tipos de sensores, así como para la creación del algoritmo.

Capítulo 2

Conocimientos Previos

2.1. Presentación WIMU

WIMU es un dispositivo inalámbrico compuesto por una serie de sensores que monitoriza la actividad física proporcionando información relevante de forma continua y en tiempo real.
Entre la información proporcionada destacan algunas variables cinemáticas como aceleración, velocidad o distancia recorrida y algunas variables fisiológicas como frecuencia cardiaca. Además, la empresa Realtrack Systems ha desarrollado un software llamado QÜIKO con el que procesar todos los datos, permitiendo trabajar con la información, tanto en tiempo real desde un dispositivo móvil como a posteriori.

Figura 2.1: Imagen dispositivo WIMU.

La utilización de este dispositivo está enfocado a entrenadores físicos, médicos, terapeutas, etc, encargados de mejorar la actividad física y competitiva

de deportistas de élite.

Los datos proporcionados por WIMU son recogidos por los siguientes componentes:

- *GPS/Galileo de 5 Hz.* Es decir, proporciona 5 datos por segundo y da información acerca de la posición, velocidad, distancia, etc.

- *Acelerómetro 3D, 1000Hz a 2G.* Es decir, recoge 1000 datos por segundo de fuerzas de hasta dos veces la fuerza de la gravedad terrestre que es 1G y proporciona información sobre la aceleración, sobre caída libre, tiempo de vuelo, etc.

- *Acelerómetro 3D, 1000Hz a 8G.* Es decir, recoge 1000 datos por segundo de fuerzas de hasta ocho veces la fuerza de la gravedad terrestre y proporciona información sobre impactos, carga, tiempos de reacción, saltos, golpes, etc.
  La existencia de dos tipos de acelerómetros se debe a que los acelerómetros que recogen un rango de aceleraciones más amplio son menos precisos que los que recogen aceleraciones menores, es por ello que se busca el equilibrio entre la precisión y tener la máxima información posible de los movimientos.

- *Giroscopio 3D, 140 Hz a* $2000°/s$. Es decir, recoge 140 datos por segundo de velocidades angulares hasta de $2000°/s$ y proporciona información sobre velocidad angular en cada uno de los ejes, etc.

- *Magnetómetro 3D, 50 Hz a 4 Gauss.* Es decir, recoge 50 datos por segundo e intensidades de campos magnéticos de hasta 4 Gauss, con ello nos proporciona información sobre la dirección del movimiento, diferencia entre carrera de frente y de espalda, etc.

- *Barómetro 9 Hz a 120 kPa.* Es decir, recoge 50 datos por segundo de presiones de, a lo sumo, 120 Kilopascales y proporciona información sobre la presión atmosférica, altura barométrica, previsión meteorológica, etc.

En la próxima sección explicaremos con más detalle el principio de funcionamiento y las características técnicas necesarias de dos de esos sensores, concretamente el acelerómetro y el giroscopio.

2.2. Sensores

El dispositivo con el que hemos tratado para la realización de este trabajo, WIMU, está compuesto por una unidad de medición inercial (IMU, inertial measurement unit) que consiste en un módulo electrónico que envía al procesador principal una colección de datos en los que recogen la velocidad angular y la aceleración lineal.

Una unidad de medición inercial está compuesta por dos tipos de sensores, el acelerómetro y el giroscopio.

- **Descripción de un acelerómetro**

  Un acelerómetro es un aparato electromecánico que nos permite medir fuerzas de aceleración. Estas fuerzas pueden ser estáticas, como la fuerza constante de la gravedad, o pueden ser dinámicas, causadas por el movimiento o la vibración del acelerómetro.
  Hay muchos tipos de acelerómetros, la mayoría de ellos basados en cristales piezoeléctricos (es decir, cristales que cuando se comprimen o se golpean generan una carga eléctrica). Sin embargo, este tipo de acelerómetros es demasiado grande y por ello se han desarrollado los acelerómetros MEMs (micro electromechanical systems), [2] [4] [5], mucho más pequeños, lo que permite una gran aplicabilidad en la microelectrónica. Aunque también hay varios tipos de acelerómetros dentro de los MEMs, los utilizados en nuestro dispositivo proporcionan la señal detectando cambios en la capacitancia o capacidad eléctrica.
  Este tipo de acelerómetro está compuesto por celdas capacitivas, formadas por placas fijas y placas móviles unidas a una masa, como podemos ver en la siguiente imagen, [2].

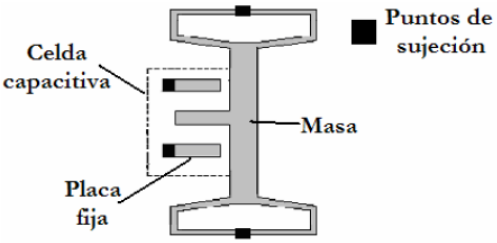


Figura 2.2: Esquema del acelerómetro MEMS.

Cuando el acelerómetro sufre una aceleración la masa se mueve y este movimiento modifica el valor de la capacitancia, de tal manera, que se obtiene un voltaje de salida proporcional a la magnitud de la fuerza a la que está siendo sometido el acelerómetro.

En ausencia de aceleración el sistema se encuentra en reposo, como en la figura 2.3.a, [2]. Cuando se presenta una fuerza en sentido negativo (los sentidos se han elegido de forma arbitraria), se mueve todo el sistema, pero la masa opone una resistencia al cambio de posición, de tal manera que se presenta un movimiento relativo entre el marco fijo y la masa como se muestra en la Figura 2.3.b , [2]. Para el caso de la Figura 2.3.c, [2], en el que se aplica una fuerza en sentido contrario, la masa presenta nuevamente una resistencia al cambio de posición, y de nuevo se manifiesta un movimiento relativo entre el marco fijo y la masa, lo que provoca una variación en la capacitancia.

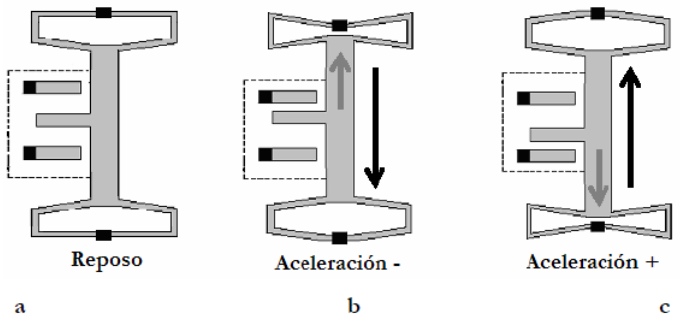


Figura 2.3:
Comportamiento del sensor MEMS con aceleración.

- **Descripción de un giroscopio**

Un giroscopio es un dispositivo capaz de medir la velocidad angular basado en el principio de conservación del momento angular.

La mayoría de giroscopios basan su funcionamiento en el efecto de Coriolis, un fenómeno que se observa en un sistema de referencia en rotación cuando un cuerpo se encuentra en movimiento respecto de dicho sistema de referencia. Este efecto consiste en la existencia de una

aceleración relativa del cuerpo en el sistema en rotación, que se caracteriza por ser siempre perpendicular al eje de rotación del sistema y a la velocidad del cuerpo.

El efecto Coriolis hace que un objeto que se mueve sobre el radio de un disco en rotación tienda a acelerarse con respecto a ese disco según si el movimiento es hacia el eje de giro o alejándose de éste.

El valor de la fuerza de Coriolis, F_c, es:

$$\vec{F_c} = -2m\left(\vec{\omega} \times \vec{v}\right),$$

donde:

- m es la masa del cuerpo.
- v es la velocidad del cuerpo en el sistema en rotación.
- ω es la velocidad angular del sistema en rotación vista desde un sistema inercial.
- $\times$ indica producto vectorial.

Veámos en la siguiente figura cómo se comporta dicha fuerza

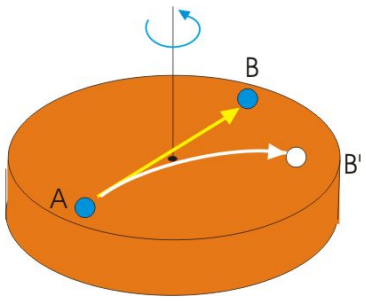


Figura 2.4: Fuerza de Coriolis. Reproducido de [7]

El giroscopio MEMS más utilizado y en concreto, el que utiliza WIMU, es el que tiene una configuración de horquilla vibrante. Esto quiere decir que está compuesto por dos masas que oscilan y se mueven de forma constante en direcciones opuestas (veáse figura 2.5, [3]). Cuando se aplica una velocidad angular, las fuerzas de Coriolis resultantes en cada masa tienen también sentidos opuestos, lo que provoca un cambio en la capacitancia.
Este cambio en la capcitancia es proporcional a la velocidad angular ω y se transforma en un voltaje, que será la señal que recibamos del giroscopio.

Cuando la aceleración aplicada a las dos masas es lineal, dichas masas se mueven en el mismo sentido, luego no se producen cambios en la capacitancia y el voltaje proporcionado ha de ser cero.

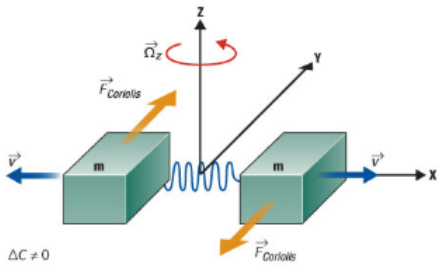


Figura 2.5: Esquema giroscopio MEMS.

2.3. Errores de los sensores

Hemos de tener en cuenta que para obtener la información que necesitamos del acelerómetro y el giroscopio debemos utilizar métodos de integración, pues para calcular la posición debemos integrar dos veces la aceleración y para la orientación hemos de integrar una vez la velocidad angular.
Esto supone que pequeños errores en las mediciones pueden tener grandes errores en el resultado final.
En esta sección vamos a presentar los tipos de errores que afectan a nuestros sensores, [8].

- **Error de sesgo constante.**
  - En el giroscopio, el error de sesgo en la velocidad angular es la salida que nos proporciona, medido en $^\circ/h$, cuando no está siendo

sometido a ninguna rotación. El error de sesgo constante, ϵ, cuando se integra provoca un error angular que crece de forma lineal con el tiempo.

$$\theta(t) = \epsilon t.$$

Es posible estimar este error de sesgo tomando mediciones en periodos donde el giroscopio no esté sometido a ninguna rotación.

- En el acelerómetro es también el «offset» en la señal de salida del valor real, medido en m/s^2. Un error de sesgo constante, ϵ, si se integra dos veces provoca un error en la posición que crece de forma cuadrática respecto al tiempo. El error acumulado en la posición viene dado por

$$s(t) = \epsilon \frac{t^2}{2},$$

donde t es el tiempo.
Es posible estimar este error de sesgo tomando mediciones en periodos donde el acelerómetro no esté sometido a ninguna aceleración pero teniendo en cuenta la gran complicación de que la fuerza de gravedad le afecta en todo momento.

- **Ruido blanco termodinámico.**

  - Las salidas proporcionadas por el giroscopio están perturbadas por secuencias de ruido termodinámico, que fluctúa con una velocidad mayor a la toma de datos del giroscopio. Como consecuencia, los datos obtenidos del sensor están perturbados por una secuencia de ruido blanco, que es básicamente una secuencia de variables aleatorias incorreladas y de media cero. En este caso, las variables aleatorias son idénticamente distribuidas y con varianza finita σ^2. Para hacernos una idea de cuál es el efecto del ruido blanco durante la integración de la señal, podemos asumir que el método de integración utilizado es el de la regla del rectángulo.

    Sea N_i la i-ésima variable aleatoria de la secuencia de ruido blanco. Cada N_i está idénticamente distribuida con media $E(N_i) = E(N) = 0$ y varianza finita $Var(N_i) = Var(N) = \sigma^2$. Por definición de una secuencia de ruido blanco $Cov(N_i, N_j) = 0,\ \forall i \neq j$.
    El resultado de integrar la señal de ruido blanco $\epsilon(t)$ utilizando la

regla del rectángulo sobre un periodo de tiempo $t = n \cdot \delta t$ es

$$\int_0^t \epsilon(\tau)d\tau = \delta t \sum_{i=1}^{n} N_i,$$

donde n es el número de datos recogidos por el aparato durante el periodo de tiempo y δt es el tiempo transcurrido entre la recogida de un dato y el siguiente.
Teniendo en cuenta la linealidad de la esperanza y que la varianza verifica $Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X, Y)$ (donde a y b son constantes y X e Y variables aleatorias) tenemos que

$$\begin{aligned} E\left(\int_0^t \epsilon(\tau)d\tau\right) &= \delta t \cdot nE(N) = 0 \\ Var\left(\int_0^t \epsilon(\tau)d\tau\right) &= \delta t^2 \cdot nVar(N) = \delta t \cdot t \cdot \sigma^2. \end{aligned}$$

Por tanto, el ruido introduce un error en la señal integrada cuya desviación estándar es

$$\sigma_\theta(t) = \sigma\sqrt{\delta t \cdot t}.$$

- La salidas proporcionadas por el acelerómetro también están perturbadas por una secuencia de ruido blanco. Para ver cuánto afecta este ruido en la posición podemos realizar un análisis similar al del giroscopio, pero ahora integrando dos veces los datos proporcionados por el sensor.
  Sea N_i la i-ésima variable aleatoria en la secuencia de ruido blanco, con $E(N_i) = E(N) = 0$ y $Var(N_i) = Var(N) = \sigma^2$. El resultado de la doble integral de la señal de ruido blanco $\epsilon(t)$ en un periodo de tiempo $n \cdot \delta t$ es

$$\begin{aligned} \int_0^t \int_0^t \epsilon(\tau)d\tau d\tau &= \delta t \sum_{i=1}^{n} \delta t \sum_{j=1}^{i} N_j \\ &= \delta t^2 \sum_{i=1}^{n} (n - i + 1)N_i, \end{aligned}$$

donde n es el número de datos tomados durante el periodo de

tiempo y δt es el tiempo que transcurre entre dos datos consecutivos. Así, el error esperado es

$$E\left(\int_0^t \epsilon(\tau)d\tau\right) = \delta t^2 \sum_{i=1}^{n}(n-i+1)E(N_i) = 0.$$

Y su varianza es

$$\begin{aligned} Var\left(\int_0^t \epsilon(\tau)d\tau\right) &= \delta t^4 \sum_{i=1}^{n}(n-i+1)^2 Var(N_i) \\ &= \frac{\delta t^4 n(n+1)(2n+1)}{6} Var(N_i) \\ &\approx \frac{1}{3}\cdot\delta t\cdot t^3\cdot\sigma^2 \end{aligned}$$

asumiendo en el último paso que δt es pequeño. Por tanto, el ruido introduce un error a la señal integrada cuya desviación estándar es

$$\sigma_s(t) \approx \sigma \cdot t^{3/2}\sqrt{\frac{\delta t}{3}}.$$

- **Efectos de la temperatura.**

  Las fluctuaciones de temperatura debido a los cambios ambientales o al calentamiento propio del sensor provoca un nuevo error de sesgo. Estas fluctuaciones, además, afectan de forma distinta a cada uno de los ejes tanto del acelerómetro como del giroscopio y van aumentando durante el tiempo que se mantiene encendido el dispositivo.

  Puesto que estos errores producidos por la temperatura se comportan como errores de sesgo, como hemos explicado anteriormente, en el caso del giroscopio crecerán de forma lineal con el tiempo y de forma cuadrática en el caso del acelerómetro.

- **Errores de calibración.**

  Este tipo de error se refiere a los errores de escala, desalineación y falta de ortogonalidad de los ejes conjuntamente. Este tipo de error afecta tanto al giroscopio como al acelerómetro y se comporta como un error de sesgo al integrar la señal y se van acumulando junto con los errores provocados por la temperatura durante el periodo de tiempo que el aparato está encendido.
  Las causas fundamentales por las que se tiene este tipo de error son causas de fabricación y envejecimiento de los sensores.

2.4. Rotaciones y transformaciones en el espacio

2.4.1. Cambio de coordenadas en un espacio vectorial

Supongamos que tenemos un espacio vectorial $\mathfrak{V}$ sobre $\mathbb{R}$ de dimensión n (más adelante nos interesará fundamentalmente el caso $\mathfrak{V} = \mathbb{R}^n$, y sobre todo, $n = 3$).

Vamos a usar las letras de tipo $\boldsymbol{a}$ para denotar *vectores* en $\mathfrak{V}$. Dada una base $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ de $\mathfrak{V}$, todo vector $\boldsymbol{w} \in \mathfrak{V}$ se puede representar de forma única como combinación lineal

$$\boldsymbol{w} = \alpha_1 \boldsymbol{v}_1 + \cdots + \alpha_n \boldsymbol{v}_n. \tag{2.1}$$

El vector columna $(\alpha_1, \ldots, \alpha_n)^T \in \mathbb{R}^n$ es el vector de coordenadas de $\boldsymbol{w}$ en la base ordenada $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$, y lo denotaremos por $[\boldsymbol{w}]_{\boldsymbol{V}}$:

$$[\boldsymbol{w}]_{\boldsymbol{V}} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}.$$

Si formalmente entendemos por la base $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ un vector fila (de vectores $\boldsymbol{v}_j$), entonces (2.1) se puede escribir como

$$\boldsymbol{w} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \boldsymbol{V}[\boldsymbol{w}]_{\boldsymbol{V}}.$$

Observación 2.4.1. Si $\mathfrak{V} = \mathbb{R}^n$, cada vector de la base $\boldsymbol{v}_j$ es a su vez un vector (columna), de modo que en ese espacio $\boldsymbol{V}$ es en realidad una *matriz* de $n \times n$.

Supongamos ahora que tenemos otra base ordenada, $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$, de $\mathfrak{V}$, y queremos calcular el vector de coordenadas $[\boldsymbol{w}]_{\boldsymbol{U}}$ de $\boldsymbol{w}$ en la nueva base $\boldsymbol{U}$. En otras palabras, queremos que

$$\boldsymbol{U}[\boldsymbol{w}]_{\boldsymbol{U}} = \boldsymbol{w} = \boldsymbol{V}[\boldsymbol{w}]_{\boldsymbol{V}}.$$

Pero cada elemento $\boldsymbol{v}_j$ de la *vieja* base $\boldsymbol{V}$ es un elemento de $\mathfrak{V}$, y $\boldsymbol{U}$ es una base de $\mathfrak{V}$, de modo que, tal como hemos acordado,

$$\boldsymbol{v}_1 = \boldsymbol{U}[\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, \boldsymbol{v}_n = \boldsymbol{U}[\boldsymbol{v}_n]_{\boldsymbol{U}},$$

lo que se puede agrupar en una sola expresión,

$$\boldsymbol{V} = \boldsymbol{U}\,([\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, [\boldsymbol{v}_n]_{\boldsymbol{U}}). \tag{2.2}$$

Se puede observar que independientemente de lo abstracto que sea el espacio vectorial $\mathfrak{V}$, la expresión

$$([\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, [\boldsymbol{v}_n]_{\boldsymbol{U}})$$

es una matriz de $\mathbb{R}^{n\times n}$, donde la columna j es el vector coordenadas del vector $\boldsymbol{v}_j$ de la base antigua en la base nueva U. Vamos a denotar esta matriz (conocida como *matriz de cambio de base de* $\boldsymbol{V}$ *a* $\boldsymbol{U}$) por

$$T_{\boldsymbol{U}\leftarrow\boldsymbol{V}} = ([\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, [\boldsymbol{v}_n]_{\boldsymbol{U}}).$$

Con esta notación la identidad (2.2) se escribe como

$$\boldsymbol{V} = \boldsymbol{U}\,T_{\boldsymbol{U}\leftarrow\boldsymbol{V}}.$$

Entonces tenemos esta cadena de identidades:

$$\boldsymbol{w} = \boldsymbol{V}[\boldsymbol{w}]_{\boldsymbol{V}} = (\boldsymbol{U}\,T_{\boldsymbol{U}\leftarrow\boldsymbol{V}})\,[\boldsymbol{w}]_{\boldsymbol{V}} = \boldsymbol{U}\,\left(T_{\boldsymbol{U}\leftarrow\boldsymbol{V}}\,[\boldsymbol{w}]_{\boldsymbol{V}}\right).$$

O sea, lo que está entre paréntesis en el miembro derecho es el vector de coordenadas de $\boldsymbol{w}$ en la nueva base $\boldsymbol{U}$. En otras palabras, hemos obtenido la conocida expresión para el cambio de base:

$$\boxed{[\boldsymbol{w}]_{\boldsymbol{U}} = T_{\boldsymbol{U}\leftarrow\boldsymbol{V}}\,[\boldsymbol{w}]_{\boldsymbol{V}}}$$

Donde recordemos, la columna j es el vector coordenadas del vector $\boldsymbol{v}_j$ de la base antigua en la base nueva U:

$$\boxed{T_{\boldsymbol{U}\leftarrow\boldsymbol{V}} = ([\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, [\boldsymbol{v}_n]_{\boldsymbol{U}})}$$

2.4.2. Cambio de coordenadas en un espacio euclídeo

Supongamos que $\mathfrak{V}$ es además un espacio euclídeo; en otras palabras, en $\mathfrak{V}$ está definido el producto escalar $\langle \boldsymbol{u}, \boldsymbol{v}\rangle$ entre dos vectores $\boldsymbol{u}, \boldsymbol{v} \in \mathfrak{V}$. Por ejemplo, la definición estándar del producto escalar en $\mathbb{R}^n$ es

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}, \quad \boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \quad \Rightarrow \quad \langle \boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{u}\cdot\boldsymbol{v} = \boldsymbol{u}^T\boldsymbol{v} = \sum_{j=1}^{n} u_j v_j.$$

En un espacio euclídeo un concepto importante es el de vectores *ortogonales* (tales que $\boldsymbol{u} \cdot \boldsymbol{v} = 0$) y de *ortonormales* (que adicionalmente cumplen que $\|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1$, donde como es habitual, $\|\boldsymbol{u}\| = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle}$). Una base ortonormal $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ está formada por vectores ortonormales; una de las ventajas de trabajar con bases así es la simplicidad de calcular las coordenadas $[\boldsymbol{w}]_{\boldsymbol{U}}$ para cualquier vector $\boldsymbol{w} \in \mathfrak{V}$:

$$\begin{aligned} \boldsymbol{w} = \alpha_1 \boldsymbol{u}_1 + \ldots \alpha_n \boldsymbol{u}_n \qquad &\Rightarrow \qquad \alpha_j = \boldsymbol{w} \cdot \boldsymbol{u}_j, \quad j = 1, \ldots, n, \\ &\Rightarrow \qquad [\boldsymbol{w}]_{\boldsymbol{U}} = \begin{pmatrix} \langle \boldsymbol{w}, \boldsymbol{v}_1 \rangle \\ \vdots \\ \langle \boldsymbol{w}, \boldsymbol{v}_n \rangle \end{pmatrix}. \end{aligned}$$

Supongamos que tenemos dos bases ordenadas, $\boldsymbol{V}$ y $\boldsymbol{U}$, donde $\boldsymbol{U}$ *es ortonormal*, y queremos hallar la matriz de cambio de coordenadas de la base $\boldsymbol{V}$ a la base $\boldsymbol{U}$, en otras palabras, $T_{\boldsymbol{U} \leftarrow \boldsymbol{V}}$. Por lo que acabamos de ver,

$$T_{\boldsymbol{U} \leftarrow \boldsymbol{V}} = ([\boldsymbol{v}_1]_{\boldsymbol{U}}, \ldots, [\boldsymbol{v}_n]_{\boldsymbol{U}}) = \begin{pmatrix} \langle \boldsymbol{v}_1, \boldsymbol{u}_1 \rangle & \langle \boldsymbol{v}_2, \boldsymbol{u}_1 \rangle & \ldots & \langle \boldsymbol{v}_n, \boldsymbol{u}_1 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \boldsymbol{v}_1, \boldsymbol{u}_n \rangle & \langle \boldsymbol{v}_2, \boldsymbol{u}_n \rangle & \ldots & \langle \boldsymbol{v}_n, \boldsymbol{u}_n \rangle \end{pmatrix}.$$

Una matriz así, cuyo elemento (i, j) es igual a $\langle \boldsymbol{v}_i, \boldsymbol{u}_j \rangle$, se llama *matriz de Gram*. Obviamente, ella es simétrica.

Recordemos que para dos vectores cualesquiera $\boldsymbol{u}$ y $\boldsymbol{v}$ de $\mathfrak{V}$,

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|},$$

de modo que si *las dos bases,* $\boldsymbol{V}$ *y* $\boldsymbol{U}$*, están constituidas por vectores unitarios* (y $\boldsymbol{U}$ es ortonormal), entonces esa matriz tiene la siguiente interpretación geométrica:

$$T_{\boldsymbol{U} \leftarrow \boldsymbol{V}} = \begin{pmatrix} \cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_1) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_1) & \ldots & \cos \angle(\boldsymbol{v}_n, \boldsymbol{u}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_n) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_n) & \ldots & \cos \angle(\boldsymbol{v}_n, \boldsymbol{u}_n) \end{pmatrix}. \tag{2.3}$$

Esto es lo que se conoce en ingeniería como *direction cosine matrix* o *DCM*. Por tanto, DCM es la matriz de cambio de una base ortonormal a otra base de vectores unitarios (ortogonales o no).

Si la base $\boldsymbol{V}$ también fuese *ortonormal*, entonces $T_{\boldsymbol{U} \leftarrow \boldsymbol{V}}$ *es una matriz ortogonal* (su inversa = su transpuesta). En efecto, basta comprobar que sus

columnas forman una base ortonormal en $\mathbb{R}^n$. Tenemos que

$$\begin{pmatrix} \langle \boldsymbol{v}_i, \boldsymbol{u}_1 \rangle \\ \vdots \\ \langle \boldsymbol{v}_i, \boldsymbol{u}_n \rangle \end{pmatrix} \cdot \begin{pmatrix} \langle \boldsymbol{v}_j, \boldsymbol{u}_1 \rangle \\ \vdots \\ \langle \boldsymbol{v}_j, \boldsymbol{u}_n \rangle \end{pmatrix} = \sum_{k=1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{u}_k \rangle \, \langle \boldsymbol{v}_j, \boldsymbol{u}_k \rangle = \sum_{k=1}^{n} \langle \boldsymbol{v}_j, \langle \boldsymbol{v}_i, \boldsymbol{u}_k \rangle \, \boldsymbol{u}_k \rangle$$

$$= \left\langle \boldsymbol{v}_j, \sum_{k=1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{u}_k \rangle \, \boldsymbol{u}_k \right\rangle = \langle \boldsymbol{v}_j, \boldsymbol{v}_i \rangle = \begin{cases} 0, & \text{si } i \neq j, \\ 1, & \text{si } i = j. \end{cases}$$

En particular, si las dos bases, $\boldsymbol{V}$ y $\boldsymbol{U}$, son ortonormales, la matriz de transformación inversa, $T_{\boldsymbol{V} \leftarrow \boldsymbol{U}}$, se obtiene simplemente como la transpuesta de $T_{\boldsymbol{U} \leftarrow \boldsymbol{V}}$.

En el caso de $\mathbb{R}^n$, la transformación de coordenadas que conserva la ortonormalidad de las bases es una rotación en el espacio (teorema de Euler). En ese caso las matrices DCM son de hecho las *matrices de rotación*.

En el caso particular de $\mathfrak{V} = \mathbb{R}^3$, las matrices DCM (2.3) tienen la forma

$$T_{\boldsymbol{U} \leftarrow \boldsymbol{V}} = \begin{pmatrix} \cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_1) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_1) & \cos \angle(\boldsymbol{v}_3, \boldsymbol{u}_1) \\ \cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_2) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_2) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_3) \\ \cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_3) & \cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_3) & \cos \angle(\boldsymbol{v}_3, \boldsymbol{u}_3) \end{pmatrix}.$$

Si asumimos que todos los $\angle(\boldsymbol{v}_i, \boldsymbol{u}_i)$ son pequeños (lo que corresponde a una pequeña "pérdida de alineación" de la base original $\boldsymbol{V}$, siendo la nueva base aún unitaria), una aproximación razonables es

$$\cos \angle(\boldsymbol{v}_i, \boldsymbol{u}_i) \approx 1.$$

Por otro lado, eso significa que los ángulos restantes, $\angle(\boldsymbol{v}_j, \boldsymbol{u}_i) \approx \pi/2$, $i \neq j$, de modo que si denotamos por ejemplo $\angle(\boldsymbol{v}_2, \boldsymbol{u}_1) = \pi/2 - \theta_{12}$, tenemos que $\theta_{12} \approx 0$, y

$$\cos \angle(\boldsymbol{v}_2, \boldsymbol{u}_1) = \sin \theta_{12} \approx \theta_{12}.$$

Por convenio, se denota $\angle(\boldsymbol{v}_1, \boldsymbol{u}_2) = \pi/2 + \theta_{21}$, y por el mismo argumento

$$\cos \angle(\boldsymbol{v}_1, \boldsymbol{u}_2) = -\sin \theta_{21} \approx -\theta_{21}.$$

Haciendo lo mismo para los demás elementos de la matriz, obtenemos que *en el caso de una perturbación del sistema ortonormal inicial* $\boldsymbol{V}$ *que conserva las normas unitarias de la base* (sin conservar necesariamente la ortogonalidad), la DCM se aproxima bien por la expresión

$$T_{\boldsymbol{U} \leftarrow \boldsymbol{V}} \approx \begin{pmatrix} 1 & \theta_{12} & -\theta_{13} \\ -\theta_{21} & 1 & \theta_{23} \\ \theta_{13} & -\theta_{32} & 1 \end{pmatrix} = I + \begin{pmatrix} 0 & \theta_{12} & -\theta_{13} \\ -\theta_{21} & 0 & \theta_{23} \\ \theta_{13} & -\theta_{32} & 0 \end{pmatrix}, \tag{2.4}$$

donde I es la matriz identidad.

2.4.3. Ángulos de Euler

Las matrices de rotación son fáciles de utilizar, pero sus elementos son difíciles de interpretar. Vamos a restringirnos al caso de $\mathbb{R}^3$. Euler probó que para especificar la rotación de un sistema de coordenadas en el espacio se necesitan 3 parámetros reales, y que toda rotación en $\mathbb{R}^3$ puede obtenerse como una composición de tres *rotaciones elementales* consecutivas: seleccionamos el orden de los ejes coordenados y efectuamos la rotación alrededor de ese eje de forma consecutiva. Los tres ángulos correspondientes forman los *ángulos de Euler*. Puesto que hay $3! = 12$ formas de ordenar 3 ejes, a cada rotación espacial le corresponden 12 tripletes distintos de ángulos de Euler; lo que supone una desventaja.

Sin embargo, si se conocen los ángulos de Euler, la matriz DCM es fácil de obtener, multiplicando de forma consecutiva las *matrices de rotación elementales*. Por ejemplo, una rotación alrededor del eje OX (que deja la coordenada x invariante) un ángulo θ_x está dada por su matriz DCM (ver (2.3))

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{pmatrix}.$$

Análogamente, a una rotación alrededor del eje OY (resp., OZ) un ángulo θ_x (resp., θ_z) le corresponde la matriz

$$R_y = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \quad \text{y} \quad R_z = \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

respectivamente. Entonces, a una transformación que se obtiene de realizar primero una rotación un ángulo θ_x alrededor del eje OX, seguida por una rotación un ángulo θ_y alrededor del (nuevo) eje OY, y finalmente por una rotación un ángulo θ_z alrededor del nuevo eje OZ le corresponde la matriz de rotación (o DCM)

$$R = R_z R_y R_x.$$

Si los ángulos de rotación θ son pequeños, una aproximación razonable es $\cos\theta \approx 1$, $\sin\theta \approx \theta$, de modo que

$$R_x \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \theta_x \\ 0 & -\theta_x & 1 \end{pmatrix}, \quad R_y \approx \begin{pmatrix} 1 & 0 & \theta_y \\ 0 & 1 & 0 \\ -\theta_y & 0 & 1 \end{pmatrix} \quad \text{y} \quad R_z \approx \begin{pmatrix} 1 & \theta_z & 0 \\ -\theta_z & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Realizando la multiplicación $R_z R_y R_x$ y descartando los términos de orden

superior (despreciables), nos queda que

$$R \approx \begin{pmatrix} 1 & \theta_z & \theta_y \\ -\theta_z & 1 & \theta_x \\ -\theta_y & -\theta_x & 1 \end{pmatrix},$$

lo que es un caso particular de la formula linealizada (2.4).

2.4.4. Rotaciones en $\mathbb{R}^3$. Cuaterniones

La matriz de rotación planteada en la seccion 2.4.3 presenta algunos problemas como son la existencia de singularidades y que resulta redundante porque solo cuatro de sus nueve elementos son independientes, además de que su interpretación geométrica no es clara, pues se necesitan numerosos cálculos para obtener el eje de rotación y el ángulo. Por tanto, con el objetivo de eliminar estos problemas se propone otra forma de representar las rotaciones en $\mathbb{R}^3$ mediante cuaterniones.

Los cuaterniones fueron desarrollados por W. R. Hamilton en 1843 y consisten en 4-tuplas que permiten representar rotaciones de una forma más concisa y eficaz que la presentada anteriormente, con un significado geométrico mucho más claro puesto que, tanto el ángulo como el eje de rotación se puede recuperar de forma inmediata.

Cuaterniones. Definición y propiedades

El conjunto de los cuaterniones, junto con las operaciones de suma y multiplicación, forma un anillo no conmutativo. La base ortonormal usual en $\mathbb{R}^3$ viene dada por tres vectores unitarios $\mathbf{i} = (1,0,0)$, $\mathbf{j} = (0,1,0)$, $\mathbf{k} = (0,0,1)$. Un *cuaternion* q se define como la suma de un escalar q_0 y un vector $\mathbf{q} = (q_1, q_2, q_3)$, es decir,

$$q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}.$$

La suma de dos cuaterniones se realiza componente a componente. Consideremos el cuaternión $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ y otro cuaternión $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$, entonces

$$p + q = (p_0 + q_0) + (p_1 + q_1)\mathbf{i} + (p_2 + q_2)\mathbf{j} + (p_3 + q_3)\mathbf{k}.$$

Todo cuaternión q tiene opuesto $-q$ con componentes $-q_i$, $i = 0, 1, 2, 3$.

El producto de dos cuaterniones satisface las siguientes igualdades fundamentales introducidas por Hamilton:

$$\begin{array}{rcrcrcl} \mathbf{i}^2 & = & \mathbf{j}^2 & = & \mathbf{k}^2 & = & \mathbf{ijk} = -1, \\ \mathbf{ij} & = & \mathbf{k} & = & \mathbf{\text{-}ji}, & & \\ \mathbf{jk} & = & \mathbf{i} & = & \mathbf{\text{-}kj}, & & \\ \mathbf{ki} & = & \mathbf{j} & = & \mathbf{\text{-}ik}. & & \end{array}$$

Así, el producto de dos cuaterniones p y q queda como:

$$\begin{aligned} pq &= (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ &= p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) + p_0(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + q_0(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \\ &\quad +(p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} + (p_1q_2 - p_2q_1)\mathbf{k}. \end{aligned}$$

Este producto lo podemos reescribir, de una forma más sencilla, utilizando el producto escalar y el producto vectorial de $\mathbb{R}^3$. Si $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ y $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$ se tiene

$$pq = p_0q_0 - \mathbf{p}\cdot\mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + p \times q. \tag{2.5}$$

Por tanto, el producto de dos cuaterniones es un cuaternión con parte escalar $p_0q_0 - \mathbf{p}\cdot\mathbf{q}$ y parte vectorial $p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}$. Así el conjunto de los cuaterniones es cerrado bajo la suma y la multiplicación, donde la multiplicación es distributiva respecto de la suma. El elemento identidad del conjunto de los cuaterniones es el cuaternión que tiene la parte real 1 y la parte vectorial 0.

También podemos definir el conjugado de un cuaternión, su norma y su inverso como haremos a continuación.

Sea $q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ un cuaternión, se define el *conjugado* de q, denotado como q^*, como

$$q^* = q_0 - \mathbf{q} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}.$$

De esta definición se sigue sin dificultad que

$$\begin{aligned} (q^*)^* &= q_0 - (-\mathbf{q}) = q, \\ q + q^* &= 2q_0, \\ q^*q &= (q_0 - \mathbf{q})(q_0 + \mathbf{q}) \\ &= q_0q_0 - (-\mathbf{q})\cdot\mathbf{q} + q_0\mathbf{q} + (-\mathbf{q}q_0) + (-\mathbf{q})\times\mathbf{q} \\ &= q_0^2 + \mathbf{q}\cdot\mathbf{q} \\ &= q_o^2 + q_1^2 + q_2^2 + q_3^2 \\ &= qq^*. \end{aligned}$$

Pasemos ahora a definir la *norma* de un cuaternión, q, denotada por $|q|$ como

$$|q| = \sqrt{q^* q}.$$

Diremos entonces que un cuaternión es *unitario* cuando su norma sea 1. Y definimos por último el *inverso* de un cuaternión, q, como

$$q^{-1} = \frac{q^*}{|q|^2}.$$

Es sencillo ver que $qq^{-1} = q^{-1}q = 1$ y si q es un vector unitario, su inverso coincide con su conjugado.

Operador rotación

Nos podemos hacer la pregunta de cómo puede un cuaternión, que «vive» en $\mathbb{R}^4$, operar sobre un vector de $\mathbb{R}^3$, pero hemos de recordar, en primer lugar, que un vector $\mathbf{v} \in \mathbb{R}^3$ es un *cuaternión puro*, es decir, que tiene parte real cero.
Consideremos un cuaternión unitario $q = q_0 + \mathbf{q}$ ($q_0^2 + \|\mathbf{q}\|^2 = 1$) lo que implica que existe un único existe $\theta \in [0, \pi]$ que verifica que $\cos\theta = q_0$ y $\operatorname{sen}\theta = \|\mathbf{q}\|$. Entonces podemos escribir el cuaternión unitario en términos del ángulo θ y del vector unitario $\mathbf{u} = \mathbf{q}/\|\mathbf{q}\|$:

$$q = \cos\theta + \mathbf{u}\operatorname{sen}\theta.$$

Utilizando el cuaternión unitario q podemos definir un operador sobre los vectores $\mathbf{v} \in \mathbb{R}^3$

$$\begin{aligned} L_q(0+\mathbf{v}) &= q(0+\mathbf{v})q^* \\ &= (q_0^2 - \|\mathbf{q}\|)(0+\mathbf{v}) + 2(\mathbf{q}\cdot(0+\mathbf{v}))\mathbf{q} + 2q_0(\mathbf{q}\times(0+\mathbf{v})). \end{aligned}$$

Hemos de hacer dos observaciones sobre el operador definido. En primer lugar, dicho operador no cambia la longitud del vector $\mathbf{v}$, pues

$$\begin{aligned} \|L_q(0+\mathbf{v})\| &= \|q(0+\mathbf{v})q^*\| \\ &= |q|\cdot\|(0+\mathbf{v})\||q^*| \\ &= \|(0+\mathbf{v})\|. \end{aligned}$$

En segundo lugar, la dirección de $(0+\mathbf{v})$ sobre $\mathbf{q}$, es invariante bajo la acción del operador L_q. Para probarlo, supongamos que $\mathbf{v} = k\mathbf{q}$ y

$$\begin{aligned} q(0+\mathbf{v})q^* &= q(k\mathbf{q})q^* \\ &= (q_0^2 - \|\mathbf{q}\|^2)(k\mathbf{q}) + 2(\mathbf{q}\cdot k\mathbf{q})\mathbf{q} + 2q_0(\mathbf{q}\times k\mathbf{q}) \\ &= k(q_0^2 + \|\mathbf{q}\|^2)\mathbf{q} \\ &= k\mathbf{q}. \end{aligned}$$

Probemos ahora que el operador L_q es lineal en $\mathbb{R}^3$. Sean $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^3$ y sean $a_1, a_2 \in \mathbb{R}$,

$$\begin{aligned}
L_q(a_1(0+\mathbf{v}_1) &+ a_2(0+\mathbf{v}_2)) \\
&= q(a_1(0+\mathbf{v}_1)+a_2(0+\mathbf{v}_2))q^* \\
&= (q_0^2-\|\mathbf{q}\|^2)(a_1(0+\mathbf{v}_1)+a_2(0+\mathbf{v}_2)) \\
&\quad +2(\mathbf{q}\cdot(a_1(0+\mathbf{v}_1)+a_2(0+\mathbf{v}_2)))\mathbf{q} \\
&\quad +2(\mathbf{q}\times(a_1(0+\mathbf{v}_1)+a_2(0+\mathbf{v}_2))) \\
&= a_1(q_0^2-\|\mathbf{q}\|^2)(0+\mathbf{v}_1)+a_2(q_0^2-\|\mathbf{q}\|^2)(0+\mathbf{v}_2) \\
&\quad +2a_1(\mathbf{q}\cdot\mathbf{v}_1)\mathbf{q}+2a_2(\mathbf{q}\cdot\mathbf{v}_2)\mathbf{q} \\
&\quad +2q_0a_1(\mathbf{q}\times(0+\mathbf{v}_1))+2q_0a_2(\mathbf{q}\times(0+\mathbf{v}_2)) \\
&= a_1[(q_0^2-\|\mathbf{q}\|)(0+\mathbf{v}_1) \\
&\quad +2(\mathbf{q}\cdot(0+\mathbf{v}_1))\mathbf{q}+2q_0(\mathbf{q}\times(0+\mathbf{v}_1))] \\
&\quad +a_2[(q_0^2-\|\mathbf{q}\|)\mathbf{v}_2+2(\mathbf{q}\cdot(0+\mathbf{v}_2))\mathbf{q}+2q_0(\mathbf{q}\times(0+\mathbf{v}_2))] \\
&= a_1L_q(0+\mathbf{v}_1)+a_2L_q(0+\mathbf{v}_2).
\end{aligned}$$

Tras las dos observaciones realizadas anteriormente podemos intuir que L_q actúa como una rotación sobre $\mathbf{q}$, esta intuición la precisamos en el siguiente teorema:

Teorema 2.4.2. *Para cualquier cuaternión unitario*

$$q = q_0 + \boldsymbol{q} = \cos\frac{\theta}{2} + \boldsymbol{u}\,\mathrm{sen}\,\frac{\theta}{2}$$

y para cualquier vector $\boldsymbol{v} \in \mathbb{R}^3$ la acción del operador

$$L_q(\boldsymbol{v}) = q\boldsymbol{v}q^*$$

sobre $\boldsymbol{v}$ es equivalente a la rotación de un vector con un ángulo θ y con $\boldsymbol{u}$ como eje de rotación.

Demostración. Dado un vector $\mathbf{v} \in \mathbb{R}^3$, lo podemos descomponer como $\mathbf{v} = \mathbf{a} + \mathbf{n}$, donde $\mathbf{a}$ es la componente sobre el vector $\mathbf{q}$ y $\mathbf{n}$ es la componente normal al vector $\mathbf{q}$. Entonces bajo el operador L_q, $\mathbf{a}$ es invariante mientras que $\mathbf{n}$ queda rotado un ángulo θ sobre $\mathbf{q}$. Como el operador es lineal, tenemos que la imagen $q\mathbf{v}q^*$ se puede interpretar por tanto, como una rotación de $\mathbf{v}$ sobre $\mathbf{q}$ por un ángulo θ.

Como hemos razonado anteriormente, sabemos que $\mathbf{a}$ es invariante bajo la acción de L_q. Por tanto, centrémonos en el efecto de L_q sobre la componente ortogonal $\mathbf{n}$. Tenemos:

$$\begin{aligned}
L_q(\mathbf{n}) &= (q_0^2-\|\mathbf{q}\|)\mathbf{n}+2(\mathbf{q}\cdot\mathbf{n})\mathbf{q}+2q_0(\mathbf{q}\times\mathbf{n}) \\
&= (q_0^2-\|\mathbf{q}\|)\mathbf{n}+2q_0(\mathbf{q}\times\mathbf{n}) \\
&= (q_0^2-\|\mathbf{q}\|)\mathbf{n}+2q_0\|\mathbf{q}\|(\mathbf{u}\times\mathbf{n})
\end{aligned}$$

en la última igualdad se ha tenido en cuenta el hecho de que $\mathbf{u} = q/\|\mathbf{q}\|$. Denotemos por $\mathbf{n}_\perp = \mathbf{u} \times \mathbf{n}$ dejando la ecuación anterior como sigue:

$$L_q(\mathbf{n}) = (q_0^2 - \|\mathbf{q}\|)\mathbf{n} + 2q_0\|\mathbf{q}\|n_\perp. \tag{2.6}$$

Podemos ver, además, que tanto $\mathbf{n}$ como $\mathbf{n}_\perp$ tienen la misma longitud,

$$\|n_\perp\| = \|\mathbf{u} \times \mathbf{n}\| = \|\mathbf{n}\| \cdot \|\mathbf{u}\| \operatorname{sen} \frac{\pi}{2} = \|n\|.$$

Donde en la penúltima igualdad hemos utilizado la propiedad del producto vectorial que dice que si tenemos dos vectores $a, b \in \mathbb{R}^3$, se verifica

$$a \times b = (|a||b| \operatorname{sen} \theta)\widehat{n},$$

donde $\widehat{n}$ es el vector unitario y ortogonal a los vectores a, b y θ es el ángulo entre dichos vectores.

Así, como $\mathbf{u}$ y $\mathbf{n}$ son ortogonoles por serlo $\mathbf{q}$ y $\mathbf{n}$, el ángulo formado es $\pi/2$ y teniendo en cuenta que $\mathbf{u}$ es unitario llegamos a la última igualdad, que nos prueba que $\mathbf{n}$ y $\mathbf{n}_\perp$ tienen la misma longitud.

Finalmente, teniendo en cuenta que $\cos^2\theta = q_0^2$ y $\operatorname{sen}^2\theta = \|q\|^2$, podemos reescribir la ecuación 2.6 como

$$\begin{aligned} L_q(\mathbf{n}) &= \left(\cos^2\tfrac{\theta}{2} - \operatorname{sen}^2\tfrac{\theta}{2}\right)\mathbf{n} + \left(2\cos\tfrac{\theta}{2}\operatorname{sen}\tfrac{\theta}{2}\right)\mathbf{n}_\perp \\ &= \cos\theta\mathbf{n} + \operatorname{sen}\theta\mathbf{n}_\perp, \end{aligned}$$

donde hemos utilizado las siguientes identidades trigonómetricas

$$\begin{aligned} \cos^2 x - \operatorname{sen}^2 x &= \cos(x+y)\cos(x-y). \\ \operatorname{sen} x - \operatorname{sen} y &= 2\cos(\tfrac{x+y}{2})\operatorname{sen}(\tfrac{x-y}{2}). \end{aligned}$$

Por tanto, el resultado obtenido al aplicar el operador L_q es una rotación de $\mathbf{n}$ de un ángulo θ en el plano generado por $\mathbf{n}$ y $\mathbf{n}_\perp$ (veáse figura 2.6, [17]).

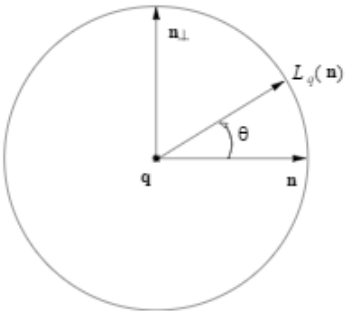


Figura 2.6: Acción del operador L_q sobre $\mathbf{n}$.

□

Desarrollando $q\mathbf{v}q^*$ como una matriz obentemos que si tenemos $\mathbf{v} = (x\ y\ z)$, el operador queda de la forma

$$L_q(\mathbf{v}) = \begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+1_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

Y será la forma matricial que acabamos de presentar la que utilizaremos en el algoritmo de calibración del giroscopio.

2.5. Regularización.

La regularización es un método que permite calcular soluciones estables de problemas mal planteados.
El concepto de problema mal planteado proviene de Hadamard, quien definió un problema mal planteado como cualquier problema que no estuviera bien planteado, considerando el mismo como cualquier problema abstracto constituido por un conjunto de datos X, un conjunto de soluciones Y y una aplicación $f : X \to Y$ que verifica las siguientes condiciones:

1. f está definida en todo X;
2. para todo $x \in X$, la solución $y = f(x)$ es única;
3. f es continua respecto a x.

Las primeras dos condiciones se resumen diciendo que f es una función $X \to Y$, mientras que la última condición se conoce como estabilidad del problema respecto a los datos.

Además, Hadamard pensaba que estos problemas eran «artificiales» y que no representaban fenómenos físicos, sin embargo, hoy en día hay una enorme cantidad de problemas de este tipo en distintas áreas de la ciencia y la ingeniería.

La teoría de la regularización surgió para problemas lineales como el que se presenta a continuación

$$Ax = b \;\; A \in \mathbb{R}^{n\times n} \tag{2.7}$$

o para problemas de mínimos cuadrados lineales

$$\min_x \|Ax - b\|_2, \;\; A \in \mathbb{R}^{m\times n}, \;\; m > n \tag{2.8}$$

en los que podemos precisar aún más la definición de problema mal planteado. Diremos que un problema está mal planteado, en este caso, si se verifican las siguiente condiciones:

- Los valores singulares de A van tendiendo a cero.
- La proporción entre el valor singular más grande y el más pequeño, distinto de cero, es grande.

Donde la segunda condición implica que A está mal condicionada, es decir, que la solución será muy sensible a pequeñas perturbaciones.

La dificultad fundamental de estos problemas es que son subdeterminados debido al grupo de valores singulares pequeños de A. Por tanto, es necesario incorporar información adicional sobre la solución que necesitamos para poder estabilizar el problema y conseguir una solución útil y estable. Éste será exactamente el objetivo de la regularización.

La regularización más conocida y utilizada es la *Regularización de Tikhonov*. Es claro que resolver el problema 2.8 es equivalente a resolver

$$\min_x \|Ax - b\|_2^2, \;\; A \in \mathbb{R}^{m\times n}, \;\; m > n$$

entonces, la regularización de Tikhonov propone añadir un término a la función objetivo del problema de minimización, que depende de un parámetro no negativo λ y de la norma del vector x:

$$\min_x \|Ax - b\|_2 + \lambda^2 \|x\|^2. \tag{2.9}$$

Así, si el valor del parámetro de regularización, λ, es el adecuado podemos encontrar una solución cercana al problema de mínimos cuadrados estándar, 2.8, pero de forma que la norma del vector solución sea pequeña.

Si λ es cero estamos ante el problema estándar, sin embargo, si λ tiende a infinito el vector x se irá acercando a cero y no tendríamos una solución adecuada.

Pero la regularización no es sólo aplicable a problemas lineales si no que se puede aplicar a problemas no lineales, como será nuestro caso.

Diremos que nos encontramos ante un problema de mínimos cuadrados no lineales si queremos encontrar un mínimo $\mathbf{x}^*$ de una función objetivo no lineal, f, dada de la siguiente forma:

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{\mathbf{x}} \frac{1}{2} r(x)^T r(x) = \min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{m} r_i(x)^2, \quad m \geq n, \tag{2.10}$$

donde $r : \mathbb{R}^n \to \mathbb{R}^m$ es una función no lineal de x. El problema de mínimos cuadrados no lineales se puede ver como un problema de minimización sin restricciones con una estructura particular. De hecho, este problema se puede interpretar como resolver un sistema de m ecuaciones no lineales

$$r_i(x) = 0, \quad i = 1, 2, \cdots, m,$$

donde $r_i(x)$ se conoce como función residual. Cuando $m > n$ el sistema es sobredeterminado, mientras que si $m = n$ el problema está bien determinado.

Las condiciones necesarias y suficientes para que un problema de mínimos cuadrados no lineales tenga un mínimo, como en el caso lineal, son las siguientes:

1. **Condición necesaria de primer orden.** Si $\mathbf{x}^*$ es un punto estacionario, es decir,
$$\nabla f(\mathbf{x}^*) = 0.$$

2. **Condición suficiente de segundo orden.** El hessiano, $\nabla^2 f(\mathbf{x}^*)$ es definido positivo.

Donde recordemos, el gradiente de una función escalar de n variables, $f(x)$ se define como el vector dado por

$$\frac{\partial f(\mathbf{x})}{\partial x_j}, \quad j = 1, \cdots, n.$$

y su hessiano, $\nabla^2 f(\mathbf{x})$ como la matriz simétrica dada por

$$\left[\nabla^2 f(\mathbf{x})\right]_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}, \quad i, j = 1, \cdots, n.$$

2.6. Algoritmos para resolver problemas de mínimos cuadrados no lineales. Algoritmo de Levenberg-Marquardt

Normalmente, para resolver problemas de mínimos cuadráticos no lineales se utiliza el método de Newton con el objetivo de resolver el sistema de ecuaciones normales.

Supongamos que nos encontramos ante el problema

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{\mathbf{x}} \frac{1}{2} r(x)^T r(x) = \min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{m} r_i(x)^2, \ \ m \geq n.$$

Sea $J(x)$ el jacobiano de $r(x)$, entonces el gradiente de $f(x)$ es

$$g(x) = \sum_{i=1}^{m} r_i(x) \nabla r_i(x) = J(x)^T r(x)$$

y su hessiano es

$$\begin{aligned} G(x) &= \sum_{i=1}^{m} (\nabla r_i(x) \nabla r_(x)^T + r_i(x) \nabla^2 r_i(x)) \\ &= J(x)^T J(x) + S(x), \end{aligned}$$

donde

$$S(x) = \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x).$$

Por tanto, el modelo cuadrático de la función objetivo $f(x)$ es

$$\begin{aligned} q^{(k)}(x) &= f(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T G(x_k)(x - x_k) \\ &= \frac{1}{2}r(x_k)^T r(x_k) + (J(x)^T r(x_k))^T(x - x_k) \\ &\quad + \frac{1}{2}(x - x_k)^T(J(x_k)^T J(x_k) + S(x_k))(x - x_k). \end{aligned}$$

Entonces, el método de Newton viene dado por

$$x_{k+1} = x_k - (J(x_k)^T J(x_k) + S(x_k))^{-1} J(x_k)^T r(x_k).$$

Pero el método de Newton nos plantea un problema en algunos casos, y es la gran dificultad y coste computacional que supone calcular el término de segundo orden $S(x)$. Entonces se plantea un método de forma que este término $S(x)$ de $G(x)$ es eliminado en el modelo cuadrático de la función objetivo $f(x)$, quedándo lo siguiente:

$$\begin{aligned} q^{(k)}(x) &= \frac{1}{2}r(x_k)^T r(x_k) + (J(x)^T r(x_k))^T(x - x_k) && (2.11) \\ &\quad + \frac{1}{2}(x - x_k)^T(J(x_k)^T J(x_k))(x - x_k). && (2.12) \end{aligned}$$

Entonces, tendríamos:

$$x_{k+1} = x_k + s_k = x_k - (J(x_k)^T J(x_k))^{-1} J(x_k)^T r(x_k).$$

Y el algoritmo sería el siguiente:

1. Dados x_0, $\epsilon > 0$, $k := 0$.
2. If $\|g_k\| \leq \epsilon$, entonces acaba.
3. Resolver
$$J(x_k)^T J(x_k) s_k = -J(x_k)^T r(x_k) \text{ para } s_k. \qquad (2.13)$$
4. Calculamos $x_{k+1} = x_k + s_k$, $k := k + 1$. Y volvemos al segundo paso.

Donde 2.13 se conoce como ecuación de Gauss-Newton y se observa que la única diferencia con el método de Newton es que el término del Hessiano $G(x_k)$ es reemplazado por el término de primer orden $J(x_k)^T J(x_k)$.

Observemos que el paso tres de nuestro algoritmo se corresponde con las ecuaciones normales de un problema de mínimos cuadrados lineales. Además, el modelo 2.12 es equivalente a considerar

$$M_k = r(x_k) + J(x_k)(x - x_k) \qquad (2.14)$$

y resolver el problema de mínimos cuadrados lineal

$$\text{mín}\,\frac{1}{2}\|M_k(x)\|^2.$$

Estas obervaciones nos muestran que el método de Gauss-Newton es, de hecho, un método para linealizar problemas de mínimos cuadrados no lineales.

Sin embargo, este método también plantea problemas si $J(x)$ no tiene rango máximo y por ello se propone el algoritmo de Leveberg-Marquardt que utilizaremos en la calibración del acelerómetro.

2.6.1. Algoritmo de Levenberg-Marquardt

Para solventar los problemas que nos plantean con el método de Gauss-Newton se utiliza la ténica de «trust-region», considerando el modelo de «trust-region»

$$\begin{aligned} \text{mín} \quad & \tfrac{1}{2}\|J(x_k)(x-x_k)+r(x_k)\|_2^2 \\ s.a \quad & \|x-x_k\|_2 \leq \Delta_k, \end{aligned} \tag{2.15}$$

que puede ser reescrito como vemos a continuación

$$\begin{aligned} \text{mín} \quad & q_k(x) = \frac{1}{2}\|r_k\|^2 + r_k^T J(x_k)(x-x_k) + \frac{1}{2}(x-x_k)^T J(x_k)^T J(x_k)(x-x_k) \\ s.a \quad & \|x-x_k\|_2 \leq \Delta_k. \end{aligned}$$

Sea $s = x - x_k$. Entonces la solución del problema 2.15 se puede obtener resolviendo el problema

$$(J(x_k)^T J(x_k) + \mu_k I)s = -J(x_k)^T r(x_k),$$

luego,

$$x_{k+1} = x_k - (J(x_k)^T J(x_k) + \mu_k I)^{-1} J(x_k)^T r(x_k).$$

Cuando $\|(J(x_k)^T J(x_k))^{-1} J(x_k)^T r(x_k)\| \leq \Delta_k$, entonces $\mu_k = 0$ y la solución viene dada por s_k. En otro caso, existe $\mu_k > 0$ tal que la solución s_k satisface $\|s_k\| = \Delta_k$ y

$$(J(x_k)^T J(x_k) + \mu_k I)s = -J(x_k)^T r(x_k),$$

Así, obtendríamos el método de Levenberg-Marquard que utilizaremos en la calibración del acelerómetro y que se encuentra ya implementado en «Matlab».

Para más información se puede consultar [22],[23],[24].

2.7. Algoritmo de Nelder-Mead

El algoritmo de Nelder-Mead, que utilizaremos para la calibración del giroscopio, se publicó en 1965 y es uno de los algoritmos más utilizados para optimización multidimensional sin restricciones. Además, no necesita calcular derivadas lo que permite que se aplique a problemas que involucren funciones que no sean excesivamente suaves.

Este algoritmo está diseñado para resolver problemas de optimización sin restricciones para funciones no lineales, $f : \mathbb{R}^n \to \mathbb{R}$ y utiliza únicamente el valor de la función en algunos puntos de $\mathbb{R}^n$, sin intentar calcular el gradiente de la función en ninguno de esos puntos.

El método de Nelder-Mead se basa en simplex en $\mathbb{R}^n$ que se define como la envolvente convexa de $n+1$ vértices $x_0, \cdots, x_n \in \mathbb{R}^n$. Por ejemplo, un simplex en $\mathbb{R}^2$ sería un triángulo y en $\mathbb{R}^3$ un tetraedro.
Este tipo de métodos comienzan con un conjunto de $n+1$ puntos $x_0, \cdots, x_n \in \mathbb{R}^n$ que los consideramos como los vértices de nuestro simplex S y con los valores de la función en los vértices $f_j := f(x_j)$ para $j = 0, \cdots, n$. El simplex inicial ha de ser no degenerado, es decir, los puntos $x_0 \cdots, x_n$ no pueden estar en un mismo hiperplano.

Entonces el método consiste en realizar una serie de transformaciones sobre el simplex S haciendo que el valor de la función en los vértices vaya siendo más pequeño y termina cuando el simplex es lo suficientemente pequeño o cuando los valores f_j están lo sufcientemente cerca (suponiendo que f es continua).

Veámos a grandes rasgos cómo sería una iteración del algoritmo:

1. En primer lugar, hemos de determinar los índices h, s, l correspondientes al peor, al segundo peor y al mejor vértice, respectivamente, en el simplex S. Es decir,

$$f_h = max_j f_j, \quad f_s = max_{j \neq h} f_j, \quad f_l = min_{j \neq h} f_j.$$

2. A continuación hemos de calcular el centroide del mejor lugar, que será el opuesto al peor vértice x_h:

$$c := \frac{1}{n} \sum_{j \neq h} x_j.$$

3. Y por último construir un nuevo simplex a partir del actual.
Primero, intentamos reemplazar el peor vértice x_h por otro punto mejor utilizando ciertas transformaciones como reflexión, expansión o contracción con respecto al mejor lugar. Todos los puntos que se utilizan para testear la función han de estar en la recta determinada por x_h y por c y se calculan a lo sumo dos en cada iteración.
Si encontramos un punto mejor, se convierte en el nuevo vértice del simplex, en caso contrario, realizamos una transformación que consiste en reducir el simplex al mejor punto x_l y calculamos n puntos más.

   Las transformaciones que hemos planteado están controladas por ciertos parámetros: α para la reflexión, β para la contracción, γ para la expansión y δ para la reducción. Estos parámetros han de verificar las siguientes condiciones

$$\alpha > 0, \quad 0 < \beta < 1, \quad \gamma > 1, \quad \gamma > \alpha, \quad 0 < \gamma < 1.$$

   Y los valores que se suelen tomar son

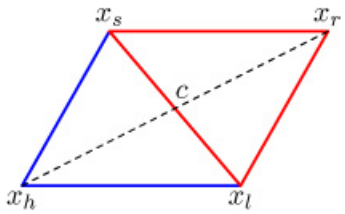
$$\alpha = 1, \quad \beta = \tfrac{1}{2}, \quad \gamma = 2, \quad \delta = \tfrac{1}{2}.$$

   Veámos finalmente cómo se realizan dichas transformaciones y mostraremos cómo se va transfomando el simplex, que aparecerá en color rojo en cada figura.

   - **Reflexión**: calculamos el punto de reflexión como

$$x_r := c + \alpha(c - x_h)$$

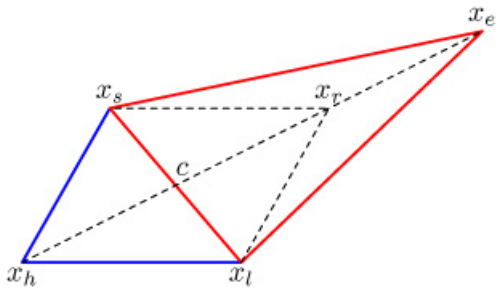
     y $f_r := f(x_r)$. Si $f_l < f_r < f_h$, nos quedamos con x_r y terminamos la iteración. El nuevo simplex quedaría como vemos en la siguiente figura:



- **Expansión**: si $f_r < f_l$, calculamos el punto de expansión de la siguiente manera

$$x_e = c + \gamma(x_r - c)$$

y $f_e = f(x_e)$. Si $f_e < f_r$, nos quedamos con x_e y terminamos la iteración, en caso contrario tomaremos x_r.
Veámos como queda el nuevo simplex



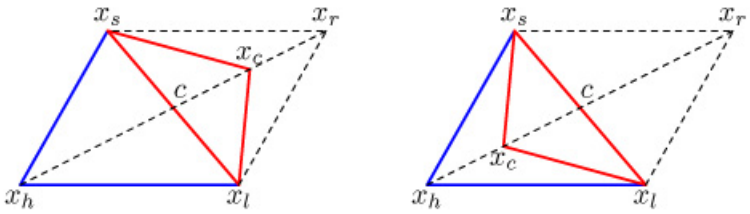
- **Contracción**: si $f_r \geq f_s$, hemos de calcular el punto de contracción, x_c, utilizando el punto x_h o x_r en función de cuál sea mejor
  - **Exterior**: Si $f_s \leq f_r < f_h$, calculamos

$$x_c := c + \beta(x_r - c) \;\; y \;\; f_c = f(x_c).$$

    Entonces si $f_c \leq f_r$, aceptamos x_c y terminamos la iteración.
  - **Interior**: si $f_r \geq f_h$, calculamos

$$x_c := c + \beta(x_h - c) \;\; y \;\; f_c = f(x_c).$$

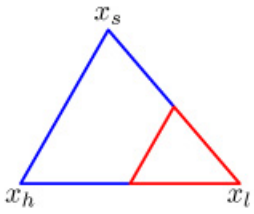
    Si $f_c < f_h$, aceptamos x_c y terminamos la iteración, en caso contrario hemos de aplicar la transformación de reducción que explicamos después de mostrar la figura de cómo varía el simplex tras aplicar las posibles transormaciones de contracción.



- **Reducción**: Calculamos n vértices nuevos:

$$x_c := c + \delta(x_j - x_l) \;\; y \;\; f_c = f(x_c)$$

para $j = 0, \cdots, n$ con $j \neq l$. Y así obtendríamos el siguiente simplex:



Este algoritmo lo encontramos ya implementado en *Matlab* bajo el comando «fminsearch» y será el algoritmo elegido en la calibración del giroscopio.

Capítulo 3

Calibración del acelerómetro

La idea fundamental utilizada para calibrar el acelerómetro consiste en que la magnitud de la aceleración cuando el acelerómetro está estático ha de ser igual a la gravedad terrestre, teniendo en cuenta que la gravedad es una magnitud muy estable.

Pasemos a describir nuestro problema de calibración del acelerómetro y desarrollar la idea fundamental que acabamos de plantear.

Para la formulación del problema hemos de definir varios sistemas de referencia. En primer lugar, el sistema de referencia del acelerómetro que denotaremos por AF y que aunque debería ser ortogonal normalmente no lo es. En segundo lugar, podemos definir un sistema de referencia ideal para el acelerómetro, que llamaremos AOF, ortogonal y al que imponemos la siguientes condiciones:

- El eje x del sistema AOF coincide con uno de los ejes del sistema del acelerómetro.
- El eje y del sistema AOF se encuentra en el plano generado por los ejes x e y del sistema del acelerómetro.
- Sobre el eje z no imponemos ninguna restricción.

Y finalmente, el sistema de referencia correspondiente al chásis de WIMU que es ortogonal, que denotaremos por BF y que generalmente difiere del sistema AF en pequeños ángulos.

Como hemos visto en la sección 2.4.1, si temenos una medición, a^s en un sistema no ortogonal (AF), lo podemos transformar al sistema de coordenadas ortogonal correspondiente al chásis de WIMU (BF) de la siguiente forma

$$s^B = Ts^s, \quad T = \begin{pmatrix} 1 & -\theta_{yz} & \theta_{zy} \\ \theta_{xz} & 1 & -\theta_{zx} \\ -\theta_{xy} & \theta_{yx} & 1 \end{pmatrix}, \tag{3.1}$$

donde s^B representa dicha medición el sistema BF.

Por otra parte, en nuestro algoritmo vamos a asumir que el sistema de coordenadas BF coincide con el ideal AOF definido anteriormente, de forma que por las condiciones que impusimos sobre AOF, θ_{xy}, θ_{xz} y θ_{yx} son cero. Entonces en el caso del acelerómetro la ecuación 3.1 nos quedaría como sigue

$$a^o = Ta^s, \quad T = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & \alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix},$$

donde hemos cambiado las θ del caso general a α para nuestro caso particular, a^o es la aceleración medida en el sistema de referencia AOF y a^s es la aceleración medida por nuestro acelerómetro.

Pero el acelerómetro no sólo se ve afectado por problemas de ortogonalidad de sus ejes, si no que también tiene errores de sesgo y de escala. Entonces si definimos la matriz de cambio de escala como

$$K = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

y el vector del error de sesgo como

$$b = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}.$$

Tendremos que

$$a^o = TK(a^s + b + v),$$

donde v es ruido blanco. Para poder eliminar el ruido blanco del modelo se decide promediar teniendo en cuenta que la esperanza de este tipo de error es cero como hemos visto en la sección 2.3.

Así, el modelo quedaría finalmente como

$$a^s = T^{-1}K^{-1}(a^o - b) = A(a^o - b), \tag{3.2}$$

donde $A = T^{-1}K^{-1} = (KT)^{-1}$ es también una matriz triangular superior.

De esta forma, para calibrar el acelerómetro necesitamos obtener el valor de los parámetros $\alpha_{yz}, \alpha_{zy}, \alpha_{xz}, s_x, s_y, s_z, b_x, b_y, b_z$.

Supongamos que tenemos $\{a_k^o\}_{k=1,\cdots,n}$ mediciones recogidas con el sensor estático y con diferentes orientaciones, es decir, en una situación en la que no hay aceleración. Entonces la aceleración de la gravedad debe ser constante, además, sin pérdida de generalidad podemos suponer que dicha aceleración g es unitaria, es decir, $\|g\| = 1$. Por tanto, la norma de g es independiente del sistema de referencia.
De acuerdo con el modelo planteado en (3.2), esta restricción la podemos reescribir de la siguiente manera:

$$(a_k^o - b)^T A_l (a_k^o - b) = 1, \ \ \forall k = 1 \cdots N, \tag{3.3}$$

con $A_l = A^T A \in \mathbb{R}^{3\times 3}$ simétrica y definida positiva.

Un punto importante de este método de calibración, [10], consiste en reconocer que la ecuación anterior, (3.3), es la ecuación de un elipsoide. De hecho, la ecuación ímplicita general de un elipsoide tridimensional viene dada por:

$$F(\bar{A}_l, c; x) = (x - c)^T \bar{A}_l (x - c) - 1 = 0, \tag{3.4}$$

donde c representa las coordenadas del centro del elipsoide y $\bar{A}_l \in \mathbb{R}^{3\times 3}$ es una matriz simétrica y definida positiva.
Como consecuencia, cada una de las mediciones a_k^o deben estar en la superficie del elipsoide cuyo centro viene dado por el error de sesgo, b, luego estamos ante un problema de ajuste elipsoidal. Para encontrar el mejor ajuste minimizaremos la función no lineal:

$$\sum_{k=1}^{N} F(A_l, b; a_k^o)^2$$

que podemos reformular como

$$\sum_{k=1}^{N} \left(1 - \|A(a_k^o - b)\|\right)^2.$$

Tal y como se definió en la sección 2.9, estamos ante un ajuste por mínimos cuadrados no lineal.
El gran inconveniente al que nos enfrentamos en este tipo de problemas es la presencia de numerosos mínimos locales que no son mínimos globales y que, con nuestros datos experimentales, se vuelve especialmente conflictivo. Es por ello que se decide utilizar la técnica de regularizar nuestra función objetivo como se muestra en la sección 2.9.

Luego añadiremos un factor de regularización de forma que obliguemos a los valores de la matriz T que sean cercanos a 0, los valores de la escala que sean positivos y cercanos a 1 y los valores de sesgo también cercanos a 0.
Es claro que este factor de regularización perderá peso cuando los datos sean más irregulares.

Con todo lo planteado, si notamos por θ al conjunto de parámetros que hemos de obtener, la función que se debe minimizar es la siguiente:

$$\begin{aligned} L(\theta,\lambda) &= \sum_{k=1}^{M} \Big\| \|g\| - \|T^a k^a (a_k^s + b)\| \Big\|^2 \\ &\quad +\lambda(\alpha_{zy}^2 + \alpha_{yz}^2 + \alpha_{xz}^2 + (s_x - 1)^2 \\ &\quad +(s_y - 1)^2 + (s_z - 1)^2 + b_x^2 + b_y^2 + b_z^2). \end{aligned}$$

Donde λ es el parámetro de regularización cuyo valor calcularemos en la sección 5. El algoritmo utilizado para la minimización será el algoritmo de Levenberg-Marquardt presentado en la sección 2.6. Este algoritmo se encuentra implementado en *Matlab*. Lo utilizaremos a través del comando *lsqnonlin* e indicándolo de forma explícita en las opciones.

3.1. Detección de periodos estáticos y búsqueda de similitudes entre bloques parados

Para nuestro método de calibración hemos de ser capaces de detectar los periodos en los que WIMU está estático.

Para ello, hemos construido un detector que nos permitirá conocer los periodos de tiempo en los que la WIMU se encuentre estático y en los que esté en movimiento y almacenar los datos correspondientes.

Este detector se va a basar en la señal emitida por el acelerómetro durante un tiempo de t_w segundos y en el cálculo de su varianza

$$\varsigma(t_w) = \sqrt{[var_{t_w}(a_x)]^2 + [var_{t_w}(a_y)]^2\,[var_{t_w}(a_z)]^2},$$

donde a_x, a_y, a_z son las aceleraciones medidas durante ese tiempo en los ejes x, y, z respectivamente y $var_{t_w}(a^t)$ es un operador que calcula la varianza de una señal a^t en un intervalo de tiempo de longitud t_w segundos centrados en t.

Entonces diremos que el dispositivo está parado o en movimiento si $\varsigma(t_w)^2$ es mayor o menor que una cierta cota. Esta cota la obtendremos calculando $\varsigma(t_w)^2$ en un periodo de tiempo inicial, en el que tenemos asegurado que la WIMU está estático. Es por ello, por lo que es necesario que al encender el dispositivo permanezca estático durante aproximadamente dos segundos.

Por tanto nuestra cota será $\varsigma(t_w)^2$ en un periodo inicial multiplicado por un entero que, de acuerdo a nuestros datos, hemos elegido como 2.

De esta forma, vamos a poder conocer los momentos en los que la WIMU esté parado y durante cuánto tiempo lo está. Y esta será la información necesaria para proceder a calibrar nuestro sensor.
Cuando los momentos de parada sean muy cortos, en nuestro caso menores a 100 datos, los desecharemos y cuando sean mayores los pasaremos a nuestra función para obtener los parámetros que nos permitirán calibrar todos los datos.

La detección de similitudes entre bloques consecutivos se ha realizado para solventar los problemas que se pudieran plantear en circunstancias como las que se presentan a continuación:

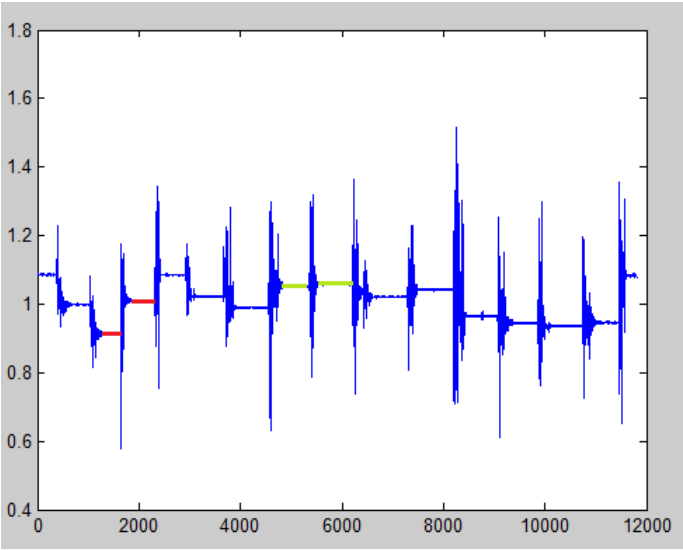


Figura 3.1: Longitud del vector aceleración.

Como podemos observar en esta figura, algunos bloques de parada difieren mucho de otros y sin embargo otros son bastante similares. Por tanto buscaremos aquellos que sean similares y consecutivos para realizar nuestra calibración y los que sean diferentes se calibrarán de forma independiente.

Para ello, diremos que dos bloques son similares si al realizar la media de todos los datos de aceleración de dos bloques consecutivos y calcular el módulo del vector resultante difieren en menos de una cierta cota que, de acuerdo a nuestros datos, hemos elegido 0.02.

3.2. Algoritmo

La calibración del acelerómetro la hemos estructurado en distintas funciones: detección y búsqueda de similitudes de bloques parados, selección de datos, construcción de la función objetivo, minimización y calibración. Pasemos a describirlos uno a uno:

1. **Detección de periodos estáticos y búsqueda de similitudes entre bloques parados**

   Como hemos dicho con anterioridad, en el algoritmo que se presenta tiene especial importancia la detección de momentos estáticos y momentos en movimiento. Esta detección se realizará estableciendo una cota a partir de la variabilidad que muestra el sensor que se necesita calibrar en los datos proporcionados en un periodo inicial en el que ha de estar en reposo.

   Una vez detectados los periodos estáticos y de movimiento, se tomarán los bloques estáticos que consideremos suficientemente largos, en nuestro caso de más de cien datos, y comprobaremos si los bloques son similares o no.

   Esta comprobación se realizará calculando la media de las longitudes de los vectores en los distintos bloques parados y viendo si la diferencia entre dos bloques contiguos es superior a un cierta cota, que hemos elegido como 0.02.

2. **Selección de datos**

   La selección de datos la iremos haciendo mientras los bloques de parada vayan siendo similares, en el momento que difieran más de lo permitido se procederá a la calibración con los datos selecionados hasta el

momento y posteriormente volveremos a seleccionar los datos necesarios. El proceso de selección consiste en escoger dentro de cada bloque estático, que como hemos dicho anteriormente ha de ser de más de 100 datos, 99 datos (la longitud mínima del bloque menos uno).

La forma de elegir los datos será a partir del primero del bloque, iremos recorriendo el vector de datos y escogeremos los datos de forma que guardemos el dato que se encuentre en la posición dada por el contador que recorre el vector multiplicado por un paso. Este paso lo tomaremos como el entero más cercano a la longitud del bloque dividido por 100.

3. **Construcción de la función objetivo y minimización**

Una vez que ya tenemos los datos seleccionados de bloques estáticos similares pasamos a la obtención de los parámetros de las matrices T, K y b.

Para ello, construimos la función descrita en la sección 3

$$\begin{aligned} L(\theta,\lambda) &= \sum_{k=1}^{M} \Big\| \|g\| - \|T^a k^a (a_k^s + b)\| \Big\|^2 \\ &\quad +\lambda(\alpha_{zy}^2 + \alpha_{yz}^2 + \alpha_{xz}^2 + (s_x-1)^2 + (s_y-1)^2 \\ &\quad +(s_z-1)^2 + b_x^2 + b_y^2 + b_z^2), \end{aligned}$$

donde M es el número de datos seleccionados dividido entre diez, pues los datos seleccionados los promediamos de diez en diez con el objetivo de eliminar el ruido blanco que nos aparecía en la ecuación del error.

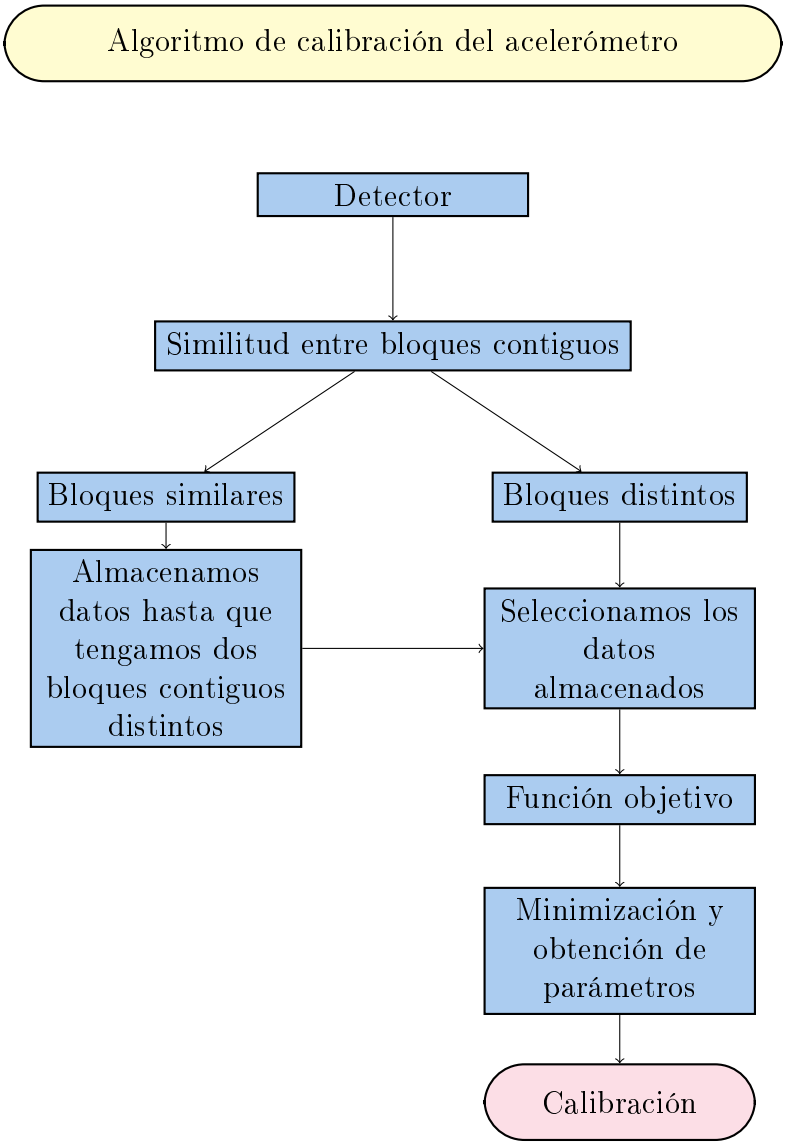
Obtendremos los parámetros buscados introduciendo dicha función en el algoritmo de minimización de Leverberg-Marquardt implementado en *Matlab*, con valores iniciales $[0,0,0,1,1,1,0,0,0]$ donde los ceros se corresponden con los parámetros de la matriz T, que nos daba el cambio de sistema de referencia y al vector b que contenía los errores de sesgo, y los unos se corresponden con los parámetros de la matriz de escala.

4. **Calibración de los datos**

Una vez que ya tenemos calculados los parámetros de las matrices T, K y b procedemos a la calibración de los datos hasta el momento en el que hemos encontrado una diferencia mayor que la permitida entre los bloques estáticos.

Esta calibración la haremos multiplicando por las matrices T, K los datos correspondientes al inicio del primer bloque parado hasta el último dato del último bloque parado que eran similares y restándoles el error de sesgo b.

En los tramos de transición entre dos bloques diferentes multiplicaremos por la media entre las matrices T, K del bloque anterior y el bloque nuevo y restándole también la media entre los errores de sesgo correspondientes a los dos bloques.


Algoritmo de calibración del acelerómetro
Detector
Similitud entre bloques contiguos
Bloques similares
Bloques distintos
Almacenamos datos hasta que tengamos dos bloques contiguos distintos
Seleccionamos los datos almacenados
Función objetivo
Minimización y obtención de parámetros
Calibración

Capítulo 4

Calibración del giroscopio

La idea fundamental utilizada para la calibración del giroscopio consiste en que el vector gravedad proporcionado por el acelerómetro calibrado ha de ser el mismo que el vector de gravedad calculado al integrar las velocidades angulares proporcionadas por el giroscopio.

Pasemos a describir formalmente nuestro problema y desarrollar la idea que acabamos de plantear.

En el caso del giroscopio tenemos de nuevo un sistema de referencia no ortogonal, GF, correspondiente al sensor, un sistema de referencia ideal ortogonal, GOF y el sistema de referencia del chásis que claramente es el mismo que para el acelerómetro BF.

Para la calibración del giroscopio haremos uso de los datos calibrados del acelerómetro, por tanto, necesitamos que ambos sensores compartan el mismo sistema de referencia, que será el sistema AOF. Por tanto, haremos el cambio de sistema de referencia, presentado en la sección 2.4.1, de un sistema de referencia no ortogonal a uno ortogonal.

Al igual que el acelerómetro, el giroscopio también se ve afectado por errores de sesgo y de escala, por tanto, el modelo de error del sensor se puede plantear de forma análoga al del acelerómetro como

$$\omega^o = T^g K^g(\omega^s + b^g + v^g),$$

donde ω^o denotan los datos tomados por el giroscopio, ω^s correponde a los valores sin errores que deberíamos tener, T^g es la matriz de cambio de

sistema de referencia que ortognaliza el sistema GF, K^g es la matriz de escala, b^g el vector que contiene los errores de sesgo y v^g es ruido blanco que podremos despreciar realizando ciertos promedios.
Pero promediando, no sólo eliminaremos el ruido blanco si no que supondremos que también podemos desechar el error de sesgo restándole a todos los datos la media de los datos durante un bloque estático suficientemente largo.

Como hemos dicho en la idea fundamental, hemos de ser capaces de calcular la orientación a partir de las velocidades angulares proporcionadas por el giroscopio. Para ello definimos Ψ como un operador que toma una secuencia de n lecturas del giroscopio ω_i^o y un vector inicial de gravedad, $u_{0,k-1}$, proporcionado por el acelerómetro calibrado y devuelve el vector de gravedad final $u_{g,k}$ calculado utilizando las mediciones del giroscopio entre el $(k-1)$-ésimo y el k-ésimo intervalo estático.

$$u_{g,k} = \Psi[\omega_i^o, u_{0,k-1}],$$

donde Ψ representa algún algoritmo de integración que calcula la orientación integrando las velocidades angulares. En este caso, la orientación vendrá representada por cuaterniones, de forma que en cada paso i, el cuaternión q_i está relacionado con el cuaternión del paso anterior, q_{i-1} por la siguiente ecuación:

$$q_i = \left[cos(0{,}5\,|\delta\beta|)I + \frac{1}{|\delta\beta|} sen(0{,}5\,|\delta\beta|)B\right] q_{i-1},$$

donde $\delta\beta = \omega_i^o$, $|\delta\beta| = \sqrt{\delta\beta_x^2 + \delta\beta_y^2 + \delta\beta_z^2}$,

$$B = \begin{pmatrix} 0 & \delta\beta_x & \delta\beta_y & \delta\beta_z \\ -\delta\beta_x & 0 & \delta\beta_z & -\delta\beta_y \\ -\delta\beta_y & -\delta\beta_x & 0 & \delta\beta_x \\ -\delta\beta_z & \delta\beta_y & -\delta\beta_x & 0 \end{pmatrix}.$$

e I es la matriz identidad de tamaño 4×4. Así, podemos utilizar el cuaternión anterior para calcular el cuaternión actual. Utilizando el cuaternión inicial $q_{0,k-1}$ correspondiente al último dato de cada bloque parado $k-1$ (para $k \geq 2$) y la secuencia de mediciones tomadas del giroscopio ω_i^o para $i = 1, \cdots, n$, se puede obtener el cuaternión q_n en el paso $i = n$ y por tanto, la matriz de rotación correspondiente a q_n (es decir, estamos aplicando al vector proporcionado por el giroscopio el teorema demostrado en la sección 2.4.4).

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + 1_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}.$$

Entonces tendremos que

$$u_{g,k} = Ru_{0,k-1}.$$

Y los parámetros correspondientes a las matrices de T^g y K^g los obtendremos minimizando la siguiente ecuación

$$\sum_{k=2}^{m-1} \|u_{a,k} - u_{g,k}\|,$$

donde $u_{a,k}$ es el primer dato proporcionado por el acelerómetro en el bloque parado k. En caso de que no hubiera errores en el sensor ni en el algoritmo de integración tendríamos que $\|u_{a,k} - u_{g,k}\| = 0$.

4.1. Algoritmo

En la calibración del giroscopio, al igual que en el acelerómetro, hemos utilizado distintas funciones: eliminar error de sesgo, función objetivo que necesita de las funciones algoritmo de integración Runge-Kutta de orden cuatro, transformación del vector dado por el giroscopio a cuaterniones y construcción de la matriz de rotación, minimización y por último calibración.

1. **Eliminar error de sesgo**

   Esta parte consiste únicamente en calcular la media en cada eje de las velocidades angulares proporcionadas por el giroscopio durante el primer bloque parado suficientemente grande, que detectamos en la parte de calibración del acelerómetro con la función «detector», y restar el resultado a todas las salidas proporcionadas por el giroscopio (en cada eje la media de los datos correspondiente al mismo).

2. **Función objetivo**

   Esta función es la que nos permite calcular los parámetros de las matrices T^g y K^g y consite en construir la función

$$\sum_{k=2}^{m-1} \|u_{a,k} - u_{g,k}\|$$

vista en la sección anterior para la que utilizaremos los datos del acelerómetro calibrados anteriormente. Pero como hemos dicho, para poder construirla hemos de hacer uso de otras funciones que presentamos a continuación:

- **Transformación del vector dado por el giroscopio a cuaterniones**

  Para evitar singularidades y para disminuir el coste computacional la integración Runge-Kutta de cuarto orden la aplicaremos sobre cuaterniones que representen las velocidades angulares dadas por el giroscopio. Para ello creamos una función que transforme el vector correspondiente al último dato de cada bloque parado en un cuaternión. Utilizando, como vimos anteriormente, la siguiente transformación

$$q_i = \left[cos(0{,}5\,|\delta\beta|)I + \frac{1}{|\delta\beta|} sen(0{,}5\,|\delta\beta|)B \right] q_{i-1},$$

  donde $\delta\beta = \omega_i^o$, $|\delta\beta| = \sqrt{\delta\beta_x^2 + \delta\beta_y^2 + \delta\beta_z^2}$ y

$$B = \begin{pmatrix} 0 & \delta\beta_x & \delta\beta_y & \delta\beta_z \\ -\delta\beta_x & 0 & \delta\beta_z & -\delta\beta_y \\ -\delta\beta_y & -\delta\beta_x & 0 & \delta\beta_x \\ -\delta\beta_z & \delta\beta_y & -\delta\beta_x & 0 \end{pmatrix}.$$

- **Integración utilizando el método de Runge-kutta de cuarto orden**

  Veámos cómo es este algoritmo de integración para la función

$$f(\mathbf{q}, t) = \frac{1}{2} B\mathbf{q}$$

  que necesitamos integrar, donde B es la matriz definida en el apartado anterior. Los pasos son los siguientes:

$$\begin{array}{rcll} \mathbf{q}_{k+1} & = & \mathbf{q}_k + \Delta t \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4), & \\ K_i & = & f(\mathbf{q}^{(i)}, t_k + c_i \Delta t), & \\ \mathbf{q}^{(i)} & = & \mathbf{q}_k, & \text{para } i = 1, \\ \mathbf{q}^{(i)} & = & \mathbf{q}_k \Delta t \sum_{j=1}^{i-1} a_{ij} K_j, & \text{para } i > 1, \end{array}$$

donde los coeficientes vienen dados por

$$\begin{array}{llll} c_1 = 0, & c_2 = \frac{1}{2}, & c_3 = \frac{1}{2}, & c_4 = 1, \\ a_{21} = \frac{1}{2}, & a_{31} = 0, & a_{41} = 0, & \\ a_{32} = \frac{1}{2}, & a_{42} = 0, & & \\ a_{43} = 1. & & & \end{array}$$

Finalmente, normalizamos cada uno de los cuaterniones, es decir,

$$\mathbf{q}_{k+1} = \frac{\mathbf{q}_{k+1}}{\|\mathbf{q}_{k+1}\|}.$$

- **Construcción de la matriz de rotación**

  Consiste en calcular la matriz de rotación que nos proporciona la información necesaria acerca de cómo ha cambiado la orientación desde el último dato proporcionado por el giroscopio en el bloque parado $k-1$ hasta el inicio del siguiente bloque estático.

  Esto lo haremos a partir del último cuaternión calculado por el algoritmo de integración y expresándolo como matriz de rotación de la siguiente forma, si $\mathbf{q}_n = [q_0\, q_1\, q_2\, q_3]$ entonces

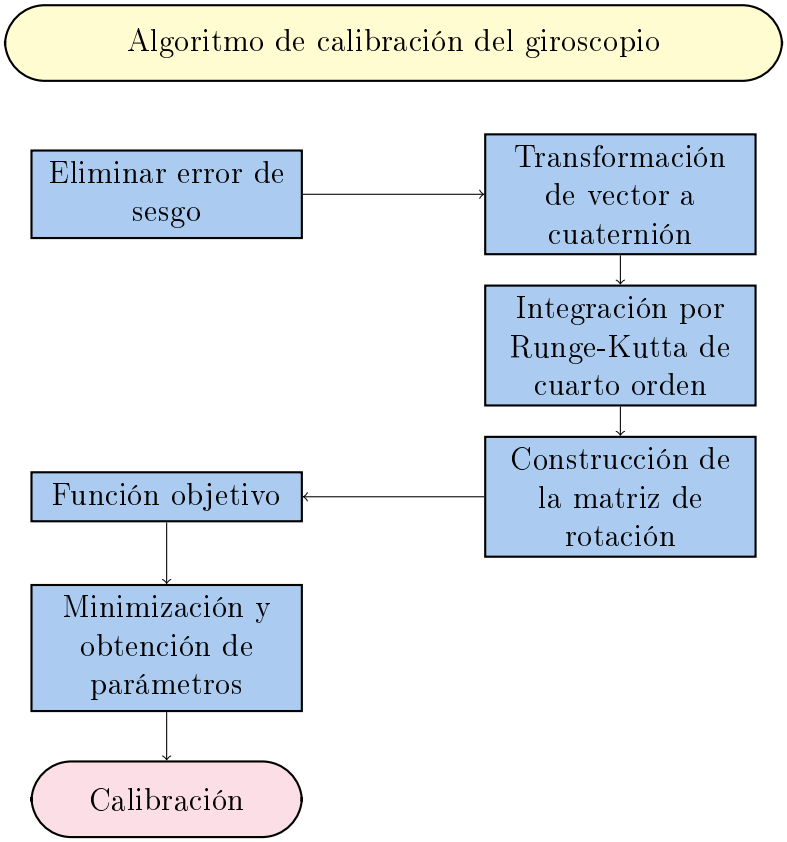
$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + 1_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}.$$

3. **Minimización**

   Una vez calculada la función objetivo, obtenemos los parámetros correspondientes a las matrices T^g y K^g aplicando el algoritmo de minimización implementado en «Matlab» dado por el comando «fminsearch» y a partir del valor inicial $[1, 0, 0, 0, 1, 0, 0, 0, 1]$ donde los ceros se corresponden a los parámetros iniciales de la matriz de ortogonalización y los unos con los parámetros iniciales de la matriz de escala.

4. **Calibración**

   Por último multiplicamos las matrices T^g y K^g por los datos proporcionados por el giroscopio quitándole el error de sesgo.


Algoritmo de calibración del giroscopio
Eliminar error de sesgo
Transformación de vector a cuaternión
Integración por Runge-Kutta de cuarto orden
Construcción de la matriz de rotación
Función objetivo
Minimización y obtención de parámetros
Calibración

Capítulo 5

Resultados experimentales

En esta sección mostraremos las distintas pruebas experimentales que se han realizado y las conclusiones obtenidas.
En primer lugar, para la calibración del acelerómetro podemos distinguir cuatro partes fundamentales: el correcto funcionamiento del detector de periodos estáticos, estimación del parámetro de regularización y calibración de datos sintéticos y datos experimentales.
Procedamos a describir cada uno de los experimentos:

- **Correcto funcionamiento del detector de periodos estáticos**

  Para la realización de este experimento hemos utilizado datos de *WIMU* proporcionados por Realtrack Systems.
  Para ver los resultados se mostrarán a continuación dos gráficas correspondientes a dos conjuntos de datos diferentes. En ambas, se observan las longitudes de los vectores de aceleración durante toda la sesión (gráfica que veremos en color azul) y aparecerá en color rojo los tramos en el que el detector considera que el sensor está en reposo.

  El primer experimento se realiza colocando en la espalda de una persona el dispositivo *WIMU*. Esta persona realiza saltos y paradas, siempre con un periodo estático inicial de dos segundos aproximadamente. Los resultados obtenidos son los siguientes:

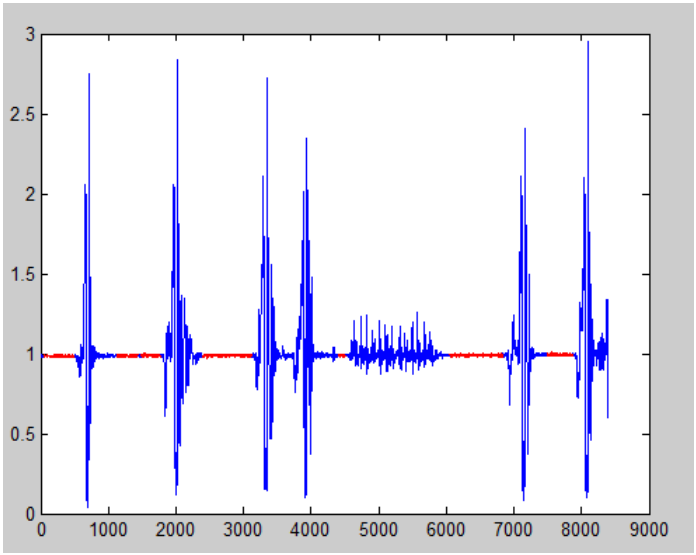


Figura 5.1: Detección de periodos estáticos.

El siguiente experimento se ha realizado colocando a *WIMU* sobre cada una de sus caras y aristas y obtenemos lo siguiente:

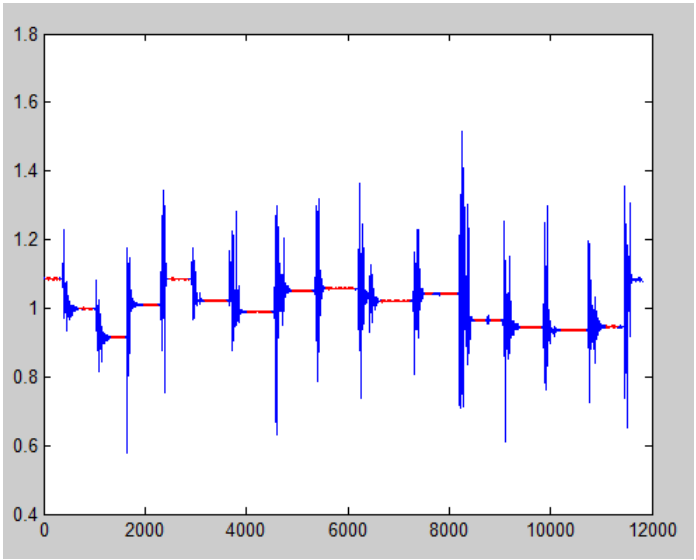


Figura 5.2: Detección de periodos estáticos.

Observamos que el detector actúa de forma satisfactoria, pues como se observa en las figuras 5.1, 5.2 las zonas coloreadas en rojo coinciden con los momentos en los que el módulo de los vectores de aceleración permanecen constantes.

- **Estimación del parámetro de regularización. Método L-Curve**

Hay distintos criterios para obtener el parámetro de regularización óptimo, aunque no siempre es posible. En nuestro caso, hemos utilizado el criterio L-Curve que consiste en una representación gráfica de la norma de la solución regularizada, $\|\theta\|$, frente a la norma nuestra función $\|L(\theta, \lambda)\|$ para distintos valores de λ.

Esta representación normalmente tiene forma de L, con un vértice diferenciado que separa la parte vertical de la horizontal de la curva y se corresponderá con el parámetro óptimo. Para hacer este vértice más prominente se suele realizar la gráfica en escala logarítmica.
Este vértice separa la parte superior (más vertical) correspondiente a valores pequeños del parámetro λ, de la parte inferior (más horizontal) correspondientes a valores más grandes de dicho parámetro.

En nuestro caso el vértice es ya visible en la escala usual, como vemos a continuación

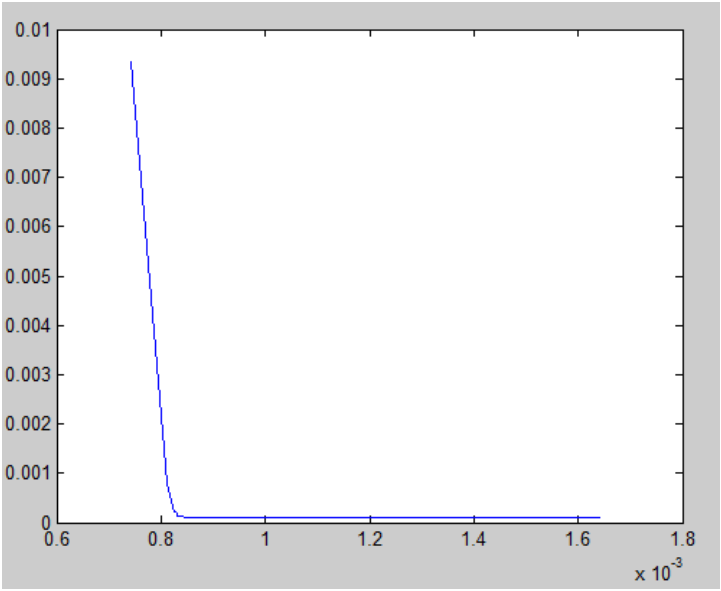


Figura 5.3: L-curve.

Aunque lo podemos detectar de fomar mucho más evidente utilizando la escala logarítmica

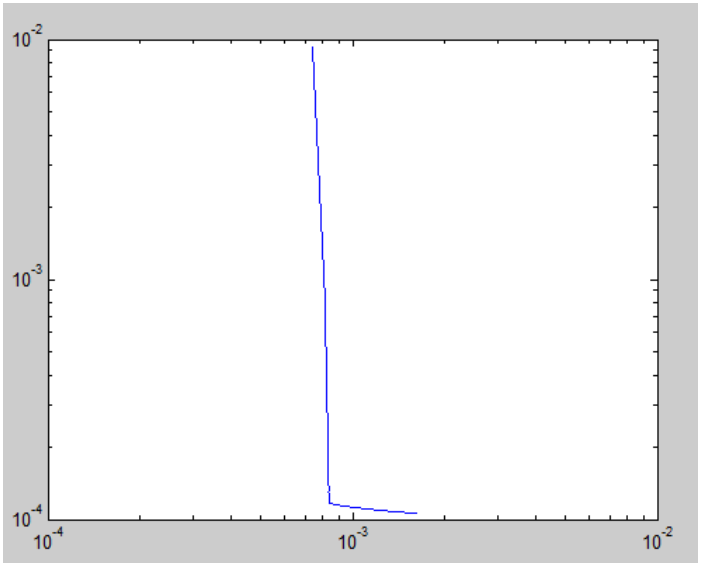


Figura 5.4: L-curve en escala logarítmica.

Así, en nuestro caso obtendremos que el valor óptimo de λ es 0,061.

- **Calibración de datos sintéticos**

  Para esta parte del experimento creamos unos datos sintéticos y le aplicamos el algoritmo de calibración propuesto.

  Los datos los generamos de la siguiente forma:

  1. Generamos una matriz de tres filas (que corresponderían a cada eje) y las columnas que consideremos necesarias en función del número de datos que queramos crear, por ejemplo, doce columnas. Esta matriz estará formada por números aleatorios que acto seguido normalizaremos. Es decir, dividiremos el valor de cada fila entre la norma del vector correspondiente.
  2. A continuación hacemos repeticiones de cada una de las columnas, el número de repeticiones dependerá de nuevo de la cantidad de datos que queramos obtener, podemos utilizar, por ejemplo, cien repeticiones de cada columna.
  3. Posteriormente rotaremos los datos, les cambiaremos la escala y le añadiremos un error de sesgo que luego al aplicar el algoritmo de calibración trataremos de recuperar.

Las matrices que se han utilizado para realizar estas transformaciones son:

- Matriz de rotación:

$$R = \begin{pmatrix} 1 & 0{,}2 & 0{,}4 \\ 0 & 1 & 0{,}3 \\ 0 & 0 & 1 \end{pmatrix}.$$

- Matriz de escala

$$E = \begin{pmatrix} 1{,}2 & 0 & 0 \\ 0 & 0{,}8 & 0 \\ 0 & 0 & 1{,}3 \end{pmatrix}.$$

- Vector de sesgo

$$B = \begin{pmatrix} 0{,}5 \\ -0{,}3 \\ 0{,}2 \end{pmatrix}.$$

4. Aplicamos las transformaciones a los datos obtenidos tras las repeticiones. Si llamamos estos datos como *repetidos* y los datos finales como *sintéticos* las transformación se aplicaría de la siguiente forma:

$$sintéticos = (R * E)^{-1} * (repetidos + B).$$

5. Finalmente le añadimos a los datos creados un ruido blanco de en torno al 10 %.

Una vez tenemos los datos aplicamos el algoritmo de calibración y los obtenemos:

- Matriz de rotación:

$$R = \begin{pmatrix} 1 & 0{,}1531 & 0{,}3605 \\ 0 & 1 & 0{,}30735 \\ 0 & 0 & 1 \end{pmatrix}.$$

- Matriz de escala

$$E = \begin{pmatrix} 1{,}0485 & 0 & 0 \\ 0 & 0{,}8999 & 0 \\ 0 & 0 & 1{,}1444 \end{pmatrix}.$$

- Vector de sesgo

$$B = \begin{pmatrix} 0{,}4782 \\ -0{,}2877 \\ 0{,}2158 \end{pmatrix}.$$

- **Calibración de datos experimentales**

  Los datos experimentales a los cuales aplicaremos la calibración coincidirán con los datos utilizados en el detector de periodos estáticos. En este caso mostraremos gráficamente los resultados, puesto que no podemos hacerlo de forma numérica.

  En el primer experimento, los datos recogidos son considerablemente buenos pues la aceleración afecta sólo a un eje. Por tanto, la mejoría se aprecia levemente. La longitud de los vectores de aceleración de los datos sin calibrar tienen la siguiente representación:

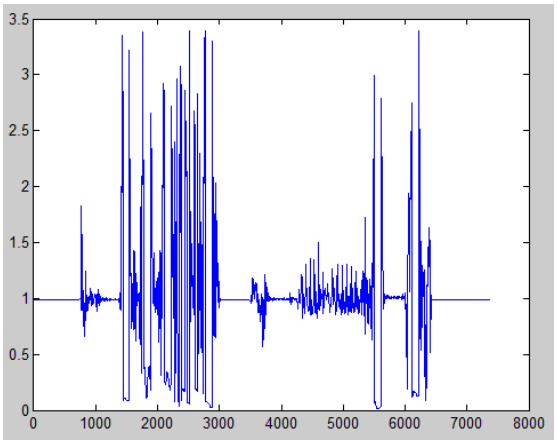


Figura 5.5: Módulo de los vectores de aceleración sin calibrar.

El resultado tras la calibración es

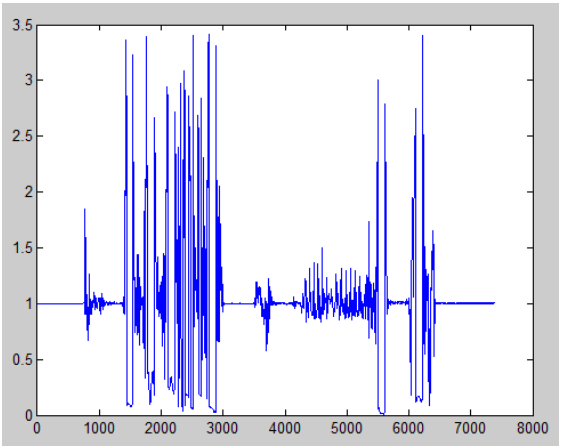


Figura 5.6: Módulo de los vectores de aceleración calibrados.

En el caso en el que se coloca el dispositivo sobre cada una de sus caras y aristas la mejoría es más evidente.

En el caso de los datos sin calibrar obtenemos

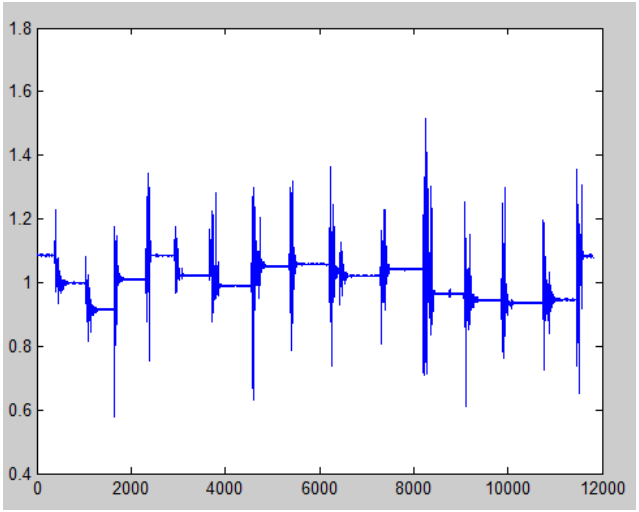


Figura 5.7: Módulo de los vectores de aceleración sin calibrar.

Y tras aplicar el algoritmo de calibración:

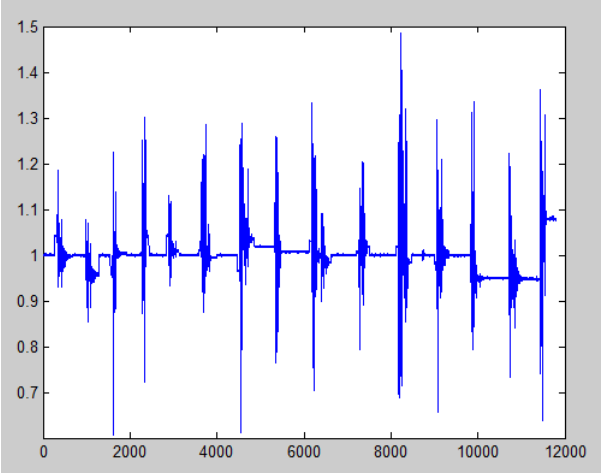


Figura 5.8: Módulo de los vectores de aceleración calibrados.

Podemos observar que en los momentos en los que el dispositivo está parado el módulo es mucho más consistente y cercano al valor de la fuerza de gravedad (que recordemos tomamos como 1).

En el caso de la calibración del giroscopio no hemos obtenido una mejora considerable respecto a los datos proporcionados por el sensor. Esto se debe a que este sensor se encuentra poco descalibrado y que nuestro algortimo introduce pequeños errores por la utilización del algoritmo de integración Runge-Kutta de cuarto orden.

Capítulo 6

Líneas de trabajo futuro

El problema principal al que nos gustaría enfrentarnos es la implementación del algoritmo de calibración en *WIMU*, que está en periodo de desarrollo, y su funcionamiento de forma autonóma y en tiempo real. Es decir, el objetivo es que el dispositivo se calibre de forma automática cuando sea necesario, impidiendo que los errores de calibración se acumulen.
Por supuesto, intentando que la calibración se haga de forma eficiente y sin que suponga un gran coste computacional.
Además, nos gustaría mejorar el algoritmo de Levenberg-Marquardt que en ciertos momentos no encuentra bien el mínimo global por la existencia de numerosos mínimos locales. Si esta medida se llevara a cabo nos permitiría eliminar la regularización.

Respecto a la calibración del giroscopio nos gustaría aplicar el algoritmo a giroscopios cuyos errores de calibración sean mayores y además mejorar el algoritmo de minimización utilizado. La necesidad de esta última mejora vendría dada por la lentitud que presenta en algunas ocasiones para encontrar el mínimo.

Y finalmente, y uno de los objetivos de mayor interés es, la fusión de la información proporcionada por el *GPS* y la proporcionada por los sensores que constituyen *IMU*, es decir, la *actitud*.
La necesidad de esta fusión se debe a que en interiores o en lugares con muchas edificaciones altas el *GPS* no es una fuente de información fiable y es, cuando debe tener un mayor papel la *actitud*.
Una de las ideas que se podría utilizar para realizar esta fusión, la podemos ver en [25]. En este artículo se plantea la utilización de un filtro de Kalman que permita integrar la información de varios sensores del mismo tipo (lo que sería especialmente adecuado en *WIMU* que integra cuatro acelerómetros)

y fusionarla con la información dada por el *GPS*. Esta fusión se haría en función de que tenga más credibilidad el *GPS*, la *actitud* o ambos por igual. Esta «credibilidad» vendrá determinada por una función *fuzzy*.

Aunque como se ha dicho podría ser una opción, habría que buscar distintas opciones y buscar aquella que se adecue más al dispositivo que estamos tratando y que sea más óptima en cuanto a las necesidades de la empresa, Realtrack Systems.

Apéndice A

Código Matlab

A.1. Calibración del acelerómetro y del giroscopio

```
1
2  %%%%%CALIBRACIÓN ACELERÓMETRO %%%%%
3  giro=importdata('F:\Realtracksystems\Funciones\
       caraaristas.mat');
4  datos=giro';
5  total_sample=length(datos(1,:));
6  a_xp = datos(1,:);
7  a_yp = datos(2,:);
8  a_zp = datos(3,:);
9   signal = [a_xp;a_yp; a_zp];
10 [matriz_intervalo_estatico,cambios_longitud]=detector(
       a_xp,a_yp,a_zp,total_sample);%Matriz que indica los
       bloques parados de más de 100 datos
11
12 todo_calibrado=[];
13 datos_para_calibrar=[];
14 last_calibrated=1;
15 parametros=[];
16 for i=1:length(cambios_longitud)-1
17      selected_data=seleccion(i,signal,
       matriz_intervalo_estatico);
18      datos_para_calibrar=[datos_para_calibrar,
```

```
    selected_data];
19    if (cambios_longitud(i)==1&i==1)|(cambios_longitud
    (1:i-1)==0&cambios_longitud(i)==1)
20    [T,K,b,vectorTK]=calibracion(datos_para_calibrar);
21        principio_calibrado=T*K*datos(1:3,
    matriz_intervalo_estatico(1,1):
    matriz_intervalo_estatico(i,2))...
22            -(diag(b)*ones(3,length(a_xp(
    matriz_intervalo_estatico(1,1):
    matriz_intervalo_estatico(i,2)))));
23      last_calibrated =i+1;
24      datos_para_calibrar=[];
25      todo_calibrado=[todo_calibrado,
    principio_calibrado];
26      parametros=[parametros,vectorTK];
27
28    elseif (cambios_longitud(i)==1&i>1)&(sum(
    cambios_longitud(1:i-1))>0)
29        Tanterior=T;
30        Kanterior=K;
31        banterior=b;
32      fin_calibracion = i+1;
33      [T,K,b,vectorTK]=calibracion(datos_para_calibrar)
    ;
34      centro_calibrado=(Tanterior*Kanterior+T*K)/2*
    datos(1:3,matriz_intervalo_estatico(last_calibrated
    -1,2):matriz_intervalo_estatico(last_calibrated,1))
    ...
35            -(diag((b+banterior)/2)*ones(3,length(a_xp(
    matriz_intervalo_estatico(last_calibrated-1,2):
    matriz_intervalo_estatico(last_calibrated,1)))));
36      bloques_calibrados=T*K*datos(1:3,
    matriz_intervalo_estatico(last_calibrated,1):
    matriz_intervalo_estatico(fin_calibracion-1,2))...
37            -(diag(b)*ones(3,length(a_xp(
    matriz_intervalo_estatico(last_calibrated,1):
    matriz_intervalo_estatico(fin_calibracion-1,2)))));
38
39      datos_para_calibrar=[];
40      last_calibrated = fin_calibracion;
41      todo_calibrado=[todo_calibrado,centro_calibrado,
```

```
    bloques_calibrados];
42          parametros=[parametros,vectorTK];
43
44        end
45
46    end
47
48  if   (cambios_longitud(1:(length(cambios_longitud)))==0)
49         [T,K,b,vectorTK]=calibracion(datos_para_calibrar);
50       todo_calibrado=T*K*datos(1:3,1:length(a_xp))-(diag(
    b)*ones(3,length(a_xp)));
51         parametros=[parametros,vectorTK];
52  else
53      [T,K,b,vectorTK]=calibracion(datos_para_calibrar);
54          final_calibrado=T*K*datos(1:3,
    matriz_intervalo_estatico(last_calibrated-1,2):
    length(a_xp))...
55              -(diag(b)*ones(3,length(a_xp(
    matriz_intervalo_estatico(last_calibrated-1,2):
    length(a_xp)))));
56    todo_calibrado=[todo_calibrado,final_calibrado];
57     parametros=[parametros,vectorTK];
58  end
59
60  figure(1)
61  hold on
62  plot(sqrt(todo_calibrado(1,:).^2+todo_calibrado(2,:)
    .^2+todo_calibrado(3,:).^2),'red')
63  plot(sqrt(datos(1,:).^2+datos(2,:).^2+datos(3,:).^2))
64  hold off
65  title('Datos calibrados y sin calibrar')
66
67  %%%%%CALIBRACIÓN GIROSCOPIO %%%%%
68
69  omega_x = datos(4,:);
70  omega_y= datos(5,:);
71  omega_z= datos(6,:);
72  omega_x_estatico=[];
73  omega_y_estatico=[];
74  omega_z_estatico=[];
```

```
75
76
77 omega_x_estatico= omega_x(matriz_intervalo_estatico
       (1,1):matriz_intervalo_estatico(1,2));
78 omega_y_estatico= omega_y(matriz_intervalo_estatico
       (1,1):matriz_intervalo_estatico(2,1));
79 omega_z_estatico= omega_z(matriz_intervalo_estatico
       (1,1):matriz_intervalo_estatico(2,1));
80
81
82 estimate_bias_x=mean(omega_x_estatico);
83 estimate_bias_y=mean(omega_y_estatico);
84 estimate_bias_z=mean(omega_z_estatico);
85
86 omega_x = omega_x - estimate_bias_x;%Quita el error de
       bias
87 omega_y = omega_y - estimate_bias_y;
88 omega_z = omega_z - estimate_bias_z;
89
90 signal=[todo_calibrado(1,:);todo_calibrado(2,:);
       todo_calibrado(3,:)];
91 matriz_info_giro=matrizgiro(matriz_intervalo_estatico,
       signal);
92
93 [Tgiro,Kgiro]=calibraciongirofmin(matriz_info_giro,
       omega_x,omega_y,omega_z)
94
95
96 giro_calibrado=Tgiro*Kgiro*[omega_x;omega_y;omega_z];
97 giro_parado=[];
98 for i=1:length(matriz_info_giro(1,:))
99       giro=[omega_x(matriz_info_giro(1,i):
       matriz_info_giro(2,i));omega_y(matriz_info_giro(1,i)
       :matriz_info_giro(2,i));omega_z(matriz_info_giro(1,i
       ):matriz_info_giro(2,i))];
100      giro_parado=[giro_parado,giro];
101 end
102 giro_parado_calibrado=Tgiro*Kgiro*giro_parado;
103
104 figure(3)
105 plot(sqrt(giro_parado(1,:).^2+giro_parado(2,:).^2+
```

```
    giro_parado(3,:).^2))
title('Datos giroscopio sin calibrar')
figure(4)
plot(sqrt(giro_parado_calibrado(1,:).^2+
    giro_parado_calibrado(2,:).^2+giro_parado_calibrado
    (3,:).^2))
title('Datos giroscopio calibrados')
```

A.1.1. Funciones necesarias para la calibración del acelerómetro

```
%%%%%FUNCIÓN DETECTOR DE PERIODOS ESTÁTICOS Y
    BÚSQUEDA DE SIMILITUDES %%%%%

function [matriz_intervalo_estatico,cambios_longitud]=
    detector(a_xp,a_yp,a_zp,total_sample)

var_3D = (var(a_xp(50:300))^2 + var(a_yp(50:300))^2 +
    var(a_zp(50:300))^2);

w_d = 101;
normal_x = zeros(1, total_sample);
normal_y = zeros(1, total_sample);
normal_z = zeros(1, total_sample);

half_w_d = floor(w_d/2);

for i = (half_w_d + 1):total_sample - (half_w_d + 1)

    normal_x(i) = var(a_xp(i - half_w_d:i + half_w_d));
    normal_y(i) = var(a_yp(i - half_w_d:i + half_w_d));
    normal_z(i) = var(a_zp(i - half_w_d:i + half_w_d));

end

s_square = (normal_x.^2 + normal_y.^2 + normal_z.^2);

filter = zeros(1, total_sample);

multiplicador= 2;

```

```
28 for i = half_w_d:total_sample - (half_w_d + 1)
29
30         if s_square(i) < multiplicador*var_3D
31
32           filter(i) = 1;
33
34         end
35
36 end
37
38     k = 1;
39     matriz = zeros(1,3);
40     samples = 0;
41     start = 0;
42
43
44
45     if filter(1) == 1
46         start = 1;
47
48     end
49
50
51     for i = 2:length(filter)
52
53         if filter(i-1) == 0 && filter(i) == 0
54
55
56         elseif filter(i-1)== 1 && filter(i) == 1
57
58             samples = samples + 1;
59
60         elseif filter(i-1) == 1 && filter(i) == 0
61
62            matriz(k,:) = [start, i - 1, samples];
63
64             k= k + 1;
65
66         elseif filter(i-1) == 0 && filter(i) == 1
67
68             start = i;
```

```
69                 samples = 1;
70
71             end
72
73         end
74         v= find(matriz(:,3)>=100);
75         matriz_intervalo_estatico(:,:)=matriz(v,:);
76
77         for i=1:(length(matriz_intervalo_estatico(:,1)))
78          datos_parados_x(i)=mean(a_xp(
           matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)));
79          datos_parados_y(i)=mean(a_yp(
           matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)));
80          datos_parados_z(i)=mean(a_zp(
           matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)));
81          modulo_bloque(i)=sqrt(datos_parados_x(i)^2+
           datos_parados_y(i)^2+datos_parados_z(i)^2);
82         end
83
84         cambios_longitud=(abs(diff(modulo_bloque))>0.01);
85         figure(2)
86         plot(sqrt(a_xp.^2+a_yp.^2+a_zp.^2))
87         hold on
88          for i=1:(length(matriz_intervalo_estatico(:,1)))
89              plot(matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2),...
90                  (sqrt(a_xp(matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)).^2+...
91              a_yp(matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)).^2+...
92              a_zp(matriz_intervalo_estatico(i,1):
           matriz_intervalo_estatico(i,2)).^2)),'r')
93
94          end
95          title('Detección bloques estáticos')
96         hold off

1  %%%%%FUNCIÓN DE SELECCIÓN DE DATOS %%%%%
```

```
function selected_data=seleccion(num_bloques,signal,
    matriz_intervalo_estatico)

     selected_data = zeros(3, 1);
     k = 1;
     long_bloque=100;

               selection_step = floor(
    matriz_intervalo_estatico(num_bloques,3)/long_bloque
    );
               for i = 1:(long_bloque - 1)
                    selected_data(1:3, k) = signal(1:3,
    matriz_intervalo_estatico(num_bloques, 1) + (i - 1)*
    selection_step);
                   k= k + 1;

               end
```

```
%%%%%FUNCIÓN DE MINIMIZACIÓN DEL ACELERÓMETRO,
    OBTENCIÓN DE PARÁMETROS %%%%%
function [T,K,b,vectorTK]=calibracion(selected_data)
     datosmedia=zeros(3,floor(length(selected_data(1,:))
    /10)-1);

     for k=1:floor(length(selected_data(1,:))/10)
          datosmedia(1,k)=mean(selected_data(1,10*(k-1)
    +1:10*(k-1)+10));
          datosmedia(2,k)=mean(selected_data(2,10*(k-1)
    +1:10*(k-1)+10));
          datosmedia(3,k)=mean(selected_data(3,10*(k-1)
    +1:10*(k-1)+10));
     end

theta_pr = [0, 0, 0,1,1,1, 0, 0, 0];


     ObjectiveFunction = @(theta) accCostFunctLSQNONLIN2
    (theta, datosmedia);
    %options = optimset('MaxFunEvals', 150000, 'MaxIter
    ', 6000, 'TolFun', 10^(-10));
     options = optimset('MaxFunEvals', 150000, 'MaxIter'
```

```
    , 6000, 'TolFun', 10^(-10),'Algorithm',{'levenberg-
    marquardt',.005});
      %Algoritmo de minimizacion:

      [theta_pr, rsnorm] = lsqnonlin(ObjectiveFunction,
    theta_pr, [], [], options);

   T = [1, -theta_pr(1), theta_pr(2); 0, 1, -theta_pr(3)
    ; 0, 0, 1];
   K = diag([theta_pr(4) , theta_pr(5) , theta_pr(6) ]);
   b = [theta_pr(7); theta_pr(8); theta_pr(9)];

   vectorTK=theta_pr';
```

```
%%%%%FUNCIÓN OBJETIVO DEL ACELERÓMETRO %%%%%

function [ res_vector ] = accCostFunctLSQNONLIN2( E,
    a_hat )

misalignmentMatrix = [1, -E(1), E(2); 0, 1, -E(3); 0,
    0, 1];
scalingMtrix = diag([E(4) , E(5) , E(6) ]);

a_bar = misalignmentMatrix*scalingMtrix*(a_hat) - (diag
    ([E(7), E(8), E(9)])*ones(3, length(a_hat)));

magnitude = 1;

residuals = zeros(length(a_bar(1,:)), 1);

for i = 1:length(a_bar(1,:))
     residuals(i,1) = magnitude^2 - (a_bar(1,i)^2 +
    a_bar(2,i)^2 + a_bar(3,i)^2);
end

res_vector = residuals.^2+0.061*(E(1)^2+E(2)^2+E(3)^2+(
    E(4)-1)^2+(E(5)-1)^2+(E(6)-1)^2+E(7)^2+E(8)^2+E(9)
    ^2);

end
```

A.1.2. Funciones necesarias para la calibración del giroscopio

```
%%%%%SELECCIÓN DE DATOS DEL GIROSCOPIO %%%%%
 function matriz_info_giro=matrizgiro(
    matriz_intervalo_estatico,signal)
selected_data = zeros(3, 1);
qsTime = 1;
sample_period = 0.01;
num_samples = qsTime/sample_period;

k = 1;
matriz=matriz_intervalo_estatico';
for g = 1:length(matriz(1,:))

     selected_acc_data = zeros(3,1);
     selection_step = floor(matriz(3,g)/num_samples);

     for i = 1:(num_samples - 1)

          selected_acc_data(1:3, i) = signal(1:3, matriz
    (1, g) + (i - 1)*selection_step);

     end

     selected_data(1:3, num_samples) = signal(1:3,
    matriz(2, g));
     matriz_info_giro(4:6, k) = mean(selected_acc_data,
    2);
     k = k + 1;

end
matriz_info_giro(1:3,:)=matriz;


%%%%%FUNCIÓN DE MINIMIZACIÓN DEL GIROSCOPIO,
    OBTENCIÓN DE PARÁMETROS %%%%%
function [Tgiro,Kgiro]=calibraciongirofmin(
    matriz_info_giro,omega_x,omega_y,omega_z)

```

```
parametros.matriz=matriz_info_giro;
parametros.x=omega_x;
parametros.y=omega_y;
parametros.z=omega_z;

theta_pr_gyro = [1,0,0,0,1,0,0,0,1];
funcionmin=@(x,parametros)gyroCostfmin(x,parametros.
    matriz,parametros.x,parametros.y,parametros.z)

theta_pr_gyro=fminsearch(funcionmin,theta_pr_gyro,[],
    parametros);

Tgiro = [1, theta_pr_gyro(2), theta_pr_gyro(3);
    theta_pr_gyro(4), 1, theta_pr_gyro(6); theta_pr_gyro
    (7), theta_pr_gyro(8), 1];
Kgiro = [theta_pr_gyro(1), 0, 0; 0, theta_pr_gyro(5),
    0; 0, 0, theta_pr_gyro(9)];


%%%%%FUNCIÓN OBJETIVO DEL GIROSCOPIO %%%%%
function [res_vector] = gyroCostfmin(x,
    QS_time_interval_info_matrix, omega_x_hat,
    omega_y_hat, omega_z_hat)

omega_hat = [omega_x_hat; omega_y_hat; omega_z_hat];

misalignmentMatrix = [1, x(2), x(3); x(4), 1, x(6); x
    (7), x(8), 1];
scalingMatrix = diag([x(1), x(5), x(9)]);

omega_bar = misalignmentMatrix*scalingMatrix*omega_hat;

omega_x = omega_bar(1,:);
omega_y = omega_bar(2,:);
omega_z = omega_bar(3,:);

vector = zeros(3,5);

for pr = 1:size(QS_time_interval_info_matrix, 2) - 1

    vector((pr-1)*3 + 1:(pr)*3, 1) =
```

```
    QS_time_interval_info_matrix(4:6,pr);
21   vector((pr-1)*3 + 1:(pr)*3, 5) =
    QS_time_interval_info_matrix(4:6,pr + 1);
22   gyroUnbiasUncalibratedValues = [omega_x(
    QS_time_interval_info_matrix(2,pr) + 1:
    QS_time_interval_info_matrix(1,pr + 1) - 1); omega_y
    (QS_time_interval_info_matrix(2,pr) + 1:
    QS_time_interval_info_matrix(1,pr + 1) - 1); omega_z
    (QS_time_interval_info_matrix(2,pr) + 1:
    QS_time_interval_info_matrix(1,pr + 1) - 1)];
23   R = rotationRK4(gyroUnbiasUncalibratedValues);
24
25   vector((pr-1)*3 + 1:(pr)*3, 2:4) = R;
26
27
28 end
29
30 residuals = zeros(length(vector(:,1))/3, 1);
31
32 for i = 1:length(vector(:,1))/3
33
34   v = vector((i-1)*3 + 1:(i)*3, 5)/(vector((i-1)*3 +
    1, 5)^2 + vector((i-1)*3 + 2, 5)^2 + vector((i)*3,
    5)^2)^(1/2) - vector((i-1)*3 + 1:(i)*3, 2:4)*vector
    ((i-1)*3 + 1:(i)*3, 1)/(vector((i-1)*3 + 1, 5)^2 +
    vector((i-1)*3 + 2, 5)^2 + vector((i)*3, 5)^2)^(1/2)
    ;
35
36   v = v';
37  residuals(i,1) = (v(1)^2 + v(2)^2 + v(3)^2);
38
39
40 end
41 residuals= sum(residuals);
42 res_vector = residuals+((x(1)-1)^2+x(2)^2+x(3)^2+(x(5)
    -1)^2+(x(5)-1)^2+x(6)^2+x(7)^2+x(8)^2+(x(9)-1)^2);
43 end
```

```
1 %%%%%FUNCIÓN PARA TRANSFORMAR EL VECTOR VELOCIDAD
    ANGULAR A CUTARNIÓN %%%%%
2 function [ q, angularRotation, direction ] =
```

```
    fromOmegaToQ( omega, intervals )
3
4  direction = zeros(3,1);
5  q = zeros(1,4);
6   angularRotation = (omega(1)^2 + omega(2)^2 + omega(3)
      ^2)^(1/2)*intervals;
7   direction(:,1) = omega(:,1)*(intervals/angularRotation
      );
8  dir = direction(:,1)';
9   q(1,:) = [cos(angularRotation/2), sin(angularRotation
      /2)*dir];
10
11 end

1  %%%%%ALGORITMO DE INTRGRACIÓN RUNGE-KUTTA DE CUARTO
     ORDEN %%%%%
2
3  function [ R ] = rotationRK4( omega )
4
5
6  dt = 0.01;
7
8  omega_x = omega(1,:);
9  omega_y = omega(2,:);
10 omega_z = omega(3,:);
11
12 num_samples = length(omega_x);
13
14 q_k = fromOmegaToQ([omega_x(1); omega_y(1); omega_z(1)
     ], [0.01])';
15 q_next_k = [0; 0; 0; 0];
16
17 for i = 1:num_samples - 1
18
19      % first Runge-Kutta coefficient
20      q_i_1 = q_k;
21      OMEGA_omega_t_k = ...
22          [0              -omega_x(i)  -omega_y(i)  -omega_z
    (i);
23          omega_x(i)    0              omega_z(i)   -omega_y
    (i);
```

```
24          omega_y(i)    -omega_z(i)   0                 omega_x(
    i);
25          omega_z(i)    omega_y(i)    -omega_x(i)   0
        ];
26      k_1 = (1/2)*OMEGA_omega_t_k*q_i_1;
27
28      %second Runge-Kutta coefficient
29      q_i_2 = q_k + dt*(1/2)*k_1;
30      OMEGA_omega_t_k_plus_half_dt = ...
31          [0                                          -(omega_x(i)
    + omega_x(i + 1))/2    -(omega_y(i) + omega_y(i + 1))
    /2   -(omega_z(i) + omega_z(i + 1))/2;
32          (omega_x(i) + omega_x(i + 1))/2    0
                              (omega_z(i) + omega_z(i + 1))
    /2    -(omega_y(i) + omega_y(i + 1))/2;
33          (omega_y(i) + omega_y(i + 1))/2    -(omega_z(i)
    + omega_z(i + 1))/2    0
         (omega_x(i) + omega_x(i + 1))/2;
34          (omega_z(i) + omega_z(i + 1))/2    (omega_y(i) +
     omega_y(i + 1))/2      -(omega_x(i) + omega_x(i + 1))
    /2   0                                          ];
35      k_2 = (1/2)*OMEGA_omega_t_k_plus_half_dt*q_i_2;
36
37      %third Runge-Kutta coefficient
38      q_i_3 = q_k + dt*(1/2)*k_2;
39      OMEGA_omega_t_k_plus_half_dt = ...
40          [0                                          -(omega_x(i)
    + omega_x(i + 1))/2    -(omega_y(i) + omega_y(i + 1))
    /2   -(omega_z(i) + omega_z(i + 1))/2;
41          (omega_x(i) + omega_x(i + 1))/2    0
                              (omega_z(i) + omega_z(i + 1))
    /2    -(omega_y(i) + omega_y(i + 1))/2;
42          (omega_y(i) + omega_y(i + 1))/2    -(omega_z(i)
    + omega_z(i + 1))/2    0
         (omega_x(i) + omega_x(i + 1))/2;
43          (omega_z(i) + omega_z(i + 1))/2    (omega_y(i) +
     omega_y(i + 1))/2      -(omega_x(i) + omega_x(i + 1))
    /2   0                                          ];
44      k_3 = (1/2)*OMEGA_omega_t_k_plus_half_dt*q_i_3;
45
46      %forth Runge-Kutta coefficient
```

```
47        q_i_4 = q_k + dt*1*k_3;
48        OMEGA_omega_t_k_plus_dt = ...
49             [0                  -omega_x(i + 1)  -omega_y(i +
      1)  -omega_z(i + 1);
50              omega_x(i + 1)    0                  omega_z(i +
      1)   -omega_y(i + 1);
51              omega_y(i + 1)    -omega_z(i + 1)  0
          omega_x(i + 1);
52              omega_z(i + 1)    omega_y(i + 1)   -omega_x(i +
      1)  0             ];
53        k_4 = (1/2)*OMEGA_omega_t_k_plus_dt*q_i_4;
54
55        q_next_k = q_k + dt*((1/6)*k_1 + (1/3)*k_2 + (1/3)*
      k_3 + (1/6)*k_4);
56
57        q_next_k = q_next_k/norm(q_next_k);
58
59        q_k = q_next_k;
60
61  end
62
63  R = fromQtoR(q_next_k);
64
65  end
```

```
1   %%%%%FUNCIÓN PARA CONSTRUIR LA MATRIZ DE ROTACIÓN
        %%%%%
2
3   function [R] = fromQtoR(q)
4
5   r_11 = q(1)^2 + q(2)^2 - q(3)^2 - q(4)^2;
6   r_12 = 2*(q(2)*q(3) - q(1)*q(4));
7   r_13 = 2*(q(2)*q(4) + q(1)*q(3));
8   r_21 = 2*(q(2)*q(3) + q(1)*q(4));
9   r_22 = q(1)^2 - q(2)^2 + q(3)^2 - q(4)^2;
10  r_23 = 2*(q(3)*q(4) - q(1)*q(2));
11  r_31 = 2*(q(2)*q(4) - q(1)*q(3));
12  r_32 = 2*(q(3)*q(4) + q(1)*q(2));
13  r_33 = q(1)^2 - q(2)^2 - q(3)^2 + q(4)^2;
14
15  R = [r_11, r_12, r_13; r_21, r_22, r_23; r_31, r_32,
```

```
    r_33 ];
16
17 end
```

Bibliografía

[1] Christopher Jekeli. *Inertial Navigation Systems with Geodetic Applications.* Walter de Gruyter, 2001.

[2] Rodrigo Iván Paredes Portador, Vicente Capistrán Gómez.*Tésis: Diagnóstico de condiciones de operación de rodamientos en máquinas usando espectros de alto orden.*

[3] Jay Esfandyari, Roberto De Nuccio, Gang Xu, «Introduction to MEMS gyroscopes», Blog de la revista «Sold State Technology», entrada del 15/11/2010, url: http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/

[4] Matej Andrejašič, Igor Poberaj. *Seminar: Mems Accelerometers.* University of Ljubliajna.

[5] David Retana Pérez, Aldo Enrique Vargas. *Tésis: Estabilización de un helicóptero a escala mediante sistemas neuro-difusos.*

[6] «Efecto Coriolis», artículo de Wikipedia, url:
http : //es.wikipedia.org/wiki/Efecto_Coriolis

[7] «¿Por qué no sopla el viento desde una zona de Alta Presión hacia una de Baja Presión?», artículo online, url:
http : //www.fondear.org/infonautic/mar/Meteo/Presion_AltaBaja/Presion_AltaBaja.htm

[8] Oliver J. Woodman.*An introduction to inertial navigation.* Artículo técnico publicado por la Universidad de Cambridge, 2007.

[9] Alberto Pretto, David Tedaldi, Emanuele Menegatti. *A Robust and Easy to Implement Method for IMU calibration without External Equipments.* Robotics and Automation (ICRA),IEEE International Conference, 3042 - 3049, 2014 .

[10] S. Bonnet, C. Bassompierre, C. Godin, S. Lesecq, A. Barraud. *Calibration methods for inertial and magnetic sensors.* Sensors and Actuators A: Physical,Vol.156,302-311 2009.

[11] J. C. Lötters, J. Schipper, P:H Veltink, W. Olthuis, P. Bergveld. *Procedure for in-use calibration of triaxial accelerometers in medical applications.* Sensor and Actuators, Vol. 68, nº1, 221-228. 1998.

[12] C.Nee, S.K.Ong, W.T.Fong. *Methods for in-field user calibration of an inertial measurement unit without external equipment.* Measurement Science and Technology, Vol. 19, nº 8, 2008.

[13] Isaac Skog, Peter Händel. *Calibration of a MEMs inertial measurement unit. XVII* Imeko World Congress.2006.

[14] Kenneth.R Britting, *Inertial navigation systems analysis.* ARTECH HOUSE, 2010.

[15] Matthew T.Mason. *Lecture 7. Quaternions.* Mechanics of Manipulation Spring 2012.

[16] «Rotaciones y cuaternios»
url: *http* : *//tesis.uson.mx/digital/tesis/docs/*21070*/Capitulo*4*.pdf*

[17] Yan-Bin Jia. *Quaternions and Rotations.* 2015.

[18] Gregory G. Slabaugh. *Computing Euler angles from a rotation matrix.* Retrieved on August, Vol.6, nº2000, 39-63, 1999. url: *http* : *//thomasbeatty.com/MATH* %20*PAGES/ARCHIVES* %20 − %20*NOTES/Applied* %20*Math/euler* %20*angles.pdf*

[19] Hansen, Per Christian. *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems.* Numerical algorithms, Vol. 6, nº 1, 1-35, 1994 .

[20] Darío Ramos López. *Tésis: Modelado matemático en óptica biomédica y oftalmológica.* 2014.

[21] Marten Gulliksson, Per Ake Wedin. *Optimization Tools for Tikhonov Regularization of Nonlinear Equations Using the L-Curve and its Dual.*Methods and Applications of Inversion. Springer Berlin Heidelberg, 155-170, 2000.

[22] Codela Scherer, Per Christian Hansen, Víctor Pereyra. *Least Squares Data Fitting with Applications.*The Johns Hopkins University Press, 2013.

[23] Wenyu Sun, Ya-Xiang Yuan. *Optimization theory and methods Nonlinear programming.* Springer Science & Business Media, 2006.

[24] Bjoerck A., Dahlquist G. *Numerical mathematics and scientific computation.* Philadelphia: SIAM, 2003.

[25] Fracois Caron, Emmanuel Duglos, Denis Pormorski, Philipe Vanheeghe. *GPS/IMU data fusion using multisensor filtering: introduction of contextual aspects.* Information Fusion, Vol. 7, nº 2, 221-230, 2006.

[26] Sasa Singer, John Nelder. *Scholarpedia: Nelder-Mead algorithm.*