Direction cosine matrix based IMU implementation in Matlab/Simulink

Direction cosine matrix based IMU implementation in Matlab/Simulink

J.C.D. van Zundert
0677177

CST 2013.067

Report Individual Space (4Y002)

Coach: F.B.J. Bruning

Supervisor: dr. ir. A.G. de Jager

Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology

Eindhoven, July 12, 2013

Chapter 1

Introduction

This report gives background information on the principles of the direction cosine matrix method to determine the orientation of, for example, an unmanned aerial vehicle (UAV) with respect to (the reference frame of) the earth. In combination with, for example, GPS and/or an optical flow camera, this information can be used to determine the global position of the UAV. Other methods for determining the orientation of an UAV are using a Kalman filter (see for example [2, 3]), an unscented Kalman filter (see for example [7]), a particle filter (see for example [6]) and a complementary filter (see for example [1]). In this report no new principles on the direction cosine matrix method are presented as the main principles are already present in [5] on which the major part of this report is based.

The goal of the project is to create a Matlab/Simulink implementation of the direction cosine matrix algorithm. Such an implementation allows to simulate other parts of code and/or Simulink models in combination with the direction cosine matrix algorithm. Slightly modified versions of the `APM2 Simulink Blockset` were used to create a single Simulink model for simulations within Simulink and for running the model on the hardware (an ArduPilot Mega 2560 was used). The implementation is based on the `C++` implementation of the file `AP_AHRS_DCM.cpp` (available at `https://github.com/diydrones/ardupilot/tree/master/libraries/AP_AHRS`) which makes use of the library `https://github.com/diydrones/ardupilot/tree/master/libraries`. The presented implementation is based on the on 12 July 2013 available `C++` implementation. As the code is subject to change, it might be that there are small implementation and parameter differences with the current version.

The research problem is formulated as: create a Matlab/Simulink implementation of the direction cosine method for determining the orientation of an UAV. The target is to achieve a maximum absolute difference with the original algorithm of 0.5° in terms of the Euler angles such that the quality is determined by the accuracy of the sensors and not by the algorithm used. This work differs from the rest as it gives an implementation in Matlab/Simulink rather than in `C++`.

In the remainder of this chapter background information on the direction cosine matrix algorithm in general as well as the notation that is used throughout this report is presented. Moreover, the direction cosine matrix (DCM) is introduced. As the DCM changes over time it is updated which is discussed in Chapter 2. The required drift correction is the topic of Chapter 3. The last step is to determine the Euler angles (i.e., the orientation) which is discussed in Chapter 4. In the final chapter, Chapter 5, obtained results are discussed.

1.1 Background

The direction cosine matrix describes the rotation between the reference frame of the body and the reference frame of the earth and will be formally introduced in Section 1.3. The nine elements of the DCM are a function of the Euler angles and allow to extract the Euler angles which is the topic of Chapter 4. As the body moves with respect to the earth, the DCM changes in time and should therefore be updated. This could ideally be done using only gyroscope measurements. However, due to numerical errors and gyroscope offset and drift, errors accumulate in the DCM resulting in incorrect angles. The errors in roll and pitch can be compensated by use of accelerometer measurements. Accelerometer measurements are unsuitable for the yaw correction as the plane of the yaw is perpendicular to the gravitational field. Therefore, a magnetometer is used to compensate for error in the yaw. These errors are fed back into the update of the DCM by use of negative proportional plus integral control action. An schematic overview of the algorithm is presented in Figure 1.1.

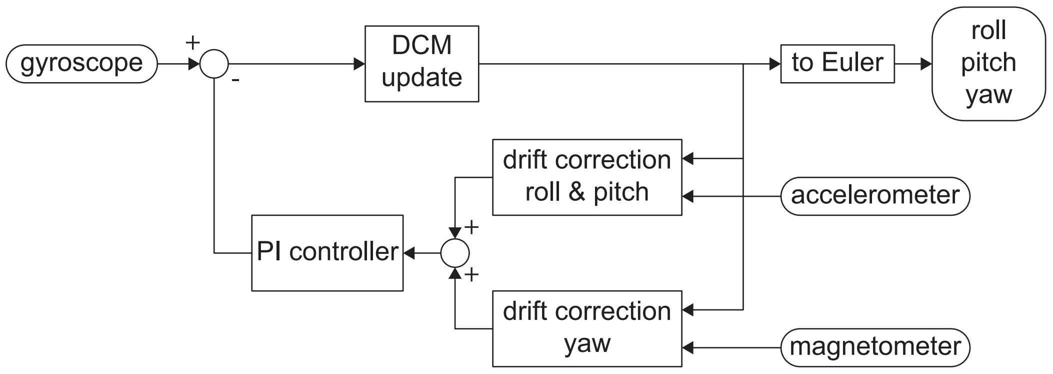

Figure 1.1: Schematic overview of the direction cosine matrix method.

In order to be able to compare the `C++` implementation with the Simulink implementation, print statements are added to the `C++` implementation for data collection. However, these print statements occasionally cause an overrun for a sample rate of $100\ Hz$. Also for a lower sample rate ($50\ Hz$) overruns occur. Therefore, the resulting variation in sample rate is taken into account in the Simulink implementation and the default sample rate of $100\ Hz$ is used.

1.2 Notation

The following notation is used throughout this report:

- A matrix is denoted by a bold capital Latin letter, e.g., $\mathbf{A}$, and parameterized as $\mathbf{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix}$.
- A column is denoted by a bold lower case Greek letter, e.g., $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}$. The length is given by the root-mean-square (RMS) value: $\|\boldsymbol{\alpha}\| = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2 1}$. A normalized column is denoted as $\hat{\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}$.
- A row is denoted by a bold lower case Latin letter, e.g., $\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix})$. The length is defined as the root-mean-square (RMS) value: $\|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. A normalized row is denoted as

$\hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}$.

- The transpose is denoted by $^\top$.
- The dot product of the columns $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of length n is defined as $\boldsymbol{\alpha} \cdot \boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i \beta_i = \boldsymbol{\alpha}^\top \boldsymbol{\beta}$. The dot product of the rows $\mathbf{a}$ and $\mathbf{b}$ of length n is defined as $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = \mathbf{a}\mathbf{b}^\top$. The dot product is commutative: $\boldsymbol{\alpha} \cdot \boldsymbol{\beta} = \boldsymbol{\beta} \cdot \boldsymbol{\alpha}$ and $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$.
- The cross product of two columns $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is defined as $\boldsymbol{\alpha} \times \boldsymbol{\beta} = \|\boldsymbol{\alpha}\| \|\boldsymbol{\beta}\| \sin\theta$ with θ the smaller angle between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (i.e., $0° \leq \theta < 180°$). The cross product of two rows $\mathbf{a}$ and $\mathbf{b}$ is defined as $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \sin\theta$ with θ the smaller angle between $\mathbf{a}$ and $\mathbf{b}$ (i.e., $0° \leq \theta < 180°$). The cross product is anticommutative: $\boldsymbol{\alpha} \times \boldsymbol{\beta} = -\boldsymbol{\beta} \times \boldsymbol{\alpha}$ and $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$.
- The frame of reference is indicated by a superscript: e for earth and b for body.

1.3 Introducing the DCM

Figure 1.2 depicts the reference frame of the earth and the reference frame of the body. The orientation vector of the body, describing the orientation with regards to its reference frame, is indicated by $\boldsymbol{\rho}^b$ with directions along the axis x^e, y^e and z^e. Similarly, the orientation vector of the earth is indicated by $\boldsymbol{\rho}^e$ with directions along the axis x^b, y^b and z^b.

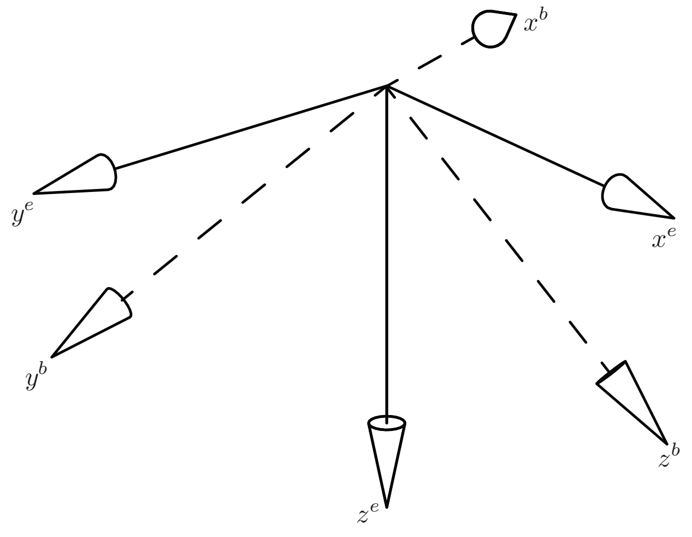

Figure 1.2: Reference frame of the earth (^{e}, solid) and of the body (^{b}, dashed).

The relation between $\boldsymbol{\rho}^e$ and $\boldsymbol{\rho}^b$ is given by

$$\boldsymbol{\rho}^e = \mathbf{R}\boldsymbol{\rho}^b \tag{1.1}$$

with $\mathbf{R}$ the direction cosine matrix (DCM):

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{r}_z \end{bmatrix}. \tag{1.2}$$

The rotation seen from the reference frame of the body ($\mathbf{R}^{-1}$) is of the same magnitude but in opposite direction from the rotation seen from the reference frame of the earth ($\mathbf{R}$) and thus it holds $\mathbf{R}^{-1} = \mathbf{R}^\top$ or equivalent $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$, i.e., the DCM is orthogonal.

Chapter 2

DCM

2.1 Updating the DCM

From the perspective of the reference frame of the body, the orientation of the earth changes over time and the orientation of the body not, i.e., $\dot{\boldsymbol{\rho}}^b = \mathbf{0}$. The time-derivative of $\boldsymbol{\rho}^e$ is given by

$$\begin{aligned} \dot{\boldsymbol{\rho}}^e &= \dot{\mathbf{R}}\boldsymbol{\rho}^b + \mathbf{R}\dot{\boldsymbol{\rho}}^b \\ &= \dot{\mathbf{R}}\boldsymbol{\rho}^b \\ &= \dot{\mathbf{R}}\mathbf{R}^{-1}\boldsymbol{\rho}^e \\ &= \dot{\mathbf{R}}\mathbf{R}^\top\boldsymbol{\rho}^e \\ &= \boldsymbol{\omega}^b_{rot} \times \boldsymbol{\rho}^e. \end{aligned}$$

Here the orthogonality property $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$ is used from which follows that $\dot{\mathbf{I}} = \dot{\mathbf{R}}\mathbf{R}^\top + \mathbf{R}\dot{\mathbf{R}}^\top = \mathbf{0}$ and thus $\dot{\mathbf{R}}\mathbf{R}^\top = -\mathbf{R}\dot{\mathbf{R}}^\top = -\left(\dot{\mathbf{R}}\mathbf{R}^\top\right)^\top$, i.e., $\dot{\mathbf{R}}\mathbf{R}$ is skew-symmetric and thus there is a column $\boldsymbol{\omega}^b_{rot}$ such that $\dot{\mathbf{R}}\mathbf{R}^\top\boldsymbol{\rho}^e = \boldsymbol{\omega}^b_{rot} \times \boldsymbol{\rho}^e$. The rotation rate column $\boldsymbol{\omega}^b_{rot}$ is given by

$$\boldsymbol{\omega}^b_{rot} = \boldsymbol{\omega}^b_P + \boldsymbol{\omega}^b_{P,yaw} + \boldsymbol{\omega}^b \tag{2.1}$$

with $\boldsymbol{\omega}^b_P$ and $\boldsymbol{\omega}^b_{P,yaw}$ the proportional terms of the controller based on roll and pitch, and yaw, respectively, and $\boldsymbol{\omega}^b$ defined as

$$\boldsymbol{\omega}^b = \boldsymbol{\omega}^b_{gyro} + \boldsymbol{\omega}^b_I \tag{2.2}$$

where $\boldsymbol{\omega}^b_{gyro}$ are the gyroscope measurements from the inertial measurement unit (IMU) and $\boldsymbol{\omega}^b_I$ is the integrating term of the controller. The terms $\boldsymbol{\omega}^b_P$, $\boldsymbol{\omega}^b_{P,yaw}$ and $\boldsymbol{\omega}^b_I$ are defined later on.

For small time steps δt, the orientation of the earth at the next time step $\boldsymbol{\rho}^e(t+\delta t)$ can be approximated by

$$\begin{aligned}
\boldsymbol{\rho}^e(t+\delta t) &= \boldsymbol{\rho}^e(t) + \dot{\boldsymbol{\rho}}^e(t)\delta t \\
&= \boldsymbol{\rho}^e(t) + \left(\boldsymbol{\omega}^b_{rot}(t) \times \boldsymbol{\rho}^e(t)\right)\delta t \\
&= \boldsymbol{\rho}^e(t) - \boldsymbol{\rho}^e(t) \times \boldsymbol{\omega}^b_{rot}(t)\delta t \\
&= \mathbf{R}(t)\left(\boldsymbol{\rho}^b(t) - \boldsymbol{\rho}^b(t) \times \boldsymbol{\omega}^b_{rot}(t)\delta t\right) \\
&= \mathbf{R}(t)\left(\boldsymbol{\rho}^b(t) + \left(\boldsymbol{\omega}^b_{rot}(t)\delta t\right) \times \boldsymbol{\rho}^b(t)\right) \\
&= \mathbf{R}(t)\begin{bmatrix} \rho^b_x + \omega^b_{rot,y}\delta t\rho^b_z - \omega^b_{rot,z}\delta t\rho^b_y \\ \rho^b_y + \omega^b_{rot,z}\delta t\rho^b_x - \omega^b_{rot,x}\delta t\rho^b_z \\ \rho^b_z + \omega^b_{rot,x}\delta t\rho^b_y - \omega^b_{rot,y}\delta t\rho^b_x \end{bmatrix} \\
&= \mathbf{R}(t)\begin{bmatrix} 1 & -\omega^b_{rot,z}\delta t & \omega^b_{rot,y}\delta t \\ \omega^b_{rot,z}\delta t & 1 & -\omega^b_{rot,x}\delta t \\ -\omega^b_{rot,y}\delta t & \omega^b_{rot,x}\delta t & 1 \end{bmatrix}\boldsymbol{\rho}^b.
\end{aligned}$$

Hence, the update of the DCM is given by

$$\mathbf{R}(t+\delta t) = \mathbf{R}(t)\begin{bmatrix} 1 & -\omega_{rot,z}\delta t & \omega_{rot,y}\delta t \\ \omega_{rot,z}\delta t & 1 & -\omega_{rot,x}\delta t \\ -\omega_{rot,y}\delta t & \omega_{rot,x}\delta t & 1 \end{bmatrix}. \tag{2.3}$$

The update of the DCM (2.3) using $\boldsymbol{\omega}^b_{rot}$ defined in (2.1) is implemented in the function `matrix_update`.

2.2 Renormalization of the DCM

Due to approximations in the update of the DCM given by (2.3), orthogonality properties will be lost over time. Therefore, the DCM is renormalized.

The dot product $e_\perp = \mathbf{r}_x \cdot \mathbf{r}_y$ is zero when $\mathbf{r}_x$ and $\mathbf{r}_y$ are orthogonal and unequal to zero if not orthogonal. $e_\perp$ is thus a measure of orthogonality between $\mathbf{r}_x$ and $\mathbf{r}_y$ and is therefore used for correction:

$$\mathbf{t}_0 = \mathbf{r}_x - \tfrac{1}{2}e_\perp\mathbf{r}_y \quad \text{and} \quad \mathbf{t}_1 = \mathbf{r}_y - \tfrac{1}{2}e_\perp\mathbf{r}_x. \tag{2.4}$$

To guarantee the third row is orthogonal to the first two rows it is based on the first two rows:

$$\mathbf{t}_2 = \mathbf{t}_0 \times \mathbf{t}_1. \tag{2.5}$$

The normalized DCM is given by

$$\mathbf{R} = \begin{bmatrix} \hat{\mathbf{t}_0} \\ \hat{\mathbf{t}_1} \\ \hat{\mathbf{t}_2} \end{bmatrix}. \tag{2.6}$$

Note that this is the exact normalization in contrast to the normalization proposed in Eqn. 21 of [5] which is based on a Taylor series expansion to decrease the computational time. As in the original C++ implementation the exact normalization is implemented, this is also done here (see the function `renorm`). In the function `normalize` (2.4) and (2.5) are implemented.

Chapter 3

Drift correction

To correct for drift, the terms $\boldsymbol{\omega}^b_{P,yaw}$, $\boldsymbol{\omega}^b_P$ and $\boldsymbol{\omega}^b_I$ are determined which are eventually used in (2.3) for updating the DCM. The drift correction for roll and pitch is based on accelerometer measurements. The drift correction for yaw is based on magnetometer measurements.

3.1 Drift correction yaw

The correction for yaw only applies to the z-direction of $\boldsymbol{\omega}^b_{P,yaw}$ and $\boldsymbol{\omega}^b_I$ as they are defined in the reference frame of the body, i.e., only $\boldsymbol{\omega}^b_{P,yaw,z}$ and $\boldsymbol{\omega}^b_{I,z}$ are updated in the function `drift_correction_yaw`. The correction is based on the error between the magnetometer reading and the actual heading based on declination measurements.

3.1.1 No new magnetometer reading

If there has been no new magnetometer reading, which may be the case if the reading of the compass was not finished in time, the proportional term $\omega_{P,yaw}$ is decreased to 97% of its previous value to decrease the influence of the yaw error that was based on previous measurements.

3.1.2 New magnetometer reading

If there has been a new magnetometer reading, the time between this reading and the previous one is denoted by ΔT_y. For the heading only the x- and y-direction are relevant in the reference frame of the earth and therefore only these values are computed from the measured compass magnitude $\boldsymbol{\eta}^b$:

$$\boldsymbol{\eta}^e_{xy} = \begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \end{bmatrix} \boldsymbol{\eta}^b. \tag{3.1}$$

Denoting the measured declination by δ, the heading is $\begin{bmatrix} \cos(\delta) \\ \sin(\delta) \end{bmatrix}$. The yaw error in the earth frame (which is in z-direction) is based on comparing this with $\boldsymbol{\eta}^e_{xy}$ after normalization, i.e., comparing with $\hat{\boldsymbol{\eta}}^e_{xy}$ which is the normalized version of the two dimensional column $\boldsymbol{\eta}^e_{xy}$. This is done by calculating the corresponding cross product:

$$e^e_{yaw} = \hat{\eta}^e_x \sin(\delta) - \hat{\eta}^e_y \cos(\delta). \tag{3.2}$$

As the correction is only based on the z-component in the body frame and the resulting z-component in the earth frame, the 'rotation' from the reference frame of the earth to the reference frame of the body is given by

$$e^b_{yaw} = R_{zz} e^e_{yaw}. \tag{3.3}$$

The proportional term $\boldsymbol{\omega}^b_{P,yaw,z}$ is given by

$$\boldsymbol{\omega}^b_{P,yaw,z} = k_{P,yaw} P_{gain}(\|\boldsymbol{\omega}^b\|) e^b_{yaw} \tag{3.4}$$

with $P_{gain}(\|\boldsymbol{\omega}^b\|)$ a nonlinear function of the spin rate $\|\boldsymbol{\omega}^b\|$ and $k_{P,yaw} = 0.3\ rad/s$ a constant defined in `AP_AHRS.cpp`.

The function $P_{gain}(\|\boldsymbol{\omega}^b\|)$ as implemented on 12 July 2013 returns $\|\boldsymbol{\omega}^b\|$ divided by $50\frac{180}{\pi}$ and saturated between 1 and 10. In the code it is stated that this variable gain is based on [4]. However, the implementation differs from the one suggested in [4], which is a mistake in the implementation. The spin rate is in $[rad/s]$ and the proposed values of 50 and 500 are in $[^\circ/s]$ so it should be `ToRad` instead of `ToDeg` for all three instances in the function `_P_gain` defined in `AP_AHRS_DCM.cpp`. In the results discussed in Chapter 5 are based on the old implementation, i.e., divide $\|\boldsymbol{\omega}^b\|$ by $50\frac{180}{\pi}$ and saturate the result between 1 and 10.

The integrating term $\boldsymbol{\omega}^b_{I,sum,yaw}$ is given by

$$\boldsymbol{\omega}^b_{I,sum,yaw} = \begin{bmatrix} 0 \\ 0 \\ \int_{\Delta T_y} k_{I,yaw} e^b_{yaw}\,\mathrm{d}t \end{bmatrix} \tag{3.5}$$

if the time $\Delta T_y < 2\ s$ (there has been a yaw reference in the last two seconds) and if the spin rate $\|\boldsymbol{\omega}^b\|$ is smaller than the spin rate limit of $20\ ^\circ/s$ (the body is not spinning too fast), with the constant $k_{I,yaw} = 0.01\ rad/s^2$ as defined in `AP_AHRS_DCM.h`. If one of these conditions is not satisfied, $\boldsymbol{\omega}^b_{I,sum,z}$ is not updated based on the yaw correction.

3.2 Drift correction roll and pitch

The drift correction for roll and pitch are based on comparing the accelerometer measurements converted to the reference frame of the earth with the actual acceleration in the reference frame of the earth.

The accelerometer measurements $\boldsymbol{\eta}^b$ are transformed to the reference frame of the earth:

$$\boldsymbol{\eta}^e = \mathbf{R}\boldsymbol{\eta}^b. \tag{3.6}$$

The accelerometer values used for correction are denoted by $\bar{\boldsymbol{\eta}}^e$ and are the average of $\boldsymbol{\eta}^e$ over time since the last update ΔT_{rp} ago, i.e.,

$$\bar{\boldsymbol{\eta}}^e = \frac{1}{\Delta T_{rp}} \int_{\Delta T_{rp}} \boldsymbol{\eta}^e. \tag{3.7}$$

$\bar{\boldsymbol{\eta}}^e$ is divided by the gravitational acceleration $g = 9.80665\ m/s^2$ and normalized when its length is larger than one. The result is denoted by ${}^*\bar{\boldsymbol{\eta}}^e$ which is compared with the accelerations at the earth given by

$$\boldsymbol{\gamma}^e = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}. \tag{3.8}$$

The error $\boldsymbol{\epsilon}^e_{accel}$ is defined as

$$\boldsymbol{\epsilon}^e_{accel} = {}^*\bar{\boldsymbol{\eta}}^e \times \boldsymbol{\gamma}^e. \tag{3.9}$$

To guarantee only the roll and pitch are corrected, the last element of $\boldsymbol{\epsilon}^e_{accel}$ is explicitly set to zero. $\boldsymbol{\epsilon}^e_{accel}$ is converted to the body frame:

$$\boldsymbol{\epsilon}^b_{accel} = \mathbf{R}^\top \boldsymbol{\epsilon}^e_{accel}. \tag{3.10}$$

The proportional term of the controller is calculated as

$$\boldsymbol{\omega}^b_P = P_{gain}(\|\boldsymbol{\omega}^b\|) k_p \boldsymbol{\epsilon}^b_{accel} \tag{3.11}$$

with the function $P_{gain}(\|\boldsymbol{\omega}^b\|)$ as defined at (3.4) and the constant $k_P = 0.3\ rad/s$ defined in `AP_AHRS.cpp`.

Integrating $k_I \mathbf{G}_{error,b}$ with $k_I = 0.0087\ rad/s^2$ a constant (defined in `AP_AHRS_DCM.h`), over time during the time span ΔT_{sum} and adding it to the term from the yaw correction $\boldsymbol{\omega}^b_{I,sum,yaw}$ gives:

$$\boldsymbol{\omega}^b_{I,sum} = \boldsymbol{\omega}^b_{I,sum,yaw} + k_I \int_{\Delta T_{sum}} \boldsymbol{\epsilon}^e_{accel}\, \mathrm{d}t. \tag{3.12}$$

The time span ΔT_{sum} is a combination of the time intervals in which the spin rate $\|\boldsymbol{\omega}^b\|$ is smaller than the spin rate limit of $20\ ^\circ/s$ since the last reset. $\boldsymbol{\omega}^b_I$ should remain within the physical bounds of the device given by the maximum gyro drift rate of $w'_{lim} = 0.5\ '/s = \frac{0.5}{60}\ ^\circ/s^2$ (defined in `AP_InertialSensor_MPU6000.cpp`). Therefore, $\boldsymbol{\omega}^b_I$ is set equal to $\boldsymbol{\omega}^b_{I,sum}$ after saturating it by $\pm w'_{lim} \Delta T_{sum}$.

Note that the conditions under which the drift correction for yaw takes place are different from the conditions under which the drift correction of the roll and pitch takes place. The update of $\boldsymbol{\omega}^b_I$ depends on the roll and pitch update.

Chapter 4

Euler angles

In this chapter the calculation of the Euler angles roll ϕ, pitch θ and yaw ψ of the body is explained.

The orientation of the reference frame of the body with respect to the reference frame of the earth can be described by Euler angles ϕ, θ and ψ as is illustrated in Figure 4.1. Starting from an orientation

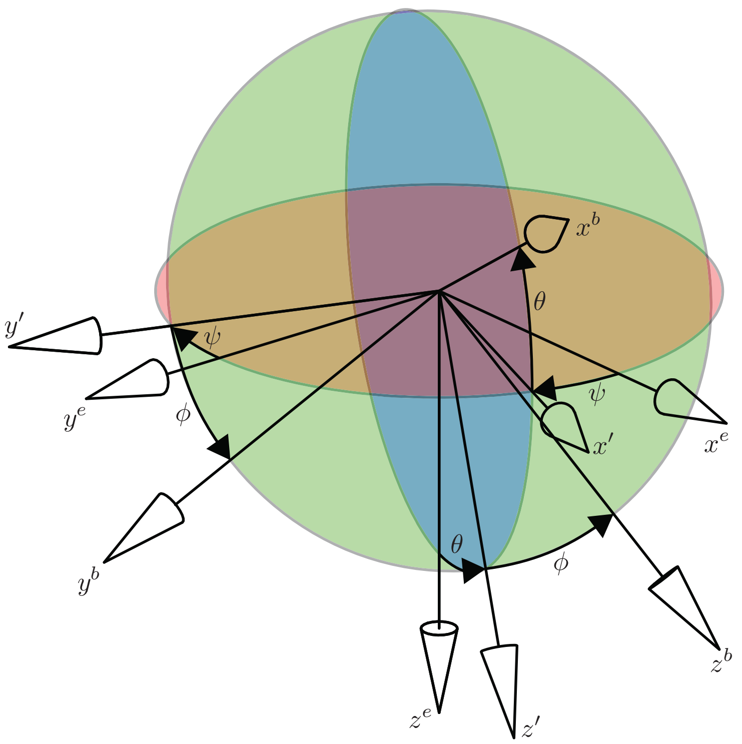

Figure 4.1: Relation between the reference frame of the earth (^{e}) and the reference frame of the body (^{b}) in terms of the Euler angles ϕ, θ and ψ.

of the reference frame of the body equal to the orientation of the reference frame of the earth, the relation is given by the following subsequently rotations (the order is of importance):

1. Rotate the body about its z-axis (z^e in Figure 4.1) through the yaw angle ψ, i.e., apply rotation

matrix

$$\mathbf{R}_1 = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2. Rotate the body about its y-axis (y' in Figure 4.1) through the pitch angle θ, i.e., apply rotation matrix

$$\mathbf{R}_2 = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}.$$

3. Rotate the body about its x-axis (x^b in Figure 4.1) through the roll angle ϕ, i.e., apply rotation matrix

$$\mathbf{R}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}.$$

Applying these rotation in this order yields the relation $\boldsymbol{\rho}^b = \mathbf{R}_3\mathbf{R}_2\mathbf{R}_1\boldsymbol{\rho}^e$. Comparing this with the relation $\boldsymbol{\rho}^e = \mathbf{R}\boldsymbol{\rho}^b$ gives (using the orthogonality property of $\mathbf{R}$):

$$\begin{aligned} \mathbf{R} &= (\mathbf{R}_3\mathbf{R}_2\mathbf{R}_1)^{-1} \\ &= (\mathbf{R}_3\mathbf{R}_2\mathbf{R}_1)^{\top} \\ &= \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}. \end{aligned}$$

From this relation between the DCM and the Euler angles, the Euler angles can be determined as follows:

$$\phi = \arctan\left(\frac{R_{zy}}{R_{zz}}\right) \tag{4.1}$$

$$\theta = -\arcsin(R_{zx}) \tag{4.2}$$

$$\psi = \arctan\left(\frac{R_{yx}}{R_{xx}}\right) \tag{4.3}$$

The roll ϕ, pitch θ, and yaw ψ angles are determined within the function `euler_angles` which calls the function `to_euler` in which (4.1), (4.2) and (4.3) are implemented.

Chapter 5

Results

In this chapter results are presented and differences are commented.

Gyroscope, accelerometer and magnetometer data was captured during motion of an ArduPilot Mega 2560 with the original direction cosine matrix algorithm implemented. This data was then used to validate the implementation of the algorithm in Matlab/Simulink.

The resulting roll, pitch and yaw angles are presented in Figure 5.1. Differences are hardly visible and therefore the difference in the resulting angles from the two algorithms is studied. To avoid discontinuities that occur when changing from $+180°$ to $-180°$ or vice versa, the sine of the difference is shown in Figure 5.2 rather than the difference itself. From this figure it can be derived that the difference is not always smaller than the target of $0.5°$. The figures show that there is no drift present, i.e., there are no structural differences between the algorithms.

Based on these results it is concluded that the in Matlab/Simulink implemented direction cosine matrix algorithm describes the original algorithm quite well but not as accurate as was aimed for in the research problem. A possible reason is that maybe not all functions are implemented (correctly). For improving the quality of the algorithm, it is advised to evaluate each function separately instead of validating the complete algorithm at once.

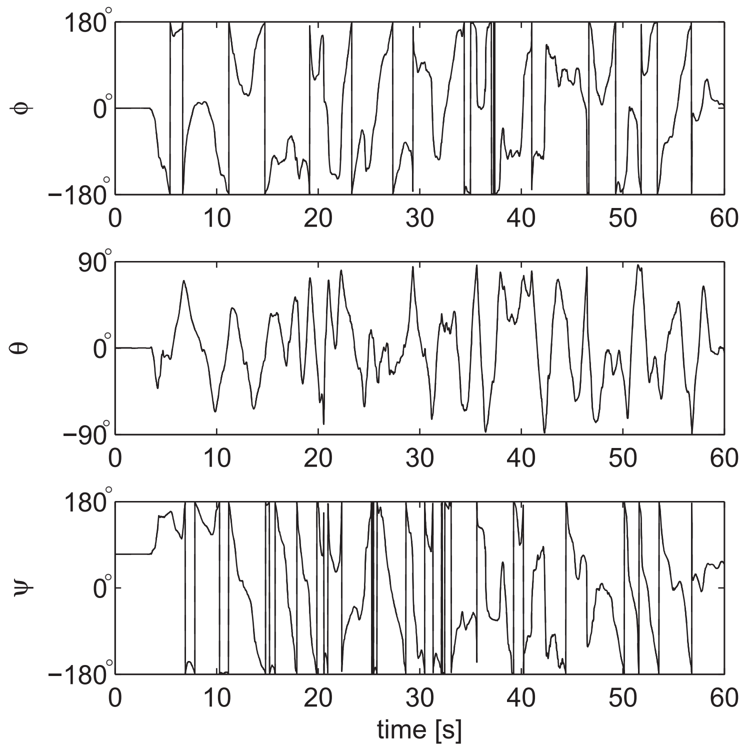

Figure 5.1: Roll ϕ, pitch θ and yaw ψ as function of time for the original implementation (solid) and for the Simulink implementation (dashed).

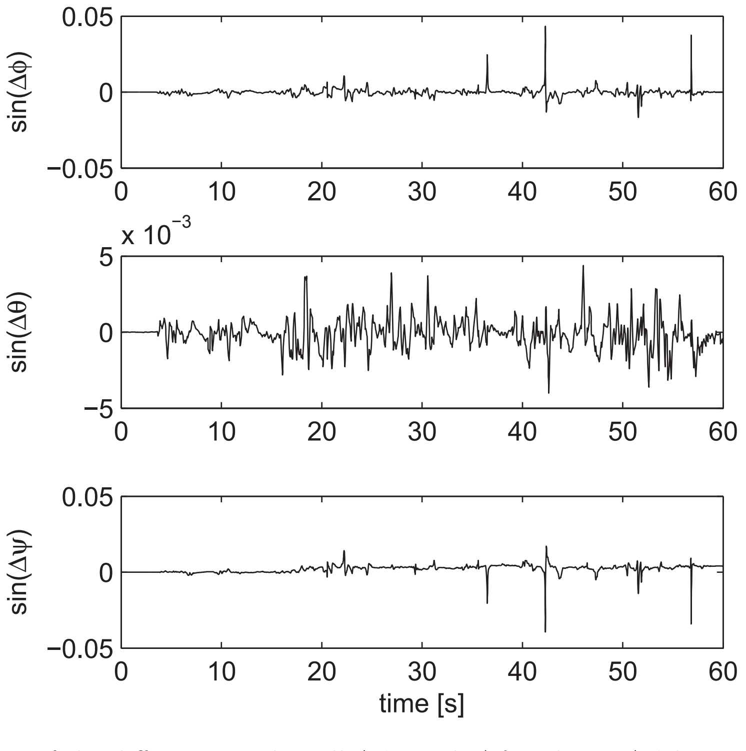

Figure 5.2: Sine of the difference in the roll $\Delta\phi$, pitch $\Delta\theta$ and yaw $\Delta\psi$ between the original implementation and the Simulink implementation.

Bibliography

[1] Shane Colton. The balance filter. `http://web.mit.edu/scolton/www/filter.pdf`, 2007.

[2] Songlai Han and Jinling Wang. A novel method to integrate IMU and magnetometers in attitude and heading reference systems. *Journal of Navigation*, 64(4):727–738, 2011.

[3] Wei Li and Jinling Wang. Effective adaptive kalman filter for MEMS-IMU/magnetometers integrated attitude and heading reference systems. *Journal of Navigation*, 66(1):99–113, 2013.

[4] William Premerlani. Fast rotations. `http://gentlenav.googlecode.com/files/fastRotations.pdf`, 2011.

[5] William Premerlani and Paul Bizard. Direction cosine matrix IMU: Theory. `https://gentlenav.googlecode.com/files/DCMDraft2.pdf`, 2009.

[6] Yafei Ren and Xizhen Ke. Particle filter data fusion enhancements for MEMS-IMU/GPS. *Intelligent Information Management*, 2(7):417–421, 2010.

[7] Zhen Shi, Jie Yang, Peng Yue, and ZiJian Cheng. Angular velocity estimation in gyroscope-free inertial measurement system based on unscented kalman filter. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 2031–2034. IEEE, 2010.