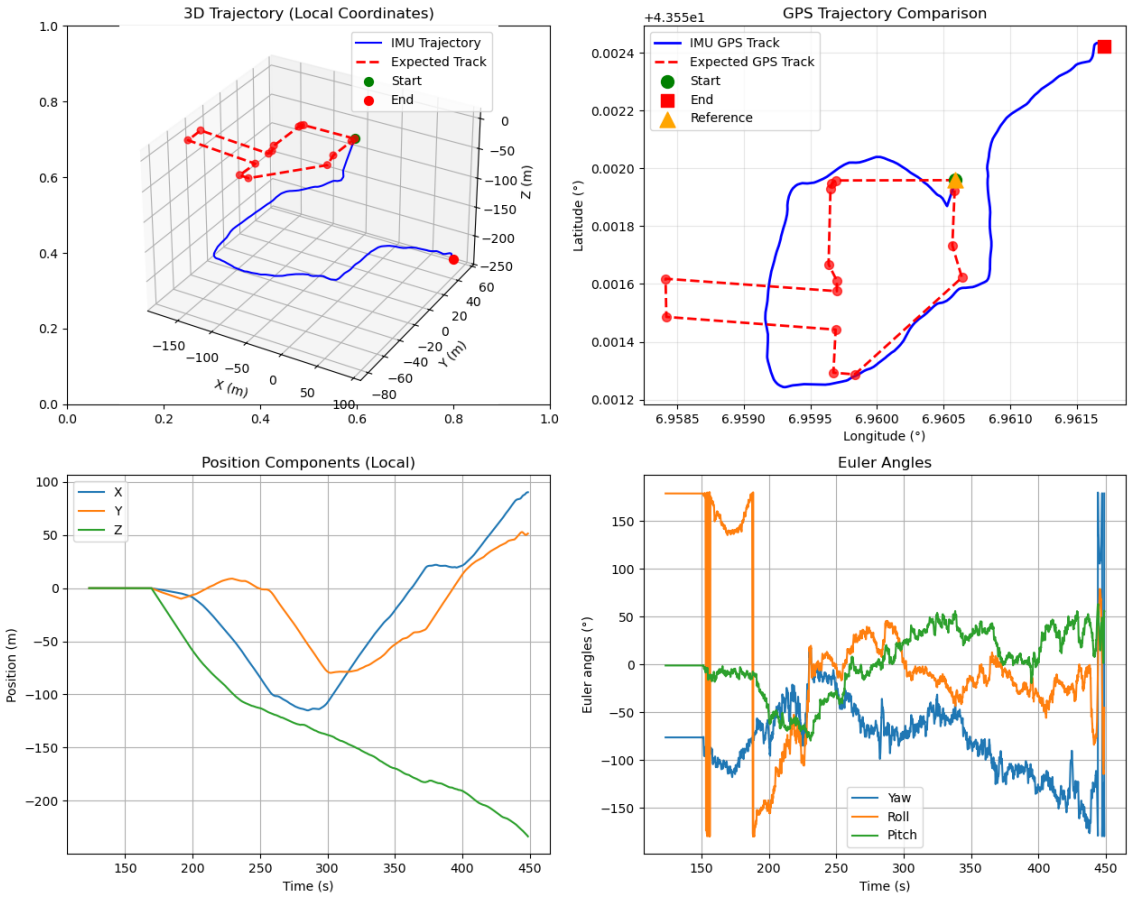
ENOVA

IMU to track - A technical review outlining code implementation and underlying mathematical principles

Larsen Mathis

Project :
Back From Space



IMU2Track v3 results

August 27, 2025

Contents

List of Figures

1 Introduction

The core objective of the `IMU2Track v3` script is to implement a sensor fusion algorithm (based on Madgwick's fusion algorithm) to estimate 3D orientation and position from IMU sensor data. It processes raw sensor inputs, applies filtering, and then outputs orientation (in quaternion form), velocity, and position over time. Core math steps include computing the gradient from the Jacobian and residual, normalizing update vectors, and integrating to track motion.

2 Applications and motivation

Inertial Measurement Units (IMUs) are widely used in applications that require precise tracking of orientation and motion in three-dimensional space. Common uses include robotics, where IMUs help stabilize and navigate mobile platforms; aerospace, for flight control and navigation; and virtual or augmented reality, where accurate head or body tracking is essential. IMUs are also critical in autonomous vehicles, drones, and wearable devices, providing real-time data about position and movement when external references such as GPS are unavailable or unreliable. By combining accelerometers, gyroscopes, and magnetometers, IMUs enable robust estimation of orientation, velocity, and displacement, making them versatile sensors for both research and industrial applications.

The primary motivation for implementing an IMU-based tracking system in the BFS project is that during reentry, traditional positioning systems like GPS may become unreliable due to signal loss, high speeds, or ionization in the atmosphere. Therefore, our IMU based tracking system can assist in estimating orientation, velocity and position in space when GPS signal is too weak, allowing for real-time prediction of its trajectory. This capability is essential to estimate the landing site accurately, ensure safety, and support mission planning. By maintaining a reliable and continuous prediction of the vehicle's path, IMU-based tracking enables more precise navigation and operational control under extreme conditions.

3 Inertial Measurement Units (IMUs)

An Inertial Measurement Unit (IMU) is an electronic device that measures linear acceleration, angular velocity, and sometimes magnetic field vectors (it is the case for our IMU sensor). It typically combines three orthogonal accelerometers, three orthogonal gyroscopes, and a three-axis magnetometer into a single sensor package. Hence why they are called 6 dof or 9 dof (degrees of freedom). These measurements are expressed in standard units: acceleration in meters per second squared (m/s^2), angular velocity in radians or degrees per second (rad/s or °/s), and magnetic field strength in microteslas (µT).

Accelerometers detect forces along the IMU's axes, including the acceleration due to gravity. Gyroscopes measure rotation rates around each axis, providing angular motion information. Magnetometers, when present, give absolute heading relative to the Earth's magnetic field. These raw sensor outputs are often noisy and subject to biases and drift, which must be corrected through calibration and filtering.

Sensor fusion algorithms combine accelerometer, gyroscope, and magnetometer data to estimate orientation in roll, pitch, and yaw angles.

IMU are exposed to a major problem if used for navigation purposes. Acceleration data is integrated twice over time to estimate velocity and position, therefore this integration accumulates errors without external references.

Implementing a fusion algorithm helps reduce this error, since inaccurate gyroscopic measurements introduce errors when converting from the body frame to the world frame.

The IMU sensor used to acquire data in our application is the **ICM 20948**. The main challenge we face is mitigating the drift from integration using a low grade MEMS IMU sensor [1].

4 Sensor Models

An Inertial Measurement Unit (IMU) typically integrates three types of sensors:

4.1 Accelerometer

The accelerometer measures the *specific force* $\mathbf{f}$ in m/s^2, which includes both linear acceleration and the acceleration due to gravity:

$$\mathbf{f} = \mathbf{a} - \mathbf{g} \tag{1}$$

where:

- $\mathbf{a}$ is the total acceleration of the sensor in the global frame,
- $\mathbf{g}$ is the gravity vector (approximately $[0, 0, 9.81]^T$ m/s^2 in ENU frame).

Most sensors are not factory-calibrated to compensate for gravity, which is acceptable since the Madgwick algorithm requires an uncompensated IMU.

[1] A MEMS IMU (Micro-Electro-Mechanical System Inertial Measurement Unit) combines accelerometers, gyroscopes, and sometimes magnetometers into a single device to measure motion and orientation. *https://guidenav.com/what-is-a-mems-imu/*

4.2 Gyroscope

The gyroscope measures the angular velocity $\boldsymbol{\omega}$, generally in radians per second (if not in degrees/s) around each sensor axis:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2}$$

4.3 Magnetometer

The magnetometer measures the local magnetic field vector $\mathbf{m}$ in microteslas (μT):

$$\mathbf{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{3}$$

For magetometer measurements in particular, both hard and soft iron calibrations are necessary to ensure accurate orientation estimation. You can find information on how this is done in the appendix.

These raw sensor measurements form the basis for estimating the orientation, velocity, and position of the system.

5 Orientation and introduction to quaternions

There are several ways to represent the orientation of an IMU:

5.1 Euler Angles

Euler angles (ϕ, θ, ψ) represent roll, pitch, and yaw. They are intuitive but can suffer from *gimbal lock*. Gimbal lock occurs when two of the three rotational axes in an Euler angle representation become aligned, causing the loss of one degree of freedom. In this situation, certain rotations cannot be distinguished, and small changes in orientation can lead to large, unpredictable changes in the Euler angles.

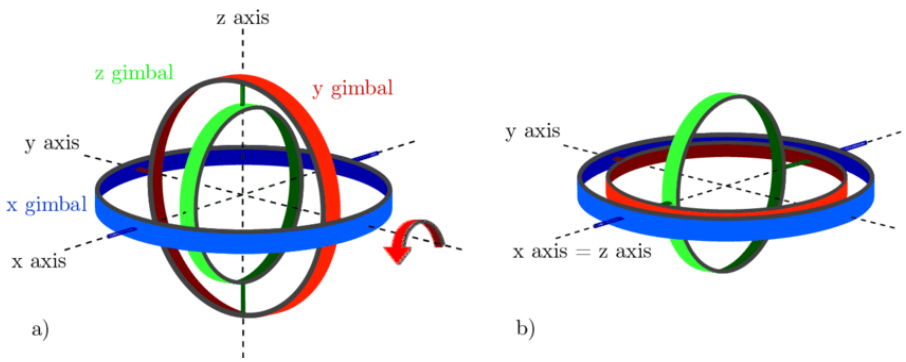


Figure 1: Gimbal lock example, *Julian Zeitlhöfler, researchgate*

This is why Euler angles can be problematic for 3D orientation tracking, especially in applications involving continuous rotations. It is however not the only method to estimate orientation, there is another that does not suffer from the gimbal lock problem : quaternions.

5.2 Quaternions

A quaternion q is represented as:

$$q = w + xi + yj + zk$$

where w, x, y, z are real numbers and i, j, k are fundamental quaternion units satisfying the following properties:

$$i^2 = j^2 = k^2 = ijk = -1$$

Quaternions are a four-component mathematical representation of rotation. They can represent any 3D rotation without the singularities and ambiguities of Euler angles:

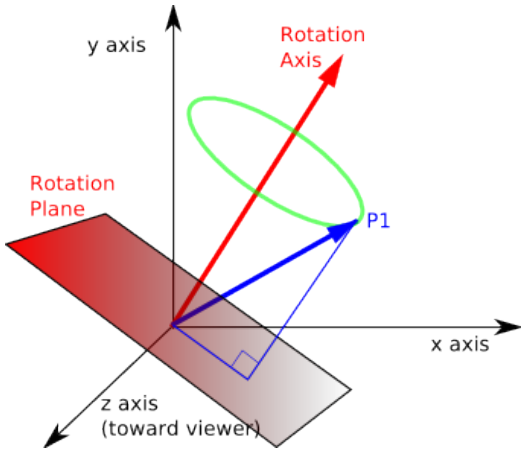


Figure 2: Quaternion representation, *ece.montana.edu*

6 Madgwick filter

The Madgwick filter is an efficient *Attitude and Heading Reference System (AHRS)* algorithm that estimates the 3D orientation of an object using measurements from an IMU: gyroscope, accelerometer, and optionally magnetometer. Below you will find a simplified explanation of the Madgwick filter implementation.

6.1 Gyroscope-based orientation prediction

The gyroscope measures angular velocity $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ in the sensor (body) frame. Using quaternion representation, the orientation derivative is:

$$\dot{q} = \frac{1}{2} q \otimes \omega_q \tag{4}$$

where:

- $q = [q_0, q_1, q_2, q_3]^T$ is the current orientation quaternion,
- $\otimes$ denotes quaternion multiplication,
- $\omega_q = [0, \omega_x, \omega_y, \omega_z]^T$ is the angular velocity expressed as a quaternion.

This provides a predicted orientation based solely on the gyroscope. However, due to gyroscope bias and drift, this prediction accumulates errors over time.

6.2 Accelerometer and magnetometer correction

The accelerometer and magnetometer provide absolute references:

- The accelerometer measures the gravity vector, allowing pitch and roll estimation.
- The magnetometer measures the Earth's magnetic field, allowing yaw (heading) estimation.

The filter compares the measured vectors $\mathbf{a} = [a_x, a_y, a_z]^T$ and $\mathbf{m} = [m_x, m_y, m_z]^T$ (normalized) with the predicted vectors from the current quaternion. The *objective function* to minimize for the accelerometer is:

$$\mathbf{f}(q) = \begin{bmatrix} 2(q_1 q_3 - q_0 q_2) - a_x \\ 2(q_0 q_1 + q_2 q_3) - a_y \\ 2(0.5 - q_1^2 - q_2^2) - a_z \end{bmatrix} \tag{5}$$

This measures the difference between the predicted gravity direction and the measured accelerometer vector.

6.3 Gradient descent correction

To correct the quaternion estimate, the filter performs a gradient descent step:

1. Compute the Jacobian J of $\mathbf{f}(q)$ with respect to q.
2. Compute the gradient:
$$\mathbf{s} = J^T \mathbf{f}(q) \tag{6}$$
3. Normalize the gradient:
$$\mathbf{s} \leftarrow \frac{\mathbf{s}}{\|\mathbf{s}\|} \tag{7}$$

This step ensures that the quaternion update moves in the direction that reduces the error between predicted and measured vectors.

6.4 Quaternion update

The quaternion derivative is corrected using the gradient:

$$\dot{q} = \frac{1}{2} q \otimes \omega_q - \beta \mathbf{s} \tag{8}$$

where β is the filter gain controlling the trade-off between convergence speed and sensitivity to measurement noise.

The quaternion is then updated using numerical integration (e.g., Euler method):

$$q_{t+\Delta t} = q_t + \dot{q}\, \Delta t \tag{9}$$

Finally, the quaternion is normalized to ensure it remains a valid rotation:

$$q \leftarrow \frac{q}{\|q\|} \tag{10}$$

6.5 Advantages

- Computational efficiency, suitable for embedded systems.
- Reduced gyroscope drift by fusing accelerometer and magnetometer measurements.
- Smooth and continuous orientation estimation without gimbal lock, thanks to quaternion representation.
- Tunable filter gain β to balance responsiveness and noise rejection.

7 Python script implementation and explanation

The code is composed of five class objects, each containing multiple methods that are explained in this section.

7.1 IMUConfig Class

The `IMUConfig` class centralizes all constants and parameters necessary for accurately computing orientation, velocity, and position, while providing the ability to fine-tune the implemented algorithm.

Here is a comprehensive list of all tunable parameters and hwo they will affect results:

- **sample_rate** [Hz]: The IMU data acquisition frequency. A higher rate captures faster dynamics but increases processing load. Too low a rate can cause aliasing and loss of motion details.

- **bias_estimation_window** [samples]: The number of initial samples used to estimate sensor biases. Larger windows yield more stable bias estimates but require longer stationary periods at startup.

- **madgwick_beta**: The gain controlling the correction strength in the Madgwick filter. Higher values allow faster convergence to sensor measurements (better for large initial errors) but amplify noise; lower values produce smoother estimates but slower response.

- **acc_lpf_cutoff** [Hz]: Low-pass filter cutoff for accelerometer data. Lower cutoffs remove more noise but can attenuate high-frequency motion signals.

- **gyro_hpf_cutoff** [Hz]: High-pass filter cutoff for gyroscope data. Higher cutoffs remove more low-frequency drift but may reduce stability for slow rotations.

- **MARG**: Boolean to indicate wether the Madgwick algorithm is to be used in MARG[2] mode or IMU-only. In other words wether you want magnetometer data to be integrated or not.

- **gravity_magnitude** [m/s^2]: Assumed gravitational acceleration magnitude. Should be close to 9.81; incorrect values bias accelerometer-based orientation estimation.

- **gravity_alignment_samples** [samples]: Number of samples used for initial gravity vector alignment. Larger values improve accuracy but delay initialization.

[2]MARG systems, also known as AHRS (Attitude and Heading Reference Systems) are able to provide a complete measure- ment of orientation relative to the direction of gravity and the earth's magnetic field. *Estimation of IMU and MARG orientation using a gradient descent algorithm. Sebastian O.H. Madgwick, Andrew J.L. Harrison, Ravi Vaidyanathan*

- **max_velocity** [m/s]: Maximum plausible velocity. Values exceeding this threshold may be rejected as unrealistic.

- **max_acceleration** [m/s^2]: Maximum plausible acceleration. Used to filter out spurious spikes from sensor noise or shocks.

- **outlier_threshold**: Number of standard deviations beyond which a measurement is considered an outlier and discarded.

- **smoothing_window** [samples]: Temporal smoothing window length. Larger windows produce smoother outputs but increase latency.

If you wish to modify those parameters, feel free to do so, as long as you know what you are doing. If you have read this document you should be able to tune these parameters in a safe and efficient way.

7.2 BiasEstimator Class

The `BiasEstimator` class is responsible for estimating and removing systematic sensor biases in IMU measurements. Biases are constant or slowly varying offsets that, if uncorrected, accumulate over time and degrade the accuracy of orientation, velocity, and position estimates. Bias is currently estimated when the sensor is at rest, therefore not dynamically updated.

7.2.1 initialize_bias method

This method estimates the sensor biases using stationary IMU data collected at the beginning:

- A window of the first n_{samples} data points is selected, with

$$n_{\text{samples}} = \min\left(n_{\text{window}},\ N\right),$$

  where n_{window} is the configured bias estimation window size and N is the number of available samples.

- **Gyroscope bias:** Since the system is stationary, the expected angular velocity is zero. The gyroscope bias is estimated as the average value:

$$\mathbf{b}_{\text{gyro}} = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} \boldsymbol{\omega}_i$$

  where $\boldsymbol{\omega}_i$ are the gyroscope measurements.

- **Accelerometer bias:** First, compute the mean acceleration vector:

$$\mathbf{a}_{\text{mean}} = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} \mathbf{a}_i$$

  Then scale it to have the correct gravity magnitude:

$$\mathbf{a}_{\text{scaled}} = \mathbf{a}_{\text{mean}} \cdot \frac{g}{\|\mathbf{a}_{\text{mean}}\|}$$

Finally, the accelerometer bias is the difference between the raw mean and the scaled vector:

$$\mathbf{b}_{\text{acc}} = \mathbf{a}_{\text{mean}} - \mathbf{a}_{\text{scaled}}$$

This approach does not assume perfect alignment of the sensor axes with gravity. Instead, it enforces that the norm of the corrected mean equals the expected gravity magnitude, thereby compensating for misalignment and ensuring consistent scaling.

7.2.2 correct_measurements method

Subtracts the computed biases from incoming measurements:

$$\mathbf{a}_{\text{corrected}} = \mathbf{a} - \mathbf{b}_{\text{acc}}$$

$$\boldsymbol{\omega}_{\text{corrected}} = \boldsymbol{\omega} - \mathbf{b}_{\text{gyro}}$$

If the biases are not yet initialized, the raw measurements are returned unchanged. It returns corrected accelerometer and gyroscope measurements for use in orientation or sensor fusion algorithms.

By removing these biases, the algorithm ensures more accurate orientation, velocity, and position estimates, especially over long periods of integration.

7.3 MadgwickFilter Class

The `MadgwickFilter` class implements a version of the Madgwick AHRS filter following the mathematical logic explained in *S. Madgwick PHD thesis* and the *AHRS library doc*[3]. It estimates 3D orientation from IMU measurements using a gradient descent algorithm to minimize the difference between predicted and measured sensor vectors.

7.3.1 Quaternion operations helper methods

- `_quaternion_product(a,b)`: Computes the Hamilton product of two quaternions a and b:

$$q = a \otimes b$$

- `_quaternion_conjugate(q)`: Computes the quaternion conjugate:

$$q^* = [q_0, -q_1, -q_2, -q_3]^T$$

These operations are used to rotate vectors between reference frames and compute the magnetic field reference direction.

7.3.2 update method

Here is a simplified approach to what this method does:

- Normalizes accelerometer and magnetometer measurements:

$$\mathbf{a}_{\text{norm}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}, \quad \mathbf{m}_{\text{norm}} = \frac{\mathbf{m}}{\|\mathbf{m}\|}$$

[3]Madgwick orientation filter library, you can find the link in the references at the end of this document

- Computes the reference direction of the Earth's magnetic field in the body frame using quaternion rotation:
$$\mathbf{h} = \mathbf{q} \otimes [0, \mathbf{m}] \otimes \mathbf{q}^*$$
- Constructs the *objective functions* for accelerometer (f_{acc}) and magnetometer (f_{mag}) to represent the difference between predicted and measured vectors.
- Builds the Jacobian matrices J_{acc} and J_{mag} and merges them to compute the gradient:
$$\mathbf{s} = \frac{J^T \mathbf{f}}{\|J^T \mathbf{f}\|}$$
- Computes the quaternion derivative from gyroscope measurements:
$$\dot{\mathbf{q}}_{\text{gyro}} = \frac{1}{2} \mathbf{q} \otimes [0, \boldsymbol{\omega}]$$
- Applies gradient descent feedback:
$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_{\text{gyro}} - \beta \mathbf{s}$$
- Integrates over time step dt and normalizes the quaternion:
$$\mathbf{q}_{t+dt} = \frac{\mathbf{q}_t + \dot{\mathbf{q}} dt}{\|\mathbf{q}_t + \dot{\mathbf{q}} dt\|}$$

7.3.3 `update_imu_only` method

If no magnetometer data is available, the filter falls back to using only accelerometer and gyroscope measurements:

- Uses only the accelerometer objective function and Jacobian to compute the gradient descent step.
- Computes quaternion derivative from gyroscope and applies feedback as before.
- Integrates and normalizes the quaternion.

7.4 MotionConstraints Class

The `MotionConstraints` class enforces kinematic limits to reduce drift in velocity and acceleration estimates from IMU data.

7.4.1 `constrain_velocity` method

- Computes the velocity magnitude:
$$\|\mathbf{v}\| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$
- If the magnitude exceeds the maximum allowed velocity (`max_velocity`), scales the velocity vector to satisfy the limit:
$$\mathbf{v}_{\text{corrected}} = \mathbf{v} \frac{\texttt{max\_velocity}}{\|\mathbf{v}\|}$$
- Stores the corrected velocity in the history buffer.

7.4.2 `detect_outliers` method

- Computes the acceleration magnitude:

$$\|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- Returns `True` if the acceleration exceeds the configured maximum (`max_acceleration`), marking it as an outlier.

7.5 IMUTracker Class

The **IMUTracker** class processes data, computes orientation using the Madgwick filter class and integrates acceleration to obtain velocity and position. It also formats the results for post processing.

Initialization When an `IMUTracker` object is instantiated, it:

- Loads configuration parameters from the `IMUConfig` object.
- Initializes the `BiasEstimator` to remove initial accelerometer and gyroscope biases.
- Sets up the `MadgwickAHRS` algorithm for sensor fusion, with the user-defined filter gain.
- Initializes `MotionConstraints` to limit physically unrealistic motion.
- Defines initial state vectors for:
  - Position $\mathbf{p}_0 \in \mathbb{R}^3$ (m)
  - Velocity $\mathbf{v}_0 \in \mathbb{R}^3$ (m/s)
  - Acceleration $\mathbf{a}_0 \in \mathbb{R}^3$ (m/s^2)
  - Orientation quaternion $\mathbf{q}_0 = [1, 0, 0, 0]^T$
- Configures low-pass and high-pass Butterworth filters for accelerometer and gyroscope measurements to remove sensor noise.

The `initialize()` method sets the IMU's initial orientation such that its magnetometer readings align with the reference North vector in the navigation (world) frame. In our setup, the IMU's X-axis is oriented toward North.

Gravity alignment The `initialize()` method computes the initial orientation so that the IMU's accelerometer readings match the reference gravity vector in the navigation (world) frame:

$$\mathbf{g}_{\text{ref}} = [0, 0, -g]^T$$

The method calculates the rotation between the measured gravity vector in the body frame and $\mathbf{g}_{\text{ref}}$, then converts it into a quaternion to initialize $\mathbf{q}_0$.

Sample processing For each incoming IMU sample, the `process_sample()` method executes:

1. **Bias correction:** Subtracts estimated biases from raw accelerometer and gyroscope measurements.
2. **Orientation update:** Passes corrected IMU data to the Madgwick filter, updating the orientation quaternion $\mathbf{q}$.
3. **Frame transformation:** Converts measured acceleration from body frame to world frame using:
$$\mathbf{a}_w = R(\mathbf{q}) \cdot \mathbf{a}_b$$
where $R(\mathbf{q})$ is the rotation matrix from the current quaternion.
4. **Gravity compensation:** Removes gravity from $\mathbf{a}_w$.
5. **Outlier detection:** Flags measurements exceeding acceleration thresholds to prevent drift from erroneous spikes.
6. **Velocity integration:** Integrates valid accelerations over Δt to obtain velocity, applying maximum velocity constraints.
7. **Position integration:** Integrates constrained velocity to update position.
8. **State logging:** Stores historical states for smoothing and later plotting.

Batch Processing The `process_data()` method applies `process_sample()` iteratively to a dataset, returning results in a `DataFrame` with expanded columns for:

- Position (x, y, z)
- Velocity (v_x, v_y, v_z)
- Acceleration (a_x, a_y, a_z)
- Quaternion (q_w, q_x, q_y, q_z)
- Euler angles (roll, pitch, yaw)

Visualization The `plot_results()` method produces:

- A 3D trajectory plot of the reconstructed motion.
- Time-series plots for position, velocity, and acceleration components.
- Orientation plots showing Euler angles over time.

7.6 `process_imu_data` Function

Reads IMU measurements from an Excel file, preprocesses them, and runs the `IMUTracker` pipeline.

Steps

1. Load `IMUConfig` (default if none provided).
2. Read dataset with `pandas`.
3. Validate presence of accelerometer and gyroscope columns; optionally load magnetometer.
4. Remove rows with missing values.
5. Extract data arrays and generate a time vector (from timestamps or sampling rate).
6. Filters data using high and low pass filters.
7. Pass data to `IMUTracker.process_data()` for processing.
8. Plot and return tracking results as a `DataFrame`.

▲ Warning: If you wish to tune the parameters, please refer to section 7.1 and 12 if you are in a hurry, elsewise read the full document.

8 IMU axis convention

In our setup, the IMU axes follow a right-handed body frame convention. The **roll** angle corresponds to rotation about the **X-axis** and is positive when banking right, so counterclockwise (trigonometric direction) when viewed from the front of the vehicle. The **pitch** angle corresponds to rotation about the **Y-axis** and is positive when the nose tilts upward, also counterclockwise when viewed from the right side. The **yaw** angle corresponds to rotation about the **Z-axis** and is positive for counterclockwise rotation when viewed from above. This ensures consistency with the standard aviation convention (and aerospace). Here is a representation of this frame in space:

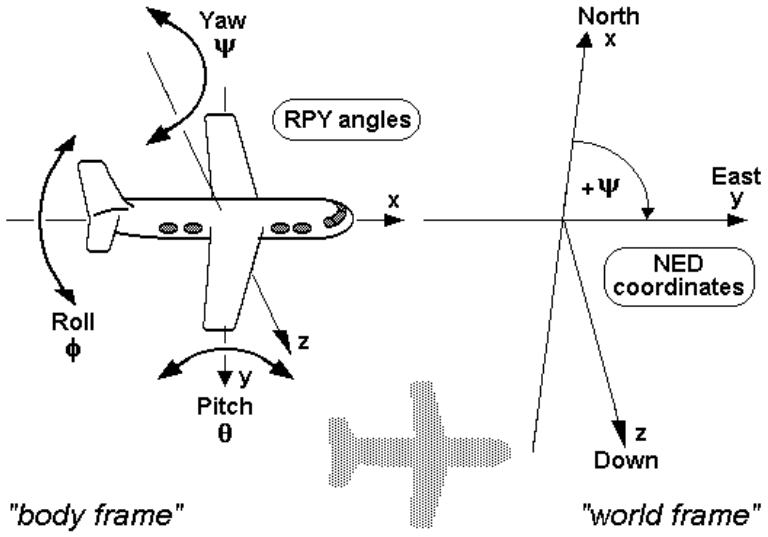


Figure 3: RPY angles of airplanes and other air vehicles, using a world frame as reference, *wikipedia.org*

9 Static IMU tests

Various static tests were performed to assess drift over time. Note that here pitch and roll axis are interchanged (they shouldn't be) but do not affect the results of the tests.

9.1 Full dead-reckoning

A full dead reckoning test was performed, to assess calibration quality. Sensor calibration was performed on 10000 samples (100 seconds). Here are the results:

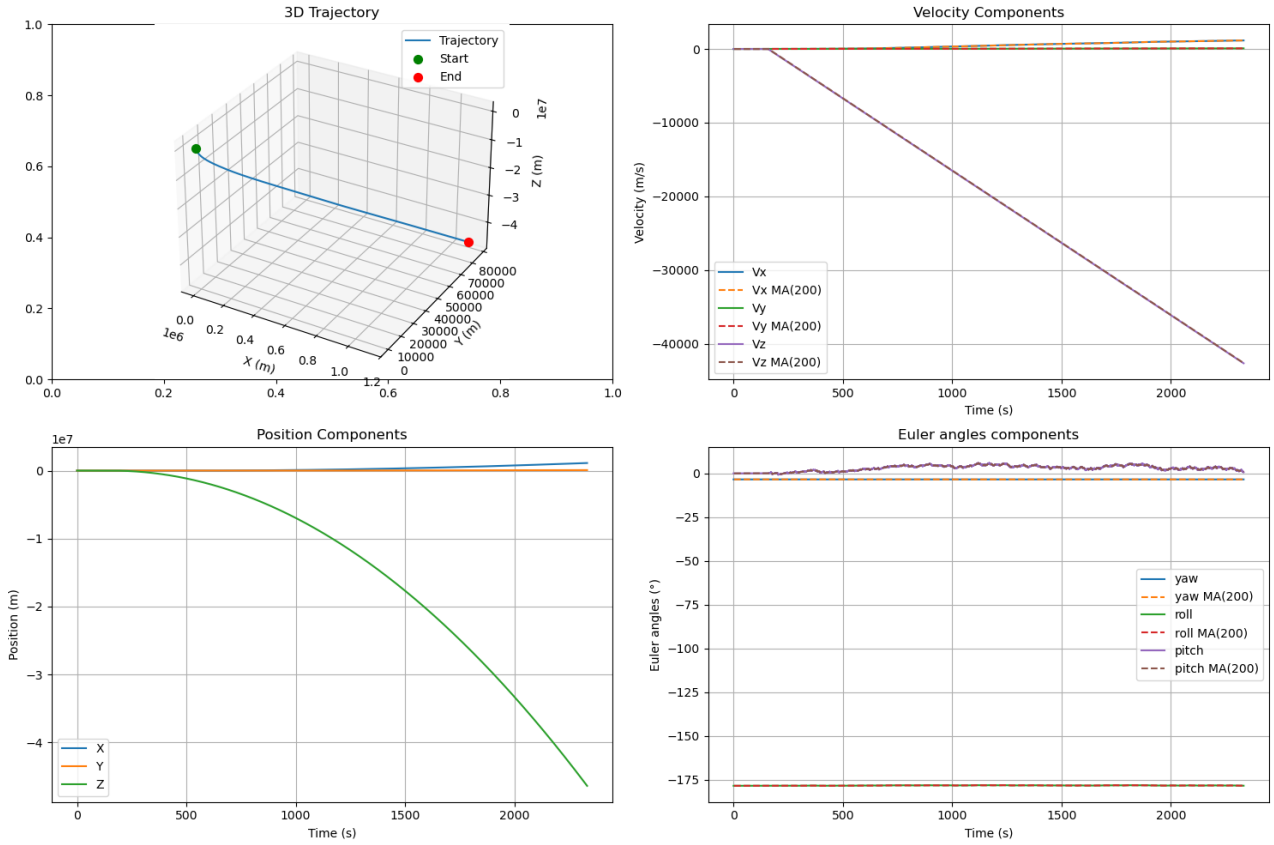


Figure 4: Tested at $100Hz$, $\beta = 0.00$, low pass filter cutoff at $10Hz$, high pass filter cutoff at $0.2Hz$, no velocity/acceleration constraints, no outlier detection

As expected in a pure dead-reckoning scenario, where the Madgwick corrective feedback is disabled ($\beta = 0$), the position estimates are obtained by integrating raw accelerometer readings over time. Any small errors in sensor bias, scale factor, or noise accumulate continuously, leading to significant drift.

From the results:

- The **Z-axis** shows a quadratic drift over time, as expected (Acceleration is integrated twice, once for velocity, which thus drifts linearly, and a second time for position, which drifts quadratically). The magnitude appears extremely high (on the order of 10^7), which translates to about **80000 km drift per hour**.
- The **Y-axis** is better calibrated, showing a drift of approximately **80 km over 2232 seconds** (about 37 minutes), which corresponds to **129 km of drift per hour**. This is consistent with expected behavior: small gyro and accelerometer biases accumulate linearly when integrating velocity and position without external corrections.
- The **X-axis** is the most poorly calibrated. The integrated acceleration (V_x) exhibits strong noise over time, resulting in a very large drift in position — nearly **2000 km per hour**. Interestingly, this drift seems to be related to noise in pitch. This is because pitch represents rotation about the lateral axis (Y-axis). Noise in pitch causes the accelerometer

measurements to be misprojected into the global frame, particularly affecting the forward (X) and vertical (Z) axes.

In summary, these results highlight why dead-reckoning alone cannot provide long-term accurate positioning. Even well-calibrated sensors produce drift that grows linearly or faster with time, with the drift rate heavily dependent on the sensor noise, bias, and axis calibration quality. The X-axis drift emphasizes the importance of orientation correction and precise gravity alignement.

On a positive note, this places our IMU between a consumer grade and an industrial grade, having a mean drift of about **27000 km**, according to the table from section 11.3.

9.2 Static test with constraints and Madgwick

The same static test was performed, this time with constraints, Madgwick activated, and outlier detection. Here are the results:

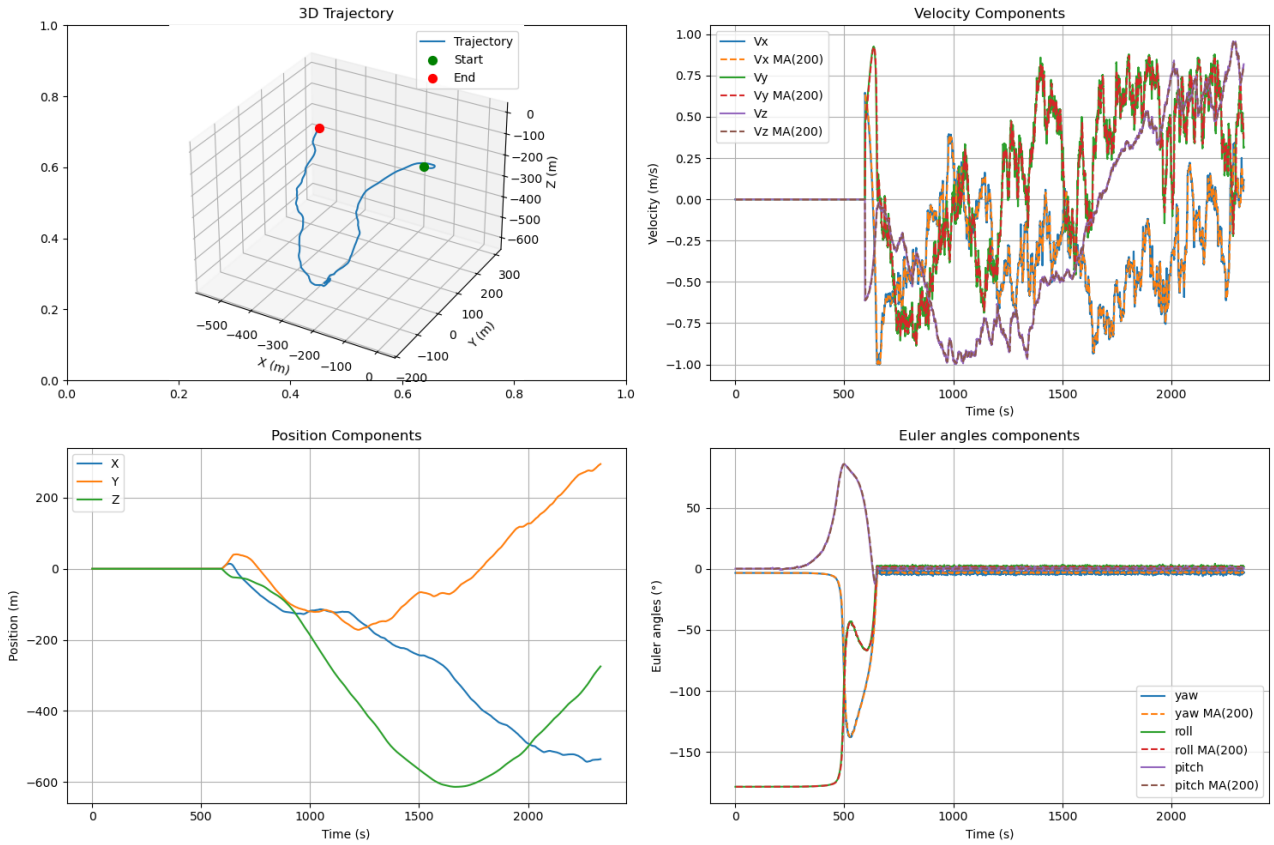


Figure 5: Tested at $100Hz$, $\beta = 0.041$, low pass filter cutoff at $10Hz$, high pass filter cutoff at $0.2Hz$, a constrained velocity to $1m.s^{-1}$ and a constrained acceleration to $12m.s^{-2}$

It is immediately clear that the drift has been drastically reduced: the total displacement error is only about 500 meters over 2232 seconds. The combination of Madgwick correction, motion constraints, and outlier rejection effectively limits the accumulation of errors, resulting in significantly improved position stability compared to the unconstrained, pure dead-reckoning test.

9.3 Conclusions on static tests and expected drift

Being of consumer to industrial grade, as is the case with our IMU, it is clear that it cannot provide highly accurate velocity and position estimates over extended periods. For precise navigation, especially in high-dynamic scenarios such as reentry, a higher-grade IMU would be necessary.

Regarding sensor calibration, the results indicate that calibration procedures generally perform well; however, performance is not uniform across all axes. Some axes, such as X, exhibit more noise and drift, while others, like Y, are comparatively stable. This uneven calibration performance is likely attributable to the intrinsic sensor quality, since the calibration procedure itself is applied identically to all axes.

10 Low dynamics IMU tests

10.1 IMU 10 Hz sample rate results / drift evaluation

It is important to note that these results are obtained under the assumption that this script will be used during reentry phases (therefore optimized for high dynamics). Consequently, techniques such as Zero-Velocity Updates (ZUPT[4]) and adaptive bias correction are not applicable in this context.

Moreover, the tests were conducted on data acquired while walking, as we do not have any reentry data (real or forged).

The algorithm was tested on real log data from our IMU system, acquired during our GPS latency tests. Velocity and acceleration constraints are applied to fit the nature of those tests. Here are the results obtained:

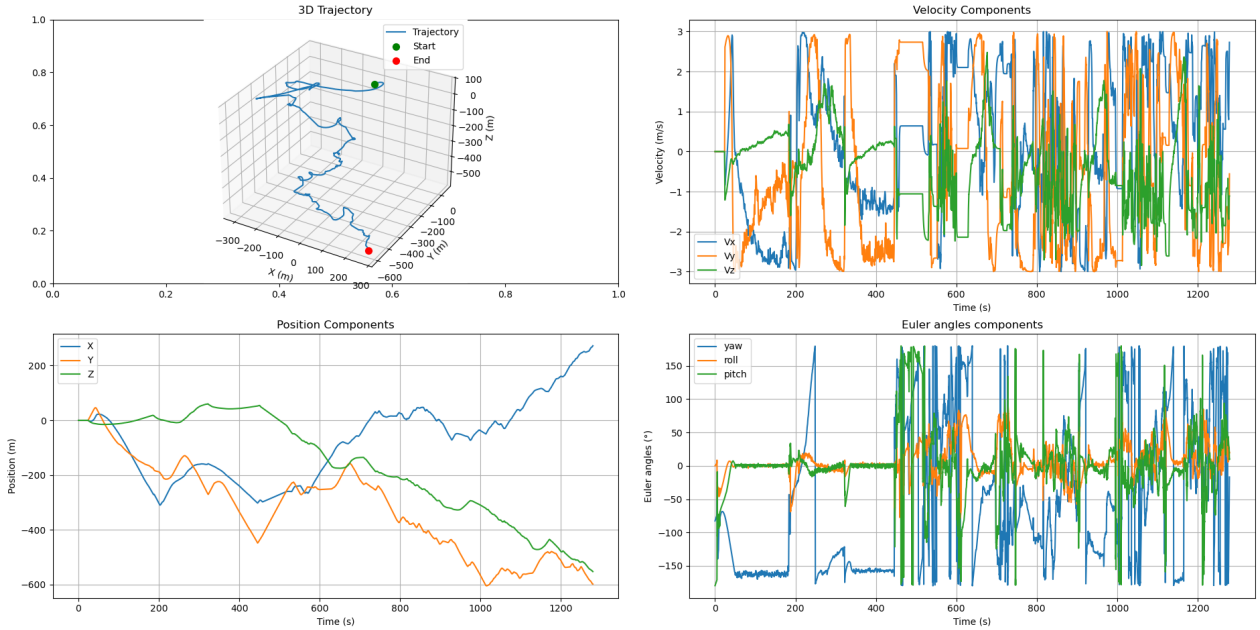


Figure 6: Tested at $10Hz$, $\beta = 0.06$, low pass filter cutoff at $10Hz$, high pass filter cutoff at $0.1Hz$, a constrained velocity to $3m.s^{-1}$ and a constrained acceleration to $8m.s^{-2}$

Multiple observations can be made from these results. Looking at the trajectory, we can see that at the start, drastic jitter appears (as well as in the velocity components). This might be due to the algorithm already running even though the first few minutes are stationary. A solution to this would be implementing a zero velocity detector or only start the algorithm when movement appears.

Position starts to drift significantly faster when movement occurs (around 600s in). This is probably because of the compounding of errors due to motion dynamics. The system is no longer integrating mostly-zero acceleration, it's integrating larger and more error-prone values.

On a positive note, the implementation of this IMU2Track script in its third version drastically reduced the drift by more than 99% compared to its second iteration (the second iteration

[4]Zero Update Position and Timing, is used to mitigate gyroscope bias drift and in turn mitigate heading drift in a GPS-Denied environment. It should be enabled when the user wants to reduce its accumulated orientation and position errors once a carrier object stops. *https://inertiallabs.com/what-is-zupt-and-when-do-i-use-it/*

had an improper fusion implementation and fundamental physics error when converting from body to world frame). The drift is now down to a couple hundred meters (on 1200 seconds), and could be reduced further by integrating adaptive bias values over different phases of flight/track and GPS recalibration.

10.2 IMU 100 Hz sample rate results / drift evaluation

To evaluate position drift, we established a set of points with known coordinates. The objective is to follow these points and then compute the position error over time during post-processing. Since the position error grows roughly quadratically, this approach allows us to estimate drift over any time interval. For this test, we were able to increase the IMU sample rate to 100 Hz. Here is the path followed to evaluate the drift:

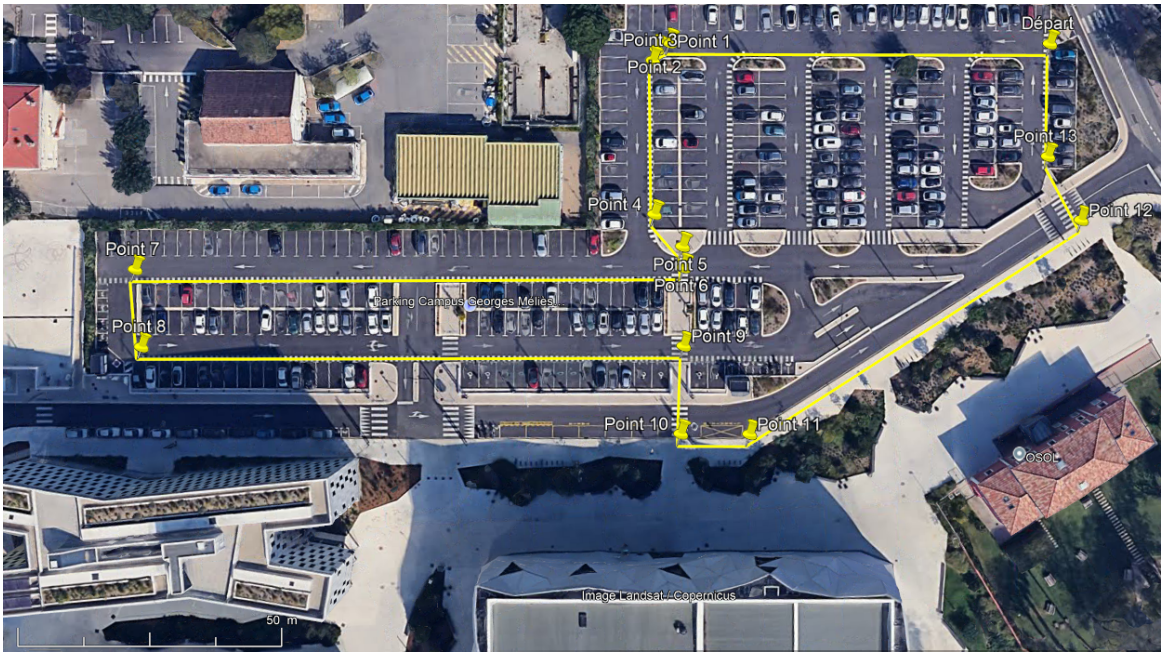


Figure 7: GPS path followed for drift evaluation tests

Here are the results observed after the test:

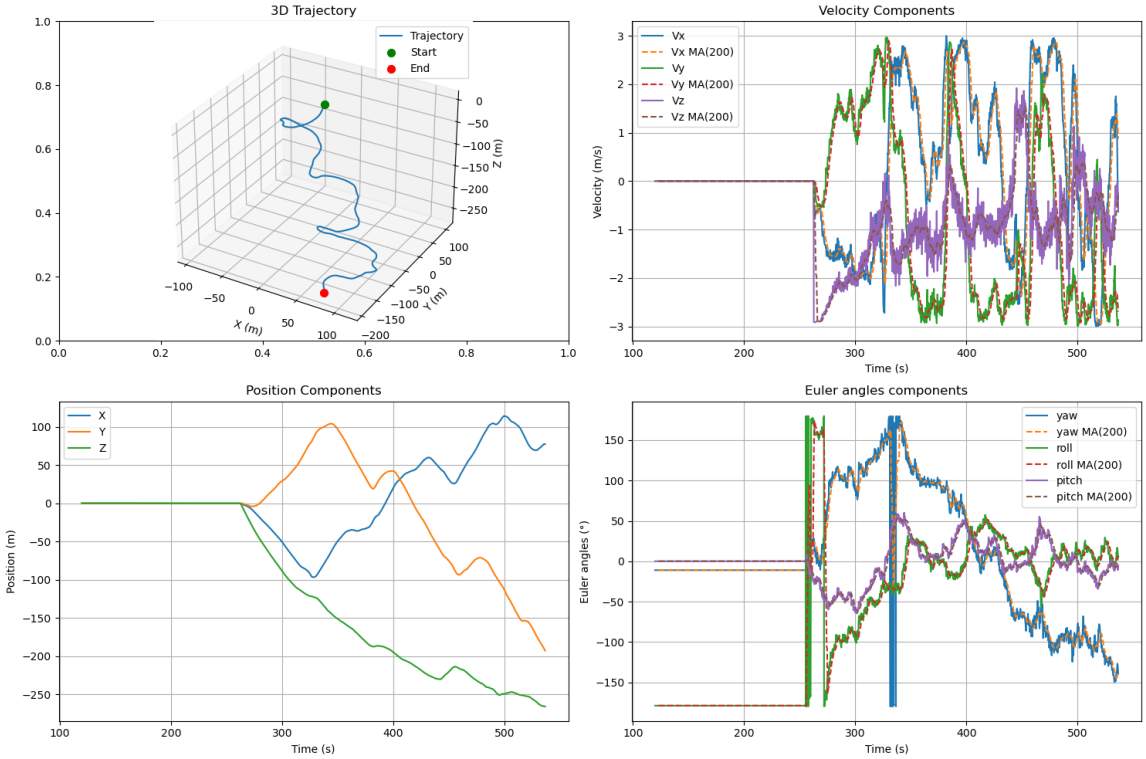


Figure 8: Tested at $100Hz$, $\beta = 0.041$, low pass filter cutoff at $25Hz$, high pass filter cutoff at $0.2Hz$, a constrained velocity to $3m.s^{-1}$ and a constrained acceleration norm of $15m.s^{-2}$

The results demonstrate a noticeable reduction in drift, particularly in the horizontal plane, where movements along the X and Y axes more closely match the expected trajectory. This improvement indicates that the implemented bias correction and filtering strategies are effective in stabilizing the position estimates in these directions.

However, a persistent drift remains in the Z axis. This vertical deviation is most likely linked to residual errors in orientation estimation, as even small inaccuracies in pitch or roll angles can significantly distort the projection of accelerometer measurements onto the gravity-compensated frame.

At the IMU sampling rate of 100 Hz, such small orientation and accelerometer biases accumulate rapidly during integration. Over the course of 325 seconds, this effect is sufficient to explain the observed 120 m drift in altitude, which highlights the sensitivity of vertical position estimates to orientation precision and bias compensation.
Here are better visuals of the results:

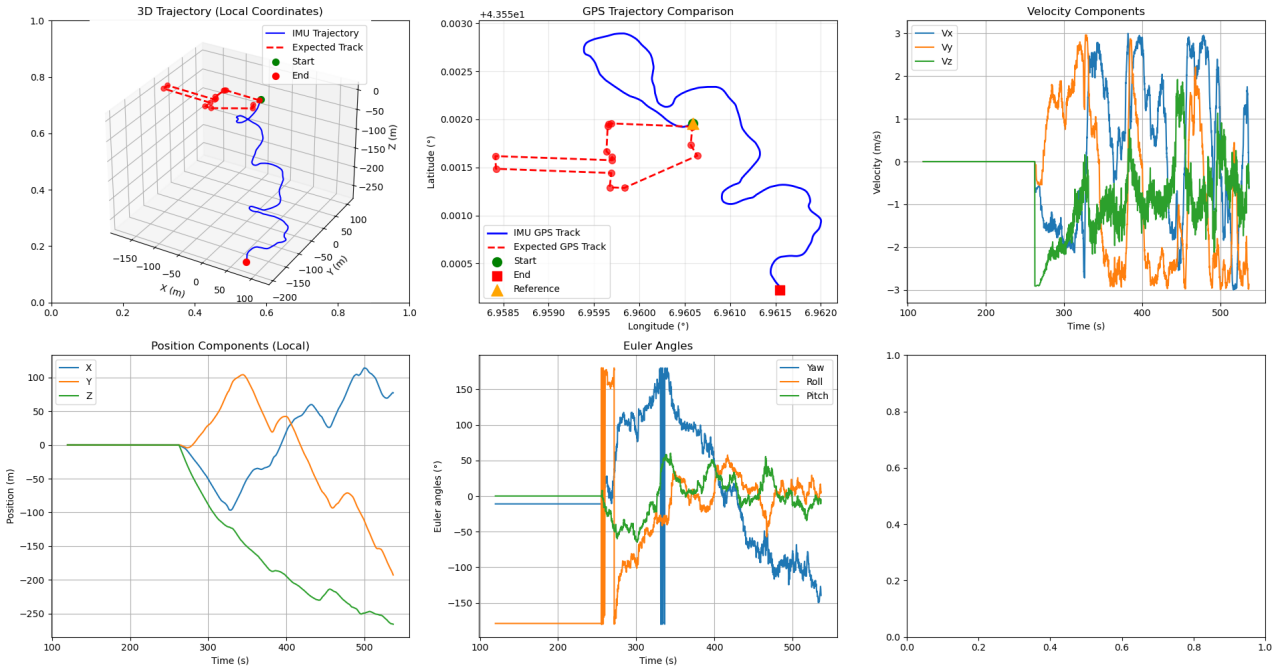


Figure 9: Tested at $100Hz$, $\beta = 0.041$, low pass filter cutoff at $25Hz$, high pass filter cutoff at $0.2Hz$, a constrained velocity to $3m.s^{-1}$ and a constrained acceleration norm of $15m.s^{-2}$ - GPS track comparison

It is evident that the reconstructed trajectory deviates significantly from the expected path. Not only does the track diverge in absolute position, but even the directions of movement are sometimes inconsistent with the true trajectory.

This misalignment suggests that the filter struggles to maintain an accurate heading estimate, leading to orientation-dependent errors when transforming accelerometer data into the global frame. As a result, accelerations may be projected into the wrong directions, causing the trajectory to bend or drift away from reality.

The reconstructed trajectory initially follows the expected path accurately for a few seconds but then begins to diverge. Interestingly, the number of turns in the reconstructed trajectory roughly matches the expected trajectory (around eight in total), but the turns are often in the wrong direction. This suggests a possible misunderstanding of the IMU's axis conventions, specifically, how positive and negative values are defined for each axis, which could lead to incorrect heading changes. Despite the divergence, this coherence in the number of turns indicates that the IMU data contains useful information. With a better understanding and interpretation of the IMU's output, we could achieve a more accurate reconstruction of the trajectory.

10.3 IMU 100 Hz results with correct axis alignment

After realizing that the previous tests had been conducted under the incorrect assumption that pitch corresponded to the X-axis rather than the Y-axis, we repeated the tests using the correct axis convention. The results obtained are presented below:

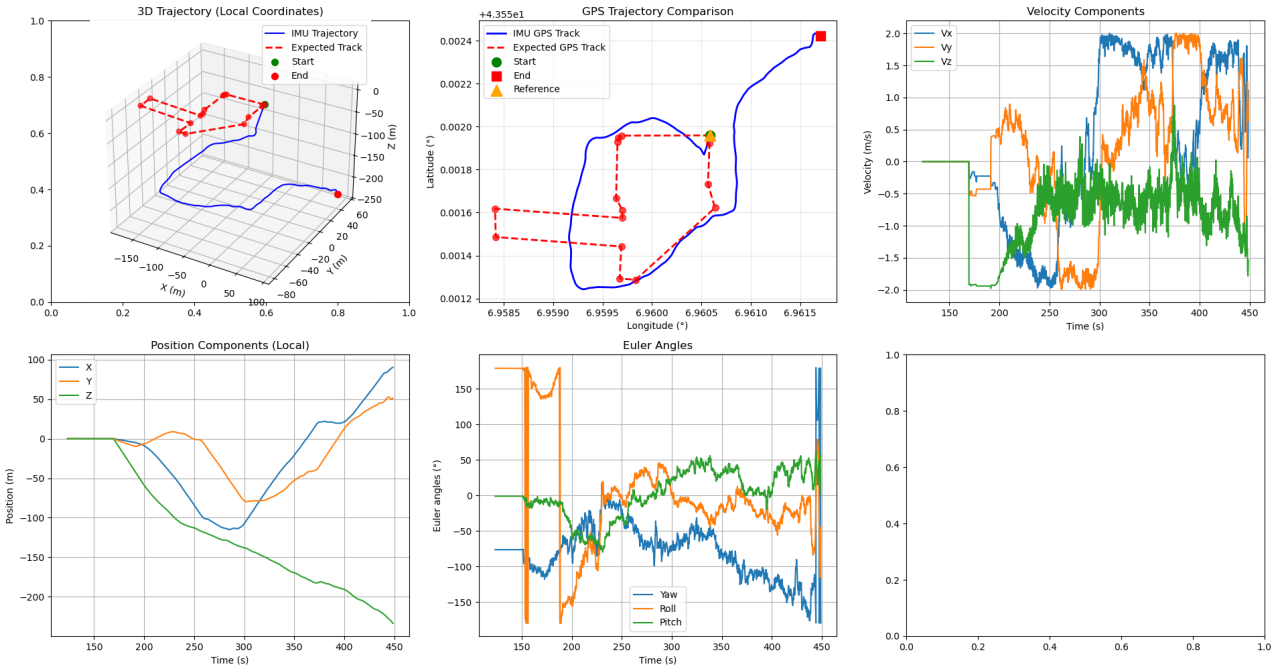


Figure 10: Tested at $100Hz$, $\beta = 0.015$, low pass filter cutoff at $25Hz$, high pass filter cutoff at $0.2Hz$, a constrained velocity to $2m.s^{-1}$ and a constrained acceleration norm of $15m.s^{-2}$ - GPS track comparison

We observe that the detected rotation directions are consistent with the expected trajectory. However, some rotations were not captured by the IMU. We also observed that, for this test, reducing the Madgwick filter parameter β, thereby giving more weight to the IMU measurements, improved the results, resulting in fewer rotations that appeared inconsistent with the expected motion. However, reducing the β also introduces more noise in the measurements, as can be seen with the computed Euler angles.

11 Conclusions on tests

Although some data have proven to be reconstructed consistently, and the number of turns is relatively consistent in every test, it becomes evident that a GPS-based recalibration mechanism is essential to correct accumulated drift and realign the trajectory (position and velocity) with precise readings.

In nearly all tests conducted with correctly aligned axes, we observed that the reconstructed trajectory remains accurate for approximately 15 seconds before consistently exhibiting drift. Additionally, our experiments showed that using a lower value of β generally yields more accurate results, whereas higher β values tend to introduce spurious rotations that do not correspond to the actual motion.

The Euler angles derived from the quaternions appear consistent and reliable, which aligns with the primary objective of the Madgwick filter. Though it has to be noted that in IMU-only case (no magnetometer data), yaw will drift arbitrarily as there is no magnetic reading to correct it.

Across multiple tests, we observed that the results are highly inconsistent, showing significant sensitivity to both parameter selection and calibration duration. This variability makes

the reconstructed trajectories unreliable and unsuitable for our application. Consequently, it is essential to implement a robust real-time recalibration mechanism to continuously correct for drift and ensure accurate trajectory estimation.

Nevertheless, the application of dynamic constraints combined with careful parameter tuning can significantly reduce drift. However, this approach becomes considerably more challenging to implement in highly dynamic environments, as it requires continuous real-time adaptation of parameters to maintain accuracy.

12 Tuning information and limitations

12.1 Beta value for Madgwick filter

The choice of the beta value (gain) for the Madgwick filter is very important in the stability of the algorithm. It must be chosen carefully. Here is an illustration that will help you chose the best value:

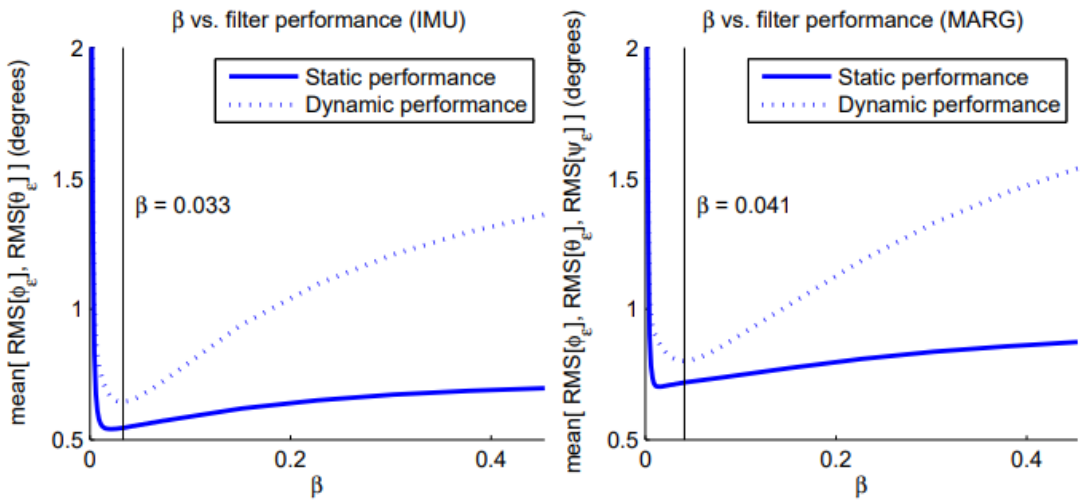


Figure 11: Effect of the adjustable parameter, β, on the performance of the proposed algorithm IMU (left) and MARG (right) implementations, *S.Madgwick PHD Thesis*

In theory, 0.041 is the best β value in dynamic conditions. Next steps would be to implement an adaptive β value over different phases of flight.

12.2 IMU sample rate

The Madgwick filter works best with IMUs capable of 100Hz or more. However, our application is limited in transmitted data to 10Hz, therefore we will never be able to have a perfect tracking estimation using transmitted data, especially in dynamic conditions. However, we found it possible to increase the IMU sample rate onboard, which would enable real-time tracking estimation directly on the system, though at the cost of higher computational demand.

Here is an illustration of this problem, from S.Madgwick thesis :

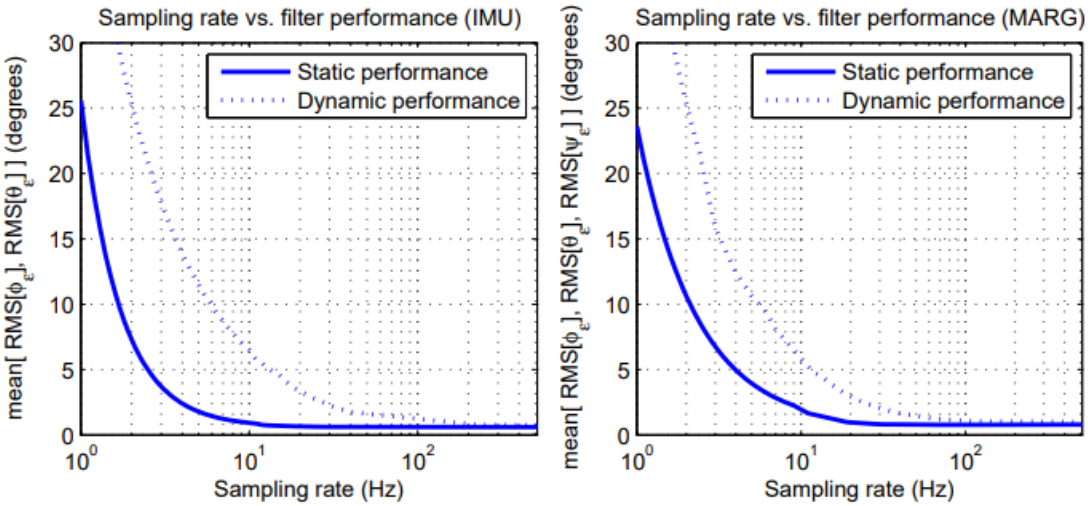


Figure 12: effect of sampling rate on the performance of the proposed algorithm IMU (left) and MARG (right) implementations, *S.Madgwick PHD Thesis*

The graphs above show that under dynamic conditions, as in our case, the Madgwick algorithm performs significantly better at a sample rate of at least $100Hz$ compared to $10Hz$. Using an IMU capable of $100Hz$ would allow us to reduce the mean orientation error by nearly a factor of five.

12.3 GPS data and trajectory plotting

In the current version of the script, no GPS data is implemented to correct velocity and position estimations. This greatly affects sensor drift, as no real time recalibration occurs, therefore integration error will accumulate over time. Here is a representation of generic error accumulation over sensor grade:

GRADE/TIME	1 s	10 s	60 s	10 min	1 hr
Consumer	6 cm	6.5 m	400 m	200 km	39,000 km
Industrial	6 mm	0.7 m	40 m	20 km	3,900 km
Tactical	1 mm	8 cm	5 m	2 km	400 km
Navigation	<1 mm	1 mm	50 cm	100 m	10 km

Figure 13: Error over time by sensor grade, *vectornav.com*

In the short term, implementing and testing GPS based data recalibration could significantly reduce drift, leading to a more accurate tracking estimate.
Today, GPS data still has to be extracted from sensor measurements data. Once extracted and converted to local frame coordinates, the trajectory will be overlayed with the estimated trajectory. This will provide us with a key metric for the validity and precision of our estimation.

12.4 Variable bias estimation

In the current version of the script, sensor bias is estimated once using a predetermined number of samples at the beginning of the dataframe. However, an enhanced version would be implementing a variable bias, either recalculated locally using a moving average method for example, or a predefined list of bias could be used over distinct phases of movements/flight. Here is a figure showing the difference between a simple average and a moving average:

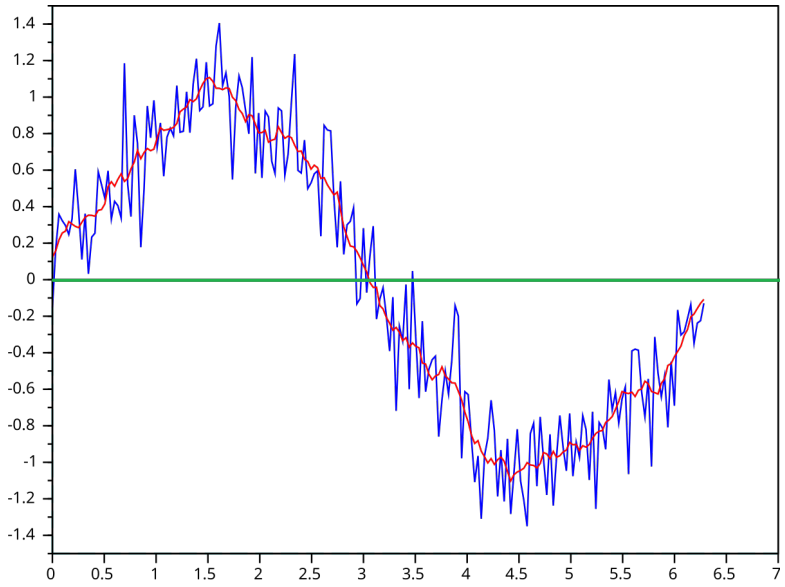


Figure 14: Moving average (red line) vs average (green line), *wikipedia.org*

We make the connection with our bias estimation, the green line being a fixed estimate, and the red line being an adaptative bias over time. It is obvious which is the best method.

12.5 Sensor redundancy

Currently, only one IMU sensor (3 tri-axis) is used in our system. However, using multiple sensors and combining the measurements can drastically reduce drift over time and provide more accurate measurements and thus estimations. Here is a graph showing how the number of sensors affects the total error (RMS):

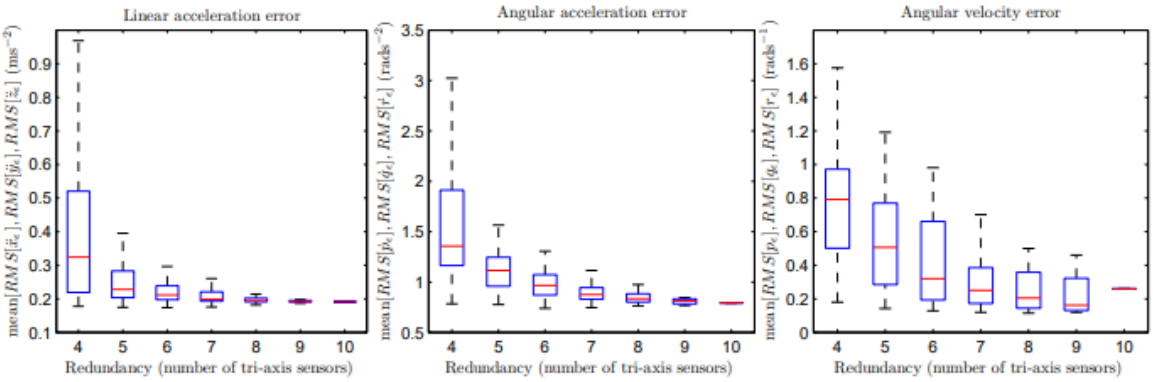


Figure 15: Relationship between error and sensor redundancy for linear acceleration (left), angular acceleration (middle), angular velocity (right). Whiskers indicate 1.5× interquartile range, *S.Madgwick PHD Thesis*

In our application, adding sensor redundancy would require more space and increase costs, but our specifications impose strict limits on both space and budget. As shown in these graphs, sensor redundancy becomes notably effective only with five or more combined IMUs, making it difficult to implement in our application.

12.6 Sensors temperature sensitivity

We know that sensors and, in particular, gyroscopes are sensitive to temperature changes, introducing a new kind of bias which will have to be accounted for, either post-processed or directly in our system. The difficulty of addressing this problem is that the temperature change and observed bias is a nonlinear relationship and unique to each device. Therefore, a specific bias evaluation model will have to be developed for our IMU. Here is a table from S.Madgwick PHD Thesis showing sensor biases over temperature, using a common low-grade sensor : MPU-6050, similar to the one we use in our system:

Parameter	Temperature sensitivity	Conditions
Gyroscope sensitivity	±2%	-
Gyroscope bias	±20°/s	-40°C to 85°C
Accelerometer sensitivity	±0.02%/°C	-40°C to 85°C
Accelerometer bias (x and y axis)	±35 mg	0°C to 70°C
Accelerometer bias (z axis)	±60 mg	0°C to 70°C
Magnetometer bias	-	-
Magnetometer sensitivity	-0.3%/°C	-

Figure 16: Temperature sensitivity of sensor parameters as specified in device datasheets. "-" indicates that the information is not provided, *S.Madgwick PHD Thesis*

We can see that the most important bias comes from the gyroscope, there is also additional bias on other sensors, though less important. Most high-grade IMUs are equipped with self-heating mechanisms, which is likely the most effective and easiest way to minimize temperature sensitivity.

12.7 Extended Kalman filter

After careful consideration, the use of an Extended Kalman Filter (EKF) may be more suitable for our application. Similar to the Madgwick filter, the EKF enables state prediction; however, it offers the additional flexibility of incorporating an arbitrary number of state variables. This makes it possible to estimate and adapt sensor biases at each step, as well as to correct position and velocity, thereby mitigating integration errors. To correct position and velocity, external measurements are needed, such as GPS readings, which would allow a much morebn precise tracking capability, adapted to our high dynamics needs.

12.8 Complementary sensors

Integrating complementary sensors such as a barometer or visual sensors could significantly enhance tracking accuracy. A barometer provides direct altitude measurements, which can

constrain drift in the vertical axis and improve height estimation. Visual sensors can be used through techniques like Visual-Inertial Odometry (VIO) or Simultaneous Localization and Mapping (SLAM), allowing the system to estimate relative motion and build a map of the environment (though more difficult to adapt in high dynamics and high altitude conditions).

13 Appendices

13.1 Definition and properties of quaternions

A quaternion q is represented as:

$$q = w + xi + yj + zk$$

where w, x, y, z are real numbers and i, j, k are fundamental quaternion units satisfying the following properties:

$$i^2 = j^2 = k^2 = ijk = -1$$

The conjugate of a quaternion $q = w + xi + yj + zk$ is given by:

$$q^* = w - xi - yj - zk$$

The norm of a quaternion is given by:

$$\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2}$$

For Madgwick filtering, the output norm should always be equal to one, since the quaternions are expected to be normalized during the process.

13.2 Rotation representation

Quaternions are useful for representing 3D rotations to avoid gimbal lock issues (which happens with euler angle use). A unit quaternion can represent a rotation about an axis $\mathbf{u} = (u_x, u_y, u_z)$ by an angle θ as follows:

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(u_x i + u_y j + u_z k)$$

13.3 Quaternion multiplication

Quaternion multiplication is used to combine rotations. For two quaternions $q1 = (w1, x1, y1, z1)$ and $q2 = (w2, x2, y2, z2)$, their product $q1 \times q2$ is given by:

$$\begin{aligned} q1 \cdot q2 = (&w1w2 - x1x2 - y1y2 - z1z2, w1x2 + x1w2 + y1z2 \\ &- z1y2, w1y2 - x1z2 + y1w2 + z1x2, w1z2 + x1y2 - y1x2 + z1w2) \end{aligned}$$

13.4 Conversion to Euler angles

Quaternions can be converted to Euler angles for more intuitive interpretation. The conversion formulas are as follows:

Let $q = (w, x, y, z)$ be a unit quaternion. The corresponding Euler angles (roll ϕ, pitch θ, yaw ψ) in radians can be found by:

$$\begin{aligned} \phi &= \text{atan2}(2(wy + zx), 1 - 2(x^2 + y^2)) \\ \theta &= \arcsin(2(wx - zy)) \\ \psi &= \text{atan2}(2(wz + xy), 1 - 2(y^2 + z^2)) \end{aligned}$$

13.5 Block diagram of the implemented Madgwick filter

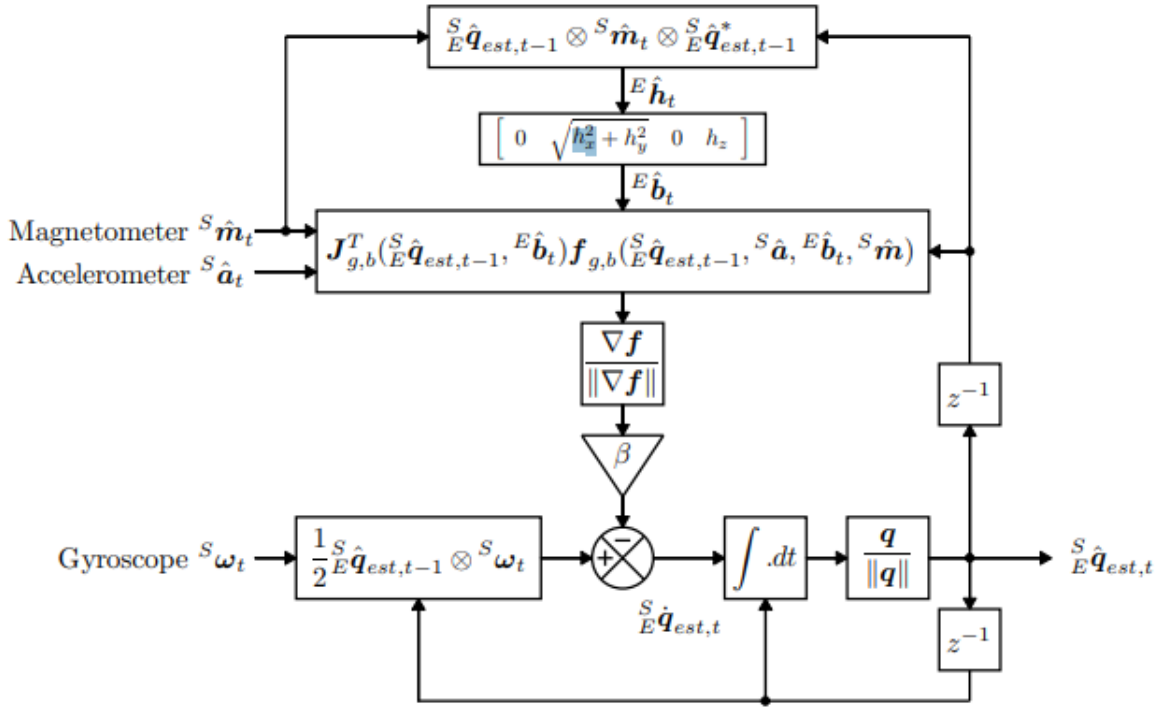


Figure 17: Block diagram of the algorithm, *S.Madgwick PHD thesis*

13.6 Equations of objective function and jacobian used in the Madgick filter implementation

$$\boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}, {}^S\hat{\boldsymbol{a}}) = \begin{bmatrix} 2(q_x q_z - q_w q_y) - a_x \\ 2(q_w q_x + q_y q_z) - a_y \\ 2(\frac{1}{2} - q_x^2 - q_y^2) - a_z \end{bmatrix}$$

$$\boldsymbol{J}_g({}^S_E\hat{\boldsymbol{q}}) = \begin{bmatrix} -2q_y & 2q_z & -2q_w & 2q_x \\ 2q_x & 2q_w & 2q_z & 2q_y \\ 0 & -4q_x & -4q_y & 0 \end{bmatrix}$$

Figure 18: Objective function (f) and jacobian (j) for IMU based Madgwick, *S.Madgwick PHD thesis*

$$\boldsymbol{f}_b({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{b}}, {}^S\hat{\boldsymbol{m}}) = \begin{bmatrix} 2b_x(0.5 - q_y^2 - q_z^2) + 2b_z(q_xq_z - q_wq_y) - m_x \\ 2b_x(q_xq_y - q_wq_z) + 2b_z(q_wq_x + q_yq_z) - m_y \\ 2b_x(q_wq_y + q_xq_z) + 2b_z(0.5 - q_x^2 - q_y^2) - m_z \end{bmatrix}$$

$$\boldsymbol{J}_b({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{b}}) = \begin{bmatrix} -2b_zq_y & 2b_zq_z \\ -2b_xq_z + 2b_zq_x & 2b_xq_y + 2b_zq_w \\ 2b_xq_y & 2b_xq_z - 4b_zq_x \\ -4b_xq_y - 2b_zq_w & -4b_xq_z + 2b_zq_x \\ 2b_xq_x + 2b_zq_z & -2b_xq_w + 2b_zq_y \\ 2b_xq_w - 4b_zq_y & 2b_xq_x \end{bmatrix}$$

Figure 19: Objective function (f) and jacobian (j) for MARG based Madgwick, *S.Madgwick PHD thesis*

13.7 Hard and soft iron magnetometer calibration

Magnetometer calibration is essential to correct measurement errors caused by hard and soft iron effects. It is therefore essential to have a well-calibrated magnetometer to have precise orientation estimates.

Hard iron distortions are constant offsets in magnetic field readings, usually caused by permanent magnets or magnetized materials near the sensor. Mathematically, these are represented as a bias vector $\mathbf{b} = [b_x, b_y, b_z]$, which is subtracted from the raw measurements $\mathbf{m}_{\text{raw}}$ to obtain a hard-iron-corrected reading $\mathbf{m}_h = \mathbf{m}_{\text{raw}} - \mathbf{b}$.

Soft iron distortions result from ferromagnetic materials that distort the local magnetic field, causing the measured field to appear as an ellipsoid rather than a sphere. These are corrected using a 3×3 transformation matrix $\mathbf{A}^{-1}$, computed by ellipsoid fitting on the hard-iron-corrected samples. The final calibrated magnetometer reading is then given by

$$\mathbf{m}_{\text{cal}} = \mathbf{A}^{-1}(\mathbf{m}_{\text{raw}} - \mathbf{b}).$$

This procedure ensures that the magnetometer outputs a normalized field vector suitable for accurate orientation estimation. Here is the output from the hard and soft iron calibration of our magnetometer:

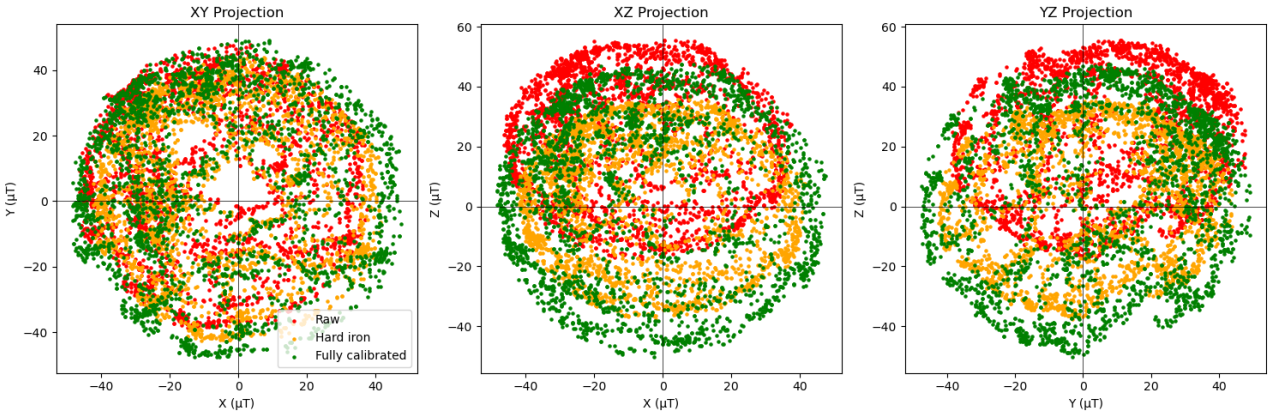


Figure 20: Graphs representing magnetometer data before and after calibrations

The raw measurements (red dots) are clearly offset and form an ellipsoidal shape. After full calibration (green dots), they form a centered sphere, indicating that the calibration was successful.

References

[1] Madgwick, S. (2010). *An efficient orientation filter for inertial and inertial/magnetic sensor arrays* (Doctoral dissertation, University of Bristol). Retrieved from `https://x-io.co.uk/downloads/madgwick-phd-thesis.pdf`

[2] AHRS. (n.d.). *AQUA filter*. AHRS documentation. Retrieved July 21, 2025, from `https://ahrs.readthedocs.io/en/latest/filters/aqua.html`

[3] Stanford University. (n.d.). *Lecture 10: Orientation representations*. EE267 Lecture Notes. Retrieved from `https://stanford.edu/class/ee267/lectures/lecture10.pdf`

[4] 3Blue1Brown. (2017, March 28). *Quaternions and 3D rotation* [Video]. YouTube. Retrieved from `https://www.youtube.com/watch?v=zjMuIxRvygQ&t=355s&ab_channel=3Blue1Brown`

[5] 3Blue1Brown. (2018, January 12). *Visualizing rotation in 3D* [Video]. YouTube. Retrieved from `https://www.youtube.com/watch?v=d4EgbgTm0Bg&ab_channel=3Blue1Brown`

[6] Wikipedia. (2025, July 18). *Quaternion*. In Wikipedia. Retrieved from `https://en.wikipedia.org/wiki/Quaternion`

[7] Wikipedia. (2025, August 18). *Moving average*. In Wikipedia. Retrieved from `https://en.wikipedia.org/wiki/Moving_average`

[8] VectorNav. (n.d.). *Specifications and error budgets: Inertial navigation primer*. Retrieved from `https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-inserrorbudget`