**MPU AXIS ALIGNMENT**

**How to draw vectors that are in or out of the plane of the page (or board)**

[[edit](https://en.wikibooks.org/w/index.php?title=Physics_Study_Guide/Vectors_and_scalars&veaction=edit&section=6) | [edit source](https://en.wikibooks.org/w/index.php?title=Physics_Study_Guide/Vectors_and_scalars&action=edit&section=6)]

How to draw vectors in the plane of the paperStandard symbols of a vector going into or out of a page

**Answer:** Vectors in the plane of the page are drawn as arrows on the page. A vector that goes into the plane of the screen is typically drawn as circles with an inscribed X. A vector that comes out of the plane of the screen is typically drawn as circles with dots at their centers. The X is meant to represent the fletching on the back of an arrow or dart while the dot is meant to represent the tip of the arrow.

[](https://en.wikibooks.org/wiki/File:Notation_for_vectors_in_or_out_of_a_plane.svg)

[AltIMU-AHRS/altimu10v3\_tiltcomp/altimu10v3\_tiltcomp.ino at master · jremington/AltIMU-AHRS (github.com)](https://github.com/jremington/AltIMU-AHRS/blob/master/altimu10v3_tiltcomp/altimu10v3_tiltcomp.ino)

/\*

This tilt-compensated code assumes that the sensor is oriented with Mag X pointing

to the North, Y pointing East, and Z pointing down (toward the ground).

With the MPU9250, the accelerometer is aligned differently, so the accelerometer axes are swapped.

The code compensates for tilts of up to 90 degrees away from horizontal.

Facing vector p is the direction of travel and allows reassigning these directions.

It should be defined as pointing forward,

parallel to the ground, with coordinates {X, Y, Z} (in magnetometer frame of reference).

\*/

[AZ3166 sensors orientation in Madgwick - Using Arduino / Sensors - Arduino Forum](https://forum.arduino.cc/t/az3166-sensors-orientation-in-madgwick/686003/8)

[**jremington**](https://forum.arduino.cc/u/jremington)

[Dec 2020](https://forum.arduino.cc/t/az3166-sensors-orientation-in-madgwick/686003/8)post #8

Hi, luc08

Hi, it's my first project with an Arduino board.

This is a very challenging "first project"! To understand 9DOF sensor data fusing, I strongly recommend [this discussion 18](http://www.olliw.eu/2013/imu-data-fusing/#chapter23).

As mentioned in reply #1, this call won't work with your sensor, because the axes of the sensors are not aligned consistently.

filter.update(gx,gy,gz,ax,ay,az,mx,my,mz);

The Madgwick/Mahony code assumes that the first sensor axis in the function call points to magnetic North, but the illustration in the sensor data sheet assumes that the accelerometer Y axis and the gyro Y (R) axis point toward magnetic North, while the magnetometer Y axis is assumed to point magnetic South.

Furthermore, the magnetometer axes are defined in a left handed coordinate system, which must be corrected.

If you want to stick with this data sheet convention, then I believe that the filter call should be as follows, and you have to **hold the sensor module as illustrated**, with accel Z up and Y North. Unfortunately, I don't have one of those sensors to test it.

filter.update(gy,gx,-gz,ay,ax,-az,-my,mx,-mz);

Notes: The Euler angles that you derive from the quaternion produced by the filter won't necessarily have the correct sign, because (1) there is no universal convention on what constitutes a positive rotation (and the mathematical definition of a positive rotation angle used in trigonometry does not agree with conventional navigation) and (2) there are around a dozen ways to define Euler angles. So feel free to change the sign of yaw, pitch or roll to suit yourself.

Finally, this is not necessary, as the filter uses normalized accelerometer and magnetometer readings (vector magnitude =1.0)

I converted ... mg to g and miligauss to µT.

**Post 10**[**jremington**](https://forum.arduino.cc/u/jremington)

[Dec 2020](https://forum.arduino.cc/t/az3166-sensors-orientation-in-madgwick/686003/10)post #10

Usually the x-axis should be aligned with the heading

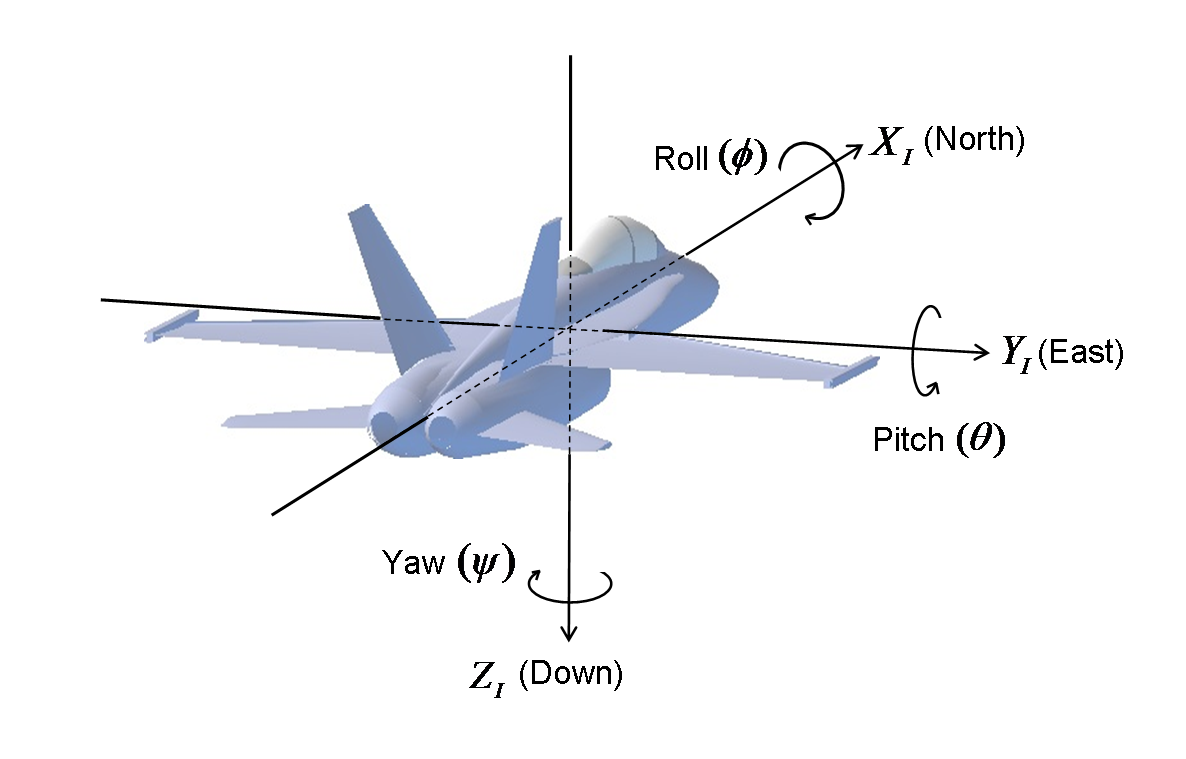
The naming of the sensor axes is completely arbitrary.

The illustration in the LSM6DSL data sheet assumes that the accel and gyro Y axes are pointing to magnetic North.

The Madgwick and Mahony filters assume that the sensor X axes are pointing to magnetic North, so you either have to orient the sensor accordingly, or swap axes in the code. And since the magnetometer axes are defined inconsistently, and in a left handed system, you have to swap those anyway or the filter won't work.

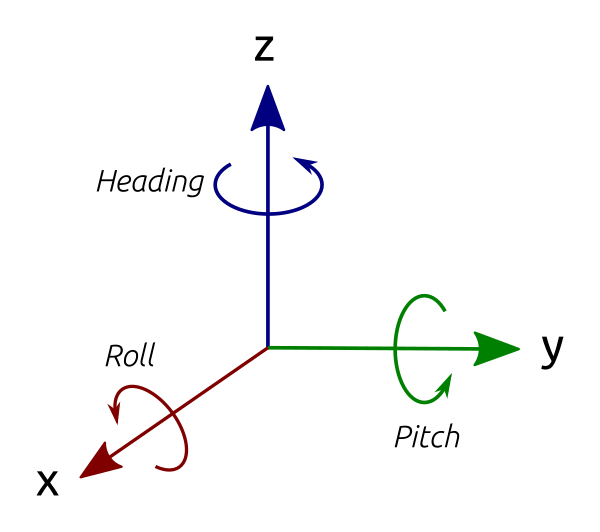
To have the accel/gyro X axis point at magnetic North for yaw=0, while holding the sensor so that Z points down, I think the call would be

filter.update(gx, gy, gz, ax, ay, az, mx, -my, mz);



[**https://www.npmjs.com/package/ahrs?activeTab=readme**](https://www.npmjs.com/package/ahrs?activeTab=readme)

The image below shows the orientation of the axis. The x-axis points north and the z-axis up. The input values should be rotated to match the orientation show. The output will be of the shown orientation.

**[](https://github.com/psiphi75/ahrs/blob/HEAD/doc/orientation.png)**

getEulerAngles()

Return an object with the Euler angles (heading/yaw, pitch, roll), in radians.

*returns:* Object where:

* heading is from north, going west (about z-axis).
* pitch is from vertical, going forward (about y-axis).
* roll is from vertical, going right (about x-axis).

The most important thing is to get your axis correct. Read the documentation of your device. Some devices (like the MPU9250) have the accelerometer and compass with a different orientation than the magnetometer. You will need to rotate the axis before you use the values for the AHRS algorithm.

<https://stackoverflow.com/questions/25902186/madgwick-sensor-fusion-on-lsm9ds0>

[Here](http://diydrones.com/forum/topics/madgwick-imu-ahrs-and-fast-inverse-square-root?id=705844%3ATopic%3A1018435&page=4#comments) is said that there is a mistake in code of madgwick filter!

Gradient decent step should look like this:

s0= -\_2q2\*(2\*(q1q3 - q0q2) - ax) + \_2q1\*(2\*(q0q1 + q2q3) - ay) + -\_4bz\*q2\*(\_4bx\*(0.5 - q2q2 - q3q3) + \_4bz\*(q1q3 - q0q2) - mx) + (-\_4bx\*q3+\_4bz\*q1)\*(\_4bx\*(q1q2 - q0q3) + \_4bz\*(q0q1 + q2q3) - my) + \_4bx\*q2\*(\_4bx\*(q0q2 + q1q3) + \_4bz\*(0.5 - q1q1 - q2q2) - mz);

s1= \_2q3\*(2\*(q1q3 - q0q2) - ax) + \_2q0\*(2\*(q0q1 + q2q3) - ay) + -4\*q1\*(2\*(0.5 - q1q1 - q2q2) - az) + \_4bz\*q3\*(\_4bx\*(0.5 - q2q2 - q3q3) + \_4bz\*(q1q3 - q0q2) - mx) + (\_4bx\*q2+\_4bz\*q0)\*(\_4bx\*(q1q2 - q0q3) + \_4bz\*(q0q1 + q2q3) - my) + (\_4bx\*q3-\_8bz\*q1)\*(\_4bx\*(q0q2 + q1q3) + \_4bz\*(0.5 - q1q1 - q2q2) - mz);

s2= -\_2q0\*(2\*(q1q3 - q0q2) - ax) + \_2q3\*(2\*(q0q1 + q2q3) - ay) + (-4\*q2)\*(2\*(0.5 - q1q1 - q2q2) - az) + (-\_8bx\*q2-\_4bz\*q0)\*(\_4bx\*(0.5 - q2q2 - q3q3) + \_4bz\*(q1q3 - q0q2) - mx)+(\_4bx\*q1+\_4bz\*q3)\*(\_4bx\*(q1q2 - q0q3) + \_4bz\*(q0q1 + q2q3) - my)+(\_4bx\*q0-\_8bz\*q2)\*(\_4bx\*(q0q2 + q1q3) + \_4bz\*(0.5 - q1q1 - q2q2) - mz);

s3= \_2q1\*(2\*(q1q3 - q0q2) - ax) + \_2q2\*(2\*(q0q1 + q2q3) - ay)+(-\_8bx\*q3+\_4bz\*q1)\*(\_4bx\*(0.5 - q2q2 - q3q3) + \_4bz\*(q1q3 - q0q2) - mx)+(-\_4bx\*q0+\_4bz\*q2)\*(\_4bx\*(q1q2 - q0q3) + \_4bz\*(q0q1 + q2q3) - my)+(\_4bx\*q1)\*(\_4bx\*(q0q2 + q1q3) + \_4bz\*(0.5 - q1q1 - q2q2) - mz);

and that code on official site is outdated and soon will be replaced.

Correcting this produced a satisfying result.

### [Madgwick IMU/AHRS and Fast Inverse Square Root](https://diydrones.com/forum/topics/madgwick-imu-ahrs-and-fast-inverse-square-root?id=705844%3ATopic%3A1018435&page=4)

After experimenting with real sensors I moved to artificial ACC input data and set up a test bed for Madgwick's algorithm ([MadgwickTests on GitHub](https://github.com/TobiasSimon/MadgwickTests)). I've figured out that in Madgwick's algorithm the **fast inverse square root** leads to **huge instabilities** when **noisy measurements** are applied.

Remarkably more stable and accurate pitch/roll angles (blue/magenta) were achieved by exchanging the inverse square root implementation. I used  the following code from [Accurate and Fast InvSqrt](http://pizer.wordpress.com/2008/10/12/fast-inverse-square-root) for computing the inverse square root of float x:

*unsigned int i = 0x5F1F1412 - (\*(unsigned int\*)&x >> 1);  
float tmp = \*(float\*)&i;  
float y = tmp \* (1.69000231f - 0.714158168f \* x \* tmp \* tmp);*

Please see the [GitHub commit](https://github.com/TobiasSimon/MadgwickTests/commit/4f76ef1475219bedbdba7afab297e3468d9e7c44) in which I've added some code for switching between the original (0), the proposed (1) and the reference (2) inverse square root implementations.

I hope that my investigations are helpful to improve the accuracy of Madgwick's C filter implementation.

Cheers, Tobias

|  |
| --- |
| **int instability\_fix = 1;**  **//---------------------------------------------------------------------------------------------------**  **// Fast inverse square-root**  **// See: http://en.wikipedia.org/wiki/Fast\_inverse\_square\_root**  **float invSqrt(float x) {**  **if (instability\_fix == 0)**  **{**  **/\* original code \*/**  **float halfx = 0.5f \* x;**  **float y = x;**  **long i = \*(long\*)&y;**  **i = 0x5f3759df - (i>>1);**  **y = \*(float\*)&i;**  **y = y \* (1.5f - (halfx \* y \* y));**  **return y;**  **}**  **else if (instability\_fix == 1)**  **{**  **/\* close-to-optimal method with low cost from http://pizer.wordpress.com/2008/10/12/fast-inverse-square-root \*/**  **unsigned int i = 0x5F1F1412 - (\*(unsigned int\*)&x >> 1);**  **float tmp = \*(float\*)&i;**  **return tmp \* (1.69000231f - 0.714158168f \* x \* tmp \* tmp);**  **}**  **else**  **{**  **/\* optimal but expensive method: \*/**  **return 1.0f / sqrtf(x);**  **}**  **}** |

[**https://github.com/kriswiner/MPU9250/issues/345**](https://github.com/kriswiner/MPU9250/issues/345)

**The Madgwick and Mahony filters (and quaternions in general I believe) work**

**in a right-handed coordinate system. So the data have to be "provided" to**

**conform to this. Thus NED, ENU (the two most common orientation**

**conventions) or even NWU will all work. As long as the sensor data is**

**provided in a manner consistent with the chosen convention.**

**So first step, the user decides which edge of the sensor board will be**

**pointing to true North when the quaternions read 1 0 0 0. This is one of**

**two absolute references in the system (the other is gravity). Once the**

**board edge facing North is decided (yes, it is your choice!), then it is a**

**simply matter to find out which accel axis point North, no? Then West, then**

**Up. Then the filter should get the data as AN, AW, AU...same for the other**

**two sensors.**

**Then once you do this, and assuming you have chosen the sensor**

**configurations reasonably, calibrated the sensors sufficiently, are running**

**the fusioniterations sufficiently fast, you have taken into account the**

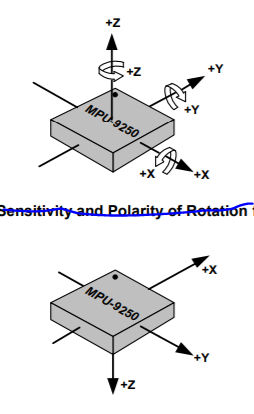
**local magnetic declination, etc you can check the behavior of your absolute**

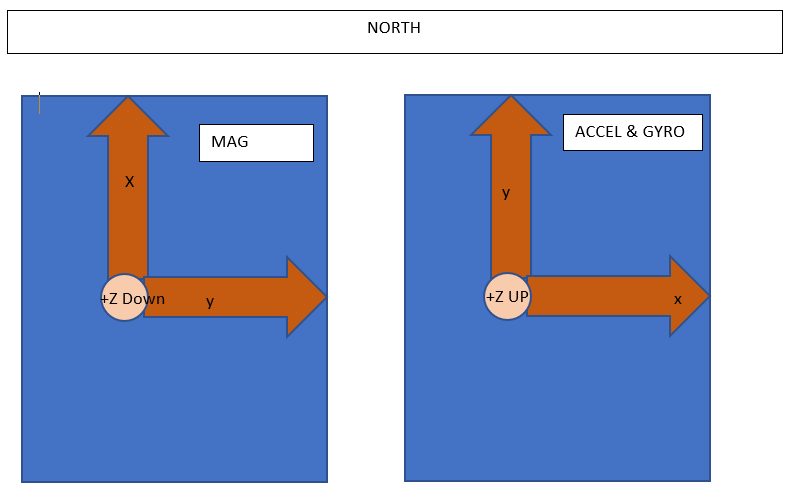
**orientation engine. What do the quaternions say when the sensors are**

**pointed North (Yaw or Heading = 0)? Should be 1 0 0 0. Try turnin the board**

**90 degrees. What happens now? etc.**

**I have just seen that the axis of Mag ang accel & gyro are misaligned on the MPU9250 datasheet.**





**so let me ask the question again with this new knowledge:**

**mpu9250 model**

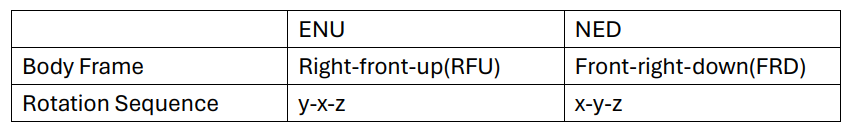
**Using NED system:**

**Given I point x-axis of my mag North, East=Y and Down is Z.**

**For ACCEL & GYRO: North = Y, EAST=X and Down = -Z**

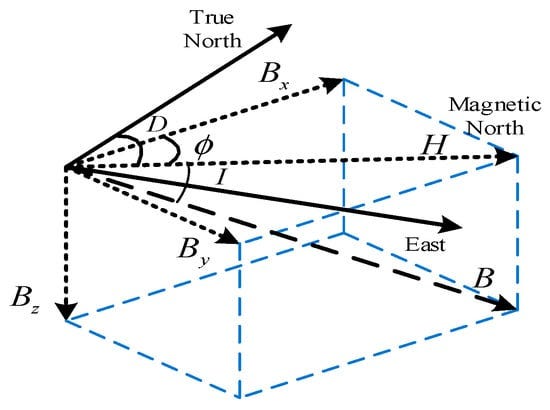
[**https://medium.com/@k66115704/imu-madgwick-filter-explanation-556fbe7f02e3**](https://medium.com/@k66115704/imu-madgwick-filter-explanation-556fbe7f02e3)

In this article, we will focus on 6-axis and 9-axis IMUs and two popular navigation frame configurations, ENU and NED. The body frame corresponding to ENU is right-front-up, and the body frame corresponding to NED is front-right-down. The ENU rotation order is y-x-z, and the NED rotation order is x-y-z. Roll, pitch, and yaw are for the X, Y, and Z axes respectively.



Below python code show the how to calculate roll, pitch, yaw angle from accelerometer and magnetometer data, which can provide initialization of euler angle and quaternion before running the madgwick filter.

For the magnetometer measurement vector, Madgwick filter assumed the body frame axis aligned to the magnetic east is negligible, which means X-axis is 0 in ENU frame, Y-axis is 0 in NED frame. Magnetometer vector defined as hx, hy, hz in body frame and bx, by, bz in navigation frame. Remember that, magnetometer measuring the magnetic north on X,Y,Z axis components instead of true north due to the effects of magnetic declination and inclination. Therefore, we need calculate magnetic north from X, Y axis of magnetometer measurement.



Magnetic field components and their relationship with true North.

[**https://github.com/xioTechnologies/Fusion/issues/24#issuecomment-1141470871**](https://github.com/xioTechnologies/Fusion/issues/24#issuecomment-1141470871)

**Your appear to be using the MPU-9250. The device datasheet confirms that the magnetometer axes are not aligned with the gyroscope and accelerometer axes. The magnetometer is effectively upside-down relative to the gyroscope and accelerometer.**

**image**

**You must therefore use the following axis assignments when passing the magnetometer values.**

**magnetometer = numpy.array([mpu9250\_mag\_y, mpu9250\_mag\_x, -1 \* mpu9250\_mag\_z])**

[**https://www.mathworks.com/help/fusion/ug/estimate-orientation-with-a-complementary-filter-and-imu-data.html**](https://www.mathworks.com/help/fusion/ug/estimate-orientation-with-a-complementary-filter-and-imu-data.html)

# Estimate Orientation with a Complementary Filter and IMU Data

### Align Axes of MPU-9250 Sensor with NED Coordinates

The axes of the accelerometer, gyroscope, and magnetometer in the MPU-9250 are not aligned with each other. Specify the index and sign x-, y-, and z-axis of each sensor so that the sensor is aligned with the North-East-Down (NED) coordinate system when it is at rest. In this example, the magnetometer axes are changed while the accelerometer and gyroscope axes remain fixed. For your own applications, change the following parameters as necessary.

 Get

% Accelerometer axes parameters.

accelXAxisIndex = 1;

accelXAxisSign = 1;

accelYAxisIndex = 2;

accelYAxisSign = 1;

accelZAxisIndex = 3;

accelZAxisSign = 1;

% Gyroscope axes parameters.

gyroXAxisIndex = 1;

gyroXAxisSign = 1;

gyroYAxisIndex = 2;

gyroYAxisSign = 1;

gyroZAxisIndex = 3;

gyroZAxisSign = 1;

% Magnetometer axes parameters.

magXAxisIndex = 2;

magXAxisSign = 1;

magYAxisIndex = 1;

magYAxisSign = 1;

magZAxisIndex = 3;

magZAxisSign = -1;

% Helper functions used to align sensor data axes.

alignAccelAxes = @(in) [accelXAxisSign, accelYAxisSign, accelZAxisSign] ...

.\* in(:, [accelXAxisIndex, accelYAxisIndex, accelZAxisIndex]);

alignGyroAxes = @(in) [gyroXAxisSign, gyroYAxisSign, gyroZAxisSign] ...

.\* in(:, [gyroXAxisIndex, gyroYAxisIndex, gyroZAxisIndex]);

alignMagAxes = @(in) [magXAxisSign, magYAxisSign, magZAxisSign] ...

.\* in(:, [magXAxisIndex, magYAxisIndex, magZAxisIndex]);

[**https://github.com/kriswiner/MPU9250/issues/368**](https://github.com/kriswiner/MPU9250/issues/368)

**OK, thanks. This is likely to be wrong as well unless you make sure the**

**sensor data is in NED or ENU order.**

**The orientation in the code is wrong for a lot of reasons.**

**The "righter" way to do this is to enforce either an NED or ENU orientation**

**convention. So choose which direction of the board will produce quaternions**

**(1, 0, 0, 0) (i.e., headion == 0 degrees) when pointed toward true North,**

**then provide the fusion filter with the sensor data as NED or ENU as**

**appropriate. In other words, filter(aN, aE, aD, gN, gE, gD, mN, mE, mD).**

**In this sense, there is no one right way to do this (one of the reasons I**

**haven't bothered to fix the sketch), but doing this inconsistently wrt one**

**of these conventions (like in this sketch) will produce errors in the**

**absolute orientation.**

For accurate orientation estimation using Madgwick and Mahony filters, proper axis alignment of the gyroscope, accelerometer, and magnetometer is crucial. Here are some key points to consider:

1. **Consistent Coordinate System**: Ensure all sensors (gyroscope, accelerometer, and magnetometer) are aligned to the same coordinate system. Typically, this means aligning the x, y, and z axes of each sensor.
2. **Calibration**: Perform a thorough calibration of each sensor to correct for any biases, scale factors, and misalignments. This includes:
   * **Accelerometer**: Calibrate for offset and scale.
   * **Gyroscope**: Calibrate for bias (drift) and scale.
   * **Magnetometer**: Calibrate for hard and soft iron distortions.
3. **Mounting Orientation**: Physically mount the sensors in such a way that their axes are aligned. This often involves ensuring that the sensor boards are mounted parallel to each other.
4. **Sensor Fusion Algorithm**: Both Madgwick and Mahony filters use quaternion-based sensor fusion algorithms to combine the data from the accelerometer, gyroscope, and magnetometer. Proper axis alignment ensures that the data from each sensor contributes correctly to the orientation estimation.
5. **Reference Frames**: Be consistent with the reference frames used. For example, if using a North-East-Down (NED) frame, ensure all sensor data is transformed accordingly.

[For more detailed guidance, you can refer to resources like the MathWorks documentation on sensor fusion1](https://www.mathworks.com/videos/sensor-fusion-part-2-fusing-a-mag-accel-and-gyro-to-estimate-orientation-1569411056638.html) [and the MDPI paper on joint calibration and axes alignment2](https://www.mathworks.com/help/nav/ref/ahrs.html).

If you have specific issues or need further assistance with your implementation, feel free to ask!

[1](https://www.mathworks.com/videos/sensor-fusion-part-2-fusing-a-mag-accel-and-gyro-to-estimate-orientation-1569411056638.html): [MathWorks - Fusing a Mag, Accel, and Gyro to Estimate Orientation](https://www.mathworks.com/videos/sensor-fusion-part-2-fusing-a-mag-accel-and-gyro-to-estimate-orientation-1569411056638.html) [2](https://www.mathworks.com/help/nav/ref/ahrs.html): [MDPI - Accelerometer and Magnetometer Joint Calibration and Axes Alignment](https://www.mdpi.com/2227-7080/8/1/11)

**IMU Data Fusing: Complementary, Kalman, and Mahony Filter**

17. Sep. 2013 | Last update 13. Sep. 2018 | Theme [Grundlagen](https://www.olliw.eu/category/knowhow/), [Multicopter](https://www.olliw.eu/category/multicopter/)  
Tag [Filter](https://www.olliw.eu/tag/filter/), [Grundlagen](https://www.olliw.eu/tag/knowhow/), [Gyro](https://www.olliw.eu/tag/t_gyro/), [IMU](https://www.olliw.eu/tag/imu/)

An ***inertial measurement unit***, or IMU, measures accelerations and rotation rates, and possibly earth’s magnetic field, in order to determine a body’s attitude. Anyone who is serious about reading this article is likely familiar with the topic, and the need of data fusing, and I shouldn’t spend more words on this. I should however – since this is going to be a long article – spend some words on what this article is about:

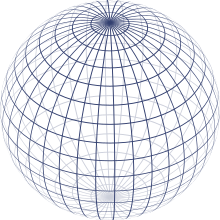
The literature on the web (and I should say that the web is my only source of information) on the topic is abundant. It appears however that it is based to a greater or lesser extent on some few works, e.g. by Colton [[SC](https://www.olliw.eu/2013/imu-data-fusing/#refSC)], Premerlani and Bizard [[PB](https://www.olliw.eu/2013/imu-data-fusing/#refPB)], Starlino [[St](https://www.olliw.eu/2013/imu-data-fusing/#refST1)], Lauszus [[La](https://www.olliw.eu/2013/imu-data-fusing/#refLA)], Mahony [[RM](https://www.olliw.eu/2013/imu-data-fusing/#refRM05)] and Madgwick [[SM](https://www.olliw.eu/2013/imu-data-fusing/#refSM1)], which – so it seems – have become standard references (for hobbiests!). The number of different algorithms and implementation details given there is somewhat confusing, but – even though different buzz words are certainly used – it is not always obvious to what extend they are different. This article presents an analysis and comparison of the data fusing filters described in these works, in order to understand better their behavior, and differences and similarities. As a corollary simplified and/or improved algorithms surface. The article considers 6DOF IMUs only.

Three basic filter approaches are discussed, the ***complementary filter***, the ***Kalman filter*** (with constant matrices), and the ***Mahony&Madgwick filter***. The article starts with some preliminaries, which I find relevant. It then considers the case of a single axis (called one dimensional or 1D). First the most simplest method is discussed, where gyro bias is not estimated (called 1st order). Then gyro bias estimation is included (called 2nd order). Finally, the complete situation of three axes (called 3D) is considered, and some approximations and improvements are evaluated.

[**https://www.olliw.eu/2013/imu-data-fusing/#codes**](https://www.olliw.eu/2013/imu-data-fusing/#codes)

## [Earth bounded axes conventions](https://en.wikipedia.org/wiki/Axes_conventions)

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=1)]

[](https://en.wikipedia.org/wiki/File:Sphere_wireframe.svg)Representation of the Earth with parallels and meridians

### World reference frames: ENU and NED

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=2)]

*Main article:*[*Local tangent plane*](https://en.wikipedia.org/wiki/Local_tangent_plane)

Basically, as lab frame or reference frame, there are two kinds of conventions for the frames:

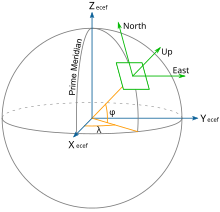
* East, North, Up (ENU), used in geography
* North, East, Down (NED), used specially in aerospace

This frame referenced w.r.t. Global Reference frames like Earth Center Earth Fixed (ECEF) non-inertial system.

#### World reference frames for attitude description

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=3)]

To establish a standard convention to describe attitudes, it is required to establish at least the axes of the reference system and the axes of the rigid body or vehicle. When an ambiguous notation system is used (such as [Euler angles](https://en.wikipedia.org/wiki/Euler_angles)) the convention used should also be stated. Nevertheless, most used notations (matrices and quaternions) are unambiguous.

[](https://en.wikipedia.org/wiki/File:ECEF_ENU_Longitude_Latitude_relationships.svg)Earth Centered Earth Fixed and East, North, Up coordinates.

[Tait–Bryan angles](https://en.wikipedia.org/wiki/Tait%E2%80%93Bryan_angles) are often used to describe a vehicle's attitude with respect to a chosen reference frame, though any other notation can be used. The positive *x*-axis in vehicles points always in the direction of movement. For positive *y*- and *z*-axis, we have to face two different conventions:

* In case of land vehicles like cars, tanks etc., which use the ENU-system (East-North-Up) as external reference (*World frame*), the vehicle's (body's) positive *y*- or pitch axis always points to its left, and the positive *z*- or yaw axis always points up. World frame's origin is fixed at the center of gravity of the vehicle.[[3]](https://en.wikipedia.org/wiki/Axes_conventions#cite_note-3)
* By contrast, in case of air and sea vehicles like submarines, ships, airplanes etc., which use the NED-system (North-East-Down) as external reference (*World frame*), the vehicle's (body's) positive *y*- or pitch axis always points to its right, and its positive *z*- or yaw axis always points down. World frame's origin is fixed at the center of gravity of the vehicle.
* Finally, in case of space vehicles like the [Space Shuttle](https://en.wikipedia.org/wiki/Space_Shuttle) etc., a modification of the latter convention is used, where the vehicle's (body's) positive *y*- or pitch axis again always points to its right, and its positive z- or yaw axis always points down, but “down” now may have two different meanings: If a so-called *local frame* is used as external reference, its positive z-axis points “down” to the center of the Earth as it does in case of the earlier mentioned NED-system,[[4]](https://en.wikipedia.org/wiki/Axes_conventions#cite_note-4) but if the *inertial frame* is used as reference, its positive z-axis will point now to the [north celestial pole](https://en.wikipedia.org/wiki/North_celestial_pole), and its positive x-axis to the Vernal [Equinox](https://en.wikipedia.org/wiki/Equinox)[[5]](https://en.wikipedia.org/wiki/Axes_conventions#cite_note-5) or some other reference meridian.

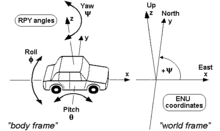
### Frames mounted on vehicles

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=4)]

Specially for aircraft, these frames do not need to agree with the earth-bound frames in the up-down line. It must be agreed what ENU and NED mean in this context.

#### Conventions for land vehicles

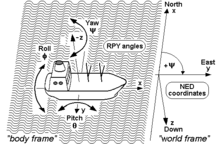
[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=5)]

[](https://en.wikipedia.org/wiki/File:RPY_angles_of_cars.png)RPY angles of cars and other land vehicles

For land vehicles it is rare to describe their complete orientation, except when speaking about [electronic stability control](https://en.wikipedia.org/wiki/Electronic_stability_control) or [satellite navigation](https://en.wikipedia.org/wiki/Satellite_navigation). In this case, the convention is normally the one of the adjacent drawing, where RPY stands for [roll-pitch-yaw](https://en.wikipedia.org/wiki/Roll-pitch-yaw).

#### Conventions for sea vehicles

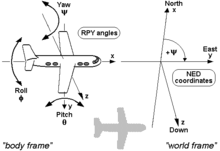
[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=6)]

[](https://en.wikipedia.org/wiki/File:RPY_angles_of_ships.png)RPY angles of ships and other sea vehicles

As well as aircraft, the same terminology is used for the [motion of ships and boats](https://en.wikipedia.org/wiki/Ship_motions). Some words commonly used were introduced in maritime navigation. For example, the *yaw* angle or heading, has a [nautical](https://en.wikipedia.org/wiki/Navigation) origin, with the meaning of "bending out of the course". Etymologically, it is related with the verb 'to go'.[[6]](https://en.wikipedia.org/wiki/Axes_conventions#cite_note-6) It is related to the concept of [bearing](https://en.wikipedia.org/wiki/Bearing_(navigation)). It is typically assigned the shorthand notation *ψ*.[[7]](https://en.wikipedia.org/wiki/Axes_conventions#cite_note-7)

#### Conventions for aircraft local reference frames

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=7)]

[](https://en.wikipedia.org/wiki/File:RPY_angles_of_airplanes.png)RPY angles of airplanes and other air vehicles[](https://en.wikipedia.org/wiki/File:Roll_pitch_yaw_mnemonic.svg)Mnemonics to remember angle names

*Main article:*[*aircraft principal axes*](https://en.wikipedia.org/wiki/Aircraft_principal_axes)

Coordinates to describe an aircraft attitude (Heading, Elevation and Bank) are normally given relative to a reference control frame located in a control tower, and therefore ENU, relative to the position of the control tower on the earth surface.

Coordinates to describe observations made from an aircraft are normally given relative to its intrinsic axes, but normally using as positive the coordinate pointing downwards, where the interesting points are located. Therefore, they are normally NED.

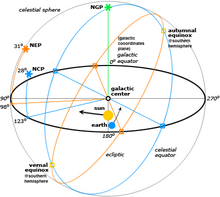
These axes are normally taken so that X axis is the longitudinal axis pointing ahead, Z axis is the vertical axis pointing downwards, and the Y axis is the lateral one, pointing in such a way that the frame is [right-handed](https://en.wikipedia.org/wiki/Right-hand_rule).

The *motion* of an aircraft is often described in terms of rotation about these axes, so rotation about the *X*-axis is called rolling, rotation about the *Y*-axis is called pitching, and rotation about the *Z*-axis is called yawing.

## Frames for space navigation

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=8)]

*Main article:*[*Celestial coordinate system*](https://en.wikipedia.org/wiki/Celestial_coordinate_system)

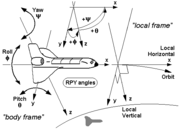
[](https://en.wikipedia.org/wiki/File:Celestial.png)Different reference systems for coordinates in space

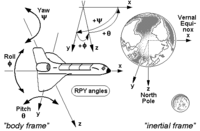
For satellites orbiting the Earth it is normal to use the [Equatorial coordinate system](https://en.wikipedia.org/wiki/Equatorial_coordinate_system). The projection of the Earth's equator onto the celestial sphere is called the [celestial equator](https://en.wikipedia.org/wiki/Celestial_equator). Similarly, the projections of the Earth's north and south geographic poles become the north and south [celestial poles](https://en.wikipedia.org/wiki/Celestial_pole), respectively.

Deep space satellites use other [Celestial coordinate system](https://en.wikipedia.org/wiki/Celestial_coordinate_system), like the [Ecliptic coordinate system](https://en.wikipedia.org/wiki/Ecliptic_coordinate_system).

### Local conventions for space ships as satellites

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=9)]

[](https://en.wikipedia.org/wiki/File:RPY_angles_of_spaceships_(local_frame).png)

[](https://en.wikipedia.org/wiki/File:RPY_angles_of_spaceships_(inertial_frame).png)

RPY angles of the Space Shuttle and other space vehicles, first using a local frame as reference and second using an inertial frame as reference.

If the goal is to keep the shuttle during its orbits in a constant attitude with respect to the sky, e.g. in order to perform certain astronomical observations, the preferred reference is the *inertial frame*, and the RPY angle vector (0|0|0) describes an attitude then, where the shuttle's wings are kept permanently parallel to the Earth's equator, its nose points permanently to the vernal [equinox](https://en.wikipedia.org/wiki/Equinox), and its belly towards the northern [polar star](https://en.wikipedia.org/wiki/Polar_star) (see picture). (Note that rockets and missiles more commonly follow the conventions for aircraft where the RPY angle vector (0|0|0) points north, rather than toward the vernal equinox).

On the other hand, if the goal is to keep the shuttle during its orbits in a constant attitude with respect to the surface of the Earth, the preferred reference will be the *local frame*, with the RPY angle vector (0|0|0) describing an attitude where the shuttle's wings are parallel to the Earth's surface, its nose points to its heading, and its belly down towards the centre of the Earth (see picture).

## Frames used to describe attitudes

[[edit](https://en.wikipedia.org/w/index.php?title=Axes_conventions&action=edit&section=10)]

Normally the frames used to describe a vehicle's local observations are the same frames used to describe its attitude with respect to the ground tracking stations. i.e. if an ENU frame is used in a tracking station, also ENU frames are used onboard and these frames are also used to refer local observations.

An important case in which this does not apply is aircraft. Aircraft observations are performed downwards and therefore normally NED axes convention applies. Nevertheless, when attitudes with respect to ground stations are given, a relationship between the local earth-bound frame and the onboard ENU frame is used.

[**https://www.instructables.com/Quaternion-Compass/**](https://www.instructables.com/Quaternion-Compass/)