

---

## Computing tilt measurement and tilt-compensated eCompass

---

By Andrea Vitali

Main components	
LSM303AGR	Ultra-compact high-performance eCompass module: ultra-low-power 3D accelerometer and 3D magnetometer
LSM303AH	Ultra-compact high-performance eCompass module: ultra-low-power 3D accelerometer and 3D magnetometer

### Purpose and benefits

This design tip explains how to compute tilt (roll and pitch angles) from accelerometer data. It also explains how to compute eCompass (yaw angle) from tilt-compensated magnetometer data. The conversion from Euler angles to Quaternions is also shown.

Benefits:

- Added functionality with respect to data fusion provided by the MotionFX library embedded in the X-Cube-MEMS1 software package which provides 9-axis Acc+Mag+Gyro and 6-axis Acc+Gyro fusion but not 6-axis Acc+Mag
- Reduction of the firmware footprint with respect to using the full-blown data fusion provided by the MotionFX library embedded in the X-Cube-MEMS1 software package - see references in design Support Material paragraph
- Short essential implementation, which enables easy customization and enhancement by the end-user (MotionFX is available only in binary format, not as source code)
- Easy to use solution, applicable to all microcontrollers, although the MotionFX library can only be run on the STM32 MCU family

### Description

Step 1: Computation of Phi (roll angle, also known as bank; see Figure 1 for reference) based on accelerometer data (Gx, Gy and Gz; unit of measure does not matter)

$$\text{Roll: } \Phi = \text{Atan2}(G_y, G_z)$$

Note: If Theta =  $\pm 90$  deg, then Gy and Gz are near-zero and Phi is unstable; if stability is desired, then Gz can be substituted by  $G_z + G_x \cdot \alpha$ , with  $\alpha = 0.01-0.05$ .

---

Step 2: Computation of Theta (pitch angle, also known as attitude; see Figure 1 for reference) based on accelerometer data (Gx, Gy and Gz; unit of measure does not matter)

$$\mathbf{Gz2} = \mathbf{Gy} * \mathbf{Sin}(\mathbf{Phi}) + \mathbf{Gz} * \mathbf{Cos}(\mathbf{Phi})$$

$$\mathbf{Pitch: Theta} = \mathbf{Atan}(\mathbf{-Gx} / \mathbf{Gz2})$$

Note: If Theta = ±90 deg, then Gy and Gz are near-zero and Gz2 is also near-zero; division by zero should not be performed, Theta = -90 deg if Gx>0, and Theta = +90 deg if Gx<0; Gx cannot be zero.

Step 3: Computation of Psi (yaw angle, also known as heading; see Figure 1 for reference) based on magnetometer data (Bx, By and Bz; unit of measure does not matter)

$$\mathbf{By2} = \mathbf{Bz} * \mathbf{Sin}(\mathbf{Phi}) - \mathbf{By} * \mathbf{Cos}(\mathbf{Phi})$$

$$\mathbf{Bz2} = \mathbf{By} * \mathbf{Sin}(\mathbf{Phi}) + \mathbf{Bz} * \mathbf{Cos}(\mathbf{Phi})$$

$$\mathbf{Bx3} = \mathbf{Bx} * \mathbf{Cos}(\mathbf{Theta}) + \mathbf{Bz2} * \mathbf{Sin}(\mathbf{Theta})$$

$$\mathbf{Yaw: Psi} = \mathbf{Atan2}(\mathbf{By2}, \mathbf{Bx3})$$

Note: If Theta = ±90 deg and if Phi is unstable, Psi will also be unstable; if Phi is made stable as mentioned above, then Psi will also be stable. Regardless of stability, the sum Phi+Psi will always be stable. See paragraph on singularities and gimbal-lock.

Step 4: Conversion from Euler angles to Quaternions (optional step)

$$\mathbf{Qw} = \mathbf{Cos}(\mathbf{Phi}/2) * \mathbf{Cos}(\mathbf{Theta}/2) * \mathbf{Cos}(\mathbf{Psi}/2) + \mathbf{Sin}(\mathbf{Phi}/2) * \mathbf{Sin}(\mathbf{Theta}/2) * \mathbf{Sin}(\mathbf{Psi}/2)$$

$$\mathbf{Qx} = \mathbf{Sin}(\mathbf{Phi}/2) * \mathbf{Cos}(\mathbf{Theta}/2) * \mathbf{Cos}(\mathbf{Psi}/2) - \mathbf{Cos}(\mathbf{Phi}/2) * \mathbf{Sin}(\mathbf{Theta}/2) * \mathbf{Sin}(\mathbf{Psi}/2)$$

$$\mathbf{Qy} = \mathbf{Cos}(\mathbf{Phi}/2) * \mathbf{Sin}(\mathbf{Theta}/2) * \mathbf{Cos}(\mathbf{Psi}/2) + \mathbf{Sin}(\mathbf{Phi}/2) * \mathbf{Cos}(\mathbf{Theta}/2) * \mathbf{Sin}(\mathbf{Psi}/2)$$

$$\mathbf{Qz} = \mathbf{Cos}(\mathbf{Phi}/2) * \mathbf{Cos}(\mathbf{Theta}/2) * \mathbf{Sin}(\mathbf{Psi}/2) - \mathbf{Sin}(\mathbf{Phi}/2) * \mathbf{Sin}(\mathbf{Theta}/2) * \mathbf{Cos}(\mathbf{Psi}/2)$$

Singularities and gimbal-lock

If Theta = ±90 deg, Phi and Psi will describe a rotation around the same vertical axis (one degree of freedom is lost, this is also known as gimbal lock); the sum Psi+Phi will be correct when Theta=-90 deg, or Psi-Phi when Theta=+90 deg.

High-g motion

Accelerometer data cannot be used for tilt measurement (Phi and Theta) if high-g is ongoing. Accelerometer data can only be used when the modulus is near g:  $Gx^2 + Gy^2 + Gz^2 = 1$ , when Gx / Gy / Gz are expressed in g.

Gyroscope data can be used to update the tilt output when the accelerometer data cannot be used.

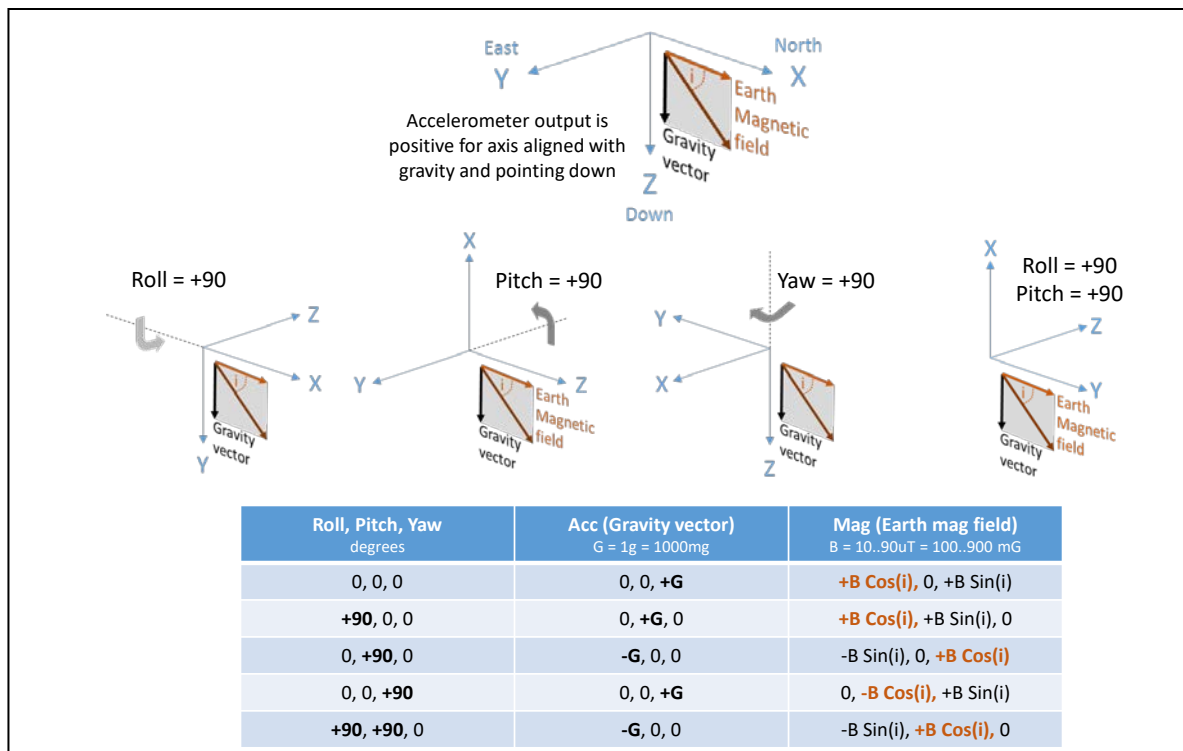
## Hard/Soft iron effects and Magnetic anomalies

Magnetometer data cannot be used for the eCompass (Psi) if hard/soft iron effects are not compensated and/or if magnetic anomalies are present. Hard-iron effects are offsets that must be subtracted from  $B_x$  /  $B_y$  /  $B_z$ . Soft-iron effects are non-unity axis sensitivities and non-zero cross-axis sensitivities that must be compensated by matrix multiplication.

Magnetometer data can only be used when the modulus is of near nominal Earth field value, which happens when there are no magnetic anomalies and hard/soft iron effects are compensated:  $B_x^2 + B_y^2 + B_z^2 = B$ .

Gyroscope data can be used to update the eCompass output when the magnetometer data cannot be used.

**Figure 1. Reference orientation for input data from accelerometer and magnetometer, and reference orientation for output data: roll, pitch and yaw angles**



## Generic tilt angle with respect to horizontal plane

The generic tilt angle, with respect to the horizontal plane, can be derived from the formula of dot-product or vector-product of current acceleration vector and gravity vector pointing downward. The gravity vector has components  $x=0$ ,  $y=0$ ,  $z=1$ .

The dot-product of two vectors A and B is  $\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}| |\mathbf{B}| \cos(\text{angle}) = A_x B_x + A_y B_y + A_z B_z$ . When A is the current acceleration (Gx, Gy, Gz) and B is the gravity pointing downward (x=0, y=0, z=1):  $|G| \cos(\text{angle}) = G_z$ , therefore tilt angle =  $\arccos ( G_z / |G| )$ .

**Tilt =  $\arccos( G_z / \sqrt{G_x^2+G_y^2+G_z^2} )$**  from the dot-product formula

The vector-product of two vectors A and B is  $\mathbf{C} = \mathbf{A} \times \mathbf{B} = |\mathbf{A}| |\mathbf{B}| \mathbf{N} \sin(\text{angle})$ , where N is the unity vector orthogonal to both vectors A and B.  $C_x = A_y B_z - A_z B_y$ ,  $C_y = A_z B_x - A_x B_z$ ,  $C_z = A_x B_y - A_y B_x$ . When A is the current acceleration (Gx, Gy, Gz) and B is the gravity pointing downward (x=0, y=0, z=1):  $|C| = |G| \sin(\text{angle}) = \sqrt{G_x^2 + G_y^2}$ , therefore tilt angle =  $\arcsin ( |C| / |G| )$ .

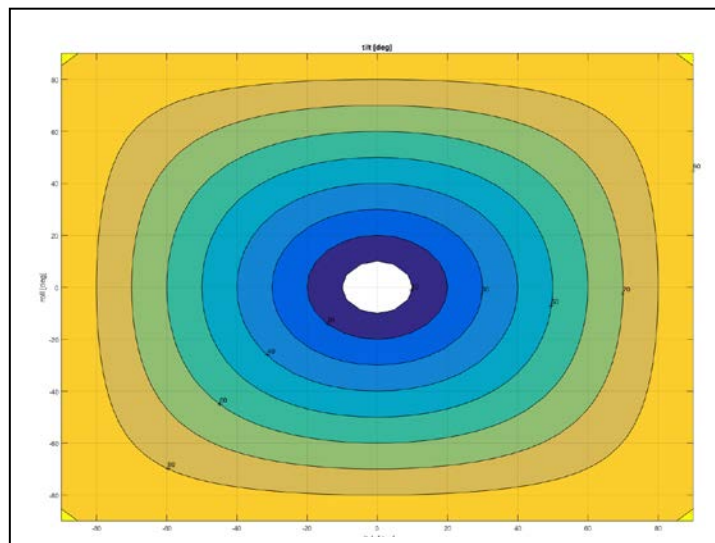
**Tilt =  $\arcsin( \sqrt{G_x^2 + G_y^2} / \sqrt{G_x^2+G_y^2+G_z^2} )$**  from the vector-product

Taking into account measurement noise, the asin function is more accurate if the angle is small, that is when  $-45 < \text{angle} < +45$  deg, while the acos function is accurate when the angle is large, that is  $\text{angle} < -45$  or  $\text{angle} > +45$ . To get highest accuracy for all angles, use the atan or atan2 function:

**Tilt =  $\arctan( \sqrt{G_x^2 + G_y^2} / G_z )$**  if  $G_z=0$ ,  $\arctan(\pm\infty) = \pm 90$  deg

**Tilt =  $\text{atan2}( \sqrt{G_x^2 + G_y^2} , G_z )$**  no need to check if  $G_z=0$

**Figure 2. Roll and pitch angles compared to generic tilt angle with respect to horizontal plane**



Optimized implementation for trigonometric functions (sin, cos, atan2)

If a floating-point unit (FPU) is available on the microcontroller (e.g. on Cortex-M4 in STM32F4 or STM32L4), then the vectorized computation may be available to speed up computations. Data is collected in buffers, and then it is processed in batches.

If single-cycle multiplication is available on the microcontroller (e.g. on Cortex-M3 in STM32F1, STM32L1, STM32F3), then polynomial approximation with Horner evaluation order can be very effective.

Finally, the simplest microcontrollers may benefit from CORDIC implementation which requires only table look-up and shift-and-add operations. Roughly speaking, the CORDIC algorithm requires as many iterations as the number of precision bits; also, two functions are computed simultaneously: sin & cos, atan2 and modulus.

## Support material

Related design support material
FP-SNS-MOTENV1, STM32Cube function pack for IoT node with BLE connectivity and environmental and motion sensors
User manual, UM2016, Getting started with the STM32 ODE function pack for IoT node with BLE connectivity and environmental and motion sensors
Quick Start Guide, STM32Cube function pack for IoT node with BLE connectivity, environmental and motion sensors (FP-SNS-MOTENV1)
X-CUBE-MEMS1, Sensor and motion algorithm software expansion for STM32Cube
User manual, UM2220, Getting started with MotionFX sensor fusion library in X-CUBE-MEMS1 expansion for STM32Cube
Documentation
Application note, AN4509, Tilt measurement using a low-g 3-axis accelerometer

## Revision history

Date	Version	Changes
05-Nov-2015	1	Initial release
22-Nov-2018	2	Added paragraph on generic tilt; updated Figure 1
07-Jan-2021	3	Updated references to reflect current software packages Corrected formula for tilt computation

---

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2021 STMicroelectronics – All rights reserved