STATIC CALIBRATION OF TRIAXIAL ACCELEROMETER USING NONLINEAR NUMERICAL METHOD

Ass. Prof. Dr. Emad Natiq Abdulwahab
Mechanical Engineering Department
University of Technology

Ass. Prof. Dr. Mohammed Idrees Mohsin
Mechanical Engineering Department
University of Technology

Ass. Lec. Akeel Ali Wannas
Mechanical Engineering Department
University of Technology

Abstract

Triaxial accelerometers are used in various applications, such as inertial navigation systems (INSs) and inclinometers. Such accelerometers must be calibrated as accurately as possible because accelerometers with even small biases could result in a very fast position drift when they are used for INS applications and could result in inaccurate tilt angle measurements. Since the calibration parameters of a triaxial accelerometer vary under different environmental conditions, including temperature change, the gain factors and biases change whenever the sensor is switched on. Thus, an accelerometer must be calibrated before use or when the temperature is significantly changed. This paper presents a triaxial accelerometer calibration method using a mathematical model of nine calibration parameters: three gain factors, three biases and three non orthogonality factors. The fundamental principle of the proposed calibration method is that the sum of the triaxial accelerometer outputs is equal to the gravity vector when the accelerometer is stationary. The proposed method requires the triaxial accelerometer to be placed in forty eight tilt angles to estimate the nine calibration parameters. Since the mathematical model of the calibration parameters is nonlinear, an iterative method is used Levenberg-marquardt algorithm (LMA). The results are verified by comparing the estimated data and corrected data with how far or close from surface of sphere that have radius 1g and center (0,0,0) by calculating the standard deviation.

Keyword: MEMS, Accelerometer, calibration, LMA.
1 INTRODUCTION:

An accelerometer is a device that measures proper acceleration (g-force). Proper acceleration is not the same as coordinate acceleration (rate of change of velocity). For example, an accelerometer at rest on the surface of the Earth will measure an acceleration \( g = 9.81 \, \text{m/s}^2 \) straight upwards. By contrast, accelerometers in free fall orbiting and accelerating due to the gravity of Earth will measure zero.

Accelerometers have multiple applications in industry and science. Highly sensitive accelerometers are components of inertial navigation systems for aircraft, missiles, UAV’s and robots [1]. Accelerometers are used to detect and monitor vibration in rotating machinery. Accelerometers are used in tablet computers and digital cameras so that images on screens are always displayed upright [2]. Accelerometers are used in drones for flight stabilisation. Pairs
of accelerometers extended over a region of space can be used to detect differences (gradients) in the proper accelerations of frames of references associated with those points. These devices are called gravity gradiometers, as they measure gradients in the gravitational field. Such pairs of accelerometers in theory may also be able to detect gravitational waves [3].

Single and multi-axis models of accelerometer are available to detect magnitude and direction of the proper acceleration (or g-force), as a vector quantity, and can be used to sense orientation (because direction of weight changes), coordinate acceleration (so long as it produces g-force or a change in g-force), vibration, shock, and falling in a resistive medium (a case where the proper acceleration changes [4], since it starts at zero, then increases). Micromachined accelerometers are increasingly present in portable electronic devices and video game controllers, to detect the position of the device or provide for game input [4].

Triaxial accelerometers supplied for the consumer market are typically calibrated by using a six-element linear model (ellipsoid formula with arbitrary center) comprising a gain and offset or a nine-element nonlinear model comprising a gain, non-orthogonally and offset each of the three axes [5]. This factory calibration will change slightly as a result of the thermal stresses during soldering of the accelerometer to the circuit board. Additional small errors, external to the accelerometer, including rotation of the accelerometer package relative to the circuit board and misalignment of the circuit board to the final product, will also be introduced during the soldering and final assembly process, therefore we looking to obtain improved accuracy [6].

The accelerometer provides information about acceleration, speed and position of the Vehicle, by using statistical filtering. The development of such a system requires the calibration of sensor, which is a challenging task [7], [8], [9], because, the data output depends on:

1. Local acceleration.
2. The sensor attitude.
3. The real gain and bias parameters.

In a 3D sensor each axis $i$ have a different gain noted as $\alpha_i$ and a bias noted as $\beta_i$. Figure 1 represents a 3D sensor with offset $\hat{\beta} = (\beta_x, \beta_y, \beta_z)^T$, gains $\alpha_x, \alpha_y, \alpha_z$ and orthogonality errors $S_{xy}, S_{xz}, S_{yz}$ for accelerometer; the model of the sensor output is given by [4] see Figure 2.
where \( \varepsilon \) is a residual error or noise, \( S_{ij} \) represents the orthogonality errors between sensor axes (cosine of angles between axes of sensor x-y, y-z, z-x where \( S_{xy} = S_{yx} \), \( S_{yz} = S_{zy} \), and \( S_{zx} = S_{xz} \)). The matrix \( \alpha \) is considered symmetrical [4].

\[
\begin{bmatrix}
\bar{a}_x(t) \\
\bar{a}_y(t) \\
\bar{a}_z(t)
\end{bmatrix} =
\begin{bmatrix}
\alpha_x & S_{xy} & S_{xz} \\
S_{xy} & \alpha_x & S_{yz} \\
S_{xz} & S_{yz} & \alpha_x
\end{bmatrix}
\begin{bmatrix}
\bar{a}'_x(t) \\
\bar{a}'_y(t) \\
\bar{a}'_z(t)
\end{bmatrix}
+ 
\begin{bmatrix}
\beta_x \\
\beta_y \\
\beta_z
\end{bmatrix}
+ 
\begin{bmatrix}
\varepsilon_x \\
\varepsilon_y \\
\varepsilon_z
\end{bmatrix}
\]

Figure 1 Sensor behavior

Figure 2 Acceleration vector \( \bar{a} \) measured as \( \bar{a}' \)

2 CALIBRATION PROCEDURE

2.1 Acquisition of a set of points

A simple test rig (see Figure 3) consisting of an adjustable platform, a cube mounted with a triaxial accelerometer and a V-block was built to perform these forty eight positions [6]. Set of measurements with various attitudes are needed, so the parameters in equation (1) can be estimated. In the case of an accelerometer, the magnitude of the local force (or acceleration) must be precisely known; it equals the magnitude of the Earth gravity force (1g).
The calibration procedure stops when the measurements cover most of the surface of equation (1) [10]. The minimum procedure is to rotate the sensor in the reference $XYZ$ coordinate system to measure a set of points which describes a closed solid.

### 2.2 2D projection of measurements

The calibration can be facilitated by a display of measured acceleration magnitude in spherical coordinates $r(t)$, $\phi(t)$ and $\theta(t)$ (see Figure 4). This transformation requires an estimation of parameters that is obtained from the first measured points [11]:

\[
\begin{align*}
    r(t) &= \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2}, \\
    \theta(t) &= \tan^{-1}\left(\frac{\sqrt{a_x(t)^2 + a_y(t)^2}}{a_z(t)}\right), \\
    \phi(t) &= \tan^{-1}\left(\frac{a_y(t)}{a_x(t)}\right),
\end{align*}
\]

![Experiment platform setup](image)

**Figure 3 Experiment platform setup**
2.3 Estimation of parameters

To estimate the matrix $\alpha$ and the vector $\beta$ in equation (1), we can minimize the following error function [12]:

$$E(p) = \sum_t (e_p(t))^2,$$

with:

$$e_p(t) = \|\tilde{a}\|^2 - (\tilde{a}'(t) - \tilde{\beta})^T (\alpha^{-1})^2 (\tilde{a}'(t) - \tilde{\beta}),$$

where

$$\|\tilde{a}\|^2 = a_x(t)^2 + a_y(t)^2 + a_z(t)^2 = 1g,$$

$$p = [a_x, a_y, a_z, S_x, S_y, S_z, \beta_x, \beta_y, \beta_z]^T,$$

the parameters vector to be estimated using LMA (Levenberg–Marquardt Algorithm).

![Figure 4 Sphere coordinates](image)

2.4 The levenberg-marquardt method for nonlinear least squares

The Levenberg-Marquardt method is a standard technique used to solve nonlinear least squares problems. Least squares problems arise when fitting a parameterized function to a set of measured data points by minimizing the sum of the squares of the errors between the
data points and the function. Nonlinear least squares problems arise when the function is not linear in the parameters. Nonlinear least squares methods involve an iterative improvement to parameter values in order to reduce the sum of the squares of the errors between the function and the measured data points. The Levenberg-Marquardt curve-fitting method is actually a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the direction of the greatest reduction of the least squares objective. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic. The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and act more like the Gauss-Newton method when the parameters are close to their optimal value.

![2D Projection Data](image)

**Figure 5 Measurement data of acceleration in 2d projection**

In fitting a function \( \hat{y}(t; p) \) of an independent variable \( t \) and a vector of \( n \) parameters \( p \) to a set of \( m \) data points \((t_i, y_i)\), it is customary and convenient to minimize the sum of the weighted squares of the errors (or weighted residuals) between the measured data \( y(t_i) \) and the curve-fit function \( \hat{y}(t_i; p) \). This scalar-valued goodness-of-fit measure is called the chi-squared error criterion [10].
The value \( w_i \) is a measure of the error in measurement \( y(t_i) \). The weighting matrix \( W \) is diagonal with \( W_{ii} = 1/w_i \). If the function \( \hat{y} \) is nonlinear in the model parameters \( p \), then the minimization of \( \chi^2 \) with respect to the parameters must be carried out iteratively. The goal of each iteration is to find a perturbation \( h \) to the parameters \( p \) that reduces \( \chi^2 \).

The Levenberg-Marquardt algorithm adaptively varies the parameter updates between the gradient descent update and the Gauss-Newton update [10],

\[
J^T W J + \lambda I \hat{h} = J^T W (y - \hat{y}) ,
\]

where \( m \times n \) Jacobian matrix \( J = \partial \hat{y} / \partial p \) and small values of the algorithmic parameter \( \lambda \) result in a Gauss-Newton update and large values of \( \lambda \) result in a gradient descent update. The parameter \( \lambda \) is initialized to be large. If iteration happens to result in a worse approximation, \( \lambda \) is increased. As the solution approaches the minimum, \( \lambda \) is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically converges rapidly to the local minimum [6]. Marquardt’s suggested update relationship [10],

\[
J^T W J + \lambda \text{diag}(J^T W J) \hat{h} = J^T W (y - \hat{y}) ,
\]

The algorithm adjusts \( \lambda \) according to whether \( \chi^2 \) is increasing or decreasing as follows:

Given an initial guess for the set of fitted parameters \( p \)

1. Compute \( \chi^2(p) \)
2. Choose a value for \( \lambda \), for instance \( \lambda = 0.001 \)
3. Calculate \( h \) and evaluate \( \chi^2(p+h) \)
4. If \( \chi^2(p+h) \geq \chi^2(p) \) increase \( \lambda \) by a factor and go to (3) and try an update again.
5. If \( \chi^2(p+h) < \chi^2(p) \) decrease \( \lambda \) by a factor, accept the updated trial solution \( p \leftarrow p+h \) and go to (3) and try an update again.

The reasoning of the method is that if the error is increasing, the quadratic approximation in the Gauss-Newton method is not working well and we are likely not near a minimum, so \( \lambda \) should be increased in order to blend more towards steepest descent.
On the other hand, if the error is decreasing, the approximation is working well, and we expect that we are getting closer to a minimum so \( \lambda \) is decreased to blend more towards Gauss-Newton. Levenberg-Marquardt’s algorithm has the disadvantage that if the value of damping factor, \( \lambda \), is large, inverting \( (J^TWJ + \lambda I) \) is not used at all. Marquardt \[12\] realized that each component of the gradient can be scaled according to the curvature so that there is larger movement along the directions where the gradient is smaller. Thereby avoiding slow convergence in the direction of small gradient. Marquardt utilized this by replacing the identity matrix, \( I \) in Levenberg-Marquardt’s original equations Equation (11) with the diagonal matrix consisting of the diagonal elements of \( J^T W J \) (the Hessian matrix), resulting in the Levenberg-Marquardt algorithm, Equation (12) which then includes an estimation of the local curvature information and uses this to move further in the directions in which the gradient is smaller.

2.5 Numerical implementation

This work including the enhancement of a rank-1 Jacobian update. In iteration \( i \), the step \( h \) is evaluated by comparing \( \chi^2(p) \) to \( \chi^2(p + h) \). the step is accepted if the metric \( \rho_i \) [10] is greater than a user-specified value, \( \epsilon_4 \),

\[
\rho_i(h) = \frac{\chi^2(p) - \chi^2(p + h)}{2h^T (\lambda_i + J^T W (y - \hat{y}(p)))},
\]

(13)

If in an iteration \( \rho_i(h) > \epsilon_4 \) then \( p + h \) is sufficiently better than \( p \), \( p \) is replaced by \( p + h \), and \( \lambda \) is reduced by a factor. Otherwise \( \lambda \) is increased by a factor, and the algorithm proceeds to the next iteration.

2.5.1. Initialization and update of the L-M parameter, \( \lambda \), and the parameters \( p \) [10]

1. \( \lambda_0 = \lambda_0; \) \( \lambda_0 \) is user-specified.
\[
[J^TWJ + \lambda I \text{ diag}[J^TWJ]]h = JTW (y - \hat{y}(p));
\]
if \( \rho_i(h) > \epsilon_4 ; p \leftarrow p + h; \lambda_{i+1} = \max[\lambda_i / L_i, 10^{-7}]; \)
otherwise: \( \lambda_{i+1} = \min[\lambda_i / L_i, 10^{7}]; \)

2. \( \lambda_0 = \lambda_0 \text{ max}[\text{ diag}[J^TWJ]]h; \) \( \lambda_0 \) is user-specified.
\[
[J^TWJ + \lambda I]h = JTW (y - \hat{y}(p));
= \left( \left( J^TW (y - \hat{y}(p)) \right)^T h \right) / \left( \left( \chi^2(p + h) - \chi^2(p) \right) / 2 + \left( J^TW (y - \hat{y}(p)) \right)^T h \right)
\]
if $\rho_i(h) > \epsilon_4; p \leftarrow p + ah; \lambda_{i+1} = \max[\lambda_i/(1 + \alpha), 10^{-7}]$
otherwise: $\lambda_{i+1} = \lambda_i + |\lambda^2(p + ah) - \lambda^2(p)|/(2\alpha)$;

3. $\lambda_0 = \lambda_0 \max[\text{diag}[J^TW]] h; \lambda_0$ is user-specified.
$[J^TW + \lambda_i I]h = J^W(y - \hat{y}(p))$
if $\rho_i(h) > \epsilon_4; p \leftarrow p + ah; \lambda_{i+1} = \lambda_i \max[1/3, 1 - (2\rho_i - 1)^3]$
otherwise: $\lambda_{i+1} = \lambda_i + |\lambda^2(p + ah) - \lambda^2(p)|/(2\alpha)$;

where $\epsilon_4$ is determines acceptance of a L-M step

2.5.2. Computation and rank-1 update of the Jacobian, $[\partial y/\partial p]$

In the first iteration, in every $2n$ iterations, and in iterations where $\chi^2(p + h) > \chi^2(p)$, the Jacobian ($J \in \mathbb{R}^{m \times n}$) is numerically approximated using forward differences,

$$J_{ij} = \frac{\partial \hat{y}_i}{\partial p_j} = \frac{\hat{y}(t_i; p + \delta p_j) - \hat{y}(t_i; p)}{\|\delta p_j\|}, \quad (14)$$

or central differences (default)

$$J_{ij} = \frac{\partial \hat{y}_i}{\partial p_j} = \frac{\hat{y}(t_i; p + \delta p_j) - \hat{y}(t_i; p - \delta p_j)}{2\|\delta p_j\|}, \quad (15)$$

where the $j$-th element of $\delta p_i$ is the only non-zero element and is set to $\Delta_i(1 + |p_j|).n$
all other iterations, the Jacobian is updated using the Broyden rank-1 update formula,

$$J = J + \frac{(\hat{y}(p + h) - \hat{y}(p) - Jh)h^T}{h^Th}, \quad (16)$$

For problems with many parameters, a finite differences Jacobian is computationally expensive. Convergence can be achieved with fewer function evaluations if the Jacobian is re-computed using finite differences only occasionally. The rank-1 Jacobian updates equation (16) requires no additional function evaluations.

2.5.3. Convergence criteria

Convergence is achieved when one of the following three criteria is satisfied,

1. Convergence in the gradient, $\max[J^TW(y - \hat{y})] < \epsilon_1$;
2. Convergence in parameters, $\max |h/p_i| < \epsilon_2$;
or
3. Convergence in $\chi^2$, $\chi^2/(m - n + 1) < \epsilon_3$. 78
where $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$ are convergence tolerance for gradient, parameters and Chi-square respectively.

Otherwise, iterations terminate when the iteration count exceeds a pre-specified limit.

To get data from sensor and solve equation (1) numerically for estimate nine parameters, Figure 6 illustrates the graphical user interface of the developed calibration software, interfaced with a Matlab.

3 RESULTS AND DISCUSSION

This section gives some calibration results for a 3D accelerometer specifications in [13]. The calibration procedure has been performed three times successively, according to section (II). Each time, 4800 data points are used for the estimation of parameters. The obtained gains, bias and orthogonality are then compared in the following tables. The estimation algorithm LMA fit performs the alpha, beta and $S_{ij}$ computation instantly, with about less than 100 iterations [4].

In Table 1 the calibration results for a 3D accelerometer sensor. The sensor must only measure the static acceleration of gravity, for which the norm is constant. We need to exclude measurements where the sensor is not static, for which dynamic acceleration is present.
by observing the temporal variance of the data signal. In static position the accelerometer has a low variance.

Figure 7 shows that the estimated parameters define a sphere according to the model equation (2) that fits the measurement [3]. The calibration results in Table 1 shows that the gains, bias and the orthogonality of sensors gives very good stability. In some applications the orthogonality errors cannot be taken into account because of the low impact on the measurements [3].

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_x$</td>
<td>-0.0103</td>
<td>-0.0099</td>
<td>-0.00101</td>
</tr>
<tr>
<td>$\beta_y$</td>
<td>0.0010</td>
<td>-0.0008525</td>
<td>-0.0094</td>
</tr>
<tr>
<td>$\beta_z$</td>
<td>-0.0186</td>
<td>-0.0186</td>
<td>-0.0186</td>
</tr>
<tr>
<td>$\alpha_x$</td>
<td>0.9936</td>
<td>0.9930</td>
<td>0.9938</td>
</tr>
<tr>
<td>$\alpha_y$</td>
<td>1.0105</td>
<td>1.0098</td>
<td>1.0102</td>
</tr>
<tr>
<td>$\alpha_z$</td>
<td>0.9914</td>
<td>0.9917</td>
<td>0.9920</td>
</tr>
<tr>
<td>$s_{xy}$</td>
<td>-0.0067</td>
<td>0.0011</td>
<td>0.00118</td>
</tr>
<tr>
<td>$s_{xz}$</td>
<td>-0.00096</td>
<td>0.000766</td>
<td>0.000806</td>
</tr>
<tr>
<td>$S_{yz}$</td>
<td>0.0072</td>
<td>0.000528</td>
<td>0.00055</td>
</tr>
<tr>
<td>standard deviation for x axis</td>
<td>0.001419</td>
<td>0.000956</td>
<td>0.0015438</td>
</tr>
<tr>
<td>standard deviation for y axis</td>
<td>0.00156</td>
<td>0.00194</td>
<td>0.00167</td>
</tr>
<tr>
<td>standard deviation for z axis</td>
<td>0.00115</td>
<td>0.00123</td>
<td>0.00145</td>
</tr>
</tbody>
</table>

### 4 CONCLUSIONS

The numerical calibration of 3D accelerometers sensors is not an expensive solution and can be implemented with no mechanical means. The calibration procedure can be performed after the integration phase of electronics to take into account the drift and uncertainty of the component itself, all the gains, bias and geometry. The calibration parameters of accelerometers are accurate at 1%.

This method of calibration has been successfully used for the preparation of MicroElectroMechanical Systems (MEMS) inertial units, in a pedestrian tracking system, robot and UAVs. It is worth noting the possibility of using this method in calibration of magnetometer.
5 REFERENCES


