UDC 004.05, 004.41

A RAPID ANALYSIS OF HARDWARE-SOFTWARE TOOLS FOR EXPLORING INERTIAL MEASUREMENT UNITS BASED ON NORMALITY TESTS

T. A. Marusenkova
Lviv Polytechnic National University

Abstract. *The work deals with a problem of testing hardware-software tools for exploring inertial measurement units. We have proposed a rapid analysis method for detecting defects in such tools, based on normality tests. We have found relationships between the nature of defects and the results of normality tests.*

Keywords: *IMU, accelerometer, gyroscope, Shapiro-Wilk test, rapid analysis, normality test.*

Introduction

Inertial measurement units (IMUs) especially those based on micro-electromechanical systems (MEMS) find their application in various spheres including but not limited to medicine [1, 2], avionics and robotics. For instance, inertial sensors are used in order to recognize a human posture/gait, distinguish between normal and pathological human movements, classify the actions of a worker as right or wrong, track the movements of a sportsman, and even detect the first signs of loosing footage and prevent damages caused by falling down [3]. Their key feature is that they are autonomous, i.e., they do not require any external information for estimation of the coordinates and orientation, in contrast to well-known GPS navigation systems. However, MEMS IMUs are susceptible to errors, which should be carefully modeled, studied and compensated for, otherwise these devices are scarcely applicable.

Typically an IMU contains a triaxial accelerometer, a triaxial gyroscope and additionally a 3D magnetometer. IMU devices supplied with a magnetometer are commonly called MARG (Magnetic, Angular Rate and Gravity). We will use mainly IMU for brevity when nothing more specific than a combination of inertial sensors is meant. In order to achieve more consistent and reliable results, they use data fusion – readings of all the sensors are combined to fight fallacies of each individual sensor.

There is a multitude of IMUs available in the market. Most commercially available modules provide either raw data, or data after mathematical processing, however processing algorithms are not open-access. The Kalman filter [4] is considered to be the standard de-facto for data fusion in IMUs. To name a few, inertial sensors by xsens, micro-strain, VectorNav, Intersense, PNI and Crossbow all use this filter. Despite their obvious effectiveness and wide-spread popularity, the Kalman filter-based solutions are not free of disadvantages. For one thing, they are rather difficult to implement and consume a lot of computational resources. Besides, they demand high sample rates in applications where one might not expect this condition. For instance, a sampling rate exceeding 512 Hz may be used for a human motion caption. The aforementioned disadvantages led to appearance of modern, less computationally demanding solutions among which we should mention those proposed by Bachman et al [5], Mahony et al [6] and, doubtlessly, the famous Madgwick filter [7].

Ready-to use solutions by well-known companies may be sufficient for a wide range of applications. Nevertheless, in order to have a full control over the output of inertial sensors one might be interested in development of custom firmware solutions whose hardware parts are commercially available accelerometers, gyroscopes and magnetometers (individual or combined into IMU/MARG modules), auxiliary hardware units and microcontrollers or alternative electronic devices that are able to serve as "the brain" of a navigation system. Programmatically, such a "brain" can be flashed by implementations of custom algorithms, not necessarily limited to the Kalman filter, for example, any of the abovementioned alternative filters or some novel solutions, which are likely to keep emerging in the near future. A key point is that custom hardware and firmware open new opportunities for researchers and engineers when seeking for a reasonable combination of hardware components suitable for a specific task/application area under a set of restrictions (weight, size, price, power consumption, sample rate, etc). The problem of testing custom hardware-software tools is underestimated in the subject literature, presumably because it is more common to take data from ready-to-use software provided by the manufacturer of IMUs.

The work shows that IMU measurement results may look completely trustworthy while being erroneous and proposes a rapid analysis method of detecting defects in custom hardware-software tools for exploring IMUs.

The hardware-software tool architecture

There are several reasons why not only firmware solutions but also PC-based applications for communication with individual inertial sensors and IMUs are of key importance. Firstly, such a hardware-software tool is fundamental for the researcher who wants to investigate into actual parameters of a specific accelerometer/gyroscope, for the characteristics of any device specimen may slightly differ from those stipulated in the datasheet for the whole family. Secondly, whatever results might have been obtained over simulated data when testing a novel data fusion algorithm, no final conclusions should be made without running the latter over a set of experimental data, obtained from a real IMU. I.e., the quality of mathematical processing of raw sensor data should be evaluated by benchmarking against some dependable reference. A typical approach is to take readings of an IMU/MARG, calculate the position and attitude of an object being tracked and compare the results with an optical measurement system (the object has marks captured and recognized by a camera [8]). Also, comparison of different approaches to data fusion is to be conducted using real sensor data. Thirdly, such a hardware-software tool may be used when deciding whether the underlying IMUs are good enough in order to be used with a specific purpose.

At least, the aforementioned hardware-software tool is comprised of the following hardware parts: 1) a set of inertial sensors, either individual or integrated into a single IMU module; 2) a microcontroller or similar smart unit for data acquisition and configuring the IMU module; 3) interfaces through which the microcontroller or other controlling logic can be accessed (I^2C, SPI, UART, etc).

As usually, any peripheral module can be configured via a set of registers.

The software part contains (as a separate module or somehow else) the following functionality: reading/writing bits of controlling registers in the underlying hardware; acquisition of raw sensor readings; UI and the underlying logic for different operation modes supported by the IMU; representation of raw sensor readings in the corresponding physical quantities (for example, milliG for accelerometers or degrees per second for gyroscopes); graphical representation of readings and dumping readings into a file. The author is a co-developer of IMUTester, a hardware-software tool for exploring IMUs, some windows of which are shown in Fig. 1.

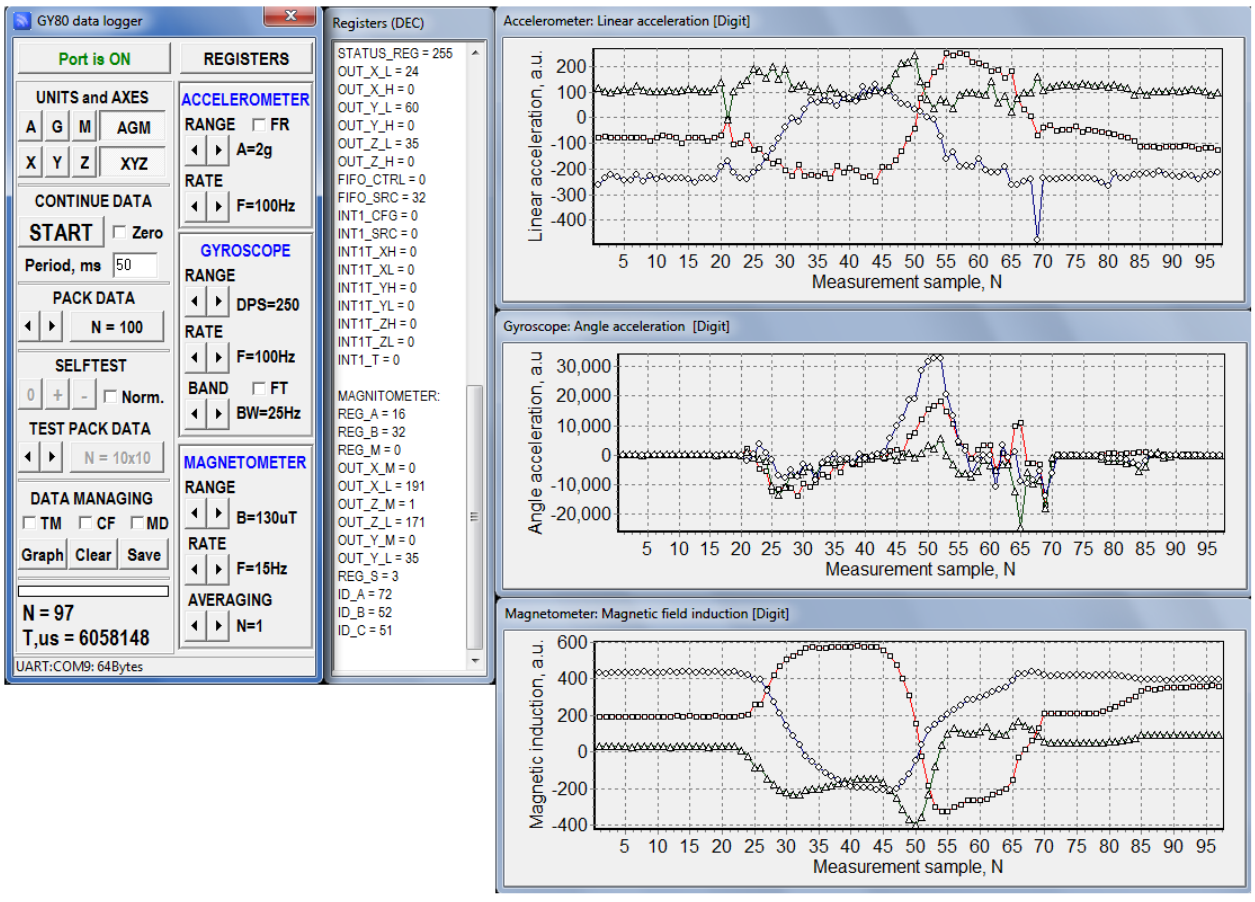


Fig. 1. Sample windows of IMUTester, a hardware-software tool for exploring IMUs

We do not enter into details of the tool here since it has been partially described and will be fully covered in our future publications for one thing and for another – it is not the focus of this work and serves only as an example. IMUTester contains all the aforementioned features and many more. Its hardware basis is GY-80, a module comprised by a triaxial digital accelerometer ADXL345, a 3D digital gyroscope L3G4200D, a 3D compass HMC5883L, and some auxiliary nodes, but again – the underlying IMU/MARG might have been some another model. The accelerometer ADXL345 supports measurement ranges ±2 g, ±4 g, ±8 g, or ±16 g and resolutions 10, 11, 12 or 13 bits.

Most of the subject literature on IMUs consider issues of compensation for their inaccuracies, error models, methods of errors identification and measurement, data filtering and data fusion and so forth and simply rely on the fact that data for analysis come directly from sensors without any distortion during transmission. This is exactly the case when the researcher uses software supplied by the manufacturer of an IMU being studied. However, if the researcher decides to develop their own hardware-software tool for investigation, as we did, they will have to deal with a problem of verification of their tool. Since, as we have shown above, operation of IMUs may be controlled by different registers (see sub-window Registers in Fig. 1) and data can be accessed via several different interfaces, there are multiple possible sources of software/firmware defects and one needs to ensure that data are transmitted from hardware part to software part correctly, without distortion.

Testing is a time-consuming and expensive process, thus a rapid analysis method detecting the presence of defects would be rather helpful. For analysis of IMU outputs they typically use two main tools – the Allan variance [9, 10] and power spectral density [11, 12]. The latter is computationally expensive and provides results difficult to interpret [13]. The former is easier to understand and compute and consists in the following steps. Keeping an accelerometer/gyroscope steady, one acquires an equidistant time series, *y(t)* of length *N*, with the sample period τ_0. For some averaging factor *m*, one takes all possible overlapping sample clusters of period $\tau = m\tau 0$. Once all the possible $(N - m)$ sample clusters have been formed for the selected value of *m* (and τ), the Allan variance is calculated as a function of τ:

$$\sigma_y^2(\tau) = \frac{1}{2\tau^2(N-2m)} \sum_{i=1}^{N-2m} [x_{i+2m} - 2x_{i+m} + x_i]^2$$

where *x* is the cumulative sum of the time series.

Then one repeats the same steps for different values of *m* (and τ) and draws the Allan deviation plot, usually in a log-log format (the Allan deviation is the square root of the Allan variance). Different parts of the plot allow us to figure out different characteristics of the noise terms in the input signal. Despite the fact that the Allan variance is a powerful tool accepted as the standard for analysis of IMU signals, it is difficult to judge of the measurement correctness on its basis.

Issues of hardware-software covalidation are widely covered in the literature [16, 17], all of them being rather universal and general than specific. Most approaches focus on test generation, not on preliminary check for most likely fault sources. Moreover, while a wide range of testing techniques exist, an idea of using expected data distribution for detecting faults is rather novel.

It is known that measurement errors tend to normality. The goal of this work is to investigate into the applicability of normality tests for detecting errors in measurements and, therefore, in hardware-software tools for exploring IMUs. Let's emphasize that we are not aimed at test generation. Our goal is a quick low-cost tool that would help figure out the possible fault sources and, therefore, draw up a testing plan focusing on most faulty software parts.

Normality tests and a rapid analysis for detecting software defects

Normal (Gaussian) distribution is one of the most common. Many random variables in practice turn out to be normally distributed, because many outcomes are the result of many forces added together, and the sum of many random variables tends to be normally distributed in accordance with the central limit theorem, no matter what distributions each individual component in the sum came from. On the other hand, multiple natural non-Gaussian distributions exist in the real world. Some methods, like discrimination analysis, Pearson test, f-test or t-test, require data being analyzed to be normally distributed, i.e., before applying these methods the researcher has to check whether their sampled data could have come from a population with Gaussian distribution. For this reason, a set of methods of testing data for normality exist. Among them are Kolmogorov-Smirnov test, Shapiro-Wilk test, box-plots, Q-Q plots. Besides, one can judge of the distribution by visual examining a histogram that shows how many readings are concentrated in each sub-range of the range of possible values. These methods were designed as tools for discovering whether a sample could have come from a normal distribution, when the nature of this sample is unknown. The same methods may be used with the opposite purpose – to

check normality of a sample about which we know for sure that it should have come from a normal distribution. Negative normality check results may be used as a testament to errors in data. The latter indicates defects in a hardware-software tool.

Typically, when an IMU is kept static, its output should be somewhere around the offset value. Moreover, the IMU readings will come from a normal distribution. The manufacturer provides typical performance characteristics in datasheets. For instance, Fig. 2 depicts how specimens of X-axis g offset at 25 ºC and supply voltage 3.3 V are distributed over their possible range. The histogram is taken from the datasheet of the accelerometer ADXL345. One can recognize a normal distribution.

It is known [14] that the Shapiro-Wilk test provides trustworthy results and many researchers prefer it to concurrent methods. It is recommended, though, to run several different normality tests in order to avoid incorrect conclusion.

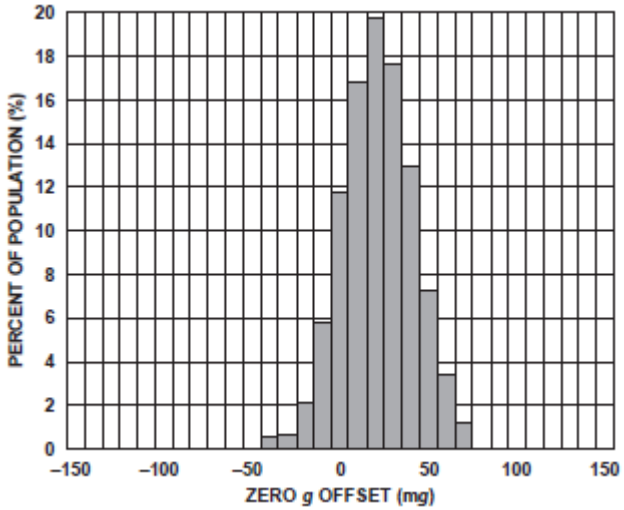


Fig. 2. Distribution of X axis zero g offset in the accelerometer ADXL345 at 25 ºC and 3.3 V

The Shapiro-Wilk test assumes that a sample $x_1,\ldots,x_n$ did come from a normally distributed population with unspecified mean and variance (the null hypothesis). Then the following test statistics should be applied:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \qquad (1)$$

where $x_{(i)}$ is the *i*-th order statistic, i.e., the *i*-th item of the sample sorted in ascending order, $\bar{x}$ is the sample mean, and a_i are the items of the vector

$$\left(a_1,\ldots,a_n\right) = \frac{m^{\mathrm{T}} V^{-1}}{C} \qquad (2)$$

Here vector $m = \left(m_1,\ldots,m_n\right)^{\mathrm{T}}$ is made of the expected values of the order statistics of IID (independent and identically distributed) random variables sampled from the standard normal distribution (a normal distribution whose mean is 0 and standard deviation is 1). *V* is the covariance matrix of the normal order statistics. *C* is a vector norm calculated by the following formula:

$$C = \left\| V^{-1} m \right\| = \left(m^{\mathrm{T}} V^{-1} V^{-1} m \right)^{1/2} \qquad (3)$$

Random variables $X_1,\ldots, X_n$ can be considered IID only if the following condition is fulfilled:

$$F_{X_1}(x) = F_{X_k}(x) \quad \forall k \in \{1,\ldots,n\} \text{ and } \forall x \in I$$

$$F_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = F_{X_1}(x_1)\cdot\ldots\cdot F_{X_n}(x_n)$$
$$\forall x_1,\ldots,x_n \in I$$

where

$F_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = \mathrm{P}\left(X_1 \le x_1 \wedge \ldots \wedge X_n \le x_n\right)$ is a joint cumulative distribution function of $X_1,\ldots, X_n$.

The Shapiro-Wilk test is more applicable for platykurtic samples whereas Shapiro-Francia test is more suitable for leptokurtic samples.

Kurtosis is defined as the fourth standardized moment and serves as a measure of the "tailedness" of the probability distribution of a real-valued random variable. One may use the following formula

$$\mathrm{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mathrm{E}\left[(X-\mu)^4\right]}{\left(\mathrm{E}\left[(X-\mu)^2\right]\right)^2} \qquad (4)$$

which is not a unique way of calculating kurtosis, though.

The classic Shapiro-Wilk test is intended for a sample of size not exceeding 5000. For a larger sample there exists an extension to the Shapiro-Wilk test – the Royston's test.

If p-value turns out to be greater than the accepted significance level, α (also called alpha level and type I error rate), the researcher fails to reject the null-hypothesis, but it does not mean that the null hypothesis can be accepted. It can be easily seen if one takes several uniformly distributed samples and applies the Shapiro-Wilk p-value test – for some of the samples the p-value will exceed the significance level, which means a failure to reject the hypothesis about normally distributed data.

In statistical test theory, they distinguish type I error (false positive, false alarm) and type II error (false negative). The former relates to situations when the null hypothesis is true but it has been erroneously rejected. On the contrary, the latter occurs

when the null hypothesis is false but the researcher has failed to reject it. The alpha level can be interpreted as the probability of a type I error. In our case, the alpha level is the probability of an incorrect conclusion that a sample could not have come from a normal distribution given that it actually did. The significance level can be chosen arbitrarily. Its most typical values are 0.05, 0.01, 0.005 and 0.001.

With respect to the goal of our work, we suggest to expose readings of an accelerometer and gyroscope to one or several tests for normality, among which there should be the Shapiro-Wilk p-value test with the significance level 0.001. If the test results show that the p-value is less than the significance level, it will be a clear indication that the hardware-software tool being tested contains defects in its part responsible for acquiring readings from the IMU.

We suggest that the following steps should be taken.

1. Acquire readings of all the accelerometers and gyroscopes incorporated in the hardware part of the tool being tested. During data acquisition the hardware should not be moved, otherwise the results cannot be considered trustworthy. A sample size does matter – if there are too few readings, the test results are likely to be false. On the other hand, it is well known that results may be inaccurate for large samples. That is why we suggest that one should take several samples of different sizes, test them and average the test results.

2. Run the Shapiro-Wilk tests based on Eqs. (1) – (3) and analyze the calculated p-value. It may be that the tool provides measurements in packages of a fixed size (for example, 10000). In this case, either Royston's test should be used or the sample is to be split so that its sub-samples have a suitable size (up to 5000 items). Moreover, if kurtosis (defined by (4)) is greater than 3, we run both Shapiro-Wilk and Shapiro-Francia tests (we still use kurtosis despite its controversial history [15]). Otherwise, we run the Shapiro-Wilk test only. Because of possible inaccuracies related to large samples, we form smaller samples, selecting each 5th, 10th, 20th, 50th and 100th item in an original sample if the latter contains 10000 items or more.

If a gyroscope outputs integrated measurements (angles instead of angular rate), then it is advised to analyze several subsamples. We take first N points (N should not be less than 500), last N points and N equidistant points throughout the whole sample. This is to ensure that rate random walk, which is likely to be present in measurements after some time elapsed and errors have cumulated, does not impact the test results. I.e. one has to compare the test results for the first and last portions of data and data uniformly scattered over the original sample.

If p-value does not exceed the significance level (0.001), we conclude that our readings could not have come from a normal distribution. The probability that this conclusion is false is only 0.001.

3. Build histograms and Q-Q plots. In addition to the numerical results of the Shapiro-Wilk test and its modifications, visualized data are able to provide an additional hint about the nature of measured data.

4. Sort the sampled data and search for discontinuities. The effectiveness of this step is illustrated with Figs.3, a and 4. The original sample (Fig. 4, a) looks much as white Gaussian noise, i.e., the measurement results seem to be pretty normal. However, when sorted and plotted (Fig. 5), the same sample provides clear evidence that the data could not have come from a normal distribution, because there are discontinuities in the data. Some similar information can be obtained from visual examination of histograms, but in the case of sorting sampled data visualization may be omitted – it is enough to find the maximum difference between adjacent items and calculate its ratio to the measurement range. Discontinuities are most likely attributable to incorrect processing of the accelerometer or gyroscope's registers that contain measurement results.

5. Calculate the variance of the sample. The variance is helpful when assessing the correctness of measurement ranges and data scaling.

6. Refer to the datasheet for performance characteristics by the manufacturer. If there is a clear indication that measurement results are expected to be normally distributed, then a test for normality has either of two meanings. First, if we failed to prove that the data are likely to have come from a normal distribution, then careful retesting of the hardware-software tool should be definitely done. Otherwise, no conclusions should be made. If the manufacturer provides no information on readings distribution then normality check, when enabled, can be used for two purposes: 1) detecting defects under a reasonable assumption that data are normally distributed and 2) exploring the output of accelerometers / gyroscopes. We suggest normality tests to be included into the software part of the hardware-software tool as an additional instrument of its smoke testing.

Common faults in hardware-software tools that deals with inertial sensors

As can be deduced from Fig. 3, raw sensor data may be corrupted inside an IMU, inside software or in between (due to incorrect transmission). Therefore, all faults of hardware-software tools for exploring IMUs fall into three groups: hardware faults, communication faults and software faults. We discard an idea of hardware faults if IMUs by reputable vendors are used and assume that we deal with either

of two issues: misinterpretation of correct sensor data or incorrect transmission of correct sensor data from hardware to software.

Taking into account the fact that each inertial sensor is controlled via several dozens of registers and all of these registers are accessed via one of the standard interfaces, there are numerous possible software faults, usually not easily distinguishable.

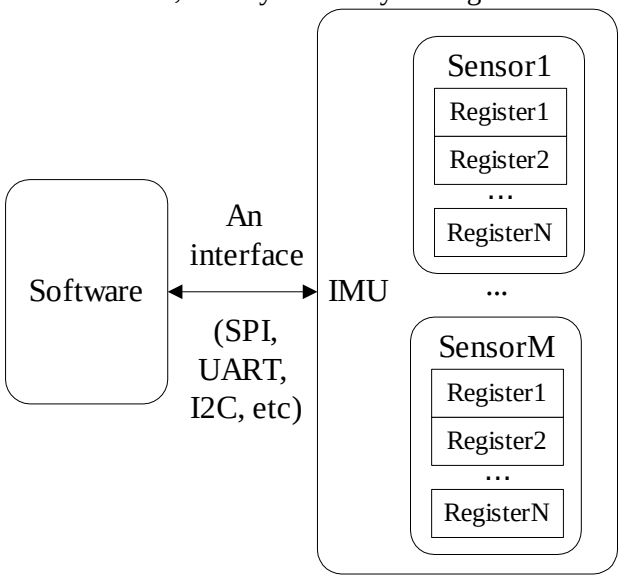


Fig. 3. Architecture of a hardware-software tool for exploring IMUs

Here we list some faults likely to happen in practice, and misinterpretation of data registers comes first. All measurement results are placed in data registers of an inertial sensor and read by software. A function for reading a data register via SPI may look like this:

```
void SPI_Read (uint8_t reg, uint8_t* val) {
    Set_State_CS(GPIO_PIN_RESET); /* This function resets the pin intended for selecting a slave device, i.e., the pin will be low. This procedure launches data exchange with the selected device */
    reg = reg | 0x80; /* here we set the highest bit to prescribe reading data from the register defined by variable reg; this is a feature of a motion sensor LIS302DL inside an stm32f4Discovery board version MB997B */
    HAL_SPI_TransmitReceive(&SPIInitStruct, &reg, val, 1, SPITIMEOUT);
/* here we are reading, hence, it does not matter which value we transmit to the slave device */
    reg = 0;
    HAL_SPI_TransmitReceive(&SPIInitStruct, &reg, val, 1, SPITIMEOUT);
    Set_State_CS(GPIO_PIN_SET); /* The function sets the pin intended for selection a slave device, thus terminating data exchange with it */
}
```

If the programmer has used a different data type accidentally, the results will be distorted. For example, if a data register is one byte long and stores signed values, and its values are dumped into an unsigned char variable, the values will lose their sign. It can be that a sensor places the measurement results into two separate registers, and the programmer should restore the true measurement result by fetching its parts from the two registers and "gluing" them together. If the programmer accidentally swaps the high and low bytes, the measurement results will be false. For example, sensor readings are random values between 384 and 511 (here we intentionally omit measurement units and focus on bare numbers only). In the binary systems these numbers are represented as 00000001 (high byte) and 1 b6 b5 b4 b3 b2 b1 b0 (low byte), where the least 7 bits in the low byte may take any binary value. If one swaps the bytes, there will be a substantial discontinuity in readings, because the part of the byte with fixed zeros means that the corresponding numbers are "fallen out" and will never be observed. Moreover, in addition to big endian and little endian issues, there can be an issue with the order of bits – most significant bit first or least significant bit first (MSB or LSB). For example, the SPI interface supports both MSB and LSB, however, most slave devices are able to communicate using either MSB or LSB but not both. Thus, if the master device uses different settings than those of a slave device, measurement results will be misinterpreted. For instance, the true measurement is 1 and the master reads 128 (binary 10000000). Therefore, numbers 1, 2, 3 and 4 will be misunderstood as 128, 64, 96 and 32, and here we can see discontinuities again. It is a likely mistake to import all low-level functions for data exchange via some common interface from some project that has been developed earlier and to forget about data type adjusting for each specific case.

One more problem with data registers is the risk that they are being updated with fresher data exactly while they are being read. One should prevent this overwriting by controlling a specific bit. If no prevention was done, data registers contain mixed bits, partly old and partly new. Obviously, together they represent a numerical value that has no sense.

All the shown examples are purely software faults related to discrepancies in data representation. In general, the described situations can be emulated by masking groups of bits, whether adjacent or not.

Another common problem is misuse of an interface. Common issues with an interface include but are not limited to 1) an attempt to read/write registers when the bus is busy without checking the bus state; 2) an attempt to read/write registers with a frequency that exceeds the maximum frequency of the slave device (an IMU sensor); 3) an attempt to read/write registers when the slave device does not

respond without checking the presence and state of the slave; 4) blocking conditions, i.e., the presence of loops that wait for some specific flag with no preset time limits. Problems related to conflicting settings of the master and slave devices were mentioned before, when discussing improper treatment of data registers. Other common mistakes include the permanent absence/unavailability of a slave device, a failure to address a slave device properly and an attempt to misaddress the register inside the slave device, but here we do not focus on these faults since they are easy to detect. Most faults related to communication interfaces are due to a lack of programmatic checks for handling run-time failures.

Let's consider the first mentioned communication fault source, a lack of checking whether the bus is free. Typically, a value from the recently read register is stored into a dedicated variable X. If no new data were read because the bus was busy, X stores its previous value. A particular case is that the bus is never free, and X will store the value it was initially assigned implicitly or explicitly (typically 0). This particular case is too easy to reveal, thus we do not focus on it. Each failure to update the value of X increases the total amount of measurements and the frequency of the previous measurement result. For example, if data are being acquired continuously during 10 seconds, and after some value Y in the middle of this period no updates were observed during 1 second, then the value Y will be at least 10 times more frequent than other values – one will observe a histogram with a bar much taller than its neighbors. However, more naturally, communication issues manifest themselves either permanently or periodically, because there is a constant amount of concurrent slave devices on the bus and they are polled with a constant frequency. These periodical failures to read data registers lead to appearance of multiple bars of nearly the same height in the histogram. As a result, the histogram will resemble a uniform rather than a normal distribution, i.e., the bell shape of a normal distribution smears, being distorted by false measurement results. Other typical communication faults have a similar effect on measurement results provided that they occur with nearly the same periodicity.

The next potential fault source relates to the fact that some IMUs support not one but several measurement ranges. A failure to set the bit(s) that prescribe which measurement range should be used causes incorrect measurement scaling.

Upon the given examples, one may expect that the considered software faults make sensor readings no longer normally distributed, and this is the grounding of our hypothesis that normality tests can help reveal these software faults.

Approbation methods and results

In order to verify our idea that normality tests are applicable for detection of defects in custom hardware-software tools for exploring IMUs, we took several steps.

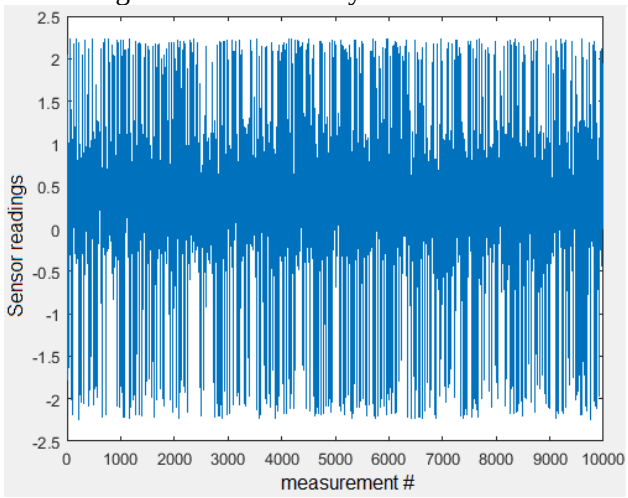
First, we have performed numerical simulation in order to find out which sample size is sufficient. We took a range of well-known distributions (continuous uniform, discrete uniform, binomial, exponential, Pareto, Bernoulli, Poisson and Rayleigh). Our choice of probability distributions was done only with the purpose to have some diversity while not overload our research with too many functions to deal with. We simulated 100 samples of sizes 10, 20, 50, 100, 200 and 500 items per each distribution. It turned out that for uniformly distributed data one can observe a false alarm almost in one third of the samples with 50 items, in approximately 10% of the samples containing 100 items each, occasionally – in samples of size 200 and never – in samples of 500 items. Samples of size 10 and 20 proved to be highly unreliable – the Shapiro-Wilk test results failed to reject the null hypothesis in nearly half of the occasions. Thus, we can conclude that 500 is the minimal size for a sample.

Then, we intentionally injected faults that are likely to be met in practice in our hardware-software tool, IMUTester. For each defect and combination of defects, we acquired real data from the accelerometer ADXL345 and gyroscope L3G4200D. Erroneous treatment of acquired data was not obvious. For instance, when rotating the device, we saw the corresponding changes for all axes in Gyroscope window, and when we stopped rotation, measurement results settled again. All the injected faults fall into three groups: 1) those simulating misinterpretation of data registers; 2) those simulating communication problems and 3) scaling faults.

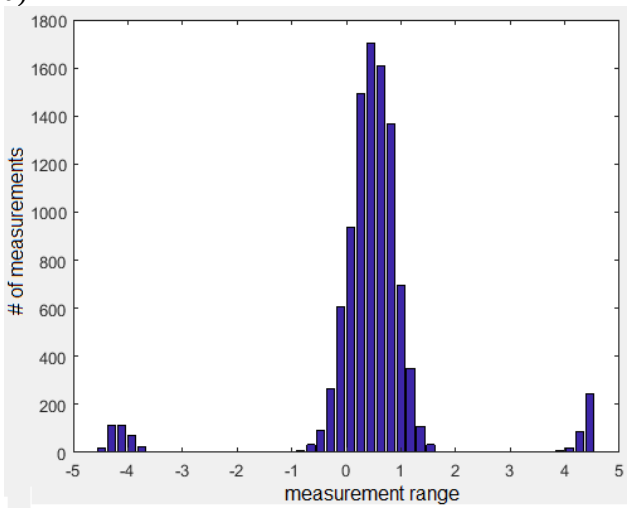
A generalized approach to simulate software faults related to misinterpreting data registers is to distort individual bits and groups of bits in the variables where the values of data registers were dumped. Thus, we formed and applied the following masks for one-byte-long data register: 01111111, 11111110, 11101111, 11100111, 11111100, 10001111, 11000011, 10011001, 10101010. Hence, we emulated situations when one, two, three or four bits (adjacent or scattered) are distorted.

We simulated faults related to communication via a standard interface as follows. In a separate thread we called a function that sets/resets flags used by a communication interface with the defined periodicity. Simulation of scaling problems consisted in using raw sensor data without multiplication by a proper scaling factor.

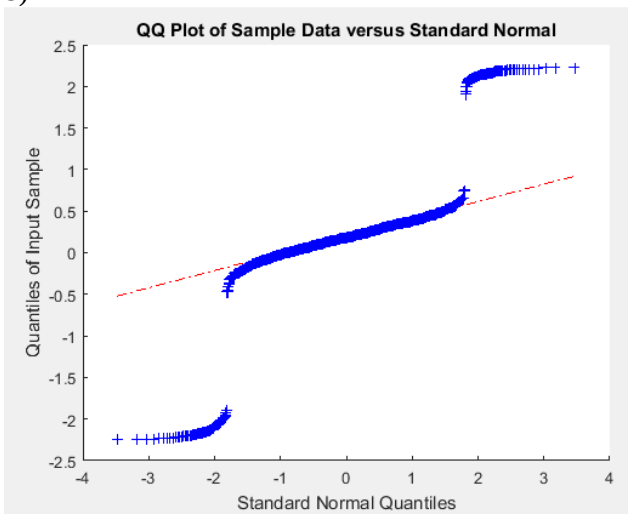
The results were as follows. Figs. 4–5 show an original sample, a histogram, a Q-Q plot and a sample plotted after sorting for the case when a single bit in a register is intentionally distorted.



a)



b)



c)

Fig. 4. A sample from the gyroscope L3G4200D with a single bit distorted (a), the distribution of measurements over the measurement range (b) and the Q-Q plot (c)

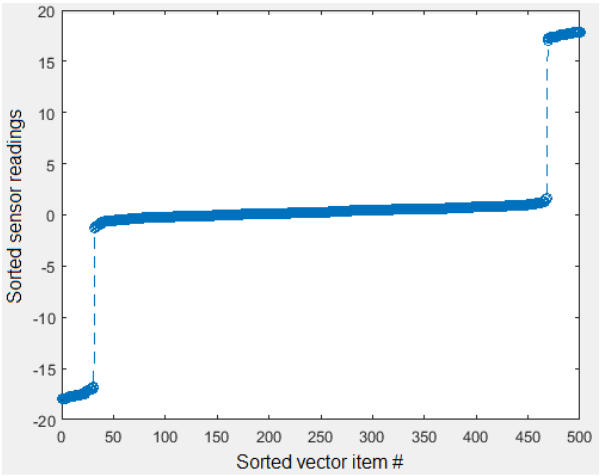


Fig. 5. A sorted sub-sample from the gyroscope L3G4200D (even items of the original sample)

For all cases where data registers were read with some bits distorted we received a clear indication that the sample could not have come from a normal distribution.

As one can see, all plots seem to exist in three parts, completely independent or loosely coupled, and there are substantial gaps in the sorted data. Neither of the plots resembles a normal distribution. Besides, the p-value for this sample was equal to 0, which undoubtedly allows us to reject the null hypothesis of the Shapiro-Wilk test. Similar results were obtained for other test cases with corrupted data registers. What these test cases have in common is zero p-value and discontinuities in all types of graphs.

Simulated communication problems resulted in smearing the bell shape of a normal distribution and non-zero p-value. The Shapiro-Wilk test still allowed us to reject the hypothesis. No discontinuities where observed in either of the plots.

Improper data scaling changed the variance, but distorted neither shape of histograms or plots nor the results of the Shapiro-Wilk test and its modifications. Thus, scaling type of errors cannot be revealed by checking for normality.

Conclusions

We have shown a problem of testing a custom hardware-software tool for exploring accelerometers and gyroscopes and proposed a rapid analysis method for discovering the presence of defects in the part of such a tool responsible for data acquisition and transmission to the software side. The proposed rapid analysis method relies on the fact that an accelerometer/gyroscope being kept static at least for some short time is likely to output normally distributed data. Thus, we check sampled data for normality using several tests. If these tests show that the data could not have come from a normal distribu-

tion, we conclude that data are sampled, transmitted or processed improperly. Otherwise, no conclusions can be made – a normal distribution does not prove the software correctness. We suggest that a set of normality tests should be included into the hardware-software tool for its self-diagnostics.

Approbation of the efficiency of normality tests in the context of software testing has been performed by injection of software faults of three groups: 1) improper treatment of data registers; 2) defects in communication interfaces; 3) incorrect scaling. It turned out that each of these groups can be characterized by some specific features. Namely, faulty registers lead to discontinuities in sampled data, well distinguishable in histograms and plots. Incorrect scaling changes the variance but do not affect other tests. Communication issues lead to smearing of the bell shape, besides, this error type can be detected by the Shapiro-Wilk test. Hence, the results of checking for normality not only testify for the presence of software defects but also give a hint where to search for them.

A list of the considered defects is rather limited – actually one can distinguish and describe possible defects in much more details. However, even this short list was sufficient to prove the main idea of the work – the applicability of normality tests for diagnostics of software defects. We are planning further investigation into other types of errors and recognition of their distinctive features.

References

1. Lee, J., Mellifont, R. and Burkett, B. (2010). The use of a single inertial sensor to identify stride, step, and stance durations of running gait. *Journal of Science and Medicine in Sport*, 13(2), pp.270–273. DOI:10.1016/j.jsams.2009.01.005

2. Tang, Z., Sekine, M., Tamura, T., Tanaka, N., Yoshida, M. and Chen, W. (2015). Measurement and Estimation of 3D Orientation using Magnetic and Inertial Sensors. *Advanced Biomedical Engineering*, 4(0), pp.135-143. DOI:10.14326/abe.4.135

3. Mao, A., Ma, X., He, Y. and Luo, J. (2017). Highly Portable, Sensor-Based System for Human Fall Monitoring. *Sensors*, 17 (9), p. 2096. DOI:10.3390/s17092096

4. Faragher, R. (2012). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]. IEEE Signal Processing Magazine, 29(5), pp.128–132. DOI: 10.1109/MSP.2012.2203621

5. Bachmann, E. R., Duman, I., Usta, U. Y., McGhee, R. B., Yun, X. P. and Zyda, M. J. (1999). Orientation tracking for humans and robots using inertial sensors. *In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'99*, pp. 187–194.

6. Mahony, R., Hamel, T. and Pimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5), pp. 1203–1218.

7. Madgwick, S., Harrison, A. and Vaidyanathan, R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. *2011 IEEE International Conference on Rehabilitation Robotics*. DOI: 10.1109/ICORR.2011.5975346

8. Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., and Savatier, X. (2017), A Study of Vicon System Positioning Performance. *Sensors*, 17(7), p. 1591. DOI:10.3390/s17071591

9. Riley, W. J. (2008). *Handbook of frequency stability analysis*. Washington: U. S. Government Printing Office, pp. 23–25.

10. El-Sheimy, N, Hou, H. and Niu, X. (2008). Analysis and modeling of inertial sensors using Allan variance. *IEEE Transactions on Instrumentation and Measurement*, 57(1), pp. 140–149. DOI: 10.1109/TIM.2007.908635

11. Krishna, A. P.R, and Andrews, J. (2015), PSD computation using modified Welch algorithm. *International Journal of Scientific Research Engineering & Technology (IJSRET)*, 4(9), pp. 951–954.

12. Sreelekha, S. and Sabi, S. (2016). Modified Welch power spectral density computation with Fast Fourier transform. *International Journal of Scientific Engineering and Research (IJSER)*, 4(10), pp. 92–96

13. Vukmirica, V., Trajkovski, I., and Asanović, N. (2010). Two Methods for the Determination of Inertial Sensor Parameters. *Scientific Technical Review*, 60(3–4), pp. 27–33.

14. Yap, B. W., and Sim, C.H. (2010). Comparisons of various types of normality tests. *Journal of Statistical Computation and Simulation*, 81 (12), pp. 2141–2155. DOI: 10.1080/00949655.2010.520163

15. Westfall, P. (2014). Kurtosis as Peakedness, 1905–2014. R.I.P. *The American Statistician*, 68(3), pp. 191–195. DOI:10.1080/00031305.2014.917055

16. Kooli, M., Benoit, P., Natale, G., Torres, L. and Sieh, V. (2014). Fault injection tools based on virtual machines," in 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip, ReCoSoC, [online], pp. 1–6. DOI: 10.1109/ReCoSoC.2014.6861351

17. Harris, I.G. 2003. Fault models and test generation for hardware-software covalidation. *IEEE Design & Test of Computers*, [online] 20(4), pp.40–47. DOI: 10.1109/MDT.2003.1214351

ЕКСПРЕС-МЕТОД ВИЯВЛЕННЯ ДЕФЕКТІВ У ПРОГРАМНО-АПАРАТНОМУ ЗАСОБІ ДЛЯ ДОСЛІДЖЕННЯ ІНЕРЦІЙНИХ ВИМІРЮВАЛЬНИХ ПРИСТРОЇВ НА ОСНОВІ КРИТЕРІЇВ НОРМАЛЬНОСТІ

Т. А. Марусенкова
Національний університет «Львівська політехніка»

Анотація. *Інерційні вимірювальні пристрої (ІВП) на основі мікроелектромеханічних систем набувають поширення завдяки їхнім численним перевагам. Однак, для таких пристроїв характерні похибки, які необхідно компенсувати за допомогою алгоритмів злиття даних. Програмно-апаратний засіб, у якому аналізуються покази інерційних сенсорів, є важливим інструментарієм для дослідження параметрів ІВП і верифікації алгоритмів злиття. Розроблення і верифікація власних програмно-апаратних засобів для дослідження ІВП є доволі складним і тривалим процесом. Дана робота спрямована на дослідження застосовності критеріїв нормальності до даних інерційних сенсорів для виявлення програмних дефектів у програмно-апаратних засобах для дослідження ІВП. Гіпотеза про застосовність критеріїв нормальності у цих цілях базується на тому, що більшість результатів вимірювання інерційних сенсорів у стані спокою тяжіють до нормального розподілу. Для дослідження нами вибрано критерій Шапіро-Вілка, графіки Q-Q, гістограми та додаткові перевірки на наявність розривів у відсортованих результатах вимірювань. Для верифікації цієї ідеї нами внесено дефекти у наш власний програмно-апаратний засіб для дослідження ІВП, IMUTester. А саме, розглядалися дефекти трьох груп: некоректне опрацювання регістрів даних, неправильне масштабування даних і некоректне використання інтерфейсу. Виявлено, що кожна група дефектів має вирізняльні риси у результатах застосування критеріїв нормальності. Зокрема, помилки опрацювання регістрів даних характеризуються розривами у всіх видах графіків і гістограм, а помилки обміну даними – розтіканням форми дзвона, типової для нормального розподілу. В роботі наведено аналітичне обґрунтування одержаних результатів. Враховуючи результати апробації, пропонуємо застосовувати розглянуті критерії нормальності у програмно-апаратних засобах для дослідження ІВП як один з засобів їхньої самодіагностики.*

Ключові слова: *інерційний вимірювальний пристрій, акселерометр, гіроскоп, критерій Шапіро-Вілка, експрес-метод, критерій нормальності.*

ЭКСПРЕСС-МЕТОД ОБНАРУЖЕНИЯ ДЕФЕКТОВ В ПРОГРАММНО-АППАРАТНОМ СРЕДСТВЕ ДЛЯ ИССЛЕДОВАНИЯ ИНЕРЦИАЛЬНЫХ ИЗМЕРИТЕЛЬНЫХ УСТРОЙСТВ НА ОСНОВЕ КРИТЕРИЕВ НОРМАЛЬНОСТИ

Т. А. Марусенкова
Национальный университет «Львовская политехника»

Аннотация. *Рассмотрена проблематика тестирования програмно-аппаратных средств для исследования инерциальных измерительных устройств. Предложен экспресс-метод обнаружения дефектов в таких средствах с помощью критериев нормальности. Выявлены закономерности между типом дефекта и результатами применения критериев нормальности.*

Ключевые слова: *инерциальное измерительное устройство, акселерометр, гироскоп, критерий Шапиро-Уилка, экспресс-метод, критерий нормальности.*

Received 19.02.2019

Tetyana Marusenkova, PhD, Associate Professor, Senior Lecturer at Software Department, Lviv Polytechnic National University, Bandery Str., 12, Lviv, Ukraine. E-mail: tetyana.marus@gmail.com, phone number +380322582578

Марусенкова Тетяна Анатоліївна, кандидат технічних наук, доцент, доцент кафедри програмного забезпечення Національного університету «Львівська політехніка». Вул. Ст. Бандери, 12, Львів, Україна. E-mail: tetyana.marus@gmail.com, тел. +380322582578

ORCID ID: 0000-0003-4508-5725