

Project Title: System Verification and Validation Plan for Attitude Check

Adrian Sochaniwsky

February 13, 2024

Revision History

Date	Version	Notes
2024/02/13	1.0	Initial draft

Contents

1	Symbols, Abbreviations, and Acronyms	iv
2	Introduction	1
3	General Information	1
3.1	Summary	1
3.2	Objectives	1
3.3	Relevant Documentation	2
4	Plan	2
4.1	SRS Verification Plan	2
4.2	Design Verification Plan	2
4.3	Verification and Validation Plan Verification Plan	3
4.4	Implementation Verification Plan	3
4.5	Automated Testing and Verification Tools	3
4.6	Software Validation Plan	4
5	System Test Description	4
5.1	Tests for Functional Requirements	4
5.1.1	Inputs and Outputs	4
5.1.2	Calculations	5
5.2	Tests for Nonfunctional Requirements	7
5.2.1	Accuracy	7
5.2.2	Understandability	7
5.2.3	Maintainability	8
5.2.4	Portability	8
5.3	Traceability Between Test Cases and Requirements	9
6	Unit Test Description	9
6.1	Unit Testing Scope	9
6.2	Tests for Functional Requirements	9
6.2.1	Module 1	9
6.3	Tests for Nonfunctional Requirements	10
6.3.1	Module TBD	11
6.4	Traceability Between Test Cases and Modules	11

7	Appendix	13
7.1	Symbolic Parameters	13
7.2	IMU Test Data	13

List of Tables

1	Table of Abbreviations and Acronyms	iv
2	Relation of Test Cases and Requirements.	9
3	Sample of ground truth quaternion.	14
4	Sample of sequential sensor measurements	14

List of Figures

1 Symbols, Abbreviations, and Acronyms

For symbols and units see the Section 1 of the SRS ([Sochaniwsky, 2024](#)).
Table 1 defines the abbreviations and acronyms used in this document.

Table 1: Table of Abbreviations and Acronyms

symbol	description
accel	Accelerometer
gyro	Gyroscope
IMU	Inertial Measurement Unit
mag	Magnetometer
SRS	Software Requirements Specification
VnV	Verification and Validation

2 Introduction

A verification and validation (VnV) plan is a document that describes the objectives, scope, methods, and criteria for verifying and validating a software product. Verification is the process of checking whether the software meets the specified requirements and design specifications. Validation is the process of checking whether the software meets the user's needs and expectations. A VnV plan helps to ensure the quality, reliability, and functionality of the software, as well as to identify and correct any defects or errors before the software is released or deployed.

This document details the plan for verification and validation of Attitude Check. Section 3 provides an overview and objectives of this document. Section 4 discusses the methods of achieving the objectives. Section 5 covers the test descriptions.

3 General Information

3.1 Summary

Attitude Check is an IMU-based attitude estimation algorithm. It consumes sensor data and produces an estimate of the current orientation of the sensor relative to the Earth.

3.2 Objectives

The objectives of this Verification and Validation plan are the following:

- Build confidence in the software correctness.
- Demonstrate the software's ability to accurately estimate orientation.

Objectives that are not included in the scope of this document:

- Demonstration of adequate understandability.
- Verification of external libraries.

It will be assumed that the documentation of Attitude Check is adequate to facilitate adequate understandability for developers wishing to use it. Furthermore, it is assumed that external libraries have been independently verified and validated.

3.3 Relevant Documentation

See the Software Requirements Specification ([Sochaniwsky, 2024](#)) for Attitude Check, it details the goals, requirements, assumptions, and theory of the software.

4 Plan

This section will detail the plan for the verification of the documentation and software for Attitude Check. The primary items that will be verified are: SRS, design, VnV, implementation.

4.1 SRS Verification Plan

The SRS ([Sochaniwsky, 2024](#)) will be verified via feedback from a domain expert and assigned secondary reviewer. The following is a checklist for SRS review derived from ([Wiegers, 2002](#)):

- ☐ Are all internal cross-references to other requirements correct?
- ☐ Are all requirements written at a consistent and appropriate level of detail?
- ☐ Do any requirements conflict with or duplicate other requirements?
- ☐ Is each requirement written in clear, concise, unambiguous language?
- ☐ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ☐ Are all requirements actually requirements, not design or implementation solutions?

4.2 Design Verification Plan

To verify the design, feedback will be collected from domain expert and an assigned secondary reviewer will use the following checklist:

- ☐ Create a high-level diagram that represents the relationships between modules based on the Module Guide (MG) and Module Interface Specification (MIS) [to be cited]. This is intended to find gaps in the design.

4.3 Verification and Validation Plan Verification Plan

This VnV plan will be verified via inspection from a domain expert and a secondary review. The inspection will look for the following items:

- ☐ Confirm all sections contain a verification checklist or description.
- ☐ Inspect system-level test cases for coverage of SRS requirements and goal statements.
- ☐ Inspect all test cases for complete definitions.

4.4 Implementation Verification Plan

A combination of static and dynamic tests will be used to verify Attitude Check. Cppcheck ([cpp](#), 2023) is a static code checking tool that will be employed to analyze the code for undefined behaviour and poor design constructs.

System and unit level tests will be used to dynamically verify Attitude Check. Section 5 outlines the system-level tests. Section 6 details the unit tests.

4.5 Automated Testing and Verification Tools

- Cmake ([cma](#), 2023) is a build system tool that will be used to simplify building and testing.
- GoogleTest is a unit testing framework that support c++ code. It also includes a mocking framework. See <https://github.com/google/googletest>.
- GCOV, LCOV are tools for calculating and displaying unit test coverage metrics.
- Valgrind is a suite of tools for memory and performance profiling. It will be used to check for memory leaks and efficient memory utilization.
- Uncrustify is a tool that applies code formatting rules based on a configuration file. It will be used before every commit to the repo.

- GitHub PR checklists will help reviewers/developers see and check that all rules and tests are completed before code enter a protected branch (i.e. main).
- GitHub CI workflow will automate regression tests and checks that Attitude Check builds are passing before a PR is merged into a protected branch.
- Docker is a mechanism to containerize applications. It will be used to ensure installability and understandability of the project.

4.6 Software Validation Plan

For the scope of this project, software validation will consist of benchmarking against existing algorithms. The Root Mean Squared Error (RMSE) of Attitude Check should lie within ϵ of the RMSE of attitude estimators implemented by [Laidig \(2023\)](#). The data will come from the BROAD dataset ([Laidig et al., 2021](#)).

The derivation of RMSE can be found in the Section ??.

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Inputs and Outputs

This section covers requirements R1 and R4 of the SRS. This includes the input and output requirements for Attitude Check. For input constraints see Section 4.2.8 of the SRS.

T1: Input/Output Test

Control: Automatic

Initial State: Uninitialized

Step	Input	Output
1	Initialize with: $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Quat	Assert “get params” method equals $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Quat
2	Call “update” with: ${}^S\mathbf{a}_t = [0, 0, 9.81]$, ${}^S\boldsymbol{\omega}_t = [0, 0, 0]$ ${}^S\mathbf{m}_t = [16676.8, -3050.9, 49916.9]$	Assert “update” output is a normalized quaternion $\in \mathbb{R}^4$.
3	Initialize with: $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Rot	Assert “get params” method equals $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Rot
4	Call “update” with: ${}^S\mathbf{a}_t = [0, 0, 9.81]$, ${}^S\boldsymbol{\omega}_t = [0, 0, 0]$ ${}^S\mathbf{m}_t = [16676.8, -3050.9, 49916.9]$	Assert “update” output is a matrix $\in \mathbb{R}^{3 \times 3}$.
5	Initialize with: $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Euler	Assert “get params” method equals $\Delta t = 1.0$, ${}^E\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Euler
6	Call “update” with: ${}^S\mathbf{a}_t = [0, 0, 9.81]$, ${}^S\boldsymbol{\omega}_t = [0, 0, 0]$ ${}^S\mathbf{m}_t = [16676.8, -3050.9, 49916.9]$	Assert “update” output is vector $\in \mathbb{R}^3$.

Test Case Derivation: N/A

How test will be performed: At each step, apply the inputs and assert the output.

5.1.2 Calculations

This section covers requirements R2 and R3 of the SRS. This includes the input and output requirements for Attitude Check. For input constraints see Section 4.2.8 of the SRS.

The accuracy metric is described in [Laidig et al. \(2021\)](#), it defines the error between two quaternions (1), the total error (2), and the RMSE (3):

$$\mathbf{q}_{e,t} = {}^S_E \mathbf{q}_{gt,t}^* {}^S_E \mathbf{q}_{est,t} = [q_w, q_x, q_y, q_z]^T \quad (1)$$

$$e_q = 2 \arccos(|q_w|) \quad (2)$$

$$\text{RMSE} = \sqrt{\frac{\sum_{t=0}^n (e)^2}{n}} \quad (3)$$

Often computing separate errors for the heading and inclination components provides better insight into the performance. When magnetometer data is not available, the heading error will be much larger than the other sources of error.

The equation for the heading only error is defined in (4) and the inclination only error is defined by (5):

$$e_h = 2 \arctan\left(\left|\frac{q_z}{q_w}\right|\right) \quad (4)$$

$$e_i = 2 \arccos(\sqrt{q_w^2 + q_z^2}) \quad (5)$$

T2: Orientation with Magnetometer

Control: Automatic

Initial State: Uninitialized

Step	Input	Output
1	Initialize with: $\Delta t = 10.0, \gamma = 0.6$, ${}^E\mathbf{b} = [16676.8, -3050.9, 49916.9]$, outputType = Quat	-
2	Call “update” and loop through measurements in Table 4	Assert output quat is normalized
3	Calculate RMSE for e_q, e_h, e_i vs. ground truth in table 3	Assert average error is within the tolerance.

Test Case Derivation: Benchmark dataset with labelled ground truth orientation for each set of sensor measurements.

How test will be performed: At each step, apply the inputs and assert the output.

T3: Orientation without Magnetometer

Control: Automatic

Initial State: Uninitialized

Step	Input	Output
1	Initialize with: $\Delta t = 10.0, \gamma = 0.6$, ${}^E\mathbf{b} = [16676.8, -3050.9, 49916.9]$, outputType = Quat	-
2	Call “update” with measurements at $t = 0$ from Table 4	Assert output quat is normalized
3	Call “update” and loop through {accel, gyro} measurements in Table 4	Assert output quat is normalized
4	Calculate RMSE of e_q, e_h, e_i vs. ground truth in table 3	Assert average error is within the tolerance.

Test Case Derivation: Benchmark dataset with labelled ground truth orientation for each set of sensor measurements.

How test will be performed: At each step, apply the inputs and assert the output.

Note: Only Pitch and Roll components are evaluated.

5.2 Tests for Nonfunctional Requirements

5.2.1 Accuracy

T4: **test-a1**

Type: Automatic

This test refers to Section 5.1.2.

5.2.2 Understandability

T5: **test-u1**

Type: Manual

Initial State: N/A

Input/Condition: N/A

Output/Result: Successful integration of Attitude Check into a project.

This represents how typical users interact with the software.

How test will be performed: A manual inspection of the code, specifically the header file. Is each input documented/commented? This test passes if this is true.

5.2.3 Maintainability

T6: **test-m1**

Type: Manual

Initial State: N/A

Input/Condition: N/A

Output/Result: Successful addition of a likely change within the threshold defined in NFR3 of the SRS.

How test will be performed: If a likely change needs to be added, measure the development time and compare it to the total development time. If it is less than the threshold specified in the SRS, this test passes.

5.2.4 Portability

T7: **test-p1**

Type: Manual

Initial State: N/A

Input/Condition: One set of initialization inputs, 1 set of measurements

Output/Result: Result of compilation and execution.

How test will be performed: Build the project, initialize and input 1 set of measurements. Test is successful if the build has no errors and if the code produces the same result on Windows 11, macOS, and Ubuntu.

Table 2: Relation of Test Cases and Requirements.

	R1	R2	R3	R4	NFR1	NFR2	NFR3	NFR4
T1	X			X				
T2		X			X			
T3			X		X			
T4					X			
T5						X		
T6							X	
T7								X

5.3 Traceability Between Test Cases and Requirements

6 Unit Test Description

6.1 Unit Testing Scope

This will be updated when the MG and MIS are complete.

6.2 Tests for Functional Requirements

This will be updated when the MG and MIS are complete.

6.2.1 Module 1

UT1: Inputs: Invalid time

Initial State: ...

Input: ...

Output: ...

Test Case Derivation: ...

How test will be performed: ...

UT2: Inputs: Invalid mag ref

Initial State: ...

Input: ...
Output: ...
Test Case Derivation: ...
How test will be performed: ...

UT3: Inputs: Invalid gamma

Initial State: ...
Input: ...
Output: ...
Test Case Derivation: ...
How test will be performed: ...

UT4: Inputs: Invalid output type

Initial State: ...
Input: ...
Output: ...
Test Case Derivation: ...
How test will be performed: ...

UT5: Inputs: Valid

Initial State: ...
Input: ...
Output: ...
Test Case Derivation: ...
How test will be performed: ...

6.3 Tests for Nonfunctional Requirements

This will be updated when the MG and MIS are complete.

6.3.1 Module TBD

UT6: **test-id1**

Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

6.4 Traceability Between Test Cases and Modules

This will be updated when the MG and MIS are complete.

References

- Cmake. <https://cmake.org/>, 2023.
- Cppcheck. <https://cppcheck.sourceforge.io/>, 2023.
- Daniel Laidig. Berlin Robust Orientation Estimation Assessment Dataset (BROAD). <https://github.com/dlaidig/broad/tree/main>, 2023.
- Daniel Laidig, Marco Caruso, Andrea Cereatti, and Thomas Seel. Broad—a benchmark for robust inertial orientation estimation. *Data*, 6(7), 2021. ISSN 2306-5729. doi: 10.3390/data6070072. URL <https://www.mdpi.com/2306-5729/6/7/72>.
- Adrian Sochaniwsky. System requirements specification for attitude check. https://github.com/adrian-soch/attitude_check/blob/7c8cd7387ab120dd1267b56902499314c696e42b/docs/SRS/SRS.pdf, 2024.
- K.E. Wiegers. *Peer Reviews in Software: A Practical Guide*. Addison-Wesley information technology series. Addison-Wesley, 2002. ISBN 9780201734850. URL <https://books.google.com/books?id=d7BQAAAAMAAJ>.

7 Appendix

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Param	Value
ϵ	5 %

7.2 IMU Test Data

Table 3: Sample of ground truth quaternion.

t	q_w	q_x	q_y	q_z
0	0.999926	0.001208	-0.002004	-0.011972
1	0.999923	0.001884	-0.002375	-0.012036
2	0.999922	0.002302	-0.002478	-0.012057
3	0.999925	0.002024	-0.001862	-0.011960
4	0.999927	0.001747	-0.001245	-0.011863
5	0.999927	0.001672	-0.001145	-0.011925
6	0.999926	0.001621	-0.001108	-0.012006
7	0.999926	0.001454	-0.001265	-0.012036
8	0.999926	0.001176	-0.001609	-0.012017
9	0.999926	0.001012	-0.001949	-0.012006

Table 4: Sample of sequential sensor measurements

t	\mathbf{acc}_x	\mathbf{acc}_y	\mathbf{acc}_z	\mathbf{gyro}_x	\mathbf{gyro}_y	\mathbf{gyro}_z	\mathbf{mag}_x	\mathbf{mag}_y	\mathbf{mag}_z
0	0.090941	-0.031273	9.759028	0.005327	-0.000000	-0.004260	0.554144	14.948225	-41.506577
1	0.130181	-0.001843	9.793363	0.005327	0.001065	-0.008522	0.035031	15.327949	-41.656347
2	0.072302	-0.006748	9.864976	0.003196	0.001065	-0.007458	-0.484083	15.707673	-41.806118
3	0.062492	0.002081	9.855166	0.000000	0.002131	-0.005327	0.183349	15.707673	-40.982379
4	0.090941	0.069770	9.754123	0.005327	0.003196	-0.001065	0.850781	15.707673	-40.158640
5	0.077207	0.021701	9.735484	0.003196	0.002131	-0.005327	0.850781	15.707673	-40.158640
6	0.072302	-0.112696	9.812002	0.003196	0.001065	-0.004260	0.850781	15.707673	-40.158640
7	0.115466	0.069770	9.893425	0.004260	0.001065	-0.002131	0.183349	15.555783	-40.308411
8	0.052682	0.021701	9.807097	0.001065	0.004260	-0.003196	-0.484083	15.403894	-40.458182
9	0.029138	0.006986	9.816907	0.004260	0.001065	-0.005327	-0.261606	15.631728	-41.057264