A Fourier Series Library: System Verification and Validation Plan for FSL

Bo Cao

October 23, 2019

1 Revision History

Date	Version	Notes
Oct. 28, 2019	1.0	First draft.
Date 2	1.1	Notes

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

Some symbols, abbreviations and acronyms are defined in the Common Analysis (CA) document [1]. For simplicity and maintainability, they are not re-defined here. Readers shall refer to the CA documents when a certain item is not defined here.

symbol	description
T	Test

[1] This document is available at https://github.com/caobo1994/FourierSeries/blob/master/docs/SRS/CA.pdf.

This document provides an overview of the Verification and Validation (VnV) plan for the Fourier Series Library (FSL). It lays out the purpose, methods, and test cases for the VnV procedure.

3 General Information

3.1 Summary

The library to be tested is called the Fourier Series Library (FSL). This library performs a set of computations, transformations, and/or input/output at the request of the library user.

3.2 Objectives

The intended objective of the VnV procedure is to verify that this library has generally met the requirements described in the CA document. These requirements include the functional requirements (FRs) and the non-functional requirements (NFRs).

Note that if a small part of the NFRs has not been met, the library is still acceptable when the not-met NFRs' impact has been analyzed and deemed non-essential.

3.3 Relevant Documentation

As we said before, this document relies on the CA document. This document is also the base of the Unit Test Plan document.

4 Plan

4.1 Verification and Validation Team

The major member of the team is the author himself. Other contributors might assist in the VnV procedure, but their contributions are not guaranteed.

4.2 CA Verification Plan

The verification of the CA document mainly consists of the feedback from reviewers, and the author's experience in developing and verifying this library.

4.3 Design Verification Plan

The design of this library will be verified by reviewing how the functions in this library relies on each other, and how they are integrated.

4.4 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

The verification of the implementation of this library is mainly done by unit testing. The detail of unit testing can be found in the Unit Test Plan document. Mainly, the unit test will be done by first testing the basic functions in this library, and then testing the advanced functions. Please note that the test result of any function in this library is acceptable, if and only if the reliant functions of this function is tested to be right.

4.5 Software Validation Plan

The transformation part of this library will be validated by comparing its result with the MATLAB pseudo-oracle.

5 System Test Description

5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS]

5.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

5.1.2 Area of Testing2

...

5.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. —SS]

[Tests related to usability could include conducting a usability test and survey. —SS]

5.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

References

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

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

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]