Verification and Validation Plan for PolyHarmonics

Nolan Driessen

August 26, 2015

Contents

1 General Information

The following section provides an overview of the Verification and Validation (V&V) Plan for PolyHarmonics. This section explains the purpose of this document, the scope of the system, common definitions, acronyms and abbreviations that are used in the document, and an overview of the following sections

1.1 Purpose

The main purpose of this document is to describe the verification and validation process that will be used to test PolyHarmonics. This document is intended to be used as a reference for all future testing and will be used to increase confidence in the software implementation.

This document will be used as a starting point for the verification and validation report. The test cases presented within this document will be executed and the output will be analyzed to determine if the software is implemented correctly.

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

symbol	description
QA	Quality assurance
SRS	Software requirements specification
V&V	Verification and validation
V&VP	Verification and validation plan
V&VR	Verification and validation report

1.4 Overview of Document

The following sections provide more detail about the V&V of PolyHarmonics. Information about the testing process is provided, and the software specifications that were discussed in the SRS document are stated. The evaluation process that will be followed during testing is outlined, and test cases for both the system testing and unit testing are provided

2 Plan

This section provides a description of the software that is being tested, the team that will perform the testing, the milestones for the testing phase, and the budget allocated to the testing.

2.1 Software Description

The software being tested is designed for acoustic analysis. Given a LabVIEW file for input, the software will analyze the signals present within the file in terms of its frequency and amplitude.

2.2 Test Team

The team that will execute the test cases, write and review the V&VR consist of:

- Nolan Driessen
- Dr. Spencer Smith
- Alex Halliwushka.

2.3 Milestones

2.3.1 Location

The location that the testing will be performed is Hamilton Ontario. The institution that will be performing the testing is McMaster University.

2.3.2 Dates and Deadlines

Test Case:
The creation of the test cases for both system testing and unit testing is scheduled to begin June 1st 2015. The deadline for the creation of the test cases is June 15th 2015.

Test Case Implementation:
Implementing code for the automation of the unit testing is scheduled to begin June 15th 2015. The implementation period is expected to last approximately two weeks and has a deadline of June 30th 2015.

Verification and Validation Report:
The writing of the V&VR is scheduled to begin July 1st 2015 and end on July 15th 2015.

2.4 Budget

The budget for the testing of this system is being funded by McMaster University and NSERC

3 Software Specification

This section provides the functional requirements, the business tasks that the software is expected to complete, and the non-functional requirements, the qualities that the software is expected to exhibit.

3.1 Functional Requirements

- The product shall receive a LabVIEW TDMS file as input.
- The product shall verify the input contains acceptable information.
- PolyHarmonics shall use the input file to calculate the DFT.
- Using the transformed data, create a plot and text file holding information about the signal.

3.2 Non-functional Requirements

Games are very resource intensive, so performance is a high priority. Other non-functional requirements that are a priority are: correctness, understandability, portability, reliability, and maintainability.

4 Evaluation

This section first presents the methods and constraints that are to be used during the evaluation process. This is followed by how the data obtained by the testing will be evaluated, which includes: how the data will be recorded, how to move from one test to the next, and how to determine if the test was successful.

4.1 Methods and Constraints

4.1.1 Methodology

The testing of PolyHarmonics will be separated into two sections: system testing and unit testing.

The system testing will be done manually, the tester will set up the initial conditions as described in the test cases and compare the actual results of the test to the expected results. If the results match then the test passed, otherwise the test failed.

The unit testing will be automated. The tester will implement the unit tests into the code using a unit testing framework. Once the unit tests are implemented, the software will be run and any incorrect results will be outputted by the system

4.1.2 Extent of Testing

The extent of testing that will be employed is extensive testing. The unit test cases below provide complete code coverage and will increase confidence in the verification of the software. The system test cases increase confidence in the validation of the system

4.1.3 Test Tools

A unit testing framework will be used to implement the unit test cases and run them automatically.

4.1.4 Testing Constraints

There are currently no anticipated limitations on the testing

4.2 Data Evaluation

4.2.1 Data Recording

After each test is run the results of the test should be recorded in the following format:
Test ID:
Input:
Expected Output:
Actual Output:
Result:

4.2.2 Test Progression

For the system test cases: Follow the preparation instructions given for the test case to get the system initialized correctly. Follow the procedures given for the test case and use the inputs provided. Run the test and record the results, record any discrepancies between the actual output and the expected output. Move on to the next test case and repeat the process again.

4.2.3 Testing Criteria

The actual results of each test will be evaluated against the expected results to see if the software is working as intended.

4.2.4 Testing Data Reduction

The results of the test data will be evaluated on a PASS/FAIL basis. If the actual results match the expected results the test will be considered a PASS, otherwise the test is considered a FAIL.

5 System Test Description

5.1 Test 1

5.1.1 Means of Control

Manual

5.1.2 Input

Input files found within the testing/set 1/ Grade 0 folder
Start_Freq = 100
Stop_Freq = 1000
Step_Freq = 100

5.1.3 Expected Output

The set of text files and plots shall pass the following assert statements:
assert filecmp.cmp('output/newPlot','standardTestCaseOutput/oldPlot'), 'Error: Entry i failed the comparison'

5.1.4 Procedure

Determine if the plots and text files produced match the expected output.

5.1.5 Preparation

The user shall enter the directory and frequency values defined in the Input section.

6 Unit Test Description

6.1 Input Finding Module

6.1.1 Module Inputs

Start_Freq
Stop_Freq
Step_Freq
Path to a set of TDMS files

6.1.2 Module Outputs

None

6.1.3 Related Modules

Control Module
Input Data Module

6.1.4 Test Data

6.1.5 Inputs

Start_Freq = 100
Stop_Freq = 1000
Step_Freq = 100

6.1.6 Expected Outputs

Transitions InputData class to fully populate its state.

6.2 Input Data Module

6.2.1 Module Inputs

Start_Freq
Stop_Freq
Step_Freq
TDMS_Data
TDMS_Time

6.2.2 Module Outputs

Start_Freq
Stop_Freq
Step_Freq
TDMS_Data
TDMS_Time

6.2.3 Related Modules

Signal Transform Module
Input Finding Module
Filtering Module
Output Module
Plot Data Module

6.2.4 Test Data

6.2.5 Inputs

Start_Freq = 100
Stop_Freq = 1000
Step_Freq = 100

6.2.6 Expected Outputs

InputData class such that none of the below tests error:
assert Get_Start_Freq() == 100
assert Get_Stop_Freq() == 1000
assert Get_Step_Freq() == 100

6.3 Signal Transform Module

6.3.1 Module Inputs

6.3.2 Module Outputs

6.3.3 Related Modules

6.3.4 Test Data

6.3.5 Inputs

6.3.6 Expected Outputs

6.4 Transformed Signal Data Module

6.4.1 Module Inputs

List

6.4.2 Module Outputs

List

6.4.3 Related Modules

Signal Transform Module
Output Module
Plot Data Module

6.4.4 Test Data

6.4.5 Inputs

List

6.4.6 Expected Outputs

List

6.5 Plot Data Module

6.5.1 Module Inputs

Start_Freq
Stop_Freq
Step_Freq
TDMS_Time

TDMS_Data
Original_Transformed_Data Original_Frequency_Data

6.5.2 Module Outputs

6.5.3 Related Modules

Input Data Module Transformed Signal Data Module Control Module

6.5.4 Test Data

6.5.5 Inputs

6.5.6 Expected Outputs

A set of png files such that for each file the following will return true when compared with the plots from the standard test case:

filecmp.cmp(newplot,testplot,False)

6.6 Output Module

6.6.1 Module Inputs

Start_Freq
Stop_Freq
Step_Freq
TDMS_Data
Filtered_Data

6.6.2 Module Outputs

6.6.3 Related Modules

Input Data Module Filtered Data Module

6.6.4 Test Data

6.6.5 Inputs

Start_Freq = 100
Stop_Freq = 1000
Step_Freq 100
TDMS_Data
Filtered_Data

6.6.6 Expected Outputs

A set of text files such that for each file the following will return true when compared with the text from the standard test case:

filecmp.cmp(newtxt,testtxt)

6.7 Filtering Module

6.7.1 Module Inputs

TDMS_Data

6.7.2 Module Outputs

Transitions FilterData class to fully populate its state.

6.7.3 Related Modules

Input Data Module
Filtered Data Module
Control Module

6.7.4 Test Data

6.7.5 Inputs

6.7.6 Expected Outputs

6.8 Filtered Data Module

6.8.1 Module Inputs

List

6.8.2 Module Outputs

List

6.8.3 Related Modules

Signal Transform Module
Filtering Module
Output Module

6.8.4 Test Data

6.8.5 Inputs

6.8.6 Expected Outputs