SpecSearch: Unit Verification and Validation Plan

Robert E. White

December 8, 2018

1 Revision History

Date	Version	Notes
2018-12-07	1.0	Creation of first draft.

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
R	Requirement
NFR	Non-functional Requirement

Refer to the SRS Symbols, Abbreviations and Acronyms for a more complete list (White (2018c)) https://github.com/whitere123/CAS741_REW.

Contents

List of Tables

List of Figures

This document will briefly summarize the unit tests outlined in the Unit Verification and Validation plan, explain the automated testing set-up, summarize the results of the tests and list the changes made in response to the tests. After reading this document one should be able to determine whether or not SpecSearch satisfied its functional and nonfunctional requirements with respect to the unit tests. They should also be able to trace a unit test to a particular module and that module to a particular requirement.

3 Functional Requirements Evaluation

The three functional requirements for SpecSearch are Rin, Rfind and Rplt. More details on these requirements can be found in the System Requirements Specification, White (2018c). There are nine tests in the Unit Verification and Validation plan (https://github.com/whitere123/CAS741_REW) that cover functional requirements: test-InParams-N, test-InParams-P, test-InParams-k, test-NumParams-Dom, test-NumParams-Dom-Bound, test-NumParams-EllipMat, test-SpecMAT-Tr1, test-Plotting-Inspect and test-Plotting-Inspect-Bound. The first seven tests are automated. These tests inspect the InParams, NumParams and SpecMat modules. Tests with "-Bound" in their names cycle through the boundary analysis input table and verify that an assert statement returns true. The other tests cycle through a table of common inputs within their constraints and check to see that an assert statement returns true. A summary of these tests will be covered in 8.

Test-Plotting-Inspect and test-Plotting-Inspect-Bound are similar to Test-Rplt in the SystVnVReport. The exception is that we are testing the output module and plotting module. We are more concerned with checking that theoretical eigenvalues overlap with the appropriate numerical eigenvalues instead of basic aspects of the plot like symmetry and correct number of eigenvalues. These tests involve a visual analysis of the six spectral plots by my thesis supervisor using the common input and the boundary input. A summary of these tests are presented at the end of this section.

A binary decision will be used to evaluate the tests. Each of the automated tests loop through a list of input and return a pass or fail message for each input. A test will be considered a failure if it returns at least one fail message. In this case the code will have to be modified until non of the tests return a fail message. The evaluation of the plotting tests is dependant on the standards of my supervisor. The code will have to be modified if he does

not consider the plots to be adequate enough for his research. The evaluation of test-Rplt is given in the table below. A list of passing input (inputs that do not throw exceptions) and passing boundary inputs was given to my supervisor. He ran the software with these input tables and gave a check mark if the theoretical eigenvalues overlapped with the appropriate ones. A check mark in the furtherst right column will indicate that my supervisor approved of the plot for that particular input.

Input ID	k	N	P	Result
I1	0.6	100	2	✓
I2	0.1	120	2	✓
I3	0.9	500	2	✓
I4	0.88	550	2	✓
I5	0.99	200	2	✓
I6	0.65	700	2	✓
I7	0.4	100	2	✓
I8	0.8	400	4	✓
I9	0.9	500	4	✓
I10	0.2	700	4	✓
I11	0.3	200	4	✓
I13	0.8	100	4	✓
I14	0.89	150	4	✓
I15	0.69	500	4	✓
I16	0.55	300	2	✓
I17	0.9	400	4	✓

Table 1: Test-Plotting-Inspect

Input ID	k	N	P	Result
I1	0.9	100	2	✓
I2	0.99	120	2	✓
I3	0.9999	500	2	✓
I4	0.99999	550	2	✓
I5	0.999999	2	4	✓
I6	0.01	700	2	✓
I7	0.0001	100	2	✓
I8	0.000001	400	4	✓
I9	0.0000001	500	4	✓

Table 2: test-Plotting-Inspect-Bound

4 Nonfunctional Requirements Evaluation

Planning for nonfunctional tests of units will not be relevant to SpecSearch. For system tests related to Nonfunctional requirements please see the System Verification and Validation Plan at https://github.com/whitere123/CAS741_REW.

5 Comparison to Existing Implementation

My supervisor and I could not find any software that performs the same tasks as SpecSearch.

6 Unit Testing

This section will provide a brief summary of each unit test. It will list the data table used, assert statement and expected value of assert statement. Traces from tests to requirements and tests to modules are presented in 9 and 10.

1. test-InParams-N

Input: Table 1

Assert Statement: $assert(InParams.N, N)$

Output: assert=True

2. test-InParams-P

   Input: Table 1

   Assert Statement: $assert(InParams.P, P)$

   Output: assert=True

3. test-InParams-k

   Input: Table 1

   Assert Statement: $assert(InParams.k, k)$

   Output: assert=True

4. test-NumParams-Dom

   Input: Table 1

   Assert Statement: $assert(NumParams.Dom(1), -NumParams.xend)$ & $assert(NumParams.Dom(end), NumParams.xend)$

   Output: assert=True

5. test-NumParams-Dom-Bound

   Input: Table 2

   Assert Statement: $assert(NumParams.Dom(1), -NumParams.xend)$ & $assert(NumParams.Dom(end), NumParams.xend)$

   Output: assert=True

6. test-NumParams-EllipMat

   Input: Table 1

   Assert Statement: assert$(Numpars.ellipjdn(2), Numpars.EllipMat(2, 2))$

   Output: assert = True

7. test-SpecMAT-Tr1

   Input: Table 1

   Assert Statement: assert($SpecMat.NUM_x, -SpecMat.NUM_x^T$)

   Output: assert=True

Test-Plotting-Inspect and test-Plotting-Inspect-Bound are explained in 3.

7 Changes Due to Testing

All of the tests passed. No changes were made to the software package.

8 Automated Testing

The tests in 6 were automated. Data tables were organized into MATLAB matrices, with input as rows. A for loop was constructed to cycle through the rows to be used as arguments in SpecSearch. The aforementioned assert statements were called in each iteration of the loop. The results of these automated tests are summarized in the following tables. Details about the intial state, Input, Output, expected result and result of running the test are given in the following tables. A check mark indicates that the test returned the expected result. A link to a description of the unit test is provided in the table caption.

Input ID	k	N	P	Expected Result	Test Result
I1	1	100	2	True	✓
I2	0	100	4	True	✓
I3	-10	50	2	True	✓
I4	-55	200	4	True	✓
I5	2	150	2	True	✓
I6	1.0001	100	2	True	✓
I7	-0.001	200	4	True	✓
I8	1.01	200	2	True	✓
I9	-0.01	100	4	True	✓
I10	0.7	-10	2	True	✓
I11	0.9	20	4	True	✓
I12	0.9	5	2	True	✓
I13	0.7	0	4	True	✓
I14	0.5	-1	2	True	✓
I15	0.9	100	0	True	✓
I16	0.9	500	-2	True	✓
I17	0.7	150	-4	True	✓
I18	0.5	200	100	True	✓
I19	0.7	0.5	100	True	✓

Table 3: test-InParams-N 1, test-InParams-P 2, test-InParams-k 3, test-NumParams-Dom 4

Input ID	k	N	P	Expected Result	Result
I1	0.9	100	2	True	✓
I2	0.99	120	2	True	✓
I3	0.9999	500	2	True	✓
I4	0.99999	550	2	True	✓
I5	0.999999	2	4	True	✓
I6	0.01	700	2	True	✓
I7	0.0001	100	2	True	✓
I8	0.000001	400	4	True	✓
I9	0.0000001	500	4	True	✓

Table 4: test-NumParams-Dom-Bound 5

Input ID	k	N	P	Expected Result	Test Result
I1	1	100	2	True	✓
I2	0	100	4	True	✓
I3	-10	50	2	True	✓
I4	-55	200	4	True	✓
I5	2	150	2	True	✓
I6	1.0001	100	2	True	✓
I7	-0.001	200	4	True	✓
I8	1.01	200	2	True	✓
I9	-0.01	100	4	True	✓
I10	0.7	-10	2	True	✓
I11	0.9	20	4	True	✓
I12	0.9	5	2	True	✓
I13	0.7	0	4	True	✓
I14	0.5	-1	2	True	✓
I15	0.9	100	0	True	✓
I16	0.9	500	-2	True	✓
I17	0.7	150	-4	True	✓
I18	0.5	200	100	True	✓
I19	0.7	0.5	100	True	✓

Table 5: test-NumParams-EllipMat 6

Input ID	k	N	P	Expected Result	Test Result
I1	1	100	2	True	✓
I2	0	100	4	True	✓
I3	-10	50	2	True	✓
I4	-55	200	4	True	✓
I5	2	150	2	True	✓
I6	1.0001	100	2	True	✓
I7	-0.001	200	4	True	✓
I8	1.01	200	2	True	✓
I9	-0.01	100	4	True	✓
I10	0.7	-10	2	True	✓
I11	0.9	20	4	True	✓
I12	0.9	5	2	True	✓
I13	0.7	0	4	True	✓
I14	0.5	-1	2	True	✓
I15	0.9	100	0	True	✓
I16	0.9	500	-2	True	✓
I17	0.7	150	-4	True	✓
I18	0.5	200	100	True	✓
I19	0.7	0.5	100	True	✓

Table 6: test-SpecMAT-Tr1 7

9 Trace to Requirements

This section shows the traceability between the modules and the test cases.

Tests	Rin	Rfind	Rplt
test-InParams-N	X		
test-InParams-P	X		
test-InParams-k	X		
test-NumParams-Dom		X	
test-NumParams-Dom-Bound		X	
test-NumParams-EllipMat		X	
test-SpecMAT-Tr1		X	
test-Plotting-Inspect			X
test-Plotting-Inspect-Bound			X

Table 7: Traceability Between Test

10 Trace to Modules

Test Case	Modules
test-InParams-N	M1, M2, M13
test-InParams-P	M1, M2, M13
test-InParams-k	M1, M2, M13
test-NumParams-Dom	M1, M6, M13
test-NumParams-Dom-Bound	M1, M6, M13
test-NumParams-EllipMat	M1,M4, M13
test-SpecMAT-Tr1	M1,M4, M13
test-Plotting-Inspect	M1,M11,M3
test-Plotting-Inspect-Bound	M1,M11,M3

Table 8: Trace Between Test Cases and Modules

11 Code Coverage Metrics

No code covering metrics were used in the creation of SpecSearch.

References

Robert White. *Module Guide*. 2018a.

Robert White. *Module Interface Specification*. 2018b.

Robert White. *System Requirements Specification*. 2018c.