SpecSearch: System Verification and Validation Plan

Robert White

December 2, 2018

1 Revision History

Date	Version	Notes
2018-10-13	1.0	Creation of first draft for VnV plan presentation.
2018-10-20	1.1	Edit of 1.0. I considered all of the feedback from my presentation and made necessary changes. I also prepared the document for submission.
2018-10-22	1.2	Edit of 1.1. Removed comments, updated test cases, added references, improved writing and added test-NFR2c.
2018-10-23	1.3	Edit of 1.2. Minor corrections and revisions.
2018-11-28	1.4	Edit of 1.3. First wave of corrections for final documenation.

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 (2018)) https://github.com/whitere123/CAS741_REW.

Contents

List of Tables

This document discusses the verification and validation requirements for SpecSearch. The Project Management Body of Knowledge (PMBOK) guide provides unambigious definitions for verification and validation (Institute (2017)). PMBOK defines verification as "the evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process."

PMBOK defines validation as "the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers" (Institute (2017)). After reading this document one should be able to create and run test cases to verify and validate SpecSearch.

3 General Information

3.1 Summary

The software that is going to be tested is called SpecSearch. SpecSearch will search for the spectrum (set of eigenvalues) of a particular lax equation from a lax pair that is compatible with solutions to the Non-Linear Schrödinger (NLS) Equation. SpecSearch will also use the spectral information to determine the stability of the solutions. Refer to the instance models in the (White (2018)) for more details.

3.2 Objectives

The qualities that are most important for SpecSearch are highlighted in this section. Many of these qualities are also presented in the (White (2018)).

My supervisor, Dr. Dmitry Pelinovsky, and me intend to use this program as a tool to search for the continuous spectrum of the previously mentioned spectral problem. SpecSearch will only provide the approximate location of the spectrum and a finite number of elements in the spectrum. We will use this output as a guide in analytically solving for the entire continuous spectrum. Therefore, the code should be reliable and accurate within my supervisor's standards.

There are various numerical methods that can be used to approximate the derivatives of the eigenfunctions [Graselli and Pelinovksy, 2007]. My supervisor and me intend to experiment with these different methods in an attempt to create a more accurate picture of the spectrum. Therefore, it is important that SpecSearch is maintainable and manageable.

SpecSearch will also be used by a team of researchers studying rogue waves. These users may not have a strong software developing background. Therefore, it is important that the code is easy to use and has a simple user interface.

The objectives are summarized in the following points for easy reference. The objectives are to:

- Build confidence in software correctness.
- Ensure maintainability and manageability
- Satisfy the requirements of my thesis supervisor and those outlined in the SRS (White (2018)).
- Verify usability.

3.3 References

The following reference will be used to build confidence in software correctness. This research papers explores the same spectrum that my supervisor and I are studying. They were successful at analytically finding the set of eigenvalues for particular boundary conditions. These eigenvalues will be plotted alongside our numerical spectrum. These theoretical results define the standard of our software correctness. Our confidence in software correctness will be higher if our spectrum overlaps with these theorectical results.

- Bernard Deconinck and Benjamin L.Segal (Deconinck and L.Segal) The stability spectrum for elliptic solutions to the focusing NLS equation. PhysicaD, 2017.

The following reference will be frequently mentioned in this document. The test cases outlined in this document are meant to test the necessary qualities and requirements outlined in the SRS (White (2018)).

- Robert White. System Requirements Specification for SpecSearch (White (2018)). Github, 2018.

4 Plan

4.1 Verification and Validation Team

The verification and validation team consists of my thesis supervisor, Dr. Dmitry Pelinovsky and me, Robert White.

4.2 SRS Verification Plan

The SRS verification plan consists of feedback from Dr. Dmitry Pelinovsky (supervisor), Dr. Spencer Smith and CAS 741 classmates. My supervisor will provide feedback regarding mathematical theory, model assumptions, constraints and research goals. Feedback from classmates and Dr. Smith will criticize the document outline, readability and requirements. In particular, the contents of this document will be reviewed by Jennifer Garner https://github.com/PeaWagon/Kaplan.

4.3 Design Verification Plan

The design verification plan will simply involve inspection of the software by my thesis supervisor, Dr. Smith and CAS 741 classmates.

4.4 Implementation Verification Plan

The implementation verification plan consists of three parts. The first part is a software verification checklist. The checklist will be completed by researchers, Dr. Pelinovsky and me. The checklist verifies that basic software features have been implemented successfully. For example, it checks if the user was able to fulfill their responsibilities outlined in the SRS (White (2018)). This checklist can be found in the 6.3.

The second part involves running the software tests outlined in sections 5 and 6. Unit testing will also be performed.

The third part is summarized as follows:

- Code inspection : I will go through the code to see if each part is correct with respect to the mathematical theory. In particular I will ensure that:
  - Variables are being used in the right context.
  - Functions from other packages are being used in the right context. Some packages have different standards for constants. For instance, one of the complete elliptic integral functions in MATLAB does not square the inputted variable. Theoretical convention dictates that this constant should be squared.
  - The dimensions of the vectors and matrices are appropriate. For example, multiplication of row with column versus column with row.
  - Equations are translated correctly into MATLAB syntax.
- Supervisor Inspection: My supervisor and me will go through the code together to ensure that:
  - I correctly implemented the mathematical theory and numerical algorithms.
  - I made the code manageable and maintainable for future use.
- Linter: A linter is a program that checks the source code for programmatic and stylistic errors

4.5 Software Validation Plan

The first reference in 3.3 will be used to validate SpecSearch. The boundary value inputs from this will be checked against the output of SpecSearch. This test is outlined in test-find of 5.

5 System Test Description

5.1 Tests for Functional Requirements

1. test-Rin1NonNumeric

   Initial State: -

   Input: Table 2 in 6.2.

   Output: Exception (Non-Numeric)

   How test will be performed: Combinations of inputing non-numerical values for different inputs (such as strings) will be considered. A non-numerical value is denoted by 'X' in 2. These tests should catch an exception. 'X' can only be a string, character or vector of dimension greater than 2. Failure to type in enough information or writing undefined symbols is covered by MATLAB's built in exception handler.

2. test-Rin1kBounds

   Initial State: -

   Input: I1 to I9 of Table 3 in 6.2.

   Output: Exception (k out of constraint).

   How test will be performed: Variations of k being out of its respective bound will be considered in each test. I will test negative k values, values larger than 1 and the endpoints of the constraint set. I will also test k values approaching the endpoints. The variable k should be contained in $(0, 1)$ (see SRS for more details (White (2018))). A successful test will involve catching an exception. This test is ensuring that Rin is satisfied (White (2018)).

3. test-Rin1NBounds

   Initial State: -

   Input: I10 to I14 of Table 3 in 6.2.

Output: Exception (N out of constraint).

How test will be performed: Variations of N being out of its respective bound will be considered in each test. I will test fractions and negative numbers. The variable N should be a natural number. A successful test will involve catching an exception. This test is ensuring that Rin is satisfied (White (2018)).

4. test-Rin1PBounds

   Initial State: -

   Input: I15 to I18 of Table 3 in 6.2.

   Output: Exception (P out of constraint).

   How test will be performed: Variations of P being out of its respective bound will be considered in each test. I will test 0, unallowable positive numbers, and negative numbers. The variable P can only be 2 or 4. A successful test will involve catching an exception. This test is ensuring that Rin is satisfied (White (2018)).

5. test-Rfind-Rin
   Initial State: -

   Input: Table 4 in 6.2

   Output: $4N$

   How the test will be performed: $4N$ eigenvalues should be returned. Having this many eigenvalues is evidence that Rfind is being satisfied (White (2018)). This test is also checking whether or not the code ran smoothly when given inputs within their constraints. Eigenvalues are only calculated when the other modules do not throw exceptions. Therefore, the functional requirement Rin is also being validated (White (2018)).

6. test-Rplt

   Initial State: -

Input: Table 4 in 6.2 .

Output: A plot of the spectrum.

How test will be performed: Numerical parameters within their respective constraints will be inputted into SpecSearch. The program should run smoothly and generate six plots; a cn and dn spectrum for each of the three numerical algorithms. This test will involve visual inspection by my supervisor to see if the spectrum plots are adequate. This test is ensuring that the functional requirement Rplt is satisfied.

5.2 Tests for Nonfunctional Requirements

1. test-NFR1

   Type: Static

   Initial State: -

   Input/Condition: SpecSearch MATLAB code

   How test will be performed: The software will be manually read by Dr. Pelinovsky and I to see if there is a more effective code structure to allow implementation of new numerical algorithms. The purpose of this test is to ensure that the code is maintainable and manageable. This is related NFR1 in the SRS (White (2018)).

2. test-NFR2

   Type: Manual

   Initial State: -

   Input: Table 4 in 6.2 .

   Output: Six Spectrum plots.

   How test will be performed: The numerically calculated eigenvalues from SpecSearchwill be expressed as unfilled circles on a complex plane. Four theoretical eigenvalues from (segal et al) will be plotted their appropriate figure. These theoretical eigenvalues will be denoted by an asterix. My supervisor will inspect these plots to see if the SpecSearchis

accurate. This is related to NFR2 in the SRS (White (2018)). The theoretical eigenvalues should lie within the circles.

3. test-UserPerformance

   Type: Usability Survey (see 6.1)

   How test will be performed: The survey will be administered to the future users of SpecSearch. Feedback from the survey will be considered when updating SpecSearch. The survey is in 6.1. This test is also related to NFR2.

5.3 Traceability Between Test Cases and Requirements

	Rin	Rfind	Rplt	NFR1	NFR2
test-Rin1NonNumeric	X				
test-Rin1kBounds	X				
test-Rin1NBounds	X				
test-Rin1PBounds	X				
test-RFind-Rin	X	X			
test-Rplt			X		
test-NFR1				X	
test-NFR2					X
test-UserPerformance					X

Table 1: Traceability Between Test

References

Bernard Deconinck and Benjamin L.Segal. The stability spectrum for elliptic solutions to the focusing nls equation. *PhysicaD*.

Project Management Institute. The project management body of knowledge guide. 2017.

Robert White. *System Requirements Specification*. 2018.

6 Appendix

6.1 Usability Survey Questions

- How long did it take before you could run the software? How many attempts at running SpecSeach did it take before you understood how to properly use it and interpret the output?
- Was this program useful for your research and were you able to interpret the results?
- What aspects of this software do you feel need improvement?
- How does this program compare with other software that finds this particular spectrum?
- Was it clear how and where to input the variables?
- Were the plots and stability results clear?

6.2 Data Input Tables

The following data tables list the inputs and expected outputs for the functional requirements test 5.1. Table 2 consists of inputs with non-numerical values. A non-numerical value is either a string, character or 2+ dimensional vector. Table 3 lists inputs with one of the values being out of their respective constraint set. Table 4 is a list of inputs for which no exception is thrown. Each of these test cases should produce six plots and $4N$ eigenvalues.

Input ID	k	N	P	Result
I1	X	X	X	Error
I2	X	30	2	Error
I3	X	X	2	Error
I4	X	100	X	Error
I5	0.5	X	2	Error
I6	0.8	X	X	Error
I7	X	X	4	Error
I8	X	1000	X	Error
I9	0.99	X	X	Error
I10	0.6	250	X	Error

Table 2: Combinations of non-numerical input

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

Table 3: One of the inputs is out of their respective bound

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

Table 4: All inputs within constraints

6.3 Software Verification Checklist

- Did any of the inputs you entered provide suprising results? If yes, what were they?
- Were you able to identify which numerical algorithm and wave solution the plot represented?
- Were all of the plots legible?
- Was the output useful for your research?
- Was it clear how to input the variables?