Project Title: System Verification and Validation Plan

Robert White

October 21, 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.

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
R	Requirement
NFR	Non-functional Requirement

[symbols, abbreviations or acronyms – you can simply reference the SRS tables, if appropriate —SS]

Contents

List of Tables

List of Figures

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. It 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". It 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." (PMBOK) 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 being tested is called SpecSearch. SpeacSearch will search for the spectrum of a particular lax equation from a lax pair that is compatbilible with solutions to the Non-Linear Schrodinger (NLS) Equation. It will also use the spectral information to determine the stability of the solutions. See the instance models in the SRS 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 SRS.

My supervisor and I intend to use this program as a tool to search for the continuous spectrum of the previously mentioned specral 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. An unreliable and inaccurate output will provide a misleading direction for the next stage of our research.

There are various numerical methods that can be used to locate the spectrum. My supervisor and I intend to experiment with these different methods in an attempt to create a more accurate picture of the spectrum. Therefore, it is important that SpecSeach is maintainabilible and manageable. This will allow for easy implementation of new algorithms.

SpecSearch will also be used by a team of researchers studying modulated wave packets and 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.
- Verify effectiveness and ease of use (usability).

3.3 References

- Bernard Deconinck and Benjamin L.Segal. The stability spectrum for elliptic solutions to the focusing NLS equation. PhysicaD, 2017.
- Robert White. System Requirements Specification for SpecSearch. Github, 2018.

4 Plan

4.1 Verification and Validation Team

The verification and validation team consists of my theis supervisor, Dr. Dmitry Pelinovsky, and I.

4.2 SRS Verification Plan

The SRS verification plan consists of feedback from my thesis supervisor, Dr. 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.

4.3 Design Verification Plan

The design verification plan will simply involve inspection of the software by my thesis supervisor.

4.4 Implementation Verification Plan

The implementation and verification plan consists of two parts. The first part is a software verification checklist. The checklist will be completed by researchers who use SpecSearch for their research, my supervisor and I. The checklist sees if basic software features have been implemented successfully, such as allowing the user to fufill their responsibilities or if the software fufilled its basic responsibilities (ie plotting something that resembles a spectrum). The checklist can be found in the appendix. The second part involves running software tests outlined in sections 5 and 6. Unit testing will also be performed.

4.5 Software Validation Plan

This section does not apply to SpecSearch.

5 System Test Description

5.1 Tests for Functional Requirements

1. test-Rin1

   Initial State: -

Input: See "data input tables" in the appendix for more detail.

Output: Error or pass message.

How test will be performed: Combinations of inputing non-numerical values as input (such as letters), or numerical values outside of their respective constraints, will be considered. A successful test in these instances will be an error message.
I will also test cases with each variable in the input having an acceptable numerical value. A successful test in this case will be a pass message.

2. test-Rfind1

   Initial State: -

   Input: Similar to test-Rin1

   Output: Binary output (Full or non-full array)

   How the test will be performed: A test is successful if the amount of eigenvalues is equal to $2n$ and if each element of the eigenvalue array is non-empty. Unsuccessful cases from test-Rin1 should produce no eigenvalue array or an empty eigenvalue array. We are not testing for accuracy in this test; see nonfunctional tests for more detail.

3. test-Rcon1

   Initial State: -

   Input: Similar to test-Rin1

   Output: Binary Variable (Connected or disconnected)

   How test will be performed: This test will check to see if there is a sufficient amount of points between the tagged portions of the spectrum. Tagged portions are the explicitly calculated eigenvalues from the previous test. The spectrum array should have approximated values between these "tagged portions". Failed test cases from test-Rin1 should have no data for this section. We are not testing for accuracy in this test.

4. test-Rplt

   Initial State: -

   Input: Similar to test-Rin1

Output: Pass of fail message (Is there a graph or not)

How test will be performed: This test will simply check to see if the spectrum is plotted and if the picture agrees with the numbers in the spectrum array. These should be no plot for the failed test cases of test-Rin1.

5. test-Rstl

   Initial State: -

   Input: Similar to test-Rin1

   Output: Binary Variable (Verification of stability)

   How test will be performed: The stability results will be compared with the stability analysis in (ref1).

5.2 Tests for Nonfunctional Requirements

1. test-NFR1

   Type: Static

   Initial State: -

   Input/Condition: SpecSearch MATLAB code

   Output/Result: Pass or Fail

   How test will be performed: The software will be manually read by the developer and his supervisor to see if there is a more effective code structure to allow implementation of new numerical algorithms.

2. test-NFR2a

   Type: Manual

   Initial State: Software system with prescribed input.

   Input: Matrix derived from the time indepent lax equation.

   Output: Pass or fail (See "Matrix Structure" in appendix).

   How test will be performed: The matrix output ought to have a specific form for each instance of SpecSearch. This test will loop through each element of the matrix to ensure it has the correct form (ie two of the quadrants are diagonal matrices and the other have a certain 'diagonal pattern')

3. test-NFR2b

   Type: Manual

   Initial State: Software system with prescribed input.

   Input: See data input tables in the appendix for more detail.

   Output: Pass or fail.

   How test will be performed: The output will be tested against the boundary value eigen-values derived analytically in Deconinck and Segal. The standard of necessary accuracy will be determined by my supervisor.

5.3 Traceability Between Test Cases and Requirements

	Rin	Rfind	Rcon	Rplt	Rstl	NFR1	NFR2
test-Rin	X						
test-Rfind		X					
test-Rcon			X				
test-Rplt				X			
test-Rstl					X		
test-NFR1						X	
test-NFR2a							X
test-NFR2b							X

Table 1: Traceability Between Test

6 Static Verification Techniques

- Code inspection : I will go through the code to see if each step is correct with respect to the mathematical theory. In particular I will ensure that:
  - variables are being used in the right context.
  - any discretization of functions is performed accurately about the origin. For example, equal step sizes in either directions.
  - functions from other packages are being used in the right context. For example, some packages have different standards for constants. Theoretical convention may square a constant while the code may take the squared value directly as the constant.
  - the dimensions of the vectors and matrices are appropriate. For example, multiplication of row with column versus column with row. Or that rows in matrices are seperated with semi-colons.
  - ensure a varible is not accidently overwritten or cleared.
- Code walkthrough: My supervisor and I 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.

References

[1] Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide)–Sixth Edition. Project Management Institute, 2017.

[2] Caswell and Johnston. Design Verification. University of Calary. 2017.

[3] Bernard Deconinck and Benjamin L.Segal. The stability spectrum for elliptic solutions to the focusing NLS equation. PhysicaD, 2017.

[4] Robert White. System Requirements Specification for SpecSearch. Github, 2018.

7 Appendix

This is where you can place additional information.

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.

7.2 Usability Survey Questions

- How long did it take to learn how to run the software?
- Was it easy to interpret the output?
- Was this program useful for your research?
- What aspects of this software do you feel need improvement?
- Was the output recieved and processed in an adequate amount of time?
- How does this program compare with other software that finds this particular spectrum?

7.3 Data Input Tables

The following data table is for test-Rin1. An "X" indicates no-input or incorrect input. Incorrect input is anything that is non-numerical (ie letters or symbols).

Input ID	a	b	k	n	Result
I1	1	1	0.9	100	Runs
I2	X	1	0.9	100	Error
I3	1	X	0.9	100	Error
I4	1	1	X	100	Error
I5	1	1	0.9	X	Error
I6	1	1	1	1.2	Error
I7	1	1	1.2	100	Error
I8	1	-1	0.9	100	Error
I9	-1	2	0.9	100	Runs
I10	1	2	0.99	100	Runs
I11	1	3	0.99	500	Runs
I12	1	1	0	200	Runs

Table 2: Prescribed input for testing Rin

The following table outlines the boundary value test cases that can be verified from previous literature. A 'V' means that the parameter is free to vary.

Boundary ID	a	b	k
B1	$\sqrt{b}$	a^2	V
B2	$\sqrt{\frac{b}{k^2}}$	a^2k^2	V
B3	V	V	0

Table 3: Prescribed input for Rin

For these cases all of the eigenvalues, except for four of them, will be purely imaginary. The remaing four have the form: $\lambda = {}^{+}_{-}\frac{1}{2}a(1+\sqrt{1-k^2})$

and $\lambda = \pm\frac{1}{2}a(1-\sqrt{1-k^2})$ for B1.
As for B2 the eigenvalues have the form: $\lambda = \frac{1}{2}a(\pm k + i\sqrt{1-k^2})$ and $\lambda = \frac{1}{2}a(\pm k - i\sqrt{1-k^2})$. (source)
The plan is to run B1 with a=b=1 and k=0.999. B2 will be run with the same input. The expected eigenvalues were previously mentioned.

7.4 Software Verification Checklist

- Does the software allow the user to fufill their responsibilities (as outlined in the SRS)?
- Does the software fufill its intended responsibilities (see SRS)?
- Was the intended user able to use the software and interpret the output for their research purposes?
- Are any of the system constraints violated (see SRS)?

7.5 Correct Matrix Form

test-NFR2a will ensure that the matrix for which the eigen values are calculated, E, is of the correct form. For any given input n, the matrix is 4n by 4n. It is of the form $[A1A2; A2-A1]$. Where $A1A2$ is the first row and $A2-A1$ is the second row. A1 and A2 are matrices. A2 is a diagonal 2n by 2n matrix and A1 is 2n by 2n and defined as follows: A1(i,j) = n/2 for (1,2),(2,3),... (2n-1,2n) and A1(i,j)=-n/2 for (2,1),(3,2) ... (2n,2n-1). A1(1,2n)=-n/2 and A1(2n,1)=n/2.