Module Interface Specification for SpecSearch

Robert E. White

December 9, 2018

1 Revision History

Date	Version	Notes
2018-11-09	1.0	Created the first draft for my MIS presentation.
2018-11-15	1.1	Updated the presentation draft based on feedback from CAS 741 class. Completed half of the module interface specifications.
2018-11-20	1.2	Completion of the module interface specifications. Completion of section 2.
2018-11-22	1.3	Edit of 1.2. First submission.
2018-12-09	1.4	Creation of final draft for final submission.

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/whitere123/CAS741_REW.

Abbreviations and Acronyms

symbol	description
SRS	System Requirements Specification
:	"Contained in"
$\neg$	Negation
$\Rightarrow$	"It follows"
cn	Elliptic cosine function
sn	Elliptic sine function
dn	Elliptic delta function
NLS	Non-linear Schrödinger
$O(8)$	Eigth order central numerical differentiation method
$O(10)$	Tenth order central numerical differentiation method
$O(12)$	Twelfth numerical differentiation method
dim(in)	The dimension of item "in"
$\notin$	Not an element of

Contents

3 Introduction

The following document details the Module Interface Specifications for SpecSearch. It will describe the intended behaviour of the access routines in SpecSearch's modules. Mathematical notation will be used to explain the external behaviour of the modules and the relationships between input and output. The MIS is still abstract since it does not outline any code.

SpecSearch is scientific computing software that computes and plots the spectrum of a matrix operator from a LAX pair that is compatible for solutions of the Non-Linear Schrödinger (NLS) equation. This spectrum carries useful information regarding the stability of solutions to the NLS equation. Physicists are interested in this spectrum because the NLS equation models physical phenomena; such as rogue waves or modulated wave packets. Mathematicians are interested in the analytical behaviour of this spectrum.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/whitere123/CAS741_REW.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SpecSearch.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
complex	$\mathbb{C}$	any number $x+iy$ with $x \in \mathbb{R}$, $y \in \mathbb{R}$ and $i^2 = -1$.
real matrix	$\mathbb{R}^{mxn}$	An m by n Matrix with real elements.
complex matrix	$\mathbb{C}^{mxn}$	An m by n Matrix with complex elements.

The specification of SpecSearch uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SpecSearch uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Parameters Output Format Spectrum Matrix Exact Eigenvalue Equations Numerical Parameters Control
Software Decision	Eigenvalue and Eigenvector Solver Diagonal Matrix Generator Elliptic Integral Elliptic Functions Plotting Linspace Toeplitz

Table 1: Module Hierarchy

6 MIS of Input Parameters

The secrets of this module are the methods for verifying input and the data structure for storing the inputed parameters.

6.1 Module

InParams

6.2 Uses

-

6.3 Syntax

Name	In	Out	Exceptions 19
Load_params	$k : \mathbb{R}$ $N : \mathbb{N}$ $P : \{2, 4\}$	-	NonNumericalError
Verify_params	-	-	BadkRange, BadNRange, BadPRange
k	-	$\mathbb{R}$	-
N	-	$\mathbb{N}$	-

6.4 Semantics

6.4.1 State Variables

$k : \mathbb{R}$
$N : \mathbb{N}$
$P : \{2, 4\}$

6.4.2 Environment Variables

InputParameters: The three values entered by the user (k, N, P).

6.4.3 Assumptions

- Load_params is called before the values of any state variables are accessed.
- The user inputs the environment variables in the following order: (k, N, P).

6.4.4 Access Routine Semantics

InParams.k():

- output: $out = k$
- exception: None

InParams.N():

- output: $out = N$
- exception: None

InParams.P():

- output: $out = P$
- exception: None

Load_Params():

- transition: The data is read as a vector from the command line. The three elements of the vector correspond to (k, N, P) , respectively. This data is used to populate the previously mentioned state variables.
- exception: exec:=NonNumericalError if non-numerical data is entered.

Verify_Params():

- exception: exc:=

$\neg(k : [0, 1]) \Rightarrow$ BadkRange
$\neg(N : \mathbb{N})$ and $\neg(N > 20) \Rightarrow$ BadNRange
$\neg(P : \{2, 4\}) \Rightarrow$ BadPRange

6.4.5 Local Functions

None

7 MIS of Output Format

The secret of this module is the structure used for storing the output data.

7.1 Module

OutForm

7.2 Uses

Eigenvalue and Eigenvector solver 11, Numerical Parameters 10 and Exact Eigenvalue Equations 9.

7.3 Syntax

Name	In	Out	Exceptions
capture	$\lambda_o 1 : \mathbb{C}$	out	-
	$\lambda_o 2 : \mathbb{C}$		-
	$\lambda_o 3 : \mathbb{C}$		-
	$\lambda_o 4 : \mathbb{C}$		-
	$\Lambda : \mathbb{R}^{4nx6}$		-
label	-	char	-
theor	-	$\mathbb{R}^{2x6}$	-
Λ	-	$\mathbb{R}^{4nx6}$	-

7.4 Semantics

7.4.1 State Variables

$label : char$
$theor : \mathbb{R}^{2x6}$
$\Lambda : \mathbb{R}^{4nx6}$

7.4.2 Environment Variables

None.

7.4.3 Assumptions

- Eigenvalue and Eigenvector solver, Numerical Parameters and exact eigenvalue equations are called and their outputs are fed into Outform.capture() before any of the state variables are accessed.

7.4.4 Access Routine Semantics

OutForm.$label$():

- output: $out = label$
- exception: None

OutForm.$theor$():

- output: $out = theor$
- exception: None

OutForm.Λ():

- output: $out = \Lambda$
- exception: None

capture():

- Transition: The eigenvalues from the six spectral matrices, Λ , and theoretical eigenvalues, $\lambda_o i$,are put into an OutForm data structure. Each state variable is a six element cell structure. The ith cell of each state variable corresponds to one of the six spectral matrices (ie for cn order 10). OutForm.label$\{i\}$ is the i^{th} spectral matrices' plot title. OutForm.theor$\{i\}$ are the i^{th} spectral matrices' theoretically calculated eigenvalues. OutForm.$\Lambda\{i\}$ is the i^{th} spectral matrices' spectrum.This structure is convienient for creating all six plots in a for loop.
- Exception: None

7.4.5 Local Functions

None

8 MIS of Spectrum Matrix

The structure of the spectrum matrix, its data entries, how it is created, and the numerical method for approximating its eigenfunctions are the secrets of this module.

8.1 Module

SpecMat

8.2 Uses

This module uses Numerical Parameters 10.

8.3 Syntax

Name	In	Out	Exceptions
SpecMat	$n : \mathbb{N}$	$SpecMatdn_{O(8)}$	-
	$h : \mathbb{R}$	$SpecMatdn_{O(10)}$	-
	$elipMAT_{cn} : \mathbb{R}^{2nx2n}$	$SpecMatdn_{O(12)}$	-
	$elipMAT_{dn} : \mathbb{R}^{2nx2n}$	$SpecMatcn_{O(8)}$	-
	$k : \mathbb{R}$	$SpecMatcn_{O(10)}$	-
		$SpecMatcn_{O(12)}$	-
$SpecMatdn_{O(8)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMatdn_{O(10)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMatdn_{O(12)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMatcn_{O(8)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMatcn_{O(10)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMatcn_{O(12)}$	-	$\mathbb{R}^{4nx4n}$	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None.

8.4.3 Assumptions

- Numerical parameters is called, its state variables are validated and its state variables are fed into SpecMat() before the SpecMat function is called.

8.4.4 Access Routine Semantics

SpecMat():

- transition: The data (n,h,elipMAT,k) will be captured from numerical parameters and used to create the spectral matrices. There will be six spectral matrices. Each matrix corresponds to one of three numerical algorithms for approximating the eigenfunction derivatives and one of two boundary wave solutions to the NLS equation. The top right and bottom left quadrants of the spectral matrices are diagonal matrices with diagonal elements equal to the negative elipdn (or cn) function (14) evaluated at points in the discretized domain 10. The remaining quadrants of the matrix are equal to $-NUM_j$ or $+NUM_j$. NUM_j is an extremely large matrix and its explicit form will be omitted from this document. It is a matrix of coefficients that when multiplied by an eigenvector procude a matrix whose entries are the j^{th} order derivative's central difference approximation. For more details about NUM_j see Grasseli and Pelinovsky (2007). For the derivation of the spectral matrix please see the "justification of Compatibility conditions and reduction to spectral problem" under the Instance Model 1 (IM1) chart of the SRS document https://github.com/whitere123/CAS741_REW .
  Therefore the output matrices are:

$$SpecMatdn_{O(8)} = \begin{bmatrix} NUM_{O(8)} & -EllipMat_{dn} \\ -EllipMat_{dn} & -NUM_{O(8)} \end{bmatrix}$$

$$SpecMatdn_{O(10)} = \begin{bmatrix} NUM_{O(10)} & -EllipMat_{dn} \\ -EllipMat_{dn} & -NUM_{O(10)} \end{bmatrix}$$

$$SpecMatdn_{O(12)} = \begin{bmatrix} NUM_{O(12)} & -EllipMat_{dn} \\ -EllipMat_{dn} & -NUM_{O(12)} \end{bmatrix}$$

$$SpecMatcn_{O(8)} = \begin{bmatrix} NUM_{O(8)} & -EllipMat_{cn} \\ -EllipMat_{cn} & -NUM_{O(8)} \end{bmatrix}$$

$$SpecMatcn_{O(10)} = \begin{bmatrix} NUM_{O(10)} & -EllipMat_{cn} \\ -EllipMat_{cn} & -NUM_{O(10)} \end{bmatrix}$$

$$SpecMatcn_{O(12)} = \begin{bmatrix} NUM_{O(12)} & -EllipMat_{cn} \\ -EllipMat_{cn} & -NUM_{O(12)} \end{bmatrix}$$

- exception: None

8.4.5 Local Functions

None.

9 MIS of Exact Eigenvalue Equations

The secrets of this module are the analytical expressions for the theoretical eigenvalues corresponding to the cn and dn boundary wave solutions.

9.1 Module

TheorEigenValues

9.2 Uses

This module uses input parameters 6.

9.3 Syntax

Name	In	Out	Exceptions
$\lambda_O 1$	$k : \mathbb{R}$	$\mathbb{R}$	-
$\lambda_O 2$	$k : \mathbb{R}$	$\mathbb{R}$	-
$\lambda_O 3$	$k : \mathbb{R}$	$\mathbb{C}$	-
$\lambda_O 4$	$k : \mathbb{R}$	$\mathbb{C}$	-

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

- These are the eigenvalues computed from Deconinck and L.Segal.

9.4.4 Access Routine Semantics

$\lambda_O 1$:

- output: $\frac{1}{2}(1+\sqrt{1-k^2})$
- exception: None

$\lambda_O 2$

- output: $\frac{1}{2}(1 - \sqrt{1 - k^2})$
- exception: None

$\lambda_O 3$:

- output: $\frac{1}{2}(k + i\sqrt{1 - k^2})$
- exception: None

$\lambda_O 4$

- output: $\frac{1}{2}(k - i\sqrt{1 - k^2})$
- exception: None

9.4.5 Local Functions

None.

10 MIS of Numerical Parameters

The secrets of this module are the range of the eigenfunction domain, points in the periodic domain and equation for the numerical scaling factor that computes the eigenfunction derivatives.

10.1 Module

Numpars

10.2 Uses

This module uses Elliptic Functions 14, Diagonal Matrix 12, linspace 16, Elliptic Integral 13 and input parameters 6.

10.3 Syntax

Name	In	Out	Exceptions
Numpars	$k : \mathbb{R}$	$xend$	-
	$N : \mathbb{N}$	$Domain$	
	$P : \{2, 4\}$	$ellipjdn$	
		$ellipjcn$	
		$ellipMAT_{dn}$	
		$ellipMAT_{cn}$	
		h	
$xend$	-	$\mathbb{R}$	-
$Domain$	-	$\mathbb{R}^{2N}$	-
$ellipjdn$	-	$\mathbb{R}^{2N}$	-
$ellipjcn$	-	$\mathbb{R}^{2N}$	-
$ellipjMAT_{cn}$	-	$\mathbb{R}^{2Nx2N}$	-
$ellipjMAT_{dn}$	-	$\mathbb{R}^{2Nx2N}$	-
h	-	$\mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Assumptions

- Input parameters is called before Numerical parameters.
- Input parameters does not throw an exception.

10.4.4 Access Routine Semantics

$Numpars$:

- transition: The data (N, k) will be captured from the input parameters module and used to create a cell structure,NumPars, whose components are xend, Domain,ellipjdn, ellipjdc, $ellipMAT_{dn}$, $elliptMAT_{cn}$ and h. k will be inputted as an argument into the elliptic integral module 13. The resulting integral is equal to $xend$.
  The domain is created using linspace 16. The endpoint arguments of linspace are $-xend$ and $xend$, respectively. The distance between partition points in the resulting domain is $h = \frac{xend}{N}$. Ellipjdn (cn) is derived by computing the ellipjdn (cn) value 14 of each point in the Domain. $EllipjMAT_{cn}$ (dn) is a diagonal matrix whose diagonal is Ellipjcn (dn).
- exception: None

10.4.5 Local Functions

None.

11 MIS of Eigenvalue and Eigenvector Solver

The secret of this module is the numerical algorithm for calculating the eigenvalues and eigenvectors of an n by n matrix.

11.1 Module

eig (https://www.mathworks.com/help/matlab/ref/eig.html)

11.2 Uses

-

11.3 Syntax

Name	In	Out	Exceptions 19
solver	$A : \mathbb{C}^{nxn}$	$\mathbb{C}^n$ x $\mathbb{R}^{nxn}$	NotsquareMat

11.4 Semantics

11.4.1 State Variables

None.

11.4.2 Environment Variables

None.

11.4.3 Assumptions

- The input is a square matrix.

11.4.4 Access Routine Semantics

eig():

- output: out:= λ and $\bar{v}$ such that:
  $A\bar{v} = \lambda\bar{v}$
  $\bar{v} : \mathbb{C}^n$ and $\lambda : \mathbb{C}$
- exception: exce:= $\neg(A : \mathbb{C}^{nxn}) \Rightarrow$ NotSquareMatrix

11.4.5 Local Functions

None.

12 MIS of Diagonal Matrix

The secrets of this module are the numerical algorithm for creating an n by n diagonal matrix from an n by 1 vector and the numerical algorithm for creating an n by 1 vector from an n by n diagonal matrix.

12.1 Module

diag (https://www.mathworks.com/help/matlab/ref/diag.html)

12.2 Uses

-

12.3 Syntax

Name	In	Out	Exceptions 19
solver	A : Diagonal n by n matrix	$\mathbb{C}^n$	NotDiagMat
solver	$v : \mathbb{C}^n$	Diagonal n by n matrix	NotVector

12.4 Semantics

12.4.1 State Variables

None.

12.4.2 Environment Variables

None.

12.4.3 Assumptions

- The input is a square matrix or column vector.

12.4.4 Access Routine Semantics

diag.solver():

- output (if In=A): out:= v such that:
  $A(j, j) = v(j)$ for j=1,2,...,n

- exception (if In=A): exce:= $A \notin (DiagonalMatrix) \Rightarrow$ NotDiagonalMatrix

- output (if In=v): out:= A such that:
  $A(j,j) = v(j)$ for j=1,2,...,n and zero else.

- exception (if In=v): exce:= $v \notin \mathbb{C}^n \Rightarrow$ NotVector

12.4.5 Local Functions

None.

13 MIS of Elliptic Integral

The secret of this module is the the numerical algorithm for calculating the complete elliptic integral for some real constant k.

13.1 Module

ellipK (https://www.mathworks.com/help/symbolic/elliptick.html)

13.2 Uses

-

13.3 Syntax

Name	In	Out	Exceptions 19
solver	$k : \mathbb{R}$	$\mathbb{R}$	NonNumericalError

13.4 Semantics

13.4.1 State Variables

None.

13.4.2 Environment Variables

None.

13.4.3 Assumptions

- The user understands the physical context of the number k.

13.4.4 Access Routine Semantics

ellipK.solver():

- output :

$$\int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{1 - ksin^2(x)}}$$

- exception : exce:= $\neg(k : \mathbb{R}) \Rightarrow$ NonNumeric

13.4.5 Local Functions

None.

14 MIS of Elliptic Functions

The secret of this module is the the numerical algorithm for calculating the values of the elliptic functions.

14.1 Module

ellipj (https://www.mathworks.com/help/matlab/ref/ellipj.html)

14.2 Uses

-

14.3 Syntax

Name	In	Out	Exceptions 19
solver	$X : \mathbb{R}^n$	$\mathbb{R}^{3xn}$	NotVector
	$k : \mathbb{R}$		

14.4 Semantics

14.4.1 State Variables

None.

14.4.2 Environment Variables

None.

14.4.3 Assumptions

None.

14.4.4 Access Routine Semantics

ellipj.solver():

- output : $Y : \mathbb{R}^n$ such that $Y(1, j) = sn(X(j), k)$, $Y(2, j) = cn(X(j), k)$ and $Y(3, j) = dn(X(j), k)$. See Module link for more detail regarding jacobi ellptic functions sn,dn and cn.
- exception : exce:= $\neg(X : \mathbb{R}^n) \Rightarrow$ NotVector

14.4.5 Local Functions

None.

15 MIS of Plotting

The secret of this module is the plotting algorithm.

15.1 Module

plot (https://www.mathworks.com/help/matlab/ref/plot.html)

15.2 Uses

-

15.3 Syntax

Name	In	Out	Exceptions 19
plot	$X : \mathbb{R}^n$ $Y : \mathbb{R}^n$	graph(Y)	BadVal, DimensionErr

15.4 Semantics

15.4.1 State Variables

None.

15.4.2 Environment Variables

None.

15.4.3 Assumptions

None.

15.4.4 Access Routine Semantics

plot.plot():

- Transition: Plot accepts two vectors as input. A figure is created with the following properties. The domain ranges from the minimum of X to the maximum of X. The range ranges from the minimum of Y to the maximum of Y. The pairs $(X(j), Y(j))$ for $j = 1, 2, ..., n$ are plotted on a figure with respect to the aforementioned axes.

- exception : exce1:= $\neg(X, Y \in \mathbb{C}^n) \Rightarrow$ BadVal
  exce2:= $\neg(dim(Y) = dim(X)) \Rightarrow$ DimensionErr

15.4.5 Local Functions

None.

16 MIS of Linspace

The secret of this module is the software algorithm for creating a vector with equally spaced entries.

16.1 Module

Linspace (https://www.mathworks.com/help/matlab/ref/linspace.html)

16.2 Uses

-

16.3 Syntax

Name	In	Out	Exceptions 19
create	$a : \mathbb{R}$	$\mathbb{R}^c$	NonNumerical
	$b : \mathbb{R}$		notNat
	$c : \mathbb{N}$		

16.4 Semantics

16.4.1 State Variables

None.

16.4.2 Environment Variables

None.

16.4.3 Assumptions

None.

16.4.4 Access Routine Semantics

linspace.create():

- output : $X : \mathbb{R}^c$ such that X(1)=a, X(n)=b and $|x(k) - x(k-1)| = \frac{b-a}{n-1}$ for $k \in 2, 3, 4, .., n$.
- exception : exce:= $\neg(c \in \mathbb{N}) \Rightarrow$ notNat
- exception : exce:= $\neg(a : \mathbb{R}) \Rightarrow$ NonNumericalError

- exception : exce:= $\neg(b : \mathbb{R}) \Rightarrow$ NonNumericalError

16.4.5 Local Functions

None.

17 MIS of Toeplitz

The secret of this module is the software algorithm for creating a toeplitz matrix.

17.1 Module

Toeplitz (https://www.mathworks.com/help/matlab/ref/toeplitz.html)

17.2 Uses

-

17.3 Syntax

Name	In	Out	Exceptions 19
create	$c : \mathbb{R}^m$	$\mathbb{R}^{mxm}$	NonNumerical
	$r : \mathbb{R}^m$		notVec

17.4 Semantics

17.4.1 State Variables

None.

17.4.2 Environment Variables

None.

17.4.3 Assumptions

None.

17.4.4 Access Routine Semantics

toeplitz: Returns a nonsymmetric Toeplitz matrix with c as its first column and r as its first row.

17.4.5 Local Functions

None.

18 MIS of Control

The secret of this module is the algorithm that coordinates the overall program and interaction between modules.

18.1 Module

Main

18.2 Uses

This module uses Input Parameters 6, Numerical Parameters10, Spectrum Matrix 8, Eigenvalue solver 11, Output 7, plotting 15 and Exact Eigenvalue equations 9.

18.3 Syntax

Name	In	Out	Exceptions
main	-	-	-

18.4 Semantics

18.4.1 State Variables

None.

18.4.2 Environment Variables

None.

18.4.3 Assumptions

None.

18.4.4 Access Routine Semantics

main():

- transition: This function controls the running of the scientific software. First, input is taken from the user. The input 6 is brought to the numerical parameters module 10 where useful constants, matrices and elliptic function values are calculated. The state variables from Numerical parameters are used as arguments in Spectrum Matrix. The spectrums (eigenvalues) can be calculated 11 once the six spectral matrices 8 are created. These eigenvalues are plotted 15.

18.4.5 Local Funtions

None.

References

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

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Matheus Grasseli and Dmitry Pelinovsky. *Numerical Mathematics*. Jones and Bartlett Learning, Boston, MA, USA, 2007.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

19 Appendix

exception name	description
NonNumericalError	The input is not a number.
BadkRange	The value of k is out of its necessary range.
BadNRange	The value of N is out of its necessary range.
BadPRange	The value of P is out of its necessary range.
NotsquareMat	The input matrix is not square (or n by n for n: $\mathbb{N}$)
NotDiagMat	The input matrix is not diagonal
NotVector	The input was not a vector.
BadVal	At least one of the vector elements is not a number.
NotNat	The input was not a natural number.
NotVec	The input was not a vector.