Module Interface Specification for SpecSearch

Robert E. White

November 23, 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.

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
$O(1)$	First order central numerical differentiation method
$O(4)$	Fourth order central numerical differentiation method
cheb	Chebyshev 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 from SpecSearch's modules. Mathematical notation will be used to detail the input/output relationships and external behaviour of the modules. 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 that appears in 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. Mathematicans 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
	Spectrum Error Equation
	Numerical Parameters
	Control
Software Decision	Sequence Data Structure
	Eigenvalue and Eigenvector Solver
	Diagonal Matrix Generator
	Elliptic Integral
	Elliptic Functions
	Plotting
	Linspace

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	$k : \mathbb{R}$ $N : \mathbb{N}$	-	NonNumericalError
Verify_params	-	-	BadkRange, NotNat
k	-	$\mathbb{R}$	-
N	-	$\mathbb{N}$	-

6.4 Semantics

6.4.1 State Variables

$k : \mathbb{R}$
$N : \mathbb{N}$

6.4.2 Environment Variables

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

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).

6.4.4 Access Routine Semantics

InParams.k():

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

InParams.N():

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

Load_Params():

- transition: The data is read sequentially from the command line. The data are seperated by the return key. 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
$\neg(N : \mathbb{N}) \Rightarrow$ BadNRange

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 12, Spectrum Error Equations 10 and Numerical Parameters 11.

7.3 Syntax

Name	In	Out	Exceptions
capture	$D : \mathbb{R}^{2n}$	$Spectral$	-
	$Er_j : \mathbb{R}^2$		-
	$V_j : \mathbb{C}^{4nx4n}$		-
	$Vl_j : \mathbb{R}^{4n}$		-
$Spectral_j$	-	$\mathbb{C}^{4n(4n+2)}$	-
$Spectral_D$	-	$\mathbb{R}^{2n}$	-
Er_j	-	$\mathbb{R}$	-

7.4 Semantics

The j refers to the j^{th} numerical method; $j : \{O(1), O(4), cheb\}$. 2

7.4.1 State Variables

$Spectral_j : \mathbb{C}^{4n(4n+2)}$
$Spectral_D : \mathbb{R}^{2n}$
$Er_j : \mathbb{R}^2$

7.4.2 Environment Variables

None.

7.4.3 Assumptions

- Eigenvalue and Eigenvector solver, Spectrum Error Equations and Numerical Parameters 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.$Spectral_D()$:

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

OutForm.$Er_j()$:

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

OutForm.$Spectral_j()$:

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

capture():

- Transition: The data (D_j,Er_j,V_j,Vl_j) will be captured from the other modules and then stored into *Spectral* for convienience. Spectral is a data structure with four cell components. The first three cells are $4n + 1$ by $4n$ matrices created as follows: the $4n$ by 1 vector of eigenvalues, Vl_j, will be transposed and turned into the $(4n + 1)^{th}$ row of $spectral_j$. The previous $4n$ rows of $spectral_j$ will be the matrix,V_j, composed of the eigenvectors for the j numerical method. Each column of V_j is an eigenvector from the j numerical method. D will be stored in Spectral as the fourth cell component.
- 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 11.

8.3 Syntax

Name	In	Out	Exceptions
create	$n : \mathbb{N}$	$SpecMat_{O(1)}$	-
	$h : \mathbb{R}$	$SpecMat_{O(4)}$	-
	$elipMAT : \mathbb{R}^{2nx2n}$	$SpecMat_{Cheb}$	-
	$k : \mathbb{R}$		-
$SpecMat_{O(1)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMat_{O(4)}$	-	$\mathbb{R}^{4nx4n}$	-
$SpecMat_{Cheb}$	-	$\mathbb{R}^{4nx4n}$	-

8.4 Semantics

8.4.1 State Variables

$SpecMat_{Cheb} : \mathbb{R}^{4nx4n}$
$SpecMat_{0(4)} : \mathbb{R}^{4nx4n}$
$SpecMat_{0(1)} : \mathbb{R}^{4nx4n}$

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.create() before any of SpecMat's state variables are accessed.

8.4.4 Access Routine Semantics

SpecMat.$SpecMat_{O(1)}$:

- output: out = $SpecMat_{O(1)}$:
- exception: None

SpecMat.$SpecMat_{O(4)}$:

- output: out = $SpecMat_{O(4)}$:
- exception: None

SpecMat.SpecFour:

- output: out = $SpecMat_{cheb}$:
- exception: None

SpecMat.create():

- transition: The data (n,h,elipMAT,k) will be captured from numerical parameters and used to create the spectral matrices. There will be three spectral matrices. Each matrix corresponds to a different numerical algorithm for approximating the eigenfunction derivatives. 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 (15) evaluated at points in the discretized domain 11. The remaining quadrants of the matrix are equal to $-NUM_j$ or $+NUM_j$ such that:

$$NUM_{O(1)} = \begin{bmatrix} 0 & \frac{n}{2} & 0 & \dots & 0 & -\frac{n}{2} \\ -\frac{n}{2} & 0 & \frac{n}{2} & \ddots & \ddots & 0 \\ 0 & -\frac{n}{2} & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \frac{n}{2} & \vdots \\ 0 & \ddots & \ddots & -\frac{n}{2} & 0 & \frac{n}{2} \\ \frac{n}{2} & 0 & \dots & 0 & -\frac{n}{2} & 0 \end{bmatrix}$$

$$NUM_{O(6)} = \begin{bmatrix} 0 & \frac{3n}{4} & -\frac{3n}{20} & \frac{n}{60} & 0 & \dots & 0 & -\frac{n}{60} & \frac{3n}{20} & -\frac{3n}{4} \\ -\frac{3n}{4} & 0 & \frac{3n}{4} & -\frac{3n}{20} & \frac{n}{60} & 0 & \dots & 0 & -\frac{n}{60} & \frac{3n}{20} \\ \frac{3n}{20} & -\frac{3n}{4} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & -\frac{n}{60} \\ -\frac{n}{60} & \frac{3n}{20} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & -\frac{n}{60} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \frac{n}{60} & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -\frac{3n}{20} & \frac{n}{60} \\ \frac{n}{60} & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \frac{3n}{4} & -\frac{3n}{20} \\ -\frac{3n}{20} & \frac{n}{60} & 0 & \ddots & \ddots & -\frac{n}{60} & \frac{3n}{20} & -\frac{3n}{4} & 0 & \frac{3n}{4} \\ \frac{3n}{4} & -\frac{3n}{20} & \frac{n}{60} & 0 & \dots & 0 & -\frac{n}{60} & \frac{3n}{20} & -\frac{3n}{4} & 0 \end{bmatrix}$$

$$NUM_{Cheb} = \begin{bmatrix} 0 & & & & & -\frac{1}{2}cot(\frac{1h}{2}) \\ -\frac{1}{2}cot(\frac{1h}{2}) & \ddots & \ddots & \ddots & \ddots & \frac{1}{2}cot(\frac{2h}{2}) \\ \frac{1}{2}cot(\frac{2h}{2}) & \ddots & \ddots & \ddots & \ddots & -\frac{1}{2}cot(\frac{3h}{2}) \\ -\frac{1}{2}cot(\frac{3h}{2}) & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \frac{1}{2}cot(\frac{1h}{2}) \\ \frac{1}{2}cot(\frac{1h}{2}) & & & & & 0 \end{bmatrix}$$

Therefore the output matrices are:

$$SpecMat_{O(1)} = \begin{bmatrix} NUM_{O(1)} & -EllipMat_{O(1)} \\ -EllipMat_{O(1)} & -NUM_{O(1)} \end{bmatrix}$$

$$SpecMat_{O(4)} = \begin{bmatrix} NUM_{O(4)} & -D_{O(4)} \\ -D_{O(4)} & -NUM_{O(4)} \end{bmatrix}$$

$$SpecMat_{cheb} = \begin{bmatrix} NUM_{cheb} & -D_{cheb} \\ -D_{cheb} & -NUM_{cheb} \end{bmatrix}$$

- 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 two purely real eigenvalues.

9.1 Module

TheorEigenValues

9.2 Uses

This module uses input parameters 6.

9.3 Syntax

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

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

9.4.5 Local Functions

None.

10 MIS of Spectrum Error Equation

The secret of this module is the equation for error between exact and approximated eigenvalues.

10.1 Module

ErrCalc

10.2 Uses

This module uses Exact Eigenvalue equations 9 and Eigenvalue solver 12.

10.3 Syntax

Name	In	Out	Exceptions
$\lambda_O 1$	-	$\mathbb{R}$	-
$\lambda_O 2$	-	$\mathbb{R}$	-
$\lambda_C 1$	-	$\mathbb{R}$	-
$\lambda_C 2$	-	$\mathbb{R}$	-
$Err1$	$\lambda_O 1 : \mathbb{R}$	$\mathbb{R}$	-
	$\lambda_C 1 : \mathbb{R}$		-
$Err2$	$\lambda_C 2 : \mathbb{R}$	$\mathbb{R}$	-
	$\lambda_C 2 : \mathbb{R}$		-

10.4 Semantics

10.4.1 State Variables

$\lambda_O 1 : \mathbb{R}$
$\lambda_O 2 : \mathbb{R}$
$\lambda_C 1 : \mathbb{R}$
$\lambda_C 2 : \mathbb{R}$

10.4.2 Environment Variables

None

10.4.3 Assumptions

None.

10.4.4 Access Routine Semantics

$Err1$:

- output: $|\lambda_O 1 - \lambda_C 1|$
- exception: None

$Err2$

- output: $|\lambda_O 2 - \lambda_C 2|$
- exception: None

$ErrCalc.\lambda_O 1$:

- output: out = $\lambda_O 1$
- exception: None

$ErrCalc.\lambda_O 2$:

- output: out = $\lambda_O 2$
- exception: None

$ErrCalc.\lambda_C 1$:

- output: out = $\lambda_C 1$
- exception: None

$ErrCalc.\lambda_C 2$:

- output: out = $\lambda_C 2$
- exception: None

10.4.5 Local Functions

None.

11 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.

11.1 Module

Numpars

11.2 Uses

This module uses Elliptic Functions 15, Diagonal Matrix 13, linspace 17, Elliptic Integral 14 and input parameters 6.

11.3 Syntax

Name	In	Out	Exceptions
CreateVars	$k : \mathbb{R}$	$xend : \mathbb{R}$	-
	$N : \mathbb{N}$	$Domain : \mathbb{R}^{2N}$	
		$ellipj : \mathbb{R}^{2N}$	
		$ellipMAT : \mathbb{R}^{2Nx2N}$	
$xend$	-	$\mathbb{R}$	-
$Domain$	-	$\mathbb{R}^{2N}$	-
$ellipjdn$	-	$\mathbb{R}^{2N}$	-
$ellipjMAT$	-	$\mathbb{R}^{2Nx2N}$	-
h	-	$\mathbb{R}$	-

11.4 Semantics

11.4.1 State Variables

$h : \mathbb{R}$
$Domain : \mathbb{R}^{2N}$
$ellipjdn : \mathbb{R}^{2N}$
$ellipMAT : \mathbb{R}^{2Nx2N}$
$xend : \mathbb{R}$

11.4.2 Environment Variables

None

11.4.3 Assumptions

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

11.4.4 Access Routine Semantics

$Numpars.xend$:

- output: $xend$
- exception: None

$Numpars.Domain$

- output: $Domain$
- exception: None

$Numpars.ellipjdn$:

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

$Numpars.ellipjMAT$:

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

$Numpars.h$:

- output: out = h
- exception: None

$Numpars.CreateVars$:

- transition: The data (N, k) will be captured from the input parameters module and used to create the state variables. k will be inputted as an argument into the elliptic integral module 14. The resulting integral is equal to $xend$.
  The domain is created using linspace 17. 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 is derived by computing the ellipjdn value 15 of each point in the Domain. EllipjMAT is a diagonal matrix whose diagonal is Ellipjdn.
- exception: None

11.4.5 Local Functions

None.

12 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.

12.1 Module

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

12.2 Uses

This module uses Spectrum Matrix 8.

12.3 Syntax

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

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.

12.4.4 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

12.4.5 Local Functions

None.

13 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.

13.1 Module

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

13.2 Uses

This module uses Numerical Parameters 11.

13.3 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

13.4 Semantics

13.4.1 State Variables

None.

13.4.2 Environment Variables

None.

13.4.3 Assumptions

- The input is a square matrix or column vector.

13.4.4 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

13.4.5 Local Functions

None.

14 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.

14.1 Module

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

14.2 Uses

This module uses Numerical Parameters 11.

14.3 Syntax

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

14.4 Semantics

14.4.1 State Variables

None.

14.4.2 Environment Variables

None.

14.4.3 Assumptions

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

14.4.4 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

14.4.5 Local Functions

None.

15 MIS of Elliptic Functions

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

15.1 Module

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

15.2 Uses

This module uses Numerical Parameters 11.

15.3 Syntax

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

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

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

15.4.5 Local Functions

None.

16 MIS of Plotting

The secret of this module is the plotting algorithm.

16.1 Module

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

16.2 Uses

This module uses Output 7.

16.3 Syntax

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

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

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

16.4.5 Local Functions

None.

17 MIS of Linspace

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

17.1 Module

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

17.2 Uses

This module uses Numerical Parameters 11.

17.3 Syntax

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

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

linspace.create():

- output : $X \in \mathbb{R}^c$ 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

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, Numerical Parameters, Spectrum Matrix, Eigenvalue solver, Spectrum Error , Output and plotting.

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. This input is used to create useful numerical parameters that will drive the rest of the calculations. The input is brought to the numerical parameters module where useful constants, matrices and elliptic function values are calculated. The state variables from Numerical parameters are used as arguments in Spectrum Matrix. The spectrum (eigenvalues) can be calculated once the spectrum matrix is

created. These eigenvalues are plotted and their deviation from theoretical values are computed (err modules).

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.

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.
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.