Module Interface Specification for CFS

Malavika Srinivasan

December 24, 2018

1 Revision History

Date	Version	Notes
Nov 10, 2018	1.0	First draft by Malavika
Nov 21, 2018	1.1	Corrections based on presentation by Malavika
Nov 24, 2018	1.2	MIS submission draft by Malavika

2 Symbols, Abbreviations and Acronyms

See CA Documentation at https://github.com/Malavika-Srinivasan/CAS741/blob/master/docs/SRS/CA.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for CFS (Curve Fitting Software). The goal of CFS is to find the parameters of the curve which is the best possible fit through the given set of 'n' data points, where $n >= 2$.

Complementary documents include the Commonality Analysis and Module Guide. The full documentation and implementation can be found at the repository https://github.com/Malavika-Srinivasan/CAS741.

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 $(c_1 \implies r_1 | c_2 \implies r_2 | ... | c_n \implies r_n)$.

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

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 (-∞, ∞)
boolean	$\mathbb{B}$	a value from the set $\{True, False\}$

The specification of CFS 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, CFS 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 Module	
Behaviour-Hiding Module	Input Output
Software Decision Module	Sequence Services Interpolation Regression Plot

Table 1: Module Hierarchy

6 MIS of Input module

This module corresponds to R1, R2 and R3 in the CA document. The secrets of this module are how the data points are verified. The verify secrets are isolated to their own access programs.

6.1 Module

input

6.2 Uses

Sequence services(10)

6.3 Syntax

Name	In	Out	Exceptions
verifyInput	t:$\mathbb{R}^n$, y:$\mathbb{R}^n$	-	TypeError, LengthError, LengthMismatchError.
verifyDegree	degree:$\mathbb{N}$	$degVerify$:$\mathbb{B}$	ValueError

6.3.1 Exported Constants

NA

6.3.2 Exported Access Programs

6.4 Semantics

6.4.1 State Variables

6.4.2 Environment Variables

[I decided that my input module will only verify the input and will not load or store the input arrays. —Malavika]

6.4.3 Assumptions

6.4.4 Access Routine Semantics

verifyInput(t,y):

- transition:NA
- output: NA

- exception: $exc :=$

Name	Exception
$(\|t\| \leq 1 \vee \|y\| \leq 1)$	$\Longrightarrow$ LengthError
$(\|t\| \neq \|y\|)$	$\Longrightarrow$ LengthMismatchError
$(\exists x \in t \wedge x \notin \mathbb{R})$	$\Longrightarrow$ TypeError
$(\exists x \in y \wedge x \notin \mathbb{R})$	$\Longrightarrow$ TypeError

verifyDegree(deg):

- transition:NA
- output:NA
- exception: $exc :=$

Name	Exception
$(deg \notin \mathbb{N})$	$\Longrightarrow$ ValueError

6.4.5 Local Functions

NA

7 MIS of Interpolation module

This module corresponds to R4, R5, R6, R7 and R9 in section 7.1 of CA document.

7.1 Module

interp

7.2 Uses

Input(6), Sequence services(10)

7.3 Syntax

7.3.1 Exported Constants

NA

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
interpMonomial	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^n$	-
interpLagrange	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^n$	-
interpNewton	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^n$	-
interpHermiteCubic	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^{n\times n-1}$, int:$\mathbb{R}^n$	-
interpBSpline	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^m$	-
evalMonomial	s:$\mathbb{R}^n$,xNew:$\mathbb{R}^m$	yNew:$\mathbb{R}^m$	-
evalLagrange	s:$\mathbb{R}^n$, t:$\mathbb{R}^n$, xNew:$\mathbb{R}^m$	yNew:$\mathbb{R}^m$	-
evalNewton	s:$\mathbb{R}^n$, t:$\mathbb{R}^n$, xNew:$\mathbb{R}^m$	yNew:$\mathbb{R}^m$	-
evalHermiteCubic	xNew:$\mathbb{R}^m$, t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	yNew:$\mathbb{R}^m$	-
evalBSpline	xNew:$\mathbb{R}^m$, t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	yNew:$\mathbb{R}^m$	-

[Your input module has state information, yet you are explicitly providing inputs of this same information. You need to decide what your design is and stick to it. —SS] [Yes Dr.Smith. Makes sense. Now I don't have state variables at all. My user program handles input, my interpolation and regression modules ris called by user program. The interpolation and regression modules use Input module to verify the input data. —Malavika]

7.4 Semantics

7.4.1 State Variables

NA

[Why do you have a state variable? The connection between your inputs and your outputs is unclear. Is the state variable keeping the information that you need for evaluation? That would make sense. I think your design would make more sense if you followed the design we used for SFS. You can have one interpolating polynomial object and the specific method it uses to build its state variables comes as an input parameter. This would be similar to the 2AA4/2ME3 assignment I showed in class, where the order of interpolation was a parameter. This would mean that you only needed one eval. The object itself would know how to interpret eval —SS] [Yes, I had a bit of confusion in handling input and output. I realized I dont need any state variable at all. My Input module will only verify the inputs. User program directly receives output directly from interpolation and regression modules. Then user program can use the services provided by Output module to get variabilities in output. —Malavika]

7.4.2 Environment Variables

NA

7.4.3 Assumptions

7.4.4 Access Routine Semantics

interpMonomial(t,y):

- transition: NA
- output: $out := x$ as shown below.
  1. x: $\mathbb{R}^n$ obtained by solving for x using the equation below.

$$\begin{bmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & t_n & t_n^2 & \dots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- exception: NA

evalMonomial(s,$xNew$):

- transition: NA
- output: $out := yNew$

  $\forall t_i \in xNew$, yNew|| y_1, where y_1 is obtained as shown below.

$$y_1 = s[0] + s[1]t_i + s[2]t_i^2 + ...s[n-1]t_i^{n-1}$$

- exception: NA

interpLagrange(t,y):

- transition: NA
- output: $out \coloneqq x$ as shown below.

  1. x: $\mathbb{R}^n$ obtained by solving for x using the equation below.

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- exception: NA

evalLagrange(s,t,$xNew$):

- transition: NA
- output: $out := yNew$

  $\forall t_i \in xNew$, yNew|| y_0 where y_0 is obtained as shown below.

$$y_0 = y_1 l_1(t) + y_2 l_2(t) + \dots y_n l_n(t).$$

  where $l_j(t)$ is given by,

$$l_j(t) = \frac{\Pi_{k=1,k\neq j}^{n}(t - t_k)}{\Pi_{k=1,k\neq j}^{n}(t_j - t_k)}$$

- exception: NA

interpNewton(t,y):

- transition:NA
- output: $out \coloneqq x$ as shown below.

1. x: $\mathbb{R}^n$ obtained by solving for x using the equation below.

$$\begin{bmatrix} \pi_0(t_0) & 0 & 0 & \dots & 0 \\ \pi_0(t_1) & \pi_1(t_1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \pi_0(t_n) & \pi_1(t_n) & \pi_2(t_n) & \dots & \pi_n(t_n) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

where,
$i < j \implies \pi_j(t) = 0$, $i \geq j \implies \pi(t) = \Pi_{k=1}^{j-1}(t - t_k)$

- exception: NA

evalNewton(s, t, y):

- transition: NA
- output: $out := yNew$
  $\forall t_i \in xNew$, yNew|| y_0 where y_0 is obtained as shown below.

$$y_0 = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + ...x_n(t - t_1)(t - t_2)...(t - t_{n-1})$$

- exception: NA

interpHermiteCubic(t,y):

- transition: NA
- output: $out := (x,$ int) as shown below.
  - 1. x:$\mathbb{R}^{n \times n-1}$ obtained by solving for x using the equation below.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 & t_n^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 2t_n & 3t_n^2 \\ 0 & 0 & 2 & 6t_0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 2 & 6t_n \\ 0 & 0 & 0 & 6 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{n(m+1)} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \\ y_0^{(1)} \\ \vdots \\ y_n^{(1)} \\ y_0^{(2)} \\ \vdots \\ y_n^{(2)} \\ y_0^{(3)} \\ \vdots \\ y_n^{(3)} \end{bmatrix}$$

Where $y_0^{(3)}$ represents the third derivative of y_0.

- int: $\mathbb{R}^n = t[0]$ to $t[n-2]$, by default we make each point a breakpoint and hence there is a piecewise polynomial between $[t_{i-1}, t_i)$.

- exception: NA

evalHermiteCubic($xNew$,t,y):

- transition:
- output: $out := yNew$

  $\forall t \in xNew$, yNew|| y_0 where y_0 is obtained as shown below.

$$y_0 = x_{k0} + x_{k1}t + x_{k2}t^2 + x_{k3}t^3 \quad for\ k = 0\ to\ n-2$$

  Where k represents the interval to which t belongs.

- exception: NA

interpBSpline(t,y):

- transition:NA
- output: $out :=\ x$ as shown below.

  1. x: $\mathbb{R}^m$ obtained by using the formula below.
     For $k > 0$ to n, where n is the number of data points

$$x = v_i^k = \frac{t - t_i}{t_{i+k} - t_i}$$

     [BSpline gives the v values which will be used recursively in formula defined in IM**??** of the CA document —Malavika]

- exception: NA

evalBSpline($xNew$,t,y):

- transition:
- output: $out := yNew$

  $\forall t_i \in xNew$, yNew|| y_0 where y_0 is obtained as shown below.

$$y_0 = B_i^k(t) = v_i^k(t)B_i^{k-1}(t) + (1 - v_{i+1}^k(t))B_{i+1}^{k-1}(t)$$

  for $k > 0$.
  [I want the notation of 'B' so that it is easy to understand the coefficients, thats why I have $y_0 = B$... —Malavika]

- exception: NA

7.4.5 Local Functions

NA

8 MIS of Regression module

This module corresponds to R4, R8 and R9 in section 7.1 of CA document.

8.1 Module

reg

8.2 Uses

Input(6), Sequence services(10)

8.3 Syntax

8.3.1 Exported Constants

NA

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
regNormalEq	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$, deg:$\mathbb{N}$	x:$\mathbb{R}^{deg+1}$	-
regAugSys	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$, deg:$\mathbb{N}$	x:$\mathbb{R}^{deg+1}$	-
regOrthogonalTn	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$	x:$\mathbb{R}^n$	-
evalReg	s:$\mathbb{R}^n$, xNew:$\mathbb{R}^m$	yNew:$\mathbb{R}^m$	-
polyval	s:$\mathbb{R}^n$, xNew:$\mathbb{R}^m$	yNew:$\mathbb{R}^m$	-

[Again, it is unclear why you have an input parameters module with state information if you aren't going to use that state information. —SS][I dont have state variables anymore. —Malavika]

8.4 Semantics

8.4.1 State Variables

NA

8.4.2 Environment Variables

NA

8.4.3 Assumptions

8.4.4 Access Routine Semantics

regNormalEq(t,y,deg):

- transition: NA
- output: $out := x$ as shown below.
  - $x : \mathbb{R}^{deg+1}$, obtained by solving for x in the equation below.

$$A^T A x = A^T y$$

- exception: NA

regAugSys(t,y,deg):

- transition: NA
- output: $out := x$ as shown below.
  - $x : \mathbb{R}^{deg+1}$, obtained by solving for x in the equation below.

$$r + Ax = y$$

$$A^T r = 0$$

    Where r is the residual vector.

- exception: NA

regOrthogonalTn(t,y):

- transition: NA
- output: $out := x$ as shown below.
  - $x : \mathbb{R}^n$, obtained by solving for x in the equation below.[For QR factorization, degree of polynomial = n-1, so the length of coeff array will be the same as data length. —Malavika]

$$Ax = Py$$

    Where P is the orthogonality matrix.

- exception: NA

polyval(s,$xNew$):

- transition: NA
- output: $out := yNew$, where

$$yNew := numpy.polyval(s, xNew)$$

- exception: NA
- implemented by: Python numpy

evalReg(s,$xNew$):

- transition: NA
- output: $out := yNew$, where

$$yNew := polyval(s, xNew)$$

- exception: NA

[Is this connected to the state variable? I think so? —SS] [You should use calls to Python in your specification. Say things mathematically. —SS][Changes made —Malavika]

8.4.5 Local Functions

NA

9 MIS of Output module

This module corresponds to R10 in the CA document. The secrets of this module are how the output is given to the user program.

9.1 Module

output

9.2 Uses

Plot(11), Sequence services(10)

9.3 Syntax

Name	In	Out	Exceptions
plotDataFit	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$,yfit:$\mathbb{R}^n$	plot	-
coeffPlotScreen	t:$\mathbb{R}^n$,y:$\mathbb{R}^n$,yfit:$\mathbb{R}^n$, x:$\mathbb{R}^m$	x:$\mathbb{R}^m$, plot	-
coeffFile	x:$\mathbb{R}^m$	x:$\mathbb{R}^m$	-

9.3.1 Exported Constants

NA

9.3.2 Exported Access Programs

9.4 Semantics

9.4.1 State Variables

NA

9.4.2 Environment Variables

OutputFile: String

The OutputFile represents a file in the hardware's file system.

The name of the file will be results.txt.

9.4.3 Assumptions

- The interpolation or the regression module will be called before the output module.

9.4.4 Access Routine Semantics

plotDataFit(t,y,$yfit$):

- transition: NA
- output: *out:*= plot
    1. plot(t, y) and plot(t,$yfit$)
- exception: NA

coeffPlotScreen(t,y,$yfit$,x):

- transition : NA
- output: Out:= (x, plot), where x from the input and plot as shown below.
  - plot(t, y) and plot(t,$yfit$)
- exception: NA

coeffFile(x):

- transition: NA
- output: Out:= x
  - Create OutputFile.
  - write x
- exception: NA

9.4.5 Local Functions

NA

10 MIS of Sequence Services

10.1 Module

seqSer [Provided by numpy Python —Malavika]

10.2 Uses

NA

10.3 Syntax

10.3.1 Exported Constants

NA

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
isAscending	$arr : \mathbb{R}^n$	$\mathbb{B}$	-
unique	$arr : \mathbb{R}^n$	$uniArr : \mathbb{R}^m$	-
array_equal	$arr1 : \mathbb{R}^n$,$arr2 : \mathbb{R}^n$	$\mathbb{B}$	-

10.4 Semantics

10.4.1 State Variables

NA

10.4.2 Environment Variables

NA

10.4.3 Assumptions

NA

10.4.4 Access Routine Semantics

unique(arr):

- transition: NA
- output: $out := uniArr : \mathbb{R}^m$ where $uniArr$ is given by,
  1. $arr[i] \neq arr[j], \forall i, j \in [(0 \text{ to } (|arr| - 1)] \wedge j \neq i \implies$
     $\forall i \in [0 \text{ to } |arr| - 1], uniArr[i] := arr[i]$
- exception: NA

isAscending(arr)

- transition:NA
- output: $out := \neg\exists(i|i \in [0..|arr| - 2] : arr_{i+1} < arr_i)$
- exception: NA

array_equal($arr1$,$arr2$):

- transition: NA
- output: $out := (|arr1| == |arr2|) \wedge (\forall i \in [0 \text{ to } |arr1| - 1], arr1[i] == arr2[i]) \implies True$
- exception: NA

10.4.5 Local Functions

NA

10.4.6 Considerations

This module is provided by numpy in Python.

11 MIS of Plot Module

11.1 Module

plot

11.2 Uses

NA

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
Plot	$X : \mathbb{R}^n, Y : \mathbb{R}^n$		SeqSizeMismatch

11.4 Semantics

11.4.1 State Variables

NA

11.4.2 Environment Variables

win: 2D sequence of pixels displayed on the screen
[good —SS]

11.4.3 Assumptions

NA

11.4.4 Access Routine Semantics

Plot(X, Y)

- transition: modify win so that it displays an x-y graph of the data points showing X and the corresponding Y values. X is the independent variable and Y as the dependent variable.
- exception: $(|X| \neq |Y| \implies$ SeqSizeMismatch$)$

References

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.