CFS: Unit Verification and Validation Plan for CFS

Malavika Srinivasan

December 6, 2018

1 Revision History

Date	Version	Notes
Dec 4, 2018	1.0	First draft by Malavika

Contents

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

Also, see the table of symbols in CA at: https://github.com/Malavika-Srinivasan/CAS741/tree/master/docs/SRS/CA.pdf

This document explains the unit verification and validation plan, which aims at verifying the modules of CFS to improve the qualities such as reliability and correctness. It is organized into different sections giving a detailed description of the CFS in terms of its goals, objectives, essential qualities and test cases to verify each module listed in the MG.

3 General Information

This section explains the summary of what is being tested in this document, the objectives of this document and references for this document. The qualities which are essential for CFS are correctness, reliability, maintainability, testability and portability.

3.1 Purpose

This document will summarize the plan for verification and validation of of the modules of CFS in compliance with their functions and semantics mentioned in the MIS document found at https://github.com/Malavika-Srinivasan/CAS741/tree/master/docs/Design/MIS/MIS.pdf

The goal statement from the CA document is presented below.
"Given the set of data points, the choice of software from CFS and the variabilities of the software, the CFS should:

1. Compute the parameters of the curve which is the best possible fit through the set of data points".

3.2 Scope

Some of the modules such as sequence services and plot module are not implemented by CFS, but are a part of it. The services offered by these modules, are implemented by python and testing them are out of scope for this project. The output module of CFS does not compute anything. It essentially gives the output from the modules to the user program by using access methods from interpolation or regression module. It uses plot module for plotting the results. Hence, the testing of output module is covered by the test cases of interpolation and regression module.

4 Plan

This section represents the planning phase for unit verification and validation plan in terms of people, tasks and tools involved in this process.

4.1 Verification and Validation Team

The verification and validation team for CFS includes the following people.

- Malavika Srinivasan
- Dr.Spencer Smith
- Hanane Zlitni

4.2 Automated Testing and Verification Tools

In this project, we will be using pytest to verify the modules of CFS. It is a testing framework available in python. The results of the test including the code coverage metrics will be presented in the test report. We will be using cov package which gives the code coverage metrics.

4.3 Non-Testing Based Verification

NA

5 Unit Test Description

The test cases presented here will verify the access programs of the modules listed in MIS. Usually, each access method will have a test case associated with it. If not, the reason for why the method does not have an associated test case is explained. The MIS can be found at https://github.com/Malavika-Srinivasan/CAS741/tree/master/docs/Design/MIS/MIS.pdf

5.1 Tests for Functional Requirements

In this section, we present the test cases which are related to the functional requirements of CFS. The functional requirements can be found at https://github.com/Malavika-Srinivasan/CAS741/tree/master/docs/SRS/CA.pdf

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS] [I have parallel testing, which makes sense as a system test. But not for unit test. —Malavika]

5.1.1 Input Module

In this section, we will present the test cases for each access method of the input module. Since all the access methods are tested, the entire module will be tested. The test cases for this module are selected based on the constraints for the input data.
The typical constraints on the input data for CFS is associated with the data type and length of the input data. Some of the inputs such as the degree of the polynomial have mathematical restrictions of being a natural number. The test cases listed below are designed to ensure that the input data to CFS does not violate these constraints.

1. **T1: Test case for verify input - One data point**
   Please refer to test case T1 in section 5.1.1 in System verification and validation plan.

   Test Case Derivation: We need at least two points to draw a straight line which is the lowest order polynomial. Only one data point does not need a curve to be represented. Hence, CFS should throw an exception.

2. **T2: Test case for verify input - Length mismatch**

   Type: Automatic

   Initial State: NA

   Input: t=[1,2,3] , y=[2,4] (length mismatch)

   Output: Exception message ("t and y array must have same length.")

   Test Case Derivation: NA

   How test will be performed: Pytest

3. **T3: Test case for verify input - Type error**
   Please refer T2 in section 5.1.1 in System verification and validation plan.

   Test Case Derivation: It is necessary that the input arrays contains only numbers.

4. **T4: Test case for verify degree - value error**

   Type: Automatic

   Initial State: NA

   Input: deg = 1.5

   Output: Exception message ("Degree cannot be a real number")

   Test Case Derivation: NA

   How test will be performed: Pytest

5.1.2 Interpolation module

In this section, we will represent the test cases for each access program present in the interpolation module. This module has access programs to find the coefficients of the interpolated curve through a set of points (t_i, y_i) for $i = 0$ to n and find the value of the interpolating polynomial at a given 't' value.

5. **T5: Test case for interpMonomial**

   Please refer to test cases T4 and T5 in section 5.1.2 in System verification and validation plan.

6. **T6: Test case for interpLagrange**

   Please refer to test cases T6 and T7 in section 5.1.2 in System verification and validation plan.

7. **T7: Test case for interpNewton**

   Please refer to test cases T8 and T9 in section 5.1.2 in System verification and validation plan.

8. **T8: Test case for interpHermiteCubic**

   Please refer to test cases T10 and T11 in section 5.1.2 in System verification and validation plan.

9. **T9: Test case for interpBSpline**

   Please refer to test case T12 in section 5.1.2 in System verification and validation plan.

10. **T10: Test case for evalMonomial**

    Please refer to test case T13 and T14 in section 5.1.2 in System verification and validation plan.

11. **T11: Test case for evalLagrange**

    Please refer to test case T15 and T16 in section 5.1.2 in System verification and validation plan.

12. **T12: Test case for evalNewton**

    Please refer to test case T17 and T18 in section 5.1.2 in System verification and validation plan.

13. **T13: Test case for evalHermiteCubic**

    Please refer to test case T19 and T20 in section 5.1.2 in System verification and validation plan.

14. **T14: Test case for evalBSpline**

    Please refer to test case T21 in section 5.1.2 in System verification and validation plan.

5.1.3 Regression Module

In this section, we will represent the test cases for each access program present in the regression module. This module has access programs to find the parameters of the best fit curve through a set of points (t_i, y_i) for $i = 0$ to n and find the value of the polynomial at a given 't' value.

15. **T15: Test case for regNormalEq**

    Please refer to test cases T22, T23 and T24 in section 5.1.3 in System verification and validation plan.

16. **T16: Test case for regAugSys**

    Please refer to test cases T25 and T26 in section 5.1.3 in System verification and validation plan.

17. **T17: Test case for regOrthogonalTrans**

    Please refer to test cases T27 and T28 in section 5.1.3 in System verification and validation plan.

18. **T18: Test case for evalReg**

    Please refer to test case T29 in section 5.1.3 in System verification and validation plan.

5.2 Tests for Nonfunctional Requirements

Please see section 5.2 in System verification and validation plan.

5.3 Traceability Between Test Cases and Modules

The following table shows the traceability mapping for test cases and the modules. All the modules except the ones listed as out of scope are covered and every access program in the module is tested. This may be considered as a proof for coverage of modules.

Table 1: Modules Traceability Matrix

Test Number	**Modules**
T1	M1
T2	M1
T3	M1
T4	M1
T5	M2
T6	M2
T7	M2
T8	M2
T9	M2
T10	M2
T11	M2
T12	M2
T13	M2
T14	M2
T15	M3
T16	M3
T17	M3

References

6 Appendix

NA

6.1 Symbolic Parameters

NA