Test Report: Curve Fitting Software(CFS)

Malavika Srinivasan

December 24, 2018

1 Revision History

Date	Version	Notes
Dec 24, 2018	1.0	Final version by Malavika

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

Contents

List of Tables

List of Figures

This document represents the test report of the test cases presented in the System Verification and Validation plan of CFS.

3 Functional Requirements Evaluation

This section describes the summary of test results for the test cases which test the functional requirements of CFS. Thirty (30) test cases are presented in section 5.1 of the System VnV plan. The summary of the test results is shown below.

Test cases	**Relative error**
T1 - T30	1×10^{-1}

All the test cases passed and the code coverage was 100%.

4 Nonfunctional Requirements Evaluation

4.1 Correctness

The file named "test_correctness.py" presents the test cases chosen for parallel testing. Some of the modules such as Interpolation and Regression were tested against Python and Matlab.

4.1.1 Scope

- Most of the methods (6 out of 8) are implemented by CFS. These methods are tested against Python libraries.
- The remaining methods are tested against Matlab.

4.1.2 Test Results

All the methods except Hermite cubic interpolation passed in parallel testing. For Hermite cubic interpolation, the coefficients from CFS and Matlab were different. The test case still fails. However, the fit does not seem to be affected. The plots below are based on the following data.

Input:
t = [1,2,4,5]
y = [2,1,4,3]

The plots from both the softwares look the same.

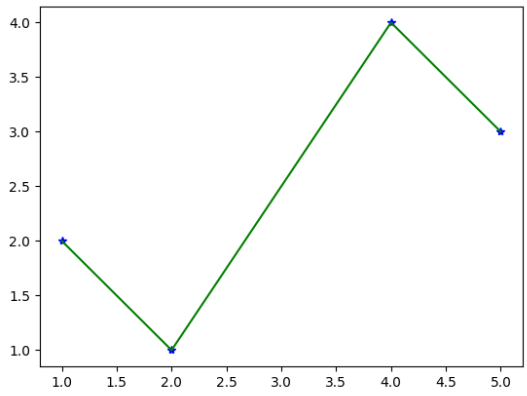

Figure 1: CFS Plot - Python wrapper

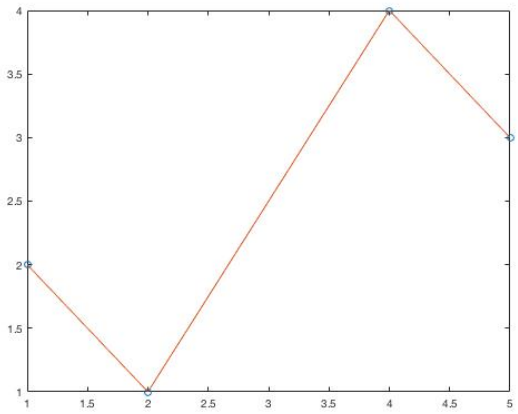

Figure 2: Matlab Plot

Both the solutions are acceptable.

4.2 Portability

CFS runs in windows and Unix. It was not tested in Linux due to lack of time and resources.

4.3 Maintainability

This test could not be conducted due to lack of resources.

5 Comparison to Existing Implementation

NA

6 Unit Testing

Unit testing is conducted in Python.

7 Changes Due to Testing

NA

8 Automated Testing

All the test cases are automated except portability and parallel testing.

9 Trace to Requirements

The following table (Table 1) shows the traceability mapping for test cases and the requirements.

Table 1: **Requirements Traceability Matrix**

Test Number	**CA Requirements**
T1(a,b)	R1, R3
T2	R1, R2
T3	R1, R3
T4	R4, R5, R6, R9, R10
T5	R4, R5, R6, R9, R10
T6	R4, R5, R6, R9, R10
T7	R4, R5, R6, R9, R10
T8	R4, R5, R6, R9, R10
T9	R4, R5, R6, R9, R10
T10	R4, R5, R7, R9, R10
T11	R4, R5, R7, R9, R10
T12	R4, R5, R7, R9, R10
T13	R4, R8, R9, R10
T14	R4, R8, R9, R10
T15	R4, R8, R9, R10
T16	R4, R8, R9, R10
T17	I R4, R8, R9, r10
T18	R4, R8, R9, R10
T19	R4, R8, R9, R10
T20	R12
T21	R13
T22	R11

Table 2: **Modules Traceability Matrix**

Test Number	**Modules**
T1	Input
T2	Input
T3	Input
T4	Input
T5	Interpolation
T6	Interpolation
T7	Interpolation
T8	Interpolation
T9	Interpolation
T10	Interpolation
T11	Interpolation
T12	Interpolation
T13	Interpolation
T14	Interpolation
T15	Regression
T16	Regression
T17	Regression

10 Trace to Modules

The following table (Table 2) 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.

11 Code Coverage Metrics

All the 3 modules mentioned in System VnV plan are tested. The code coverage metrics is 100% in all the 3 modules.