

FFT Library

Yuzhi Zhao

December 19, 2017

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
CA	Commonality Analysis
IM	Instance Module
MSE	Mean Squared Error
o_i	Output Data
e_i	Expected Output Data

Contents

List of Tables

[If there are no tables, you can comment out the above command —SS]

List of Figures

3 General Information

The following section provides an overview of the Verification and Validation (V & V) Plan for a FFT library. [\[Expand all acronyms on first usage. —SS\]](#)

3.1 Purpose

The main purpose of this document is to describe the verification and validation process that will be used to test a FFT Library. This [\[proof read —SS\]](#) document is intended to be used as a reference for all future testing and will be used to increase confidence in the software implementation.

This document will be used as a starting point for the verification and validation report. The test cases presented within this document will be executed and the output will be analyzed to determine if the library is implemented correctly.

[\[An explicit web-link to your GitHub repo would be nice. —SS\]](#)

[\[Reference your SRS document —SS\]](#)

3.2 Scope

The whole library includes four FFT or IFFT calculation functions. All tests should be applied based on this scope.

3.3 Overview of Document

The following sections provides more details about the V&V of a FFT Library. Information about verification tools, automated testing approaches will be stated. And test cases for all system testing and part of unit testing will be provided.

4 Plan

4.1 Software Description

The software being tested is a library for FFT algorithm. Users choose different FFT or IFFT functions and give proper input datas [\[data is already plural —SS\]](#) to complete a FFT or IFFT calculation. The library includes radix-2 and radix-3 FFT(and IFFT) calculation functions.

4.2 Test Team

Yuzhi Zhao

4.3 Automated Testing Approach

A unit testing framework will be implemented in both unit testing and system testing.

Script will be used to call all the test cases in test suite.

Test coverage analysis will be applied to measure code coverage.

Compiler can do syntax check automatically.

4.4 Verification Tools

1. Cutest as unit testing framework [Prior to this you should state your programming language. —SS]
2. Make as script to call test cases and execute test
3. Xcover as coverage analysis tool

4.5 Non-Testing Based Verification

Symbolic Execution

Because FFT library is based on a mathematical expression. Using Symbolic Execution can trace the path and the result can be compared with mathematical expression directly.

[The text is better for version control, and for reading in other editors, if you use a hard-wrap at 80 characters —SS]

[What tools are you going to use for symbolic execution? How are you going to do this? —SS]

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Calculation Test

Radix-2 Complex Number Calculation Function

1. T-1:Radix-2 Complex Number FFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt : Includes all the input datas. Two examples of input.txt is shown in Figure ?? and Figure ??. The floating numbers can be generated by random number generator online. The source can be reached using <http://www.meridianoutpost.com/resources/etools/calculators/generator-random-real.php>? The integer numbers can be generated using <https://andrew.hedges.name/experiments/random/>.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab FFT library. Then expectedOutput.txts are shown in Figure ?? and Figure ??.

If the numbers of data can not satisfy 2^n , program will automatically fill with 0.

[You get to decide the test case, so why not have a test case where the number of data points satisfies your constraint? You can test 0 padding separately. —SS] [Rather than figures in the Appendix, why not include the data files in your repo and then give an explicit url pointer to the documents here? —SS]

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the Mean Squared Error will be used as the algorithm. The equation is provided below:

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} (o_i - e_i)^2 \quad (1)$$

e means expected output. o means this library's output. [What does the index i mean. What is n ? I can guess, but I shouldn't have to. Also, use $MSE - SS$]

If the value of MSE is below 1% of average of input datas, then this library passed the test.

[Rather than set a specific target, you can just state that the MSE will be provided for all tests. Once all the data is together, it can be collectively judged. 1% is an arbitrary choice on your part. My guess is that the actual tests errors will be well below this. —SS]

How test will be performed:

Automated.

For validation purpose, datas should also be compared with results from normal DFT calculations as well. Do the same test as above but fill the output.txt with results from using DFT library.

2. T-2:Radix-2 Complex Number IFFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt : Includes all the input datas. The datas of input testing file can use the same datas from output.txt from T- 1 shown in Figure ?? and Figure ??.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab IFFT library.

If the numbers of data can not satisfy 2^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this IFFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

Figure 1: Complex Integer Numbers As Input

Figure 2: Complex Floating Numbers As Input

Figure 3: Output For Integer Complex Numbers As Input

Figure 4: Output For Floating Complex Numbers As Input

Figure 5: Real Integer Numbers As Input

Figure 6: Real Floating Numbers As Input

Figure 7: Output For Integer Real Numbers As Input

Figure 8: Output For Floating Real Numbers As Input

Radix-2 Real Number Calculation Function

1. T-3:Radix-2 Real Number FFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt : Includes all the input datas. Two examples of input.txt is shown in Figure ?? and Figure ?. The floating numbers and the integer numbers can be generated by on line random number generators.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab FFT library. Then expectedOutput.txts are shown in Figure ?? and Figure ??.

If the numbers of data can not satisfy 2^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

2. T-4:Radix-2 Real Number IFFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt : Includes all the input datas. The datas of input testing file can use the same datas from output.txt from T-3 showed in Figure ?? and Figure ??.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab IFFT library.

If the numbers of data can not satisfy 2^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this IFFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

Radix-3 Complex Number Calculation Function

1. T-5:Radix-3 Complex Number FFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt: Same as input.txt in T-1. Reference Figure ?? and Figure ??.

expectedOutput.txt: Same as expectedOutput.txt in T-1. Reference Figure ?? and Figure ??.

If the numbers of data can not satisfy 3^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

2. T-6:Radix-3 Complex Number IFFT Calculation Function

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt: Same as input.txt in T-2. Reference Figure ?? and Figure ??.

expectedOutput.txt: Same as expectedOutput.txt in T-2.

If the numbers of data can not satisfy 3^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

Radix-3 Real Number Calculation Function**1. T-7:Radix-3 Real Number FFT Calculation Function**

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt: Same as input.txt in T-3. Reference Figure ?? and Figure ??.

expectedOutput.txt: Same as expectedOutput.txt in T-3. Reference Figure ?? and Figure ??.

If the numbers of data can not satisfy 3^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:**2. T-8:Radix-3 Real Number IFFT Calculation Function**

Type: Functional, Dynamic, Automated

Initial State: None

Input:

input.txt: Same as input.txt in T-4. Reference Figure ?? and Figure ??.

expectedOutput.txt: Same as expectedOutput.txt in T-4.

If the numbers of data can not satisfy 3^n , program will automatically fill with 0.

Output:

output.txt : Includes the output datas using the input data computed by this FFT library.

TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:

Same as above.

[You did not include the tests suggested by Alex and Isobel in class. All of your tests rely on parallel testing with a comparison to Matlab. That is fine, but if you make an error, those tests won't help much with tracking it down. You should also include tests where you know the solution theoretically. Generate your initial data with a sine function and then verify that you get the expected frequency back. A more complex test would be like that given on the Wikipedia page for FFT in the figure at the top of the page (https://en.wikipedia.org/wiki/Fast_Fourier_transform) —SS]

5.2 Tests for Nonfunctional Requirements

5.2.1 Speed Comperation [spell check! —SS] Test

1. T-9: Compare Calculation Speed with DFT calculation

Type: Dynamic, automated, Manual

Initial State: None

Input: input.txt [proof read —SS]

Output: Time

How test will be performed:

Manually compare the time with the time using DFT Library.

[This test does not give me any useful information. It just is the time for one test. What would be more interesting is to verify the number of inputs in the file and find the time for each. How does the execution time change as the problem size increases. You could use $n = 100, 200, 400, 800, 1000$ etc. You could do the same exercise with Matlab and compare your performance to Matlab. —SS]

5.2.2 Loading Library Test

1. T-10:Under Win X86 platform

Type: Functional, Dynamic, Manual

Initial State: None

Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to an C Language compiler.

Output: output.txt

How test will be performed: Manual

[What determines success on this test? confusing? —SS]

2. T-11:Under Mac OS platform [spell check! —SS]

Type: Functional, Dynamic, Manual

Initial State: None

Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to an C Language compiler.

Output: output.txt

How test will be performed: Manual

[What determines success on this test? confusing? If you have make working properly, you can repeat all of your tests on any platform. —SS]

3. T-12:Different Compilers Under The Same Platform

Type: Functional, Dynamic, Manual

Initial State: None

Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to different compilers including different versions and different languages.

Output: output.txt

How test will be performed: Manual

[Same comments as above. —SS]

5.3 Traceability Between Test Cases and Requirements

Since CA does not include requirements part, the Test Cases will be relevant to IM.

T-1, T-2, T-3, T-4 all relevant to IM1 in CA.

T-5, T-6, T-7, T-8 all relevant to IM2 in CA.

6 Unit Testing Plan

6.1 Input Check Test

1. T-13: Check Numbers Of Input Data

Type: Functional, Dynamic

Initial State: None

Input: List of Input Datas

Output: Check whether List.length equals 2^n or 3^n according to Radix

How test will be performed: Automated Unit Test

2. T-14: Fill Input With 0

Type: Functional, Dynamic

Initial State: None

Input: List of Input Datas

Output: List (The List.length must equal to 2^n or 3^n according to Radix)

How test will be performed: Automated Unit Test

3. **T-15: Check whether the input data type is the required data type corresponding to the called Function**

Type: Functional, Dynamic

Initial State: None

Input: List of Input Datas

Output: True or False (True means the data type is right, False means the data type is wrong.)

How test will be performed: Automated Unit Test

[This is a confusing test. Are you talking about a test where you have invalid types in your input? If so, what do you expect to happen. There should likely be a requirement for an exception. Do you want to make this the expected output for this test? —SS]

6.2 Calculation Test

1. **T-16: Complex Number Multiplication Check**

Type: Functional, Dynamic

Initial State: None

Input: $(1 + 2i) * (2 + 3i)$

Output: $(-5 + 7i)$

How test will be performed: Automated Unit Test

2. **T-17: complex number Addition Check**

Type: Functional, Dynamic

Initial State: None

Input: $(1 + 2i) + (2 + 3i)$

Output: $(3 + 5i)$

How test will be performed: Automated Unit Test

3. **T-18: $e^{\frac{-2\pi ki}{N}}$ Calculation Check**

Type: Functional, Dynamic

Initial State: None

Input: $k = 1, N = 4$

Output: -i

How test will be performed: Automated Unit Test

[In your design I do not recall a module that hides the ADT for complex numbers. Is this something you are going to implement? If you are going to use an existing complex number library, you don't really have to test it, although you should make that fact explicit. —SS]

6.3 Partition Test

1. T-19:Radix-2 Partition check

Type: Functional, Dynamic

Initial State: None

Input: $[0, 1, 2, 3, 4, 5, 6, 7]$ means $[x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7]$

Output: $[0, 4, 2, 6, 1, 5, 3, 7]$ means $[x_0, x_4, x_2, x_6, x_1, x_5, x_3, x_7]$

How test will be performed: Automated Unit Test

2. T-20: Radix-2 Partition check

Type: Functional, Dynamic

Initial State: None

Input: $[0, 1, 2, 3, 4, 5, 6, 7, 8]$ means $[x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]$

Output: $[0, 3, 6, 1, 4, 7, 2, 5, 8]$ means $[x_0, x_3, x_6, x_1, x_4, x_7, x_2, x_5, x_8]$

How test will be performed: Automated Unit Test

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

[You could have used a symbolic constant for your 1% error value. If there are no constants, you don't need this section. —SS]

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.

[Your project is really quite simple. I think you should be thinking about some kind of usability testing, or something else extra, so that you can get the most out of this project. —SS]

[If you don't have a usability survey, then you shouldn't have this section. —SS]