Plotter Software

*Iteration 6: Implementation of bug fixes and enhancements*

# 1 Objectives and Testing Goals

Originally, we had split up the three packages of the program up between the members of the team for testing, one package for each of us. We had planned on doing unit testing on each of them as much as we were able. After that, we had planned to do Functional Testing on the program to try and locate as many issues as we could. Since we didn’t really have any standards by which to measure any performance tests, we decided against doing any and obviously, there weren’t any regression tests to run except to make sure the program still ran after we had finished all the updates.

# 2 Testing Results

Include subsections as appropriate to your testing strategy.

**2.1 Unit Testing**

Unit Testing proved inconclusive. The code that was tested was not easily decoupled, and therefore unapproachable. Unit testing is very important, and had we been a part of the development, modules would have been developed to allow unit testing during development.

**2.2 Integration Testing**

No integration testing was needed, as the application was integrated to start, and never was in divided into subsystems.

**2.3 System Testing**

This was the focus of our testing plan. Most tests were done from a black-box approach with the team testing the application against the expected results of its functions. All bugs reported were from this phase of testing and all enhancements were suggested from this phase of testing as well.

**2.4 Performance Testing**

No performance testing was needed as each instance of this application was an executable run on a single machine. If anything can be said of performance, it is that the application did not use up too many resources (roughly 180 MB of memory on tested machine).

**2.5 Regression Testing**

All functionality was tested of the application again by each of the team members after bug fixes and enhancements were implemented and the software worked as expected.

# 3 Summary of Bug Fixes

Sub-section descriptions of individual bug fixes (one per enhancement) that were implemented. Include operational description of the bug fix, high level software impact of the bug fix to the code base, and amount/type of testing associated with the bug fix. Include screen shots if deemed desired/appropriate by the team.

**4.1 The program originally displayed a “-0” when you would try to calculate an integral from right to left. We added an error message to prevent the program from trying to calculate the integral this way.**

**4.2 The program gave the wrong approximation for the Simpson Integral. Changes were made to the method to correctly display the approximation.**

**4.3 The program gave the wrong approximation for the Trapezium Integral. Changes were made to the method to correctly display the approximation.**

# 4 Summary of Enhancements

Sub-section descriptions of individual enhancements (one per enhancement) that were implemented. Include operational description of the enhancements, high level software impact of the enhancement to the code base, and amount/type of testing associated with the enhancement. Include screen shots if deemed desired/appropriate by the team.

**4.1 The ExportDataPanel save button originally seemed to do nothing even though it saved the data. We added code to display a confirmation message to the user that the data was saved and then exit the ExportDataPanel.**

**4.2 The Save Image button in the menu bar originally opened a dialog box whose confirmation button said “Open.” We changed the code for the dialog box to say “Save” instead.**

**4.3 A Readme file has been included with basic instructions for typing functions into bar.**

# 5 Significant Challenges

One of the greatest challenges we faced as a group was to change our original plan to do any significant amount of unit testing. Some parts of the program were too complicated to understand either because there weren’t any comments to explain what certain data represented or what a method was actually doing or because the classes were too interconnected with the rest of the program to easily test independently.

Unfortunately, we didn’t understand this entirely before we developed our testing strategy and therefore wasted a significant amount of time trying to develop unit tests. However, once we redirected our efforts towards a more integration and system testing approach, it became a lot easier to identify bugs or potential enhancements in the program.

# 6 What We Learned

We learned it is difficult to test other people’s code. We tested that the Plotter code was functioning and working correctly. The thing was, the code base was so deep in places and confusing that at times we didn’t really know what the code was doing or where it was going or what this method did or what that call was supposed to do. We had to change our approach from unit testing to a more functional black-box approach. Decoupling the code base enough to white-box test it was very difficult. We also learned that planning tests through all phases of development results in more robust testing. Writing tests after a piece of software is already complete is much more difficult that testing throughout the project. Finally we learned about the importance of commenting the code. With no comments, we didn’t know why something was written the way it was or why it was included in the first place. Having a better idea of the why would have went along way for us when we tested it.

# 7 What We Would Improve

An improvement we would make would be to check for modularity before creating unit tests. That way we would have had a better sense of the functionality of the Plotter program. We would have been able to see the independent and interchangeable modules parts. We would have known what each module was intended to do and then we could have designed our tests to suit that function better. This would have helped in our unit tests but it wasn’t feasible to spend any more time than we did. A better idea from the start may have been to perform our functional black-box tests right at the start.