**1. Introduction**

The Plotter programs was given us as a project to help us learn how to test as thoroughly, efficiently, and professionally as possible. Our goal is to eradicate any bugs, errors, or faults that we can by implementing a test plan in the best manner possible.

After testing, this program should be able to accomplish the following:

- Graph functions in Cartesian 2D, Cartesian 3D, and Polar coordinate systems.

- Graph the corresponding derivative function of the one already displayed.

- Calculate the integral of the function, using the Gaussian, Simpson, or Trapezium methods.

- Change the color scheme of the graph or background panel.

- Save the graph as an image file, or export the data in a different format.

- Zoom in or out or center on different portions of the graph.

**1.1 Test Plan Objectives**

The objectives of this test plan are

- To describe what unit tests and integration tests need to be performed.

- To enhance communication between members of the team.

- To set a schedule in place for the completion of these tests.

**1.2 Test Scope**

The unit tests will cover all classes and their dominant methods

Functionality and acceptance testing will cover

- The integral panel and calculations

- Graphing derivatives (for functions that have no derivative as well)

- The functionality of the buttons in the menu bar

- Zoom/centering functions in all coordinate systems.

**1.3 Major Constraints**

- Time is a major constraint on this project. Everything must be completed by 12/3/2016.

- Communication is also a constraint since all members of the team live in different areas of the

country and cannot have any face to face meetings.

- Another constraint is that no team member is an experienced software tester. Some aspects of

testing might be overlooked due to inexperience.

**2. Test Plan**

**2.1 Software To Be Tested**

The software to be tested is the Plotter software.

**2.2 Testing Strategy**

We will use JUnit and ECLemma to unit test each of the classes and their dominant methods and

analyze code coverage of the tests. Afterwards, we will use functionality testing to make sure

the various functions of the program. Creating and documenting the functionality tests will

probably take the most time and effort, so more effort should be applied there. Creating the

JUnit tests, on the other hand shouldn’t be too difficult, so minimal effort would be applied

here.

**2.2.1 Unit Testing**

The names of the classes to be unit tested are LineData, Point3D, Polygon3D, Renderer3D,

ZBuffer, Colorpanel, DigitTextField, ExportDataPanel, FunctionTextField, IntegralPanel,

Visualizer, AdvancedCalculator, Calculator, MathTree, ParseFunction, TNode. Most of the focus

should be given to Visualizer, Calculator, and Renderer3D and their methods.

**2.2.2 Acceptance Testing**

In the acceptance testing phase, we will develop a series of tests to demonstrate the

functionality of the software in a black box testing approach. In case of any failure of the

program, we will create a list of defects to be used for creating fixes or future implementations.

This section will test the various interfaces of the system. Namely, the ExportDataPanel, the

IntegralPanel, the Colorpanel, and the main interface.