# Analysis Section

## 1. Feasibility Study

My client is Rob Ettridge, head of the Maths department at the College of Richard Collyer. Part of my client’s job involves introducing the concept of complex numbers[[1]](#footnote-1) to students studying the Further Maths AS course. This in turn involves the drawing of Argand diagrams.

An Argand diagram is a method of plotting complex numbers and loci[[2]](#footnote-2) on a Cartesian grid[[3]](#footnote-3), by using the x axis for the real part and the y axis for the imaginary part. Drawing and manipulating Argand diagrams on a whiteboard is repetitive and inaccurate, so my client has requested a system that can draw these diagrams for him.

It will act both as an aid for teaching the subject in classrooms, and as a learning tool for students (e.g. for checking homework answers); however, nothing about its design would restrict it to a school setting. The program will have a simple input for complex equations, and a grid on which they are plotted, which can be zoomed and panned.

The program may need to store settings and preferences. Writing plots as images or serialisation of plots could also be useful, although is not integral to the working of the system. One possible solution uses the following tools:

* Python 3.
* PyQt, using the QCanvas or QGraphicsView widget to plot functions.
* cx\_Freeze to compile the Python to an executable.

## 2. Formal Investigation

### Interview

**Q: Can you tell me more about what you want the system to do?**

A: It needs to draw Argand diagrams clearly, so I can use it with a projector to demonstrate the diagrams to students. If they can’t tell what’s being displayed, I might as well just use the whiteboard.

**Q: Who will use the program?**

A: Maths teachers and students.

**Q: Does it need to run on any particular hardware?**

A: The program needs to run on the Windows 7 computers in the maths department, and it needs to be easy to understand when displayed on an overhead projector. Since it’s also supposed to be a learning tool for students, preferably it should work with as many different machines as possible.

**Q: What are the main issues with the whiteboard method?**

A: Drawing Argand diagrams can be repetitive and inaccurate, and repeatedly drawing them for students wastes time. If I actually make a mistake in a diagram, students can become confused.

**Q: Does the program need a save or an undo feature?**

A: A save feature could be useful – I could prepare a question before lesson – but it shouldn’t be too important. I’m not sure the ability to undo would be that useful.

**Q: What types of loci does the program need to be able to plot?**

A: It needs to be able to plot circles, perpendicular bisectors[[4]](#footnote-4), rays[[5]](#footnote-5), and offset versions of each. There need to be labels on the offset versions to show how the numbers in the equation relate to the graph. Also, it should be able to shade in inequalities and it would be nice if it could calculate simultaneous equalities, or just plot multiple graphs at once.

**Q: How will the user enter the loci?**

A: If possible, a text input would probably be the easiest to use.

### Observation Record

I have recorded my observations of a current lesson on Argand diagrams in the following table.

|  |  |
| --- | --- |
| **Process** | **Issues** |
| Drawing axes. | * On the first try, the teacher realised that the pen had run out of ink, and had to find another. * The axes weren’t completely straight. |
| Drawing a circle. | * Without a compass, the circle wasn’t perfect. * The circle’s centre didn’t fall in the exact correct place. * The circle didn’t cross the axes in the correct quadrants, meaning it had to be redrawn. |
| Drawing a perpendicular bisector. | * The line wasn’t completely straight. * The line didn’t cross the axis or meet the circle in the exact correct place. |
| Drawing a ray. | * The line wasn’t completely straight. * The angle wasn’t correct. |
| Labelling points and angles. | * The teacher’s handwriting wasn’t always legible from a distance. |

### Investigation of Argand Diagrams

In addition to my observation of a current lesson on the subject, I have performed personal research into Argand diagrams. I have found that the diagrams I need to plot take the form

where is one of , , , , , , or a constant. Inequalities may be described by replacing the with one of , , , or . However, in addition to the diagrams I have been asked to implement, this format allows for some other notable loci. I have shaded these loci in the following matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | A ray from , angle , when . | **Varies.** | **Archimedes’ spiral.** | **Two opposite rays.** |
|  | A circle centre , radius , when . | **Varies.** | Perpendicular bisector of and . |
|  | The point . | **If , then all points, otherwise none.** |

When , this equation represents an Archimedes’ spiral centred on . It may be possible to implement this locus when , but for other values it becomes much more complex.

This equation represents two rays, one from each , both cast away from the midpoint. must not equal , otherwise it is equivalent to the locus of all points: .

## 3. Description of the Current System

The current system is the traditional whiteboard method of teaching, as my client has not found any other software packages suitable for a classroom setting. The teacher draws Argand diagrams on the whiteboard with dry-wipe markers, which can fade. The software will be used alongside the whiteboard method, to prevent the teacher from having to draw complex diagrams – instead the diagrams are generated on a computer, and displayed by an overhead projector.

## 4. System Requirements

The system needs to provide a windowed GUI on the Windows operating system, but using Python and existing libraries could support multiple platforms. It should also be capable of running without dedicated graphics hardware, as many cheap student-oriented computers (particularly laptops) have limited graphics capabilities.

## 5. Identification of the Prospective User(s) and their skills

|  |  |  |
| --- | --- | --- |
|  | **Maths Teachers** | **Maths Students** |
| **Overview** | Teachers will use the software in the classroom in order to save time spent drawing Argand diagrams themselves. | Students will use the software as a learning tool – e.g. to check answers to their homework, or simply for experimentation. |
| **IT Competence** | Teachers are generally comfortable using education software. | Some students may not be completely comfortable with education software. Especially considering they will not completely understand complex numbers either, the software must be very accessible in order to be useful to them. |
| **Training Needs** | Some teachers may require help becoming familiar with the interface. | Students may require help becoming familiar with the interface, and may also require help understanding the inputs and output. |
| **Special Needs** | Whilst the software will have the ability to zoom in on diagrams, it may not be suitable for users with particularly poor eyesight.  Some users may have disabilities that make typing more difficult, and rely on software that switches between inputs using the tab key. Therefore, input tab indices should be properly implemented for all inputs.  Other disabilities are unlikely to affect users’ ability to use the program more than they affect the ability to use a computer in general. | |
| **Documentation Needs** | Users may need help with learning the format for entering equations. | |

## 6. Constraints/Limitations

* The software must plot complex loci and points.
* The software must plot inequalities for each type of loci.
* It must be possible to scale and pan the diagram.
* The software must run on at least an Intel x64 Windows 7 PC.
* The GUI must be legible when displayed by an overhead projector.

## 7. Data Sources and Destinations

Data will be input by the user in the form of text equations, properties of equations, and diagram preferences. Equations will be displayed in a list, as well as being plotted on the diagram itself. Diagrams can be saved to files, which will store all of this data.

## 8. Data Volumes

There will be no limit to the number of equations that can be input into one diagram. However, only one diagram may be loaded at once, and each equation may only have one set of properties.

## 9. Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Description** | **Type** | **Validation** |
| Equation | A text input used to enter equations for drawing. | Text | Takes the form , where is one of , , , , , , or a constant, and is one of , , , , or . |
| Colour | Represents the colour used to draw the associated locus. | RGB colour code | In range #000000 – #FFFFFF. |
| Inequality Alpha Transparency | How transparent inequality regions should be. | Percentage | In range 0% - 100%. |
| Point Labels | Whether labels should be drawn for points. | Boolean | Can be true or false. |
| Axis Labels | Whether labels should be drawn for axes. | Boolean | Can be true or false. |
| Label Font Size | The font size used to draw labels. | Number | In range 12pt - 28pt. |
| Line Thickness | The line thickness plots are drawn with. | Number | In range 1px - 10px. |

## 10. Entity Relationship Model

Plot

Colour

Equation

Argand

Diagram

Preferences

**Program**

Transform

## 11. Object analysis

### Argand Diagram

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Member Type** | **Description** |
| preferences | Preferences object reference | Stores a reference to the object with the current settings. |
| plots | Array of Plot object references | A list of all the points and loci which should be drawn on the diagram. |
| translate | Coordinate | The point on the diagram around which the display should be centred. |
| zoom | Number | Amount to scale the diagram around the translation centre. |

### Preferences

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Member Type** | **Description** |
| label\_points | Boolean | Whether labels should be drawn for points. |
| label\_axes | Boolean | Whether labels should be drawn for axes. |
| font\_size | Integer | Font size to use when drawing labels. |
| stroke | Integer | Thickness in pixels of plotted lines. |
| window\_size | Coordinate | Stores the last known width and height of the window. |

### Plot

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Member Type** | **Description** |
| equation | String | Stores the original equation to allow it to be changed easily. |
| type | Integer | Indicates the basic type of plot of which this is a transformation. |
| transform | Matrix | Stores the transformation from the basic type of plot to this specific instance. |
| color | RGB colour code | Colour to use when drawing the plot. |
| alpha | Number | How opaque inequality regions should be. |

## 12. Objectives

**1 - 6.** The system will be able to plot the following loci on an Argand diagram:

Re

Im

**4.** The perpendicular bisector of and :

Re

Im

**1.** A circle centred on the origin:

Re

Im

**2.** A circle centred on :

Re

Im

**3.** The perpendicular bisector of and the origin:

Re

Im

**5.** A ray from the origin:

Re

Im

**6.** A ray from :

1. The system will also shade in inequalities for each of these loci with a translucent colour fill.
2. Values of and will be labelled on the diagram. This will be configurable.
3. The system will plot individual complex numbers as points:
4. The system will display multiple plots on one diagram.
5. Diagrams can be saved to and loaded from .arg files.
6. It will be possible to zoom in on the diagram by scrolling the mouse wheel while hovering over it, or by moving a slider.
7. It will be possible to pan around the diagram, by dragging the mouse over it, or by using a coordinate input.
8. All plots will have configurable colours – either from a set collection of swatches or with a colour picker tool.
9. All inequalities will have configurable alpha transparency.
10. The program will not require a mouse to use – all inputs will have tab indices set correctly.
11. The program will have a text input for entering complex equations.
12. The program will be capable of converting strings into equations which can be plotted on the diagram.
13. The program will have built-in information (i.e. a tooltip / popup) about how to enter equations in the correct format.
14. The program will display current plots in a list, where they can be edited or removed.
15. The minimum resolution of the program will be, at most, 1024x768.
16. The diagram will be legible on a 1024x768 resolution projector from the back of an average-sized classroom.
17. The program will run on at least an Intel x64 Windows 7 PC.
18. The program will have a configuration menu which allows the following to be altered:
    1. Whether labels are displayed for points.
    2. Whether the axes are labelled.
    3. The font size used for labels.
    4. The line thickness used in the diagram.
19. The program will save the diagram configuration state to the .arg file.
20. The program will remember the last window size used, and revert to it on start-up.

## 13. Appraisal of the Feasibility of Potential Solutions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Solution A** | **Solution B** | **Solution C** |
| **Programming Language** | **Python 3**  Python is a very clean, modern language, which I am very familiar with. It has a wide range of libraries, and several convenience features built in. | **Python 3**  ″ | **C++**  C++ is an older, robust language. C++ has fewer convenience features built in than Python, and can be harder to compile for multiple platforms. |
| **GUI** | **PyQt 4**  PyQt is a python binding for the Qt library. It supports all of the features of Qt, although in some places is inconvenient to use due to being a port. | **GTK+**  GTK+ is the open source GUI library used in the GIMP project. It isn’t as well maintained as Qt, but also has a large feature set, and good cross-platform support. | **Qt**  Qt is a very popular cross-platform GUI framework, with a very large feature set. |
| **Drawing** | **QGraphicsView**  The QGraphicsView widget, built into Qt, can display QGraphicsItem instances, which can be inspected and manipulated. | **pycairo**  pycairo is a Python drawing library with support for integration with GTK+. Unlike in Qt, once shapes have been drawn to the surface, they cannot be changed. | **QGraphicsView**  **″** |
| **Deployment** | **cx\_Freeze**  A cross platform tool for compiling Python scripts into executable binaries. | **cx\_Freeze**  **″** | **Microsoft Visual C++**  A C++ compiler for Windows. Other platforms would need another compiler. |

## 14. Justification of Chosen Solution

Chosen Solution: A

I have chosen to develop with Solution A for several reasons:

* Qt is a much better framework for drawing graphics than GTK+, as it provides QGraphicsView. QGraphicsView is capable of storing objects representing the output, whereas pycairo can only rasterise graphics.
* Although Qt is native to C++, Python offers the library with a much cleaner language, more suited to developing a small project.
* Also, I am already very familiar with Python 3.
* cx\_Freeze will make deployment on multiple platforms much easier than compiling the project separately with separate C++ compilers.

1. A complex number is a number composed of a real and imaginary part: where [↑](#footnote-ref-1)
2. A locus (plural loci) is a collection of points which satisfy some common criteria. [↑](#footnote-ref-2)
3. A Cartesian grid (named after French polymath René Descartes) is a two-dimensional grid with two orthogonal axes that meet at the origin. The axes split space into four quadrants. [↑](#footnote-ref-3)
4. The perpendicular bisector of two points is a line perpendicular to the line segment between the points, which passes through the line segment’s midpoint. [↑](#footnote-ref-4)
5. A ray is a line that has a starting point but no endpoint; it continues forever in one direction. [↑](#footnote-ref-5)