Mathematica provides much utility and versatility, such as inbuilt data visualisation, data manipulation and analysis, dynamic functionality, and symbolic and numeric computation*.* Most of this inbuilt functionality is quite user-friendly and can be found in the Mathematica documentation; typically, numerous examples are provided within the documentation, and some possible issues are outlined there. Also, it’s quite easy to create dynamic objects via the inbuilt commands Manipulate or DynamicModule. Control objects, such as sliders, locator panes, and input fields, can then be used on dynamic variables, and when there is a change in the dynamic variable the dynamic objects which contain that dynamic variable will be updated. However, it can be hard to alter the behaviour of the control objects such as setting the 1D slider on a different metric(?) *(Slider increments add 10 points into the slice rather than increasing the slice height by 0.1 for example)* Also, Mathematica can be difficult to debug depending on the size of the project as there are no line numbers and there isn’t a way to backtrack evaluations to see how the state of local variables changes throughout the evaluation. *(PyCharm has this functionality)*

New methods:

Can test the Orthonormalisation Algorithms in Technical Note for example (?)

(Test algorithms that are concerned with mouse inputs?).

The penguins’ notebooks all utilise the function SliceDynamic. This function typically accepts data in the form of a matrix where the second last column details the name of the group and the last column details the group index, although this can be changed. Additionally, the initial slice height and the slice range need to be specified as function arguments. The user is then presented with an interactive(?) in which the control objects appear on the left and the slice appears on the right. The user can change the slice via the locator pane, which changes the projection matrix; a slider, which changes the slice height; and the input field, which changes the slice center. The user can also change the appearance of the plot by zooming into the center or changing the point size with sliders provided. Also, the current projection can be displayed instead of the slice and the current projection matrix can be displayed via checkboxes. The Slice Tour\* notebooks display some other functions such as ProjectionPlot, Projected2DSliderPlot, and VisualiseSliceDyanmic. ProjectionPlot is very similar to SliceDynamic, except it only displays projections and VisualiseSliceDynamic allows you to discern which points exist in the slice and which don’t. However, ProjectedLocatorPlot\*\* displays the interactive slightly different. Several locator panes are displayed above the plot and each one corresponds to a row in the projection matrix but behaves the same as ProjectionPlot. (this may be easier to implement in another language (?) )

\*may get renamed

\*\*Was Projected2DSliderPlot, this name is more accurate.