**Introduction:**

The Ellipse Calculation project is focused on the modernization of the legacy code used in the Radiocommunication Bureau of the ITU. The project is about re-writing a DLL called *ellebu.dll*, which is used in a more extensive application called GIMS, to calculate the optimal ellipse that encompasses a set of test points. The goal of this project is to produce a new version of the DLL, which will have a similar interface and the same functionality, but with improved performance and accuracy.

The GitHub repository contains the new version of the DLL: <https://github.com/cfortunylombra/Dynamic_Ellipse>

**Technical details:**

The new version of the DLL will be implemented using **C++** and it will use the same mathematical calculation to determine the optimal ellipse that encompasses a set of test points, which is applied in determining the elliptical beam of a satellite antenna by generating gain contours covering a required service area defined by a set of earth station locations.

The new version of the DLL uses several libraries to perform the calculations. These libraries are:

* **Constants**: This library contains all the constant parameters used in the calculations such as Pi, Rad, NMax, EarthRadius, GEOAlt\_EarthRad\_rat.
* **Polar2RectangularLibrary**: This library takes in theta and phi as input and returns the x, y, and z coordinates of the point in rectangular coordinates. It depends on the Constants library.
* **TransformationMatrixLibrary**: This library takes in the orbital position, the latitude, and the longitude of the antenna and returns two matrices: the matrix to convert from satellite coordinates to earth coordinates and the matrix to convert from earth coordinates to satellite coordinates. It depends on the Constants library.
* **Earth2SatLibrary**: This library takes in the matrix to convert from earth coordinates to satellite coordinates and a point in earth coordinates and returns the corresponding point in satellite coordinates. It does not depend on any other library.
* **Sat2EarthLibrary**: This library takes in the coordinates of three points in satellite coordinates and returns the coefficients of the ellipse in earth coordinates. It does not depend on any other library.
* **NormVectorLibrary**: This library takes in a point in satellite coordinates and returns the normalized coordinates of the point in the z direction. It does not depend on any other library.
* **EliminateLibrary**: This library takes in the number of points and the rectangular coordinates of the points in satellite coordinates and returns the coordinates of the remaining points that are not colinear. It does not depend on any other library.
* **PointsCheckEllipseLibrary**: This library takes in the number of points, the minimum value of the semi-major axis, the coefficients of the ellipse, the minimum value of the semi-minor axis, the center of the ellipse, and the coordinates of the remaining points and returns the semi-major and semi-minor axes and the area of the ellipse that encompasses the points. It does not depend on any other library.
* **MinimumSemiMinorAxisLibrary**: This library takes in the number of points, the semi-major axis, the semi-minor axis, the center of the ellipse and the coordinates of the remaining points and returns the semi-major and semi-minor axes, and the area of the ellipse that encompasses the points. It depends on the TrigonometricEqSolveLibrary.
* **MinimumSemiMinorAxis2Library**: This library takes in the number of points, the semi-major axis, the semi-minor axis, the center of the ellipse and the coordinates of the remaining points and returns the semi-major and semi-minor axes, and the area of the ellipse that encompasses the points. It does not depend on any other library.
* **TrigonometricEqSolveLibrary**: This library takes in the coordinates of two points, two angles, and the minimum value of the semi-major axis and returns the angle of rotation and the final angle. It does not depend on any other library.
* **3PointsEllipseLibrary**: This library takes in the number of points, the coordinates of the remaining points, the semi-major and semi-minor axes, and the center of the ellipse and returns the area of the ellipse that encompasses the points, the semi-major and semi-minor axes and the center of the ellipse. It depends on the Solution3LinearEqLibrary.
* **Solution3LinearEqLibrary**: This library takes in the coordinates of three points and returns the coefficients of the ellipse that passes through these points. It does not depend on any other library.
* **2PointsPSOUTLibrary**: This library takes in the number of points, the coordinates of the remaining points, the semi-major and semi-minor axes, and the center of the ellipse and returns the semi-major and semi-minor axes and the area of the ellipse that encompasses the points. It depends on the DeterminantLibrary.
* **DeterminantLibrary**: This library takes in the coordinates of two points and returns the coefficients of the line that passes through these points. It does not depend on any other library.
* **OrientationLibrary**: This library takes in the coefficients of an ellipse and returns the angle of rotation, the angle of inclination, and the angle of twist. It depends on the Constants library.
* **CenterGravityLibrary**: This library takes in the number of points, the orbital position, the latitude, and the longitude of the points and returns the latitude and longitude of the center of gravity of the points. It depends on the UpdateLongitudeLibrary.
* **UpdateLongitudeLibrary**: This library takes in the number of points, the orbital position, and the longitude of the points and returns the updated longitude of the points. It does not depend on any other library.
* **AddPointingErrorLibrary**: This library takes in a point in satellite coordinates and the pointing error in tangent and returns the point with the added pointing error. It depends on the Constants library.
* **AddPointingRotationalErrorLibrary**: This library takes in the number of points, the coordinates of the remaining points, the angle difference, the minimum value of the x coordinate, the pointing error in tangent, the cosine and sine of the pointing error in tangent, the sine and cosine of the station keeping error, the pointing error flag and the rotational error flag and returns the coordinates of the remaining points with the added pointing and rotational errors. It depends on the Constants library, the AddStationKeepingErrorLibrary, the AddPointingErrorLibrary, and the AddRotationalErrorLibrary.
* **AddRotationalErrorLibrary**: This library takes in a point in satellite coordinates and the cosine and sine of the rotational error and returns the point with the added rotational error. It does not depend on any other library.
* **AddStationKeepingErrorLibrary**: This library takes in a point in satellite coordinates and the station-keeping error in angle and returns the point with the added station-keeping error. It does not depend on any other library.
* **AngleDiffStationKeepingLibrary**: This library takes in the center of gravity, the number of points, the coordinates of the remaining points, the cosine and sine of the cosine and returns the angle difference. It depends on the Polar2RectangularLibrary and the Constants library.
* **ConvexLibrary**: This library takes in the coordinates of the points and the number of points and returns the coordinates of the new points that form the convex hull of the points. It depends on the FindColinearPointLibrary.
* **DisplacementEquatPlaneLibrary**: This library takes in the displacement vector of the equatorial plane and returns the displacement vector in the Z and Y directions. It depends on the Constants library.
* **EdgeEllipseLibrary**: This library takes in the angle of rotation and the angle of inclination and returns the points of the ellipse. It depends on the Constants library.
* **EllipseViewLibrary**: This library takes in the number of points, the latitude and longitude of the remaining points, the semi-major and semi-minor axes, and the center of the ellipse and returns the view of the ellipse. It depends on the Polar2RectangularLibrary and the Constants library.

The **TopCalculation** files combine all the backend libraries and contains the “Calc” function. The function "Calc" has the following input parameters:

* status: a long integer that is passed by reference, it is used to indicate the status of the function, the values returned can be 0 if the function finished correctly, or a non-zero value if there was an error.
* pointing\_error: a float value representing the pointing error of the system.
* rotational\_error: a float value representing the rotational error of the system.
* station\_keeping\_error: a float value representing the station keeping error of the system.
* minimum\_axis: a float value representing the minimum axis of the system.
* orbital\_position: a float value representing the orbital position of the system.
* n\_points: a float value representing the number of points of the system.
* points\_lat: an array of float values representing the latitude of the points.
* points\_long: an array of float values representing the longitude of the points.
* COSCOS: a two-dimensional array of float values representing the cosine of the cosine of the points.
* COSSIN: a two-dimensional array of float values representing the cosine of the sine of the points.
* SIN: a two-dimensional array of float values representing the sine of the points.

The function returns a structure called "Calc\_struct" which contains the following output parameters:

* boresight\_lat: a float value representing the boresight latitude.
* boresight\_long: a float value representing the boresight longitude.
* maj\_axis: a float value representing the major axis.
* minor\_axis: a float value representing the minor axis.
* area: a float value representing the area.
* orientation: a float value representing the orientation.

In addition, two other libraries (**Loop.h/.cpp** and **Loop1.h/.cpp**) that merge the backend libries. These two libraries are imported in the **TopCalculation** files.

The main contributions of the project are:

* The implementation of a new version of the DLL in C++.
* The use of libraries to perform the calculations, which improves the organization and maintainability of the code.
* The modernization of the legacy code.
* The possibility to improve the current functionality of the DLL.

The new DLL will have the same interface as the current one, the same input, and output parameters, and the same functionalities. However, the new version will be optimized for performance and will take advantage of modern C++ libraries and features.

**Current status:**

The project is currently in its implementation phase. Only test units for the following libraries have been developed:

* **CenterGravityLibrary**
* **UpdateLongitudeLibrary**
* **Polar2RectangularLibrary**
* **TransformationMatrixLibrary**
* **Earth2SatLibrary**
* **Sat2EarthLibrary**
* **NormVectorLibrary**
* **EliminateLibrary**

All the libraries have been implemented and are currently undergoing testing to ensure that they produce the correct results. There are currently no open issues or bugs that have been identified. Upcoming developments include further testing (the rest of the unit tests need to be performed) and verification phase and integration of the new DLL with GIMS. The project is expected to be completed by a further intern.

**Upcoming developments:**

* The next step is to test the new version of the DLL and compare the results to the current version. For this, the unit tests need to be completed and the system tests need to be developed.
* Once the tests are completed, the new version of the DLL will be integrated into GIMS and deployed to production.
* The last step is to document the new version of the DLL and its usage.

**Conclusion**:

The project aims to modernize the legacy code of the *ellebu.dll* used in GIMS by re-implementing it in C++. The new version of the DLL uses several libraries to perform the calculations, which improves the organization and maintainability of the code. The new version of the DLL has a similar interface and functionality as the current version, but the results may differ and the differences will be justified. We encourage feedback and suggestions for further improvements.