1 Programming Projects

Hereafter all Tables, Figures, Sections, and Algorithms refer to those in the paper of the cubic MARS method.

I. Write a driver program to reproduce results in Table 5.2 and Figure 5.2.

II. For the set of the modules in the matlab package, we define a partial order dependence as follows. A module \( A \) is said to be dependent on the module \( B \), written \( A \leftarrow B \), iff \( A \) calls \( B \). Then the set of modules with the dependence relation as this partial ordering is a directed graph. Plot the diagram of this graph and check that your graph is a directed acyclic graph. How many connected components are in your graph? You can ignore all modules of the forms \texttt{test*.m}, \texttt{generateResults*.m}, and \texttt{plot*.m}.

**Hint:** you might want to use the \texttt{grep} command under the \texttt{ubuntu} operating system. What is the most efficient way to complete this assignment? Can you write a shell/C/C++ program to automate this task?

III. The universe center of our matlab package is clearly the module \texttt{trackInterface.m}. For Definition 4.3, list the line numbers that roughly correspond to each of the steps (CubiMARS-1,2,3,4,5). What modules in II are used in each step? Which principles should guide your design of the modules in II?

IV. Write a subroutine \texttt{testFactoryCubicMARSn.m} to describe the test in 5.3.2. Then use \texttt{testTrackInterface.m} to reproduce results in Table 5.6 and Figure 5.8.

V. Consider an ellipse with its center at \((0.5, 0.5)\), its radii as \((0.3, 0.2)\), its rotation angle \( \phi = \frac{\pi}{3} \). Write a subroutine \texttt{testEllipseArcGeoInfo.m} to test the subroutine \texttt{ellipseArcGeoInfo.m} at 16 points on the ellipse with its parameter angle as \( \theta = 0 : \frac{\pi}{2} : 2\pi \). You can check numerically that the unit vectors as output of \texttt{ellipseArcGeoInfo.m} is sufficiently close to the exact solution. Alternatively, you can plot the calculated unit vectors along with the ellipse and check the results visually. In the latter case, you must also check numerically that the length of each calculated vector is indeed one.

**Hint:** You have to check both positive and negative orientations of the ellipse.

VI. The module \texttt{intersectSplinegons.m} in the package is a variant of the splinegon clipping algorithm (Algorithm 1 in Section 4.4). Write a module \texttt{clipSplinegonWithPolygon.m} to implement Algorithm 1. Note that you must compute the intersection points in your implementation. In other words, the input of your module consists of an arbitrary splinegon and an arbitrary simple polygon. You must also write a test file \texttt{testClipSplinegonWithPolygon.m} to verify that your program works for all possibilities of the relative position of the two input.

The above six assignments weigh 5, 20, 10, 15, 10, 40 points, respectively. The total is thus 100 points.

**Important:** All your solutions must be typeset in \LaTeX{} except for that of II. You should print out your solutions (along with your matlab code) and submit a hard copy in class.

Although you can discuss this assignment with other students, you must write up your solution by yourself. Copying another student’s solution or matlab will earn you a zero as the final grade of this class.

2 Extra credits

Based on the correctness, clarity, and elegance of your solutions, I will select three winners for this project. The first-place winner will get an additional 20% credit and the other two second-place winners will get an additional 15% credit.

You may also earn extra credit if you

- implement a module in the package with less entropy and/or more generality;
- suggest a global or local design that is better than the current one, but has the same or more general functionality.