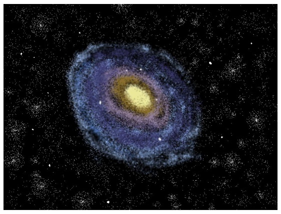

Team Andromeda

Requirements Specification Addendum

March 3, 2020

Accepted as modified requirements for the project:

For the client: ______________________ ______
Signature *Date*

______________________ ______
Signature *Date*

For the team: ______________________ ______
Signature *Date*

Clients - Dr. Audrey Thirouin and Dr. Will Grundy
Mentor - Isaac Shaffer
Members - Matthew Amato-Yarbrough, Batai Finley, Bradley Kukuk, John Jacobelli, and Jessica Smith

Table of Contents

1. Solution Statement Revision

1.1 Solution Statement Revision Explanation

In our Requirements Document we outlined that the graphical user interface for the Forward Model would be built in Python. After several weeks into development we found that this task was greater than anticipated. The existing library (Licht-cpp) had multiple problems when compiling due when attempting to interface with the shared object. As a group we came to the conclusion that it was no longer feasible to create the graphical user interface using python and have instead pivoted to using C++.

1.2 Revised Solution Statement

Due to the requirements given and the existing code base, the software that will be developed must be compatible with the existing code base. The existing code base was developed using C/C++. With that in mind the software will be built using C/C++, to keep continuity with the existing project.

For the GUI implementation for the Forward Model, the programming language we will be using is C/C++. The framework that will be used to develop the GUI will be QT, a C/C++ framework that is used to create cross platform compatible GUI. This will allow the user to use this software anywhere and at any time.

For the acceleration of the Forward Model we will be creating a Triaxial Ellipsoid class to the existing API using C/C++. The Triaxial Ellipsoid class will be dependent on the existing API's library and will use all of the same libraries.

The API will be well documented to allow users to maintain and update the code base. The design of the API will be modular to allow for new functionality to be added in the future.

2. Hamiltonian Monte Carlo (HMC) Addendum

2.1 HMC Addendum Explanation

A functional requirement we outlined in our Requirements Document was to “Produce Best Fit Parameters Using HMC”. To achieve this requirement, we decided that the best means of solving this was through the use of the RStan package as it provided the HMC functionality that was needed. However, due to the way with which RStan interfaces with external C++ code, many modifications to the lict-cpp API would be needed as well. These modifications would require the templating of all of the .h/.hpp files present within the licht-cpp API to make them compatible with the RStan package. As a result, completion of this requirement would require more development time than we have.

3. Project Requirements

3.1 Functional Requirements

There are major modifications we will be making to the functional requirements in our project. This includes the replacement of the HMC algorithm with a nonlinear minimizer known as Amoeba.

FR1. Amoeba Minimization

A new key requirement of this project is the implementation of an Amoeba minimizer module. The Amoeba module will serve the purpose of providing sets of estimated parameters for our clients. Additionally, the module will display the results of the minimization to the command prompt. Lastly, this module will save generated results to an external file. The requirements of the module are explored in further detail in the following sections.

1. Produce estimated parameters for predicted light curves
The main functionality of the Amoeba module will be to estimate parameters for predicted light curves. Initially, our clients will choose which Forward Model parameters they want to estimate by supplying their ranges to the Amoeba module. This module will then provide a multi-dimensional minimization of a chi-square function using these estimated parameters. This will then produce the minimum chi-square comparison of predicted light curves produced by the Forward Model against observed light curves. Consequently, this will determine estimated values of our client's chosen parameters with their corresponding chi-square value.

2. Display minimization results to the command line
Our clients will be able to see the results of the Amoeba module from the command line. These results will show the starting values for estimated parameters, the number of function evaluations that occured and the final values for each of the parameters the module arrived at. Additionally, data plots for the observed light curve and predicted light curves will be generated during the execution of the Amoeba module, and will be displayed after completion of the minimization.

3. Save minimization results to an external file
Our clients would also like for the Amoeba module to produce a file, such as a .csv file, that would contain final optimized parameters produced by the Amoeba module.

3.2 Performance Requirements

PR1. Amoeba: Providing Sensible Input

The runtime of the Amoeba module will depend on several factors determined by the user. These include the number of parameters they want to estimate, the number of iterations they want the Amoeba Module to execute, and the number of starting points they supply. Therefore, a minimum runtime cannot be determined due to runtime depending on user inputs. The user must be aware of what they are looking for and consider this when utilizing the Amoeba module.

3.3 Environmental Requirements

ER1. Cross-Platform Compatibility

With these modifications to the project requirements, it is important that any new implementations maintain the capability to compile and run on Linux, Windows and Mac. This is an environmental requirement met by the team that worked on the previous iteration of this project. Our clients asked that we keep this functionality with the solution we implement in order to maintain code usability and shareability.