ESF exchange Grant : final report

gaia

Maroussia ROELENS

Universitat de Barcelona

25/06/2013

Introduction

As part of my second year at the Ecole Nationale Superieure des Mines de St Etienne (institute of science and technology, graduate and doctoral school, in France), I perform a three month internship at the department of astronomy and meteorology of the University of Barcelona (Spain), from May 2013 to August 2013. This internship is related to the UB team participation to Gaia, a cornerstone mission of ESA*, approved in 2000, whose main goal is to chart the most precise and largest 3-D map of our galaxy, the Milky Way. The aim of the stage is to study how the Gaia mission will observe open clusters by using the GOG (for Gaia Object Generator) simulator, a software system able to simulate the expected results of Gaia. For this, GOG has to be be tested, improved and applied to a list of open clusters. A second part of the work consists in setting up a Graphical User Interface (GUI) in order to make the use of GOG simulator more user-friendly.

So as to prepare this internship, a preliminary two weeks visit at the UB has been planned, from May 13th to May 27th. The purpose of this visit was, firstly, to get familiar with the Gaia mission in general, and more precisely with data simulation effort so as to prepare the mission. Then the idea was to focus of GOG simulator input, running and output, in order to start working on open clusters and GOG interface as soon as possible. At the end of the stay, precise work programme and guidelines should be defined for the following stage.

I – GOG simulator

- **Simulator principle**

As mentioned before, GOG simulator (for Gaia Object Generator) is in charge of simulating data similar to those resulting from GAIA observation all along the mission. As outputs of GOG, the user can be provided, for each simulated object, a certain set of parameters: position, motion, magnitude, spectra, composition…

All in all, GOG way of working can be sum up as follows. There are two possible “modes” when using this simulator: the “universe model” mode, and the “user sources” mode. With the “universe model” mode, the user selects which type of sources he wants to simulate, in which area of the sky. Then the simulator generates the sources required for the “part of sky” simulated to be as realistic as possible, but it does not simulate exactly the real sky. The idea here is to obtain something equivalent to what Gaia will really observe. With the “user sources” mode, the user can provide a list of celestial objects, with their physical and astronomical parameters, that is to say a text file with a precise format. The idea here is to use GOG as a kind of translator : the stars and objects simulated are already known, that is to say that their characteristics can be found in existing catalogues ; by providing to GOG these parameters, the user can see how Gaia will observe these sources.

Since the needed input have been provided to GOG, simulation can be run, and at the end of the process the user can be provided a certain amount of data concerning the sky region simulated, the same data Gaia will effectively provide during the mission (astrometry, photometry, spectrometry...). There are two types of output: the combined data, that is to say the end-of-mission data, resulting from all the transits performed (here GOG simulates the 80 transits Gaia will do); and the epoch data, related to a limited number of transits. Before running the simulation, the user can choose which type of output he needs.

- **Why a Graphical User Interface for GOG simulator?**

Indeed, GOG needs many input parameters to be specified so as to run in a proper way. Since then, they were specified in a GOG configuration file, that is to say an XML* file with all the input parameters values. Therefore, when learning how to run GOG simulator, new users had to familiarize

with this GOG configuration file format, and then write directly the file corresponding to the simulation they expected. The general structure of GOG input is not very simple: understanding the meaning of choices that can be made and the links between each of them takes some time. Furthermore, the use of GOG is not really “user friendly”, not only due to this way of specifying input parameters, but also because running the simulation by itself requires launching the corresponding jar file through a command line, or using the CNES (Centre National d'Etudes Spatiales, the French government space agency) web interface in case you have a granted access to this platform. All in all, starting with GOG can be a bit “startling” for beginners. That is the reason why adding a Graphical User Interface (hereafter GUI) to the existing simulator can be considered now. One of the tasks I had to achieve during my three-month internship was to develop such a GUI for the GOG simulator, so as to make its use more user-friendly. First of all, the idea of the GUI is to handle easily the configuration of an execution, preventing the user to deal with the XML directly.

During this two-week stay, GUI requirements have been precisely defined, providing guidelines for the stage following this preparation phase. This first version had to enable the user to precise all the configuration parameters by himself, but also had to be able to read external configuration files, that is to say existing XML configuration files, in case the user already has the configuration file corresponding to the simulation he needs and just wants to run the simulator. The user should be likely to save all parameters specified through the GUI for further use. Finally it had to include the validation functionality.

- **Work performed during the stay**

First of all, I spent several days reading documentation and studying examples of configuration and properties files, so as to learn and familiarize with GOG input. During this period, I also looked for much information about Gaia mission in general, and for astrophysical and astronomical definitions so as to understand the meaning of all input parameters. Before anything else, I needed to understand how celestial objects are characterised and studied, and will be observed by Gaia, for the simulation to make sense for me. Then I focused more precisely on GOG input and all the constraints related to those parameters: format constraints, range of values, dependency between some parameters… Thus I could build a schema of the general structure, which was needed to implement the GUI.

After this documentation period, I started working on the GUI by itself. Whereas all the other programs related to GOG simulator have been developed under Eclipse environment, I used the NetBeans platform. Indeed this tool was more adapted to the building of the GUI, as it allows a rapid and instinctive user interface development.

The first step of the GUI conception was to build its general structure, so as to have a “skeleton” of the interface. The next step is to catch the information provided by the user thanks to the GUI, so as to record them. At that point the GUI is a kind of local project, that is to say that there is no link yet with the GOG simulator program. Besides, I started working on the validation of data provided by the user, modifying the code for basics errors and exceptions to be handled by the program (unexpected empty numeric fields, out-of-range values, non-existing paths to files...). All in all, at that point a first local version of the GUI was available. It was not fully operational, according to the GUI requirements, but it was likely to catch information provided by the user so as to store them, and showed all the fields needed for GOG configuration file. It is totally independent for GOG routines, algorithms and standards.

The second phase of my job was to modify the first “skeleton” of GUI so as to adapt it to the GOG format and standards. Indeed, many existing Java classes and methods in GOG simulator have been used so as to load existing XML configuration files, and store information provided by the user through the GUI under XML format (a choice based on the current way GOG simulator is run).

Another GUI requirement was to fulfil all the needed fields with defaults values: at that point, an XML default configuration file has been created, and methods have been implemented for the GUI view to be integrally initialized according to the default file read. We also started working on the GUI design, so as to make it more pleasant to use, and for its appearance to be more professional.
Thus, at the end of the stay, a first GUI prototype was available, easy to use, likely to load and create XML configuration files -which could directly be used to run GOG simulator at it has been done since then- and handling the main errors which can occur. Some views of the first GUI prototype are represented in fig. 1.

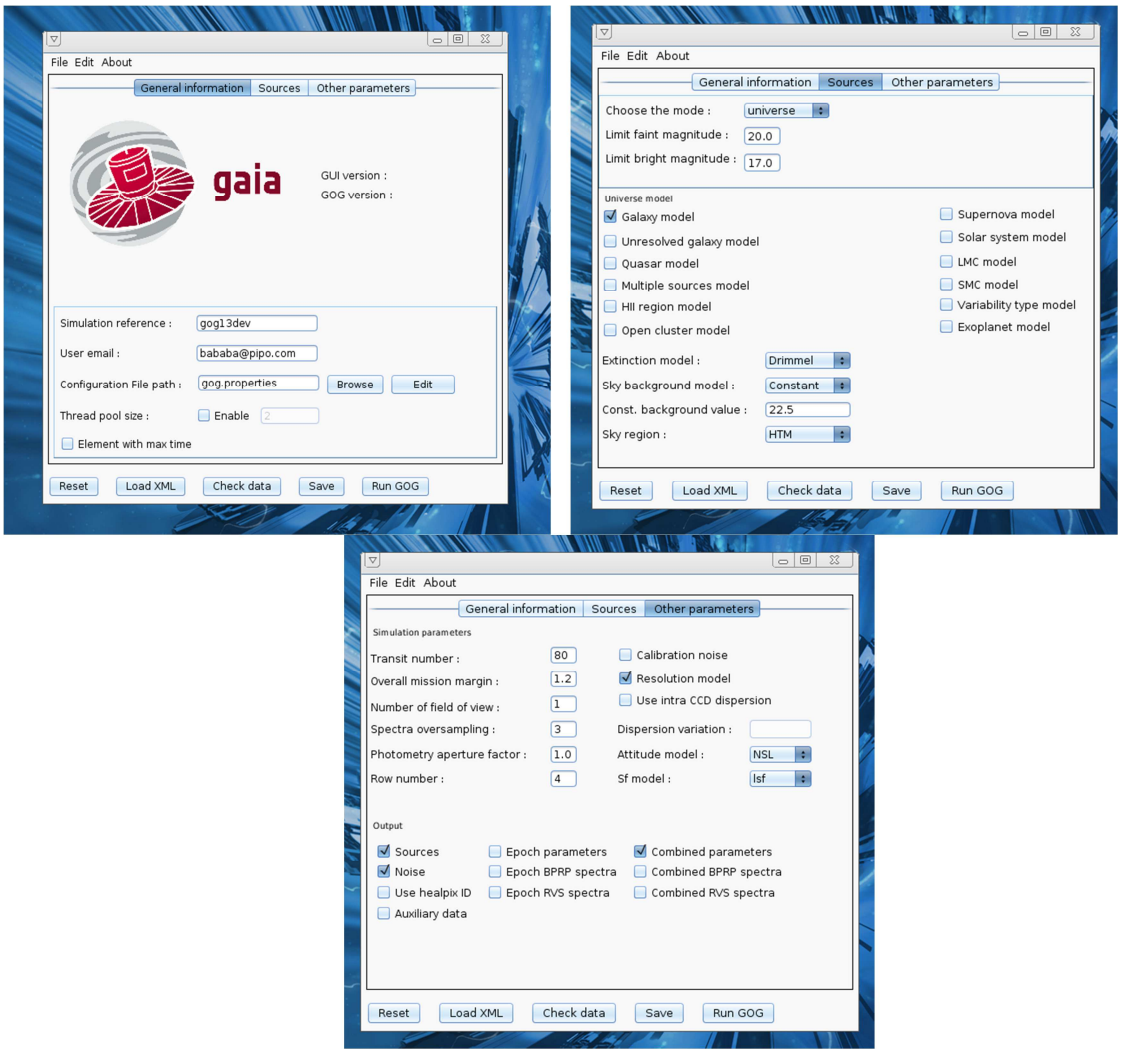

Fig. 1: First GUI prototype views

During the following stage, the next step concerning GUI development will be to make the link between the GUI and GOG simulator: the idea here is to prevent the user from dealing with GOG code in any way. Thus the user should be allowed to run GOG from the GUI, and GOG output should also be managed using the interface. Besides, a relevant point will be to enhance errors handling (which is very basic in the first prototype), and to add a kind of “help” with pertinent information concerning GOG running, so as to make the use of simulator ever more user-friendly.

II – Open clusters

- **Why open clusters?**

As mentioned before, GOG simulator is likely to simulate several types of celestial objects: stars, planets, supernovae… The latest version of the GOG simulator includes for the first time the possibility of simulating the Gaia observations of open clusters. An open cluster is a group of up to a few thousand stars, formed from the same molecular cloud, bounded to each other by gravitational attraction. All these stars roughly have the same age, distance, radial velocity and proper motion. During its five-year mission, Gaia will observe thousands of open clusters in the Milky Way, which will provide a large amount of information about the galaxy history and evolution. Indeed, open clusters position in the galaxy did not change a lot since their creation: as they are quite young in comparison with other isolated stars, they did not have time to roll away from their birth place. Thus, their age and chemical composition combined with stellar evolution models tell us about the composition of the galaxy at different ages in given positions.

Let's focus on the way open clusters are simulated. They are generated in a semi-empirical basis. For a given cluster, the age, metallicity, total mass, radius, distance and velocity are taken from the Dias et al. 2002 catalogue. The cluster is placed at its real position in the sky. Then, using these parameters, stars members of the cluster are simulated one by one, using the Padova luminosity function, and assuming a lognormal IMF (according to the Besançon model). The members of the cluster are drawn until the total mass is reached. Stars are randomly distributed around the mean position of the cluster on the sky, following a Gaussian distribution whose sigma corresponds to the core radius of the cluster, and using the tidal radius as a cut-off. For each star, the velocity is computed using the sum of velocity of the cluster, plus a peculiar velocity randomly drawn from a 3D-Gaussian. Thus, data available consist in a list of open clusters with their characteristics on the one hand, and for some of these clusters (not for all of them) a list of their members with their characteristics on the other hand. Then, the file containing stars in cluster's characteristics will be used as an input for GOG, so as to simulate these special stars.

As this feature of the simulator has not yet been validated the goal of the stage will be to simulate these objects and analyse the results obtained in order to perform such validation. For this purpose some specific open clusters will be simulated and the resulting outputs (simulated stars, astrophysical parameters...) will be checked in depth. This analysis will validate the behaviour of GOG concerning open clusters simulation by comparing input data with output data, and controlling whether the output data is generated according to specifications.

- **Work performed during the stay**

First of all, the purpose of this two-week stay was to control the coherence of available data: open clusters' characteristics on the one hand, and physical properties of stars in cluster on the other hand. Thus, focusing on a given open cluster, we can compare cluster characteristics with its members' properties, controlling whether open clusters' members are generated in the expected way. We also use the WEBDA database, a huge source of information about open clusters, to perform this validation. The open clusters chosen for the study is Melotte 22 (the Pleiades cluster).Note that validation of GOG output should be performed latter, during the stage.

Using a Python code, we calculated some relevant parameters for each star of the Pleiades, particularly right ascension, declination, proper motion components, distance and radial velocity. Then we made statistics so as to perform the validation, that is to say several plots (histograms of distance, position and velocity components, scatters of alpha against delta and muAlpha against

muDelta...) and calculations (means of position and velocity for the members of the cluster, total mass of stars in the cluster...).
Figures 2 show a set of relevant plots for Melotte 22.

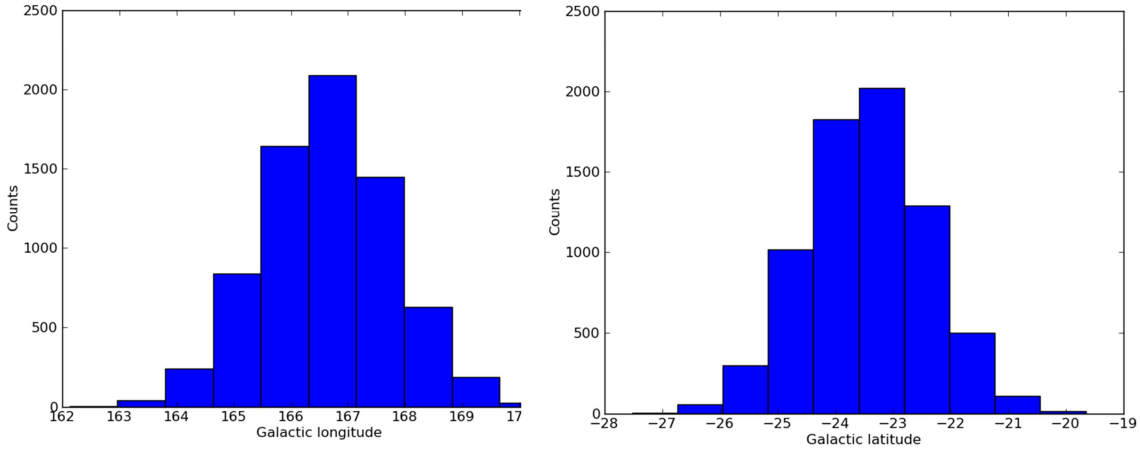

Fig. 2 a: Histograms of galactic longitude (left) and latitude (right) for Melotte 22

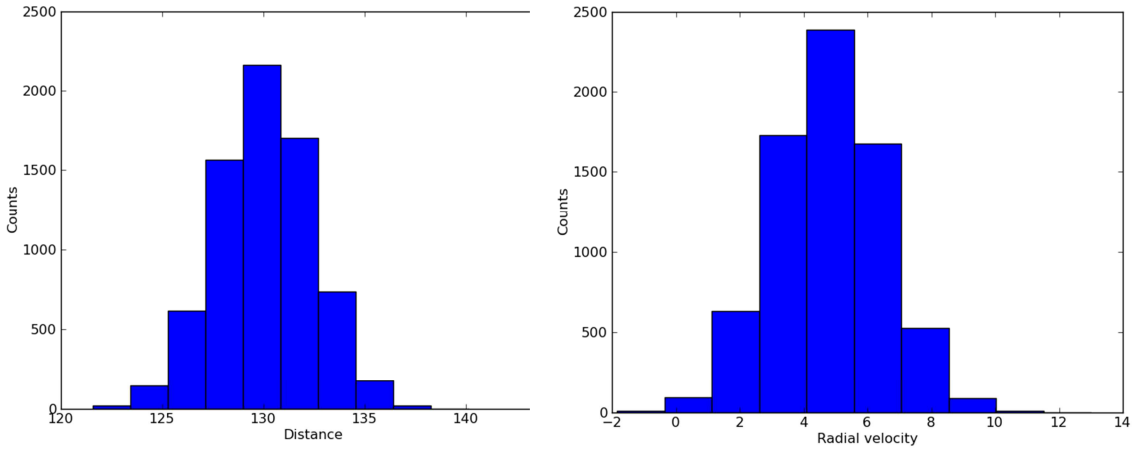

Fig. 2 b: Histograms of distance (left) and radial velocity (right) for Melotte 22

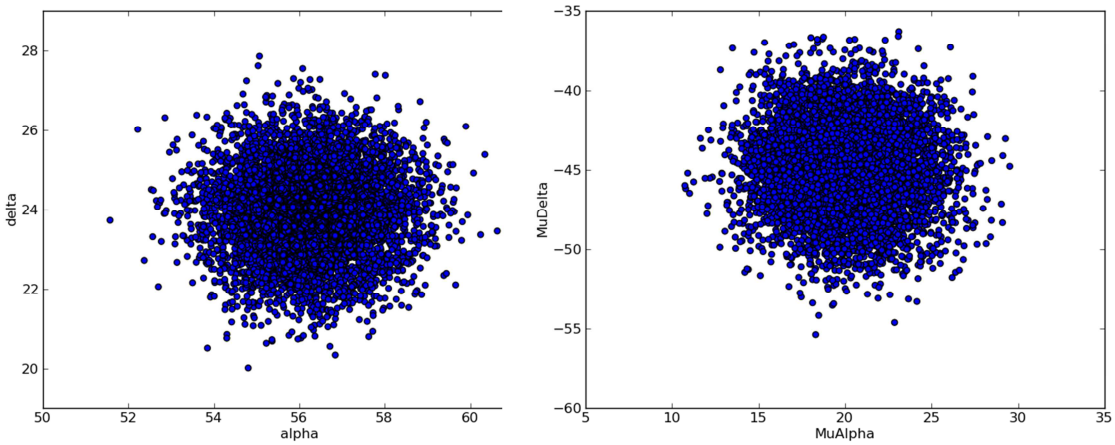

Fig. 2 c: scatters of alpha against delta (left) and muAlpha against muDelta (right) for Melotte 22

The following table sums up all the results obtained for Melotte 22.

Melotte 22			
Characteristics	**WEBDA data**	**Clusters list data**	**Data derived from stars in cluster data**
Alpha (degrees)	56.75 (03 47 00)	56.75 (03 47 00)	56.2569820877
Delta (degrees)	24.1167 (24 07 00)	24.1167 (24 07 00)	23.9648449885
Lgal (degrees)	166.571	166.63	166.628164985
Bgal (degrees)	-23.521	-23.47	-23.4605005385
Distance (pc)	150	130	130.055635931
Metallicity	?	-0.030	-0.030*
logAge	8.131	8.08	8.08*
logMass		3.540	3.53230954068**
Core radius (degrees)		2.3	2.38793776274***
MuAlpha (mas/year)		19.88	20.0911460954
MuDelta (mas/year)		-45.00	-44.9128618321
Radial velocity (km/s)		4.80	4.78570365188

Table 1: Data comparison for Melotte 22

*: All the members of a given cluster have the same age as the latter
**: Logarithm of the sum of the members' masses
***: Here the core radius is calculated as the standard deviation of distance for stars in the cluster

All in all, stars are effectively distributed around the expected mean position of the cluster, as can be seen in Table 1: both alpha, delta, longitude, latitude and distance means calculated from stars in cluster match with the expected values found in the WEBDA database and in the clusters list. Histograms of distance, latitude and longitude (fig. 2a and 2b) evidence that member's positions follow a Gaussian distribution around the mean position. The standard deviation of distance is equal to the core radius of the cluster indeed. Similarly, mean radial velocity of stars in cluster match with the assumed radial velocity of the cluster, and fig. 2b (right) validates that stars velocities have been computed according to specification. We can also validate the mass distribution, as the sum of masses of stars matches with the assumed total mass of the cluster. Finally both muAlpha and muDelta means matches with assumed values. So the conclusion is that cluster members generation can be validated. Anyway, it may be interesting to adapt the Python code used for this study (which indeed is quite basic) so as to perform automatically the same plots and tables as presented here, for all available clusters. Thus, errors when creating the clusters list could be easily detected and corrected.

During the following stage, the next step of the study is to run GOG simulation, simulating the chosen cluster (Melotte 22), so as to compare input and output data. The idea here is to control whether GOG properly simulates the cluster, with the proper position, velocity, distance... To complete the work a set of open clusters, increasingly distant from the Sun, will simulated and the outputs will also be analysed. The goal of this study is to analyse the potential contribution of Gaia to the analysis of open clusters depending on its distance.

Finally, if time permits, a library of simulated open clusters, including input data and output data resulting from simulation, will be built. This library will be made available to the community as a tool for potential users of Gaia data to assess what they will be able to do with real data resulting from the mission.

Conclusion

As a conclusion, this two-week stay was a very useful first step to the following stage itself. After this preliminary period, precise working program, requirements and guidelines had been defined, both concerning validation of open clusters simulation and the development of the GOG interface. It also allowed me to learn more about Gaia mission in general, and more particularly about data simulation. Through the documentation work I could do, I enhanced my knowledge about astrophysics. I could discover the way UB team contribute to data simulation effort, and the organisation of team work not only inside the Barcelona team, but also within the whole coordination unit in charge of data simulation. Furthermore, these two weeks allowed me to start focusing on the aim of the stage itself, by developing a first GOG interface prototype, and performing a part of validation of open clusters simulation. All this is a solid foundation for further work to be performed during the following stage.

References

- E. Antiche, R. Borrachero, F. Julbe, X. Luri, 2013, *GOG 12.1 User Guide*, UB
- Annie C. Robin, Céline Reylé, Frédéric Arenou, Carine Babusiaux, Artur Latorre i Musoll, Xavier Luri, Paola Sartoretti, Paolo Tanga, Eric Grux, 2013, *Universe model 12.0 overview*, DPAC
- ESA science and technology : Gaia
  http://sci.esa.int/gaia
  The European Space Agency webpage about Gaia mission
- Gaia at Barcelona
  http://gaia.am.ub.es/
  Webpage of the Barcelona group, with documentation and information on the tasks they are involved