QUANTIFYING MULTI STAR SYSTEMS IN STELLAR GROUPS: CALIBRATION OF BINOCS FOR THE GAIA ERA USING SYNTHETIC PHOTOMETRY OF BP/RP SPECTRA

by

John Nguyen

Submitted in partial fulfillment of the

requirements for Departmental Honors in

the Department of Physics and Astronomy

Texas Christian University

Fort Worth, Texas

May 6, 2024

QUANTIFYING MULTI STAR SYSTEMS IN STELLAR GROUPS: CALIBRATION OF BINOCS FOR THE GAIA ERA USING SYNTHETIC PHOTOMETRY OF BP/RP SPECTRA

Project Approved:

ABSTRACT

Precise assessment of binary star formation remains uncertain, yet it wields substantial influence on our understanding of galaxies' concealed dark matter. As it stands now, there are few methods to efficiently detect binary star systems in stellar groups, one of which is the Binary Information from Open Clusters Using Spectral Energy Distributions (BINOCS) fitting algorithm. With the launch of the European Space Agency *Gaia* spacecraft and its third data release, spectral data is more widely accessible for use. This project's endeavor involves enhancing the BINOCS software with *Gaia* spectroscopic data, empowering the identification of binary stars among one million celestial entities. We modified the BINOCS software to perform online data collection, have a built-in Graphical User Interface (GUI), updated for Python3, and implement modern software engineering principles while retaining algorithm accuracy. The intention of this project serves as an interdisciplinary project between my major in computer science and minor in astronomy, where I can leverage my knowledge in computer science and apply it to a real-world problem.

Table of Contents

INTRODUCTION

A binary star system is a system in which two stars are gravitationally bound and orbit a common center of mass. Valuable information may be obtained from studying binary star systems such as information about stellar masses, orbital dynamics. Detecting whether a star system in stellar group is a binary system historically has been difficult and time-consuming. This problem was addressed with one method introduced by Dr. Ben Thompson: the Binary Information from Open Clusters using Spectral Energy Distributions (SEDS) or BINOCS software algorithm (Thompson et al. 2021). The algorithm compares magnitudes from several photometric filters to synthetic star spectral energy distributions to determine component masses of binary and single star systems. It compares the observed flux over wavelength against a single model and binary model. These models use several bands/filters, 5 optical, 3 near infrared, and 4 infrared which are *UBVRI*/*ugriz*, *JHK*, and B1-B4 respectively. The system is determined to be binary or single based on how well the observed fluxes over wavelength match the models. For each star in a cluster, the magnitude and magnitude uncertainty for each filter had to be obtained manually and formatted for use in the software. This limited the number of clusters the software could run on. The clusters that have been classified through BINOCS already are NGC 188, NGC 1960, NGC 2099, NGC 2158, NGC 2420, NGC 2682, and NGC 6791. There are still millions of star systems left unclassified due to the data collection limitation. The BINOCS software, written in Python 2.7 remained untouched since 2015. However, with the launch of the European Space Agency *Gaia* spacecraft (Gaia Collaboration 2016) in 2013 and its third data release in 2022, blue and red photometer (BP/RP) spectra data for 219,224,825 sources have been made easily accessible through an officially supported Python3 library: GaiaXPy (Ruz-Mieres 2022). This data release opens the door for new classifications to be made through

BINOCS. The BINOCS software will need to be enhanced to perform online queries to *Gaia*, redesigned to work with the new data input, and refactored for Python3. Additionally, in this project, software engineering principles will be applied to make the software maintainable and extensible as well as more easily accessible to the public.

<u>DATA COLLECTION AND VERIFICATION OF GAIAXPY</u>

GaiaXPy is a Python library (Ruz-Mieres 2022) to facilitate handling Gaia BP/RP spectra as distributed from the *Gaia* archive. It comes with features to query the archive for certain photometric filter system and generates synthetic photometry which produces magnitude, flux, and flux errors for each filter (Gaia Collaboration, Montegriffo et al. 2022). Since the third data release arrived in 2022, this library is relatively new. Before using this library with BINOCS, its synthetic photometry must be tested against real photometry to prove that the library computations are reliable. To compare it against a known set of data, I used a `.fits` file containing all the BINOCS cluster members that were tested before with their respective *Gaia* source ID (Gaia Collaboration 2023). These *Gaia* source IDs were then passed to the GaiaXPy generate function along with the Sloan Digital Sky Survey (SDSS) photometric system parameter for *ugriz* filters (Fukugita et al. 1996). After receiving the magnitudes, flux, and flux errors, the magnitude error had to be calculated through this formula:

Filter Magnitude Error = 2.5 log (flux error / flux)

I added this calculation along with the magnitude, flux and, flux errors into a single Python Pandas data frame and merged it with another data frame containing the original data. I computed the differences, the mean and standard and removed outliers. Outliers were considered to be magnitude differences over 2 standard deviations. I repeated this outlier removal process for 10 iterations after recomputing each standard deviation for each iteration. I then plotted the delta plot to compare if the data floated near a zero difference.

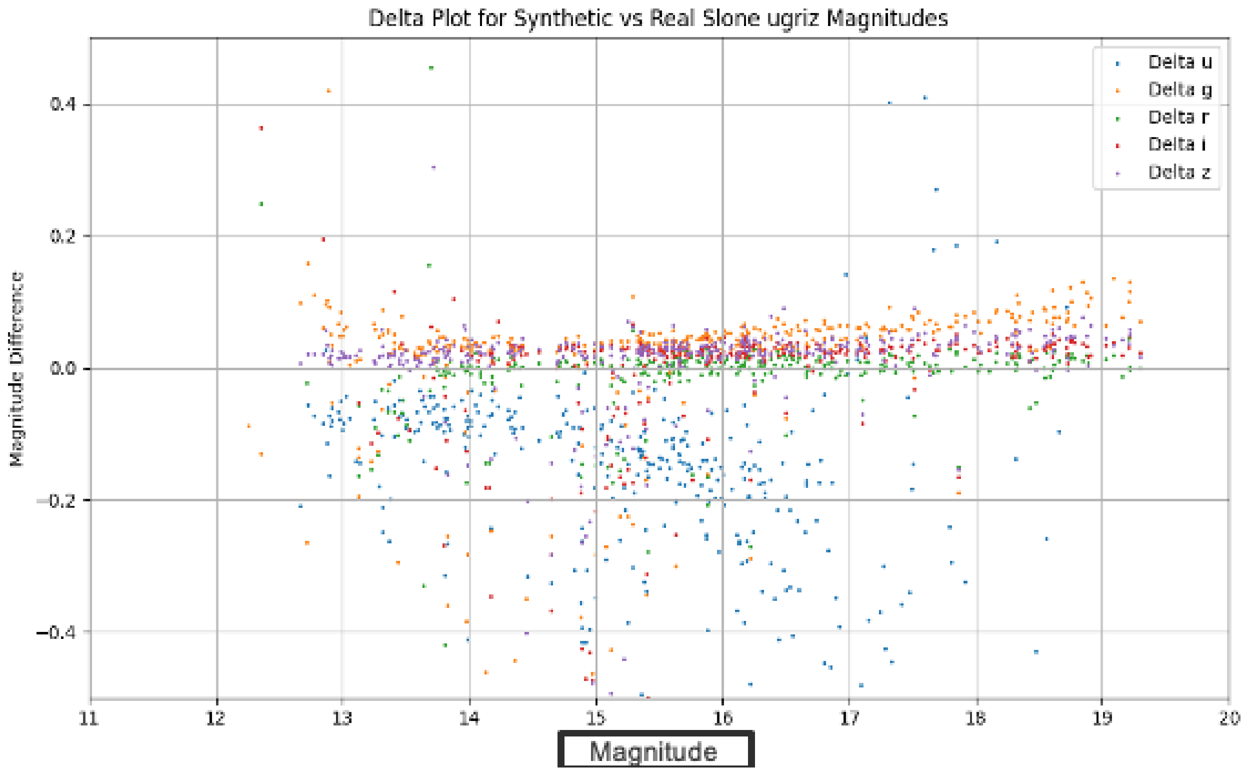


Figure 1: Delta Plot for Synthetic vs Real Slone ugriz Magnitudes

Here in **Figure 1**, we can see that the differences lie between -0.5 and 0.5. These differences are expected since the data from *Gaia* comes from space whereas the data from SDSS is from the ground. The atmosphere may influence the magnitude, amplifying this difference. I have considered this sufficient in accuracy and deem the library reliable given our limited data set of known magnitudes. However, it is still possible that for a different set of clusters, the magnitude could widely vary, but I am not able to check it. With this assumption of reliability, we can proceed to use this library with BINOCS.

SOFTWARE ENHANCEMENT 1: DATA COLLECTION ACROSS CATALOGS

Querying the *Gaia* archive has been made easy through GaiaXPy. However, a new problem arises in which we need to query other astronomical catalogs for the remaining filter sets: *JHK* and B1-B4. The catalogs we will need are the Two Micron All Mass Survey (Skrutskie et al. 2006) denoted 2MASS and the from the Wide-field Infrared Survey Explorer (Wright et al. 2010) denoted AllWISE. 2MASS contains the *JHK* filters and AllWISE contains the B1-B4 bands as well as *JHK* data from 2MASS. We can query each catalog for the filters given each star's respective designation (id) in the catalog using the astroquery library (Ginsburg, A. et al., 2017). The major problem is how do we combine this data and verify that it corresponds to the same source object? Fortunately, the *Gaia* team has developed precomputed crossmatching tables for *Gaia* to 2MASS and *Gaia* to AllWISE. Given a *Gaia* source id, we can cross match it to its matching 2MASS designation (id) in the `gaiadr3.tmass_psc_xsc_best_neighbour` table. This allows us to merge the optical filters with near infrared filter data instantly. We can repeat this method using the `gaiadr3.allwise_best_neighbor` table to crossmatch to AllWISE. Unfortunately, this method turned out to be useless since the AllWISE database does not index its designation making queries to this catalog based on designation extremely slow. A workaround had to be done to accommodate the slow query. Luckily, each source object in AllWISE contained the *JHK* attributes from 2MASS. This makes it possible to merge the 2MASS table with the AllWISE table based on *JHK* magnitudes. This workaround is still not the best solution and leads to possible mismatching errors if the *JHK* magnitudes can be matched to multiple objects or are at different significant digits. The complete algorithm looks like this: 1) provide a list of *n Gaia* source ids or 2MASS designations and perform 1 query to the crossmatching table 2) use the ids from the crossmatching table to do 1 query to *Gaia* or 2MASS to get their filters 3) For *n* source objects, perform *n* nearest match within a 0.5 arcsecond radius queries to AllWISE 4) merge the two catalog data with matching *JHK* data to the AllWISE. It is important to consider how efficient this data collection query is. In computer science, algorithm efficiency is represented in Big-O notation. This algorithm's efficiency is O(*n*) meaning given *n* *Gaia* source ids or 2MASS designations, there will be n queries. This is not that efficient, and it took roughly over an hour to collect this data and build the data file for 944 source objects in NGC 2682. A faster algorithm is worth implementing. I discovered that by inputting a different set of parameters, we can build a data file much faster with a constant number of queries. Given parameters search radius and cluster name or Right Ascension (RA) and Declination (DEC) coordinates, we can perform 1 huge radius query to AllWISE, 1 huge equal radius query to 2MASS, 1 query to the *Gaia* to 2MASS crossmatching table, and 1 query to GaiaXPy to get the filters. This is at most 4 queries meaning the efficiency is O(4) = O(1), a constant operation. Using the cluster name `m67`, I was able to build a data input file under 30 seconds. I believe this method is more likely to be used given those parameters since most users won't have a list of generic ids ready but are more likely to classify a certain cluster name or the RA and DEC of an object with a certain search radius. Now we are able to replace the manually collected data with a method to query online databases.

SOFTWARE ENHANCEMENT 2: INPUT/OUTPUT FILE FORMATS

A single input data file contains multiple columns starting from *Gaia* source id, 2MASS designation, all the filter magnitudes, and their errors. Originally, BINOCS took in data from a pure `.txt` file with no column headers, thus making the file look like a list of random numbers. Reading the input data was difficult for a user that is unfamiliar with the system. A solution to this is to convert the data file from a `.txt` to a `.csv` or `.fits` file containing column headers.

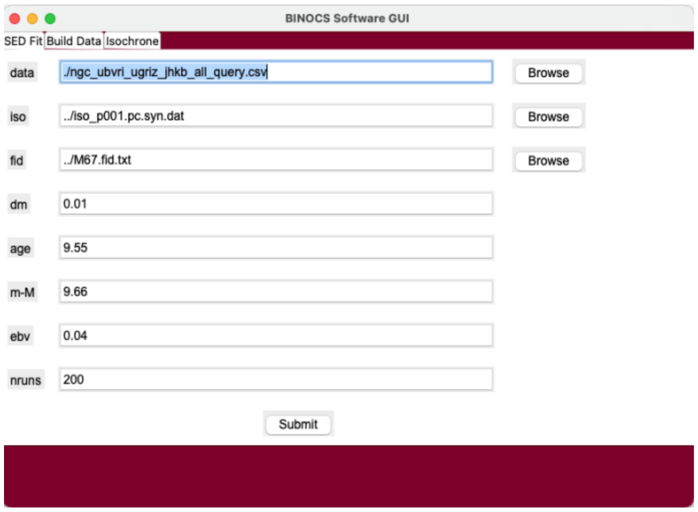
Fortunately, the Python library, Pandas, makes reading in `.csv` files easy with DataFrame objects. These objects represent tables with headers in the code. As we query data, we can write it to a DataFrame object, and export it as a `.csv`. When BINOCS wants to read in data, it can read in the same `.csv` file as a DataFrame object. Additionally, DataFrames are easy to convert to .fits files, which are common database files that astronomers use, by converting them to Astropy Tables and exporting to `.fits`. This supports ease of access to the user by giving them their preference in data input/output file formats. DataFrames make accessing columns easy through their name, eliminating the need to count the index of the column. This functionality can be further extended to support more formats and makes it easier to add new filter columns if support for them is added later.

SOFTWARE ENHANCEMENT 3: RELATIVE PATHS, PYTHON 3 & OOP

The original code was written as a Python module which was set to be found in the local machine's environment variable. This treats the module like a standard library and can be imported using the line "`import binocs`". However, this requires advanced knowledge of Python environment setup. Instead, I redesigned the module to be based on relative paths such that anyone with the source code, would only need to install required packages and not worry about environment variables. Attempting to run the existing code ran into errors. Since we are leveraging a new Python 3 library, we will need to update the old functionality to support Python 3. Minor syntax bugs were addressed but the overall functionality remained the same. As I refactored the code, I found that the modular designed could be wrapped in a single application programming interface (API), reducing the imports to only one file. I developed a single API that imports and wraps functions from submodules and allows the main software file to only import the single API file. Programming to an interface is good practice because this allows changing the implementation without directly affecting the function call in the main software. As a programmed to this interface, I felt the desire to rewrite the code in Object Oriented Programming style, a style that I am more familiar with. This treats all modules as classes with objects as instances of the class that behave with another object. While OOP is a programming style choice, I found that this makes which class the imported functions derive from clearer.

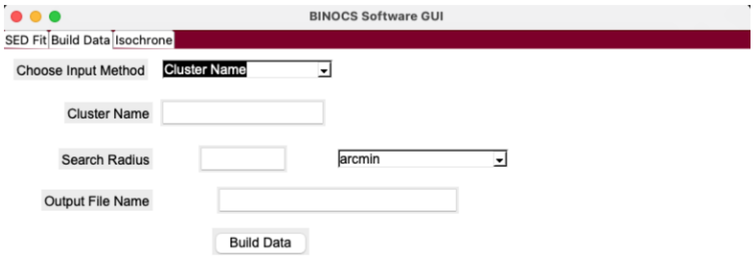
SOFTWARE ENHANCEMENT 4: GRAPHICAL USER INTERFACE

The original code was a suite of Python routines that only ran in the terminal. With the single API, we can develop a single file that utilizes this API. The average user does not perform day to day actions in a terminal, thus making accessibility an issue. I've addressed this with the introduction of the BINOCS Software Graphical User Interface (GUI). The BINOCS Software is now configurable and runnable from a visible window on screen. This window has several tabs, allowing you to configure several input parameters, build the data files, and format isochrones. Each button is set to use the API to run their functions. The old software had to be configured with an options (`.opt`) file containing its parameters. These parameters also included file paths to the data file. Having things set in the `.opt` file doesn't make the parameters transparent to the user after writing it once. We can eliminate the use of the `.opt` file through the GUI and add functionality to remove the need of knowing the data file path through a file picker dialog button. With these changes, a public user is more inclined to use the software and will have less things to worry about in regard to setting parameters and running separate Python files to build and format the data. A single window can now perform all the necessary actions.



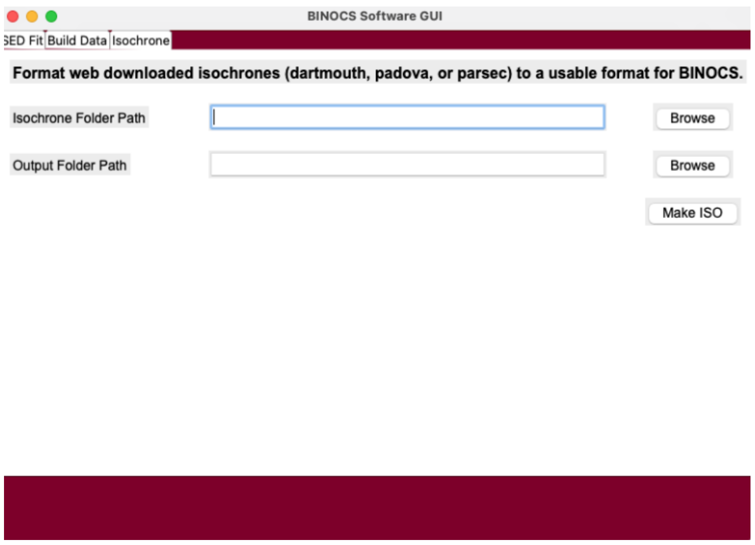
data ./ngc_ubvri_ugriz_jhkb_all_query.csv
iso ../iso_p001.pc.syn.dat
fid ../M67.fid.txt
dm 0.01
age 9.55
m-M 9.66
ebv 0.04
nruns 200

Figure 2: BINOCS Software GUI SED Fit Tab



Choose Input Method Cluster Name
Cluster Name
Search Radius arcmin
Output File Name

Figure 3: BINOCS Software GUI Build Data Tab

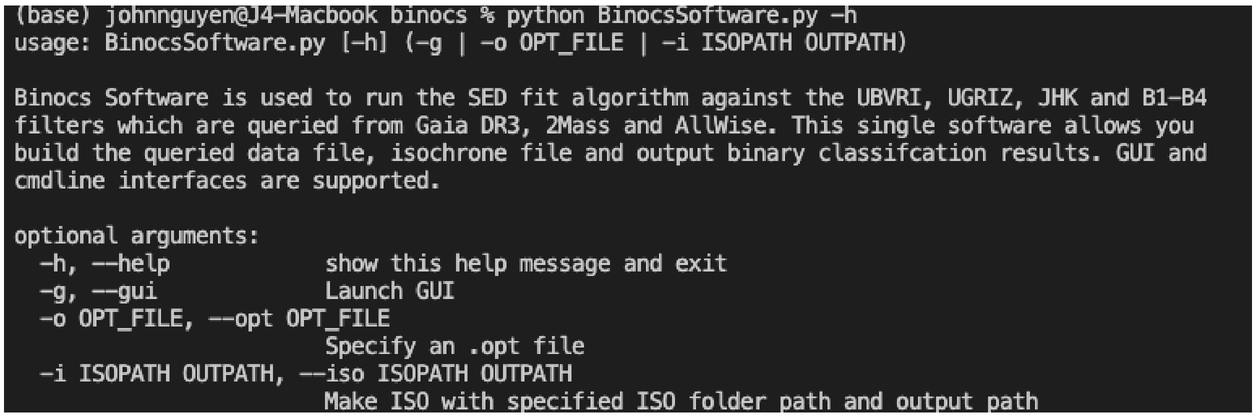


Format web downloaded isochrones (dartmouth, padova, or parsec) to a usable format for BINOCS.
Isochrone Folder Path
Output Folder Path

Figure 4: BINOCS Software GUI Isochrone Tab

SOFTWARE ENHANCEMENT 5: COMMAND LINE OPTIONS

While adding the GUI was a great feature, more options are better than no options and a good software engineer should not take away a useful feature. I assumed the need for a GUI for average users, but for existing users and those who prefer using the command line, I did not want to take away the command line use. Therefore, I have introduced command line options. Launching the GUI is possible through `-g` or `-gui`. Likewise, there are more options to run the binary fitting algorithm, format isochrones, and pass in the .opt file without the need to open the GUI. Additionally, I added a `-h` and `-help` section to explain the usage steps and options for the executable Python file and handle exceptions when the user supplies invalid options.



```
(base) johnnguyen@J4-Macbook binocs % python BinocsSoftware.py -h
usage: BinocsSoftware.py [-h] (-g | -o OPT_FILE | -i ISOPATH OUTPATH)

Binocs Software is used to run the SED fit algorithm against the UBVRI, UGRIZ, JHK and B1-B4
filters which are queried from Gaia DR3, 2Mass and AllWise. This single software allows you
build the queried data file, isochrone file and output binary classifcation results. GUI and
cmdline interfaces are supported.

optional arguments:
  -h, --help            show this help message and exit
  -g, --gui             Launch GUI
  -o OPT_FILE, --opt OPT_FILE
                        Specify an .opt file
  -i ISOPATH OUTPATH, --iso ISOPATH OUTPATH
                        Make ISO with specified ISO folder path and output path
```

Figure 5: Command Line Usage and Options

RESULTS: TESTING ACCURACY OF THE NEW SOFTWARE

Adding enhancements to a software is useless if it no longer performs its functions properly. Therefore, I performed a test comparison for cluster NGC 2682 running a newly built data file from the online databases against the old .txt data file through BINOCS. BINOCS outputs a classification `-1` for inconclusive, `1` for single system and `2` for binary system as well as the primary mass, primary mass error, secondary mass (for binary systems), and secondary mass error. I computed the differences in classifications based on the flags. First, I filtered out all inconclusive data, leaving me with 269 source objects out of 958, to view how many sources retained the same flag after running with the new dataset and how many changed from single to binary or binary to single.

Cluster	Number of Sources	Number of Same Classification	Number of Changed Classification	Ratio for Same	Ratio for Changed
NGC 2682	269	218	51	81%	19%

Then I ran a test with all data regardless of if it was inconclusive to see how many inconclusive sources became classified with the new data and how many old classifications were lost.

Cluster	Number of Sources	Number of Same Classification	Number of Changed Classification	Number of New Classifications	Number of Lost Classifications
NGC 2682	958	542	51	103	262

To further test accuracy, I created delta plots for the primary and secondary mass determinations along with their errors for sources with the same classification. This was done on the filtered data with no inconclusive data since mass is only determined for conclusive data.

Primary Mass Count	Secondary Mass Count
269	114

Cluster	Number of Primary Mass Delta >= 0.2	Number of Primary Mass Error Delta >= 0.2	Number of Secondary Mass Delta >= 0.2	Number of Secondary Mass Error Delta >= 0.2
NGC 2682	0	0	14	6

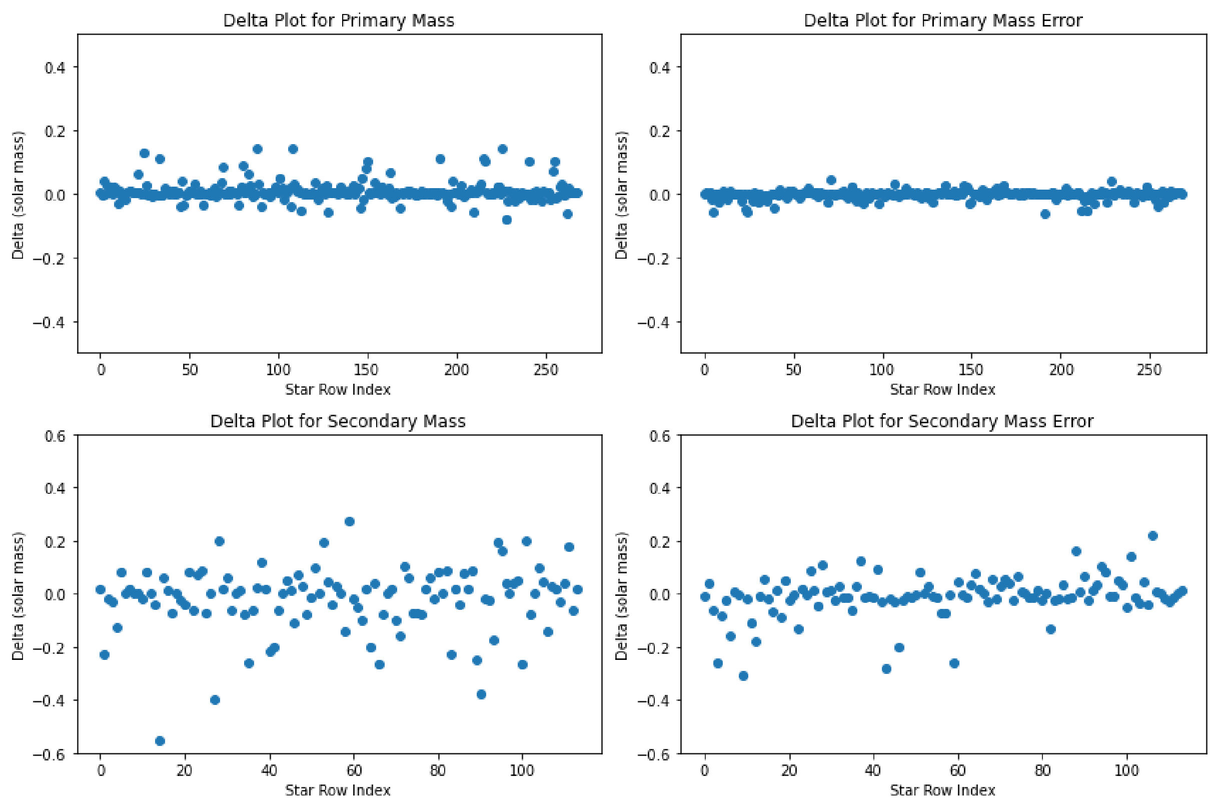


Figure 6: Delta Plots for Primary & Secondary Mass and Errors

ANALYSIS AND ERRORS

A matching ratio of 81% for the same flag is still relatively accurate. The mismatch is likely because the data catalogs might have crossmatched the wrong AllWISE data to the source object, affecting the delta. For the unfiltered data, we fortunately gained 103 classifications! Unfortunately, we lost a significant number of classifications because if any of the online databases is missing their respective set of filters, BINOCS will not be able to classify the object, leaving it inconclusive. However, this is a fair tradeoff since BINOCS will be capable to classifying millions of new objects and we still have the old classifications. The primary mass and error deltas are relatively low at under a 0.2 difference. This implies that for the conclusive data with the matching classification, the mass determination is relatively the same. These small differences might be due to slight magnitude differences from the *Gaia* space data. You might notice that the secondary masses and errors have a more varied delta plot. This is because if the primary mass goes through a change, then the secondary mass must accommodate for the change. It is important to remember that we are still under the assumption that the GaiaXPy synthetic photometry is reliable, and if it is not, then there may be errors. With a high matching ratio and minor differences in mass determination, I conclude that the new BINOCS software still performs as the original.

FUTURE ITERATIONS

The BINOCS software has gone through several enhancements, but there is always more room for improvement. Currently, no new classifications for clusters have been made. Unfortunately, I have not been able to finish the formatting of isochrones, thus no new clusters have been classified since they are dependent on a set of isochrones to be formatted. BINOCS also needs to be updated to support a newer set of isochrone catalog versions from PARSEC. The new software still utilizes the same old filter set; however, it is possible that GaiaXPy will have more filter sets accessible that can be integrated into the fitting algorithm. Future work will need to be done if a different set of filters would like to be used. A nice to have feature would be to extend this project to a public web app for accessibility. To aid astronomers around the world, I dream of this project being live on a website, with a backend supporting the API to query to databases, build data files, and run the fitting algorithm. This way, users don't have to deal with installing the software and its packages. My hope is that this project is continued if I am no longer supporting its feature development so that astronomers around the world can classify binary star systems and obtain useful information for investigation of dark matter in nearby dwarf galaxies.

REFERENCES

Fukugita M., Ichikawa T., Gunn J. E., Doi M., Shimasaku K., Schneider D. P., 1996, AJ, 111, 1748

Gaia Collaboration et al., 2016, A&A, 595, A1

Gaia Collaboration, 2023, VizieR Online Data Catalog, vol. 368, 2023. J/A+A/680/A35.

Gaia Collaboration, Montegriffo et al., 2022, VizieR On-line Data Catalog: J/A+A/674/A41

Ginsburg, A., Sipocz, B., Parikh, M., et al. 2017, astropy/astroquery: v0.4.7

Ruz-Mieres, GaiaXPy, 2022, DOI v2.1.0: 10.5281/zenodo.8239995 https://gaia-dpci.github.io/GaiaXPy-website/

Skrutskie M. F., et al., 2006, AJ, 131, 1163

Thompson, B., Frinchaboy, P. M., Spoo, T., & Donor, J. 2021, arXiv e-prints, arXiv:2101.07857. https://arxiv.org/abs/2101.07857

Wright E. L., et al., 2010, AJ, 140, 1868