README

This document is intended to help interested readers of “A von Mises-Fisher Distribution for the Orbital Poles of the Plutinos,” by IC Matheson, R Malhotra, and JT, to reproduce the calculations and figures in the paper with as little fuss and recreation from scratch as possible, given the limitations of the code used by the authors in the first place. This document and its directory are intended to be uploaded to GitHub. The paper will be submitted to MNRAS, the Monthly Notices of the Royal Astronomical Society. In theory, the journal paper itself should be sufficient to let any interested researcher reproduce the work. Any researcher who downloads this directory from GitHub should be able to reproduce the results in MNRAS nearly exactly, allowing for some small random variation due to the use of different random number seeds or small changes in the underlying software.

SOFTWARE

This work is done on a 2018 MacBook Pro running MacOS Monterey 12.6. Cluster computations are done on the University of Arizona Puma cluster, which uses SLURM as its batch submission scripting language. Both local and cluster calculations are done partly in Fortran 95 using the **gfortran** compiler and partly in Python 3.8.2. Python packages include **Astroquery, rebound, scikit-learn, urllib, shapely, random**, and **json**.

INSTRUCTIONS FOR FULL REPRODUCTION OF PAPER

Go to the JPL Solar System Dynamics Small Body Database query at https://ssd.jpl.nasa.gov/tools/sbdb_query.html.

Expand the “Custom Object/Orbit Constraints” menu.

Select “Define New Custom Constraint,” then “Orbit and Model Parameter Fields.”

In the “Select field orbit” menu, choose “a”, then “<”, and finally enter “42” in the field. This is a constraint $a < 42$ au. Repeat for $a > 38$ au. These are very rough cuts of the approximate semimajor axis limits of the 3:2 mean motion resonance (MMR) that defines the

Plutino population. These elements are heliocentric. Later the selection will be more refined through Python scripting.

Expand the menu "Output Selection Controls," click "select all", then "add." Not all available fields will be used later, but it is easier to select all and then remove unneeded fields through Python scripting than to tediously select individual fields.

In the "Output Fields" menu box, click "Full Precision."

Then click the green "Get Results" button.

Click "Download (CSV-format)". Save the csv in the working directory for this project as **00_sbdb_query_results_[YYYYMMDD].csv**. In the GitHub directory, this is **00_sbdb_query_results_20221012.csv**. This is the raw list of KBOs that will be the base of all later computation. There are 887 objects in this list.

Now go to the Minor Planet Center database at https://www.minorplanetcenter.net/data. Right-click to save **MPCORB.DAT** in the working directory, as **00_MPCORB_[YYYYMMDD].DAT**. In the GitHub directory, this is **00_MPCORB_20211202.DAT**. This will later be used to cross-reference against the Small Body database to eliminate some objects from consideration.

Save **00_MPCORB_[YYYYMMDD].DAT** in your main working folder. Save a backup copy in your Downloads folder.

Open **01_prepare_to_classify_clones.py** in your Python environment or development window. Make sure lines 11-17 are for 2022-10-12, 00:00:00. Then run the script locally.

Next, upload the entire folder to the Puma cluster (or your local cluster). Note that **00_MPCORB_[YYYYMMDD].DAT** will not upload to Puma, as it is too large. This is why you need to back it up in your Downloads folder.

From the working directory on the High Performance Computing online file browser GUI, click "Open in Terminal."

Submit the batch job using the command **sbatch slurm_classify_clones.slurm**. The SLURM file will need to be modified for your computer cluster and user account.

After the batch job has completed, delete the local folder from your computer and download the entire folder from the cluster. If the folder is too large to download using the GUI, it can be downloaded from the Puma cluster at the University of Arizona, using a MacBook command line, as follows:

```
ssh [user id]@filexfer.hpc.arizona.edu
tar czvf [folder_name].tar.gz [/path/to/folder]
```

The compressed folder is then downloaded using the High Performance Computing online interface and unzipped.

Run **02_prepare_to_check_Plutino_resonances.py** on your local machine.

Delete the entire folder from the cluster and re-upload it.

Run **sbatch slurm_check_Plutino_resonances.slurm** on the cluster.

Delete the entire folder from your local machine and re-download it from the cluster.

Examine all files **plot_False_3_2_[MPC designation].pdf** and **plot_True_3_2_[MPC designation].pdf** on your local machine. Note if any objects seem to be classified incorrectly. Enter the designations of the false negatives (objects classified False that should be classified True) on line 43 of **03_read_Plutino_resonances.py** and enter the designations of the false positives (objects classified True that should be classified False) on line 44.

Run **03_read_Plutino_resonances.py** on your local machine. This will create the plots and figures for the paper. Print commands in the code will output the numbers quoted in the paper. It saves the Plutinos to the file **plutinos_for_mnras.csv.** This is the full electronic form of Table 1 in the MNRAS paper. The Python script also saves the figures in the paper as **fig1_[# of Plutinos].pdf**, **fig2[# of Plutinos].eps**, **fig3[# of Plutinos].eps**, **fig4[# of Plutinos].eps**, and **figB1.eps**.