Dissertation Advisors:
Professor Pavlos Protopapas
Professor Christopher H. Rycroft

Author:
Michael S. Emanuel

Kepler's Sieve:
Learning Asteroid Orbits from Telescopic Observations

Abstract

A novel method is presented to learn the orbits of asteroids from a large data set of telescopic observations. The problem is formulated as a search over the six dimensional space of Keplerian orbital elements. Candidate orbital elements are initialized randomly. An objective function is formulated based on log likelihood that rewards candidate elements for getting very close to a fraction of the observed directions. The candidate elements and the parameters describing the mixture distribution are jointly optimized using gradient descent. Computations are performed quickly and efficiently on GPUs using the TensorFlow library.

The methodology of predicting the directions of telescopic detections is validated by demonstrating that out of approximately 5.69 million observations from the ZTF dataset, 3.75 million (65.71%) fall within 2.0 arc seconds of the predicted directions of known asteroids. The search process is validated on known asteroids by demonstrating the successful recovery of their orbital elements after initialization at perturbed values. A search is run on observations that do not match any known asteroids. I present orbital elements for [5] new, previously unknown asteroids.

Exact number of new asteroids presented

All code for this project is publicly available on GitHub at github.com/memanuel/kepler-sieve.

Chapter 1

Analysis of ZTF Asteroid Detections

1.1 Introduction

Zwicky Transient Facility (ZTF) is a time-domain survey of the northern sky that had first light at Palomar Observatory in 2017. It is run by CalTech. My advisor Pavlos suggested it as a data source for this project. The ZTF dataset has two major advantages for searching for asteroids:

- ZTF gives a wide and fast survey of the key, covering over 3750 square degrees an hours to a depth of 20.5 mag
- A machine learning pipeline has been developed to classify a subset of ZTF detections that are classified as probable asteroids

The data set I analyze here consists of all ZTF detections that were classified as asteroids. Data on each detection include:

- **ObjectID** an identifier of the likely ojbect associated with this detection; multiple detections often share the same ObjectID
- **CandidateID** a unique integer identifier of each detection
- **MJD** The time of the detection as an MJD
- **RA** The right ascension of the detection
- **Dec** The declination of the detection

- **mag** The apparent magnitude of the detection

Available data also includes a number of additional fields that were not used in the analysis.

ALeRCE (Automatic Learning for the Rapid Classification of Events) is an astronomical data broker. ALeRCE provides a convenient API to access the ZTF asteroid data, which can be installed with `pip`. I used ALeRCE on this project to download the ZTF asteroid data set.

1.2 Exploratory Data Analysis of ZTF Asteroid Data

Before plowing into the search for new asteroids, I conducted an exploratory data analysis (EDA) of the ZTF asteroid dataset. This can be followed interactively in the Jupyter notebook `05_ztf_data.ipynb`. I took a download of the data running through 26-Feb-2020. The first detection is on 01-Jun2018. The dataset contains 5.69 total detections. The volume of detections increases very significantly beginning in July 2019; for practical purposes the dataset consists of 8 months of detections spanning July 2019 through February 2020.

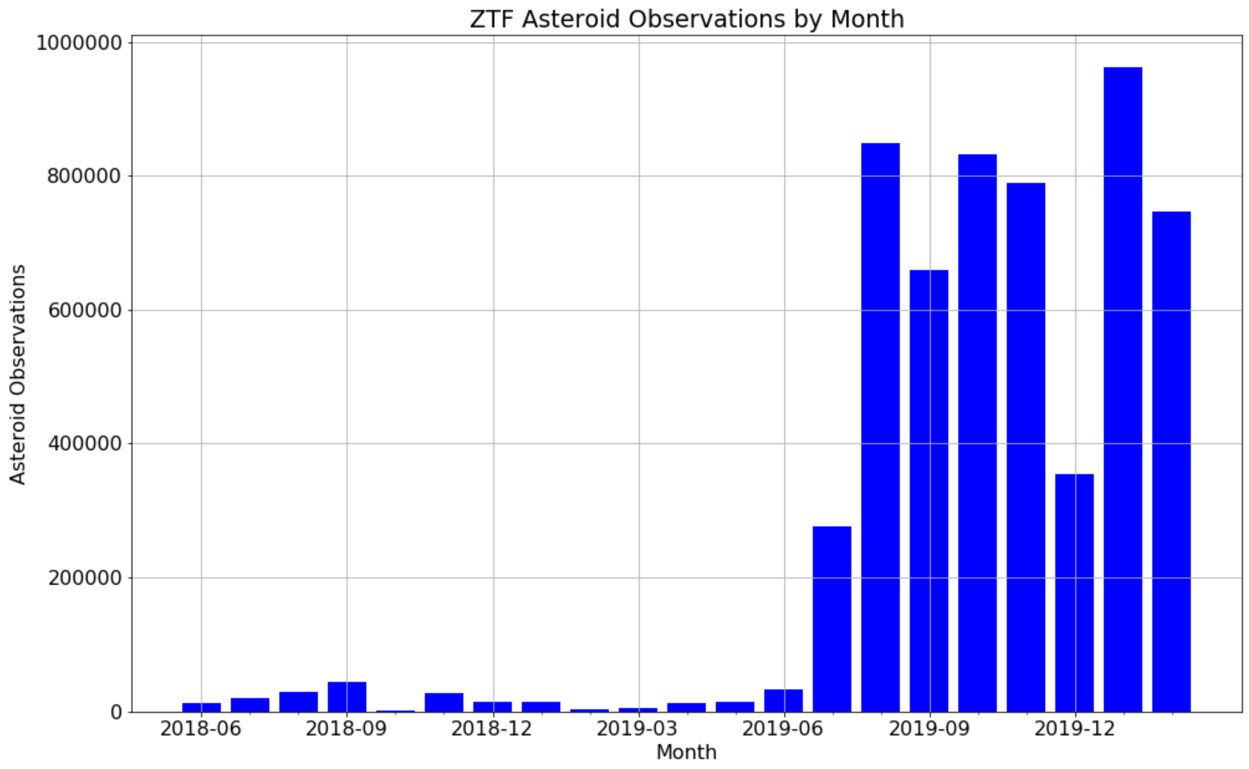
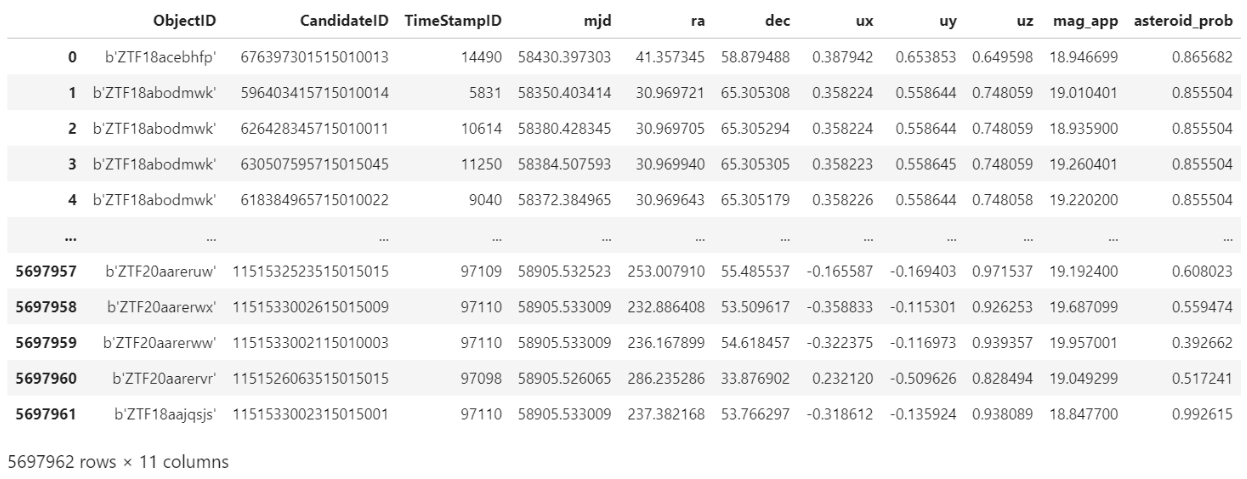


Figure 1.1: *ZTF Asteroid Detections per month*



	ObjectID	CandidateID	TimeStampID	mjd	ra	dec	ux	uy	uz	mag_app	asteroid_prob
0	b'ZTF18acebhfp'	676397301515010013	14490	58430.397303	41.357345	58.879488	0.387942	0.653853	0.649598	18.946699	0.865682
1	b'ZTF18abodmwk'	596403415715010014	5831	58350.403414	30.969721	65.305308	0.358224	0.558644	0.748059	19.010401	0.855504
2	b'ZTF18abodmwk'	626428345715010011	10614	58380.428345	30.969705	65.305294	0.358224	0.558644	0.748059	18.935900	0.855504
3	b'ZTF18abodmwk'	630507595715015045	11250	58384.507593	30.969940	65.305305	0.358223	0.558645	0.748059	19.260401	0.855504
4	b'ZTF18abodmwk'	618384965715010022	9040	58372.384965	30.969643	65.305179	0.358226	0.558644	0.748058	19.220200	0.855504
...	...	...	...	...	...	...	...	...	...	...	...
5697957	b'ZTF20aareruw'	1151532523515015015	97109	58905.532523	253.007910	55.485537	-0.165587	-0.169403	0.971537	19.192400	0.608023
5697958	b'ZTF20aarerwx'	1151533002615015009	97110	58905.533009	232.886408	53.509617	-0.358833	-0.115301	0.926253	19.687099	0.559474
5697959	b'ZTF20aarerww'	1151533002115010003	97110	58905.533009	236.167899	54.618457	-0.322375	-0.116973	0.939357	19.957001	0.392662
5697960	b'ZTF20aarervr'	1151526063515015015	97098	58905.526065	286.235286	33.876902	0.232120	-0.509626	0.828494	19.049299	0.517241
5697961	b'ZTF18aajqsjs'	1151533002315015001	97110	58905.533009	237.382168	53.766297	-0.318612	-0.135924	0.938089	18.847700	0.992615

5697962 rows × 11 columns

Figure 1.2: *Preview of Pandas DataFrame of ZTF Detections*

The fields `mjd`, `ra`, `dec`, and `mag_app` are part of the original dataset. I have populated the columns `ux`, `uy` and `uz` by running `radec2dir` on the quoted RA/Dec from ZTF.

1.3 The Angular Distance Bewteen two Directions $\vec{u}_1$ and $\vec{u}_2$

A recurring task in this thesis to compute the angular distance bewteen two directions on the unit sphere. If $\mathbf{u}_1$ and $\mathbf{u}_2$ are on the unit sphere, we can compute their Cartesian distance s in the usual way,

$$s = \|\mathbf{u}_2 - \mathbf{u}_1\|$$

s will be in the interval $[0, 2]$. We would also like to know the angular distance θ of the shortest path on the surface of the sphere (geodesic) connecting these two points. On Earth, this would be analogous to the length of the great circle route by airplane between two cities. Here is a simple picture demonstrating the derivation of the formula relating s and θ. The two directions are shown as arrows pointing up. The distance between them s is bisected by a line segment from the center of the sphere. This forms a right triangle with hypotenuse 1 and side length $s/2$

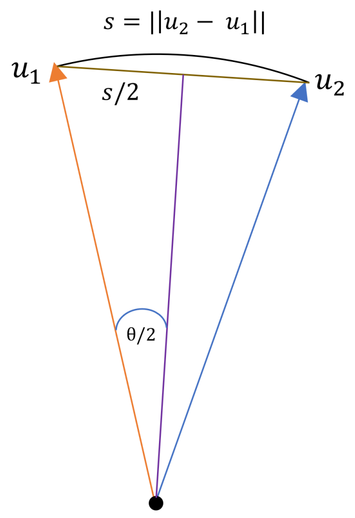


Figure 1.3: *The angular distance between two directions on the unit sphere.*

opposite angle $\theta/2$. We thus obtain the formulas

$$\sin(\theta/2) = s/2$$
$$\theta = 2\arcsin(s/2)$$
$$s = 2\sin\theta(\theta/2)$$

This function is implemented in `astro_utils` as `deg2dist` and `dist2deg` to convert between degrees and Cartesian distance in either direction.

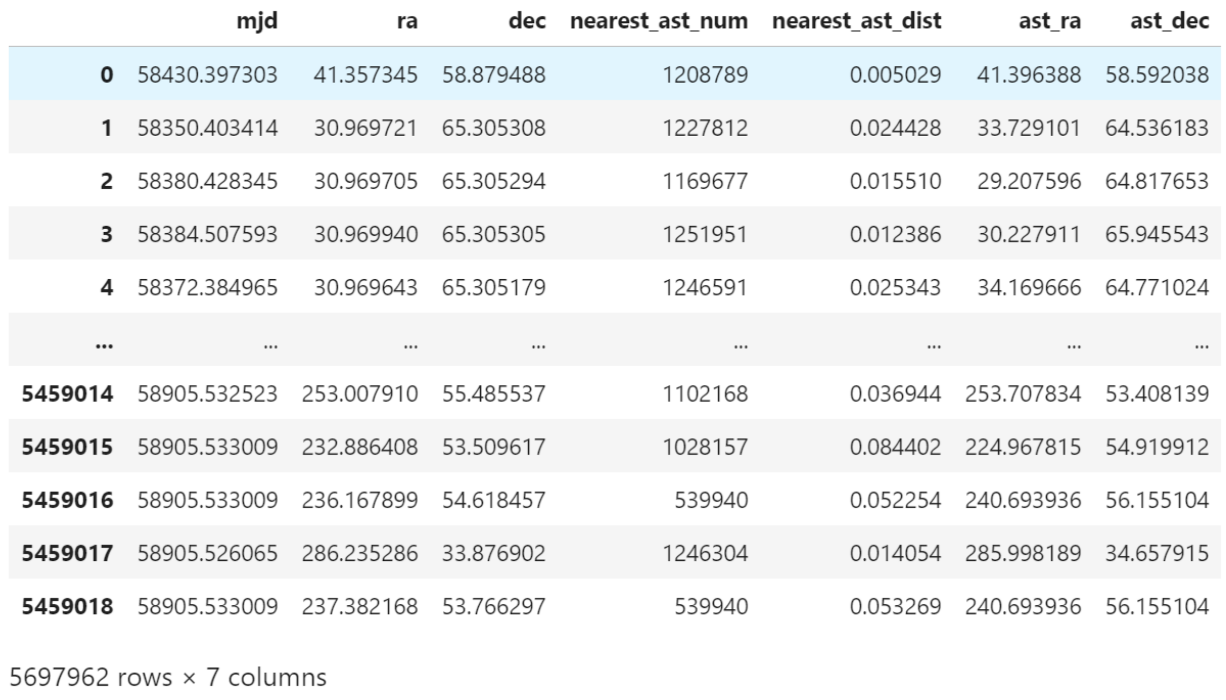
1.4 Finding the Nearest Asteroid to Each ZTF Observation

We now have in principle all the tools required to find which asteroid was closest in angular distance to each ZTF observation. To recap the key steps, each ZTF observation is converted from a RA/Dec to a direction in the BME. The position of Earth and each of the 733,489 catalogued asteroids are integrated as of the objervation time, and the direction bewteen them is calculated. There are a few problems with the brute force approach implicitly suggested above. We have 5.7E6 detections and 7.3E5 catalogued asteroids for a total of 4.16E12 (4.16 billion) interactions. Even if we work in single precision with 4 bytes per float, we need 12 bytes for a 3D direction difference translating to about 50 GB to load the matrix in memory. The bigger problem is that a brute force integration of all the asteroids at the MJDs of the all 5.7 million observations would be brutally slow.

Fortunately there are a few simple tricks we can use that together make this problem tractable. The ZTF detections come from a series of images taken through the same telescope, so they come in blocks made at the same time. The 5.7 million rows share “only” 97,111 distinct MJDs. We also don’t need to re-integrate the asteroid orbits. We’ve already done a high precision numerical integration at a 1 day frequency and saved the results into `numpy` arrays in blocks of 1,000 asteroids at a time. Our entire data span only 635 days. Loading one block of 1,000 asteroids therefore has only 3.81 million numbers (635 days x 1,000 asteroids x 6 numbers per integration point).

The code to load the basic ZTF data from ALeRCE is included in the module `ztf_data.py`. The main function used by consumers is `load_ztf_det_all`, which loads a cached copy of all the available detections from the local disk. The module `ztf_nearest_ast.py` contains a Python program that peforms the calculation described above. The block size is a parameter that can be controlled from the commandline; I ended up leaving it at 1,000. Checking back from my handwritten notes when I ran the job, it took approximately 25 hours to complete on a powerful server with 40 Intel CPU cores. The job ended up being memory bound, maxxing out all 256 GB of available RAM on the server. The initial job described above writes only computes the nearest asteroid in a block of 1,000 asteroids. A second reduction operation is then carried out to find the nearest asteroid overall. To limit memory usage, this was done in two steps. First, I took 16 chunks of 1,000 at a time to find the nearest asteroid in a block of 16,000 to each detection. Then I combined all the blocks of 16,000 (about 46) to generate one file with the nearest asteroid.

The work of splining the asteroid directions is done in the module `asteroid_dataframe.py`. It includes functions to load the asteroid data from disk; spline the positions and velocities to the requested dates; and compute the astrometric directions to these splined positions and velocities. The module `ztf_ast` does the work of comparing a block of ZTF observations to a block of splined asteroid directions and finding the nearest one. Once these calculations have been done once, you don’t need to worry about them unless you are adding a new block of ZTF data. Consumers can load the assembled DataFrame including the nearest asteroid number and distance with a single call to `load_ztf_nearest_ast` which is defined in `asteroid_dataframe`. For the motivated reader who would like a detailed an interactive review of all these calculations, please see the Jupyter notebook `04_asteroid_dataframe.ipynb`. It includes tests comparing my splined outputs of daily data to Horizons data downloaded at a 3 hour interval.



	mjd	ra	dec	nearest_ast_num	nearest_ast_dist	ast_ra	ast_dec
0	58430.397303	41.357345	58.879488	1208789	0.005029	41.396388	58.592038
1	58350.403414	30.969721	65.305308	1227812	0.024428	33.729101	64.536183
2	58380.428345	30.969705	65.305294	1169677	0.015510	29.207596	64.817653
3	58384.507593	30.969940	65.305305	1251951	0.012386	30.227911	65.945543
4	58372.384965	30.969643	65.305179	1246591	0.025343	34.169666	64.771024
...	...	...	...	...	...	...	...
5459014	58905.532523	253.007910	55.485537	1102168	0.036944	253.707834	53.408139
5459015	58905.533009	232.886408	53.509617	1028157	0.084402	224.967815	54.919912
5459016	58905.533009	236.167899	54.618457	539940	0.052254	240.693936	56.155104
5459017	58905.526065	286.235286	33.876902	1246304	0.014054	285.998189	34.657915
5459018	58905.533009	237.382168	53.766297	539940	0.053269	240.693936	56.155104

5697962 rows × 7 columns

Figure 1.4: *Preview of Pandas DataFrame of ZTF Detections Including Nearest Asteroid*

I would like to take a step back and review what has been presented thus far. Over 4 billion interactions between a ZTF asteroid detection and the predicted position of a known asteroid in the sky have been generated. These have been filtered to associate each ZTF detection with the known asteroid it is nearest to. This is a powerful enrichment of the original ZTF dataset, and might open the door to some additional work in the future. For example, it could be used to create a bulk data set linking the original image files to the asteroids they belong to. This could in turn be used to refine the machine learning pipeline used to classify detections and guess when they belong to the same object.

1.5 Analyzing the Distribution of Distance to the Nearest Asteroid

In this section I explore the statistical distribution of the Cartesian distance between observations and the nearest asteroid. I compare the observed distribution of this distance to the theoretical distribution we would obtain if either our observed or predicted directions were distributed uniformly at random on the sphere.

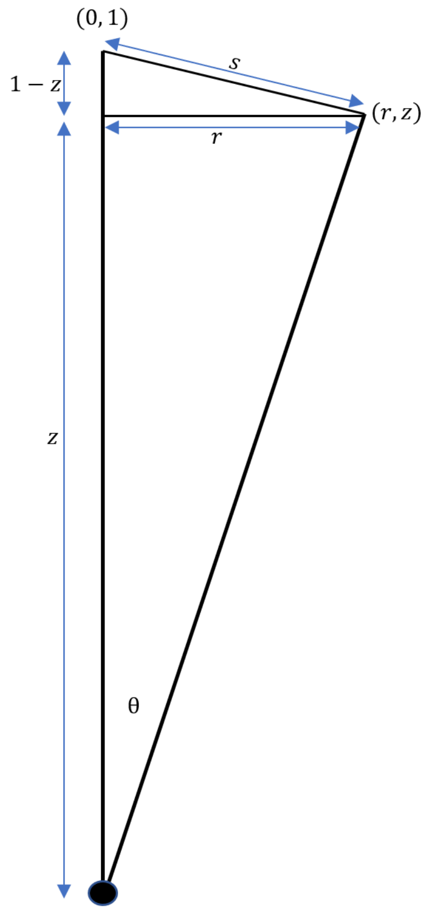


Figure 1.5: *Distance between an observation at the north pole and an arbitrary point.*
The observed direction $\mathbf{u}_{\text{obs}}$ *is assumed w.l.o.g. to be at the north pole* $(0,0,1)$.
The predicted direction (x,y,z) *is rotated in the xy plane to to* $(r,0,z)$
The Cartesian distance to from the observation to (r,z) *is s, the hypotenuse of a right triangle with sides* $1-z$ *and* r.

Suppose without loss of generality that the observed direction is at the north pole, i.e. $\mathbf{u}_{\text{obs}} = (0,0,1)$. Suppose that the direction we predict is (x,y,z). Observe that the problem is symmetric about the z axis, so we can rotate the problem into the plane containing the center, the north

pole, and our guess. Let $r^2 = x^2 + y^2$ be the squared distance from the z axis to our guess. This configuration is shown in the diagram above. We can can relate the Cartesian distance s to the height z of our guess with the following simple observations.
(x, y, z) lies on the surface of a sphere, so $x^2 + y^2 + z^2 = 1$.
$r^2 = x^2 + y^2$ by definition, so $r^2 = 1 - z^2$.
In our diagram, we can see that s is the hypotenuse of a right triangle whose other two side have length r and $1 - z$. Applying the Pythagorean Theorem, we find $s^2 = (1 - z)^2 + r^s$.
This simplifies to

$$s^2 = 2(1 - z)$$
$$z = 1 - \frac{s^2}{2}$$

It turns out that this is a very useful parameterization because there is an elegant parameterization of the differential solid angle $d\Omega$ in terms of dz. When surface integrals of a sphere are taught in introductory multivariate calculus courses, students are most likely to be exposed only to the surface element in spherical coordinates (r, θ, ϕ) which is $d\Omega = \sin\theta d\theta d\phi$. See, e.g. Wikipedia - integration in spherical coordinates. But the parameterization in terms of the height z along the z axis is especially clean and convenient on this problem.
Observe that $z = \cos\theta$ so $dz = |-\sin\theta| d\theta = \sin\theta d\theta$. We take absolute values because this is an application of the change of variables formula and measures are always positive.
Substituting this expression for the surface element, we obtain

$$d\Omega = dz d\phi$$

As a quick sanity check of this result, let's see if we can recover that the surface area of the unit sphere is 4π:

$$A = \int_{z=-1}^{1} \int_{\phi=0}^{2\pi} d\phi dz = 2\pi \int_{z=-1}^{1} dz = 4\pi$$

We can now write the probability density function (PDF) for any function that can be expressed in terms of z. Think of Z as a random variable now. The above result shows that when Z is uniformly distributed on the unit sphere,

$$Z \sim \text{Unif}(-1, 1)$$

Previously we showed that $z = 1 - s^2/2$. This is a very useful result; it tells us that if we treat the squared distance as a random variable S^2, then it is uniformly distributed on $[0, 2]$. Even more usefully, the conditional distribution of S^2, conditioned on $S^2 \leq \tau^2$, is also uniform:

$$S^2 | S^2 \leq \tau^2 \sim \text{Unif}(0, \tau^2)$$

If we apply a threshold distance τ and only consider predicted directions that are within Cartesian distance τ of an observation, then the condtional distribution of the relative distance over the threshold squared is uniform on $[0, 1]$. In mathematical notation instead of words,
Let $\mathbf{u}_{\text{pred}}$ be a random variable distributed uniformly on the unit sphere.
Let $S = \|\mathbf{u}_{\text{pred}} - \mathbf{u}_{\text{obs}}\|$ be the Cartesian distance between $\mathbf{u}_{\text{pred}}$ and $\mathbf{u}_{\text{obs}}$.
Let τ be a threshold distance in [0, 2].
Let $V = S^2/\tau^2$ be the relative squared distance of on observation vs. the threshold.
Then the conditional distribution of V, conditional on $S < \tau$ (equivalently $V < 1$) is

$$V \sim \text{Unif}(0, 1)$$

References

[1] Hanno Rein, Shang-Fei Liu
REBOUND: An open-source multi-purpose N-body code for collisional dynamics.
Astronomy & Astrophysics. November 11, 2011.
arXiv: 1110.4876v2

[2] Hanno Rein, David S. Spiegel
IAS15: A fast, adaptive high-order integrator for gravitational dynamics, accurate to machine precision over a billion orbits.
Monthly Notices of the Royal Astronomical Society. Printed 16 October 2014.
arXiv: 1405.4779.v2

[3] Murray, C. D.; Dermott, S.F.
Solar System Dynamics
Cambridge University Press. 1999