Kepler's Sieve:
Learning Asteroid Orbits from Telescopic Observations

A dissertation presented

by

Michael S. Emanuel

to

Institute of Applied Computational Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Data Science

Harvard University

Cambridge, Massachusetts

May 2020

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 by randomly choosing the six elements independently at random, either by sampling from a known asteroid or from uniformly distributed angles. A statistical model describes the distribution of angular distances between the directions of observed detections to the predicted direction from the observatory site to a body with these orbital elements. This model yields a log likelihood function, which attains large positive values when the candidate orbital elements are near to the elements of a real asteroid viewed many times in the data. 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 in multiple stages. First, I demonstrate that given an initial guess equal to a perturbation applied to the elements of asteroids present in the data, the search process is able to successfully recover the true orbital elements to a high degree of precision. Next, I demonstrate that a random initialization is able to converge to elements matching some known asteroids to high precision. Finally, I use the search 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.

Contents

Acknowledgments

I would like to thank my advisor, Pavlos Protopapas, for suggesting this topic and for his consistent support, advice and encouragement.

I would like to thank Chris Rycroft, my secondary advisor, for guiding me through a paper in Applied Math 225 in which I explored numerical integrators for solving solar system orbits.

I would like to thank Matt Holman and Matt Payne from the Center for Astrophics (CFA) for their advice on state of the art solar system integrators.

Most importantly, I would like to thank my wife Christie and my children Victor and Renée for their love and support. The Covid-19 crisis struck just as my work on this thesis kicked into high gear. It would not have been possible to complete it without extraordinary understanding and support from them.

Introduction

Determining the orbits of asteroids is one of the oldest problems in astronomy. Classical methods are based on taking multiple observations of the same body through a telescope. For an object that is large and bright enough, the human eye can ascertain the continuity of the motion, i.e. that the data are multiple sightings of the same object. Once enough sightings have been obtained, orbital elements can be solved using traditional numerical methods, such as a least squares fitting procedure that seeks elements to minimize the sum of squares error to all of the observations.

State of the art techniques for solving this problem are remarkably similar in spirit to the classical method. Indeed, the first interstellar object, 'Oumuamua, was discovered when astronomer Robert Weryk saw it in images captured by the Pan-STARRS1 telescope on Maui. [1] [2] More automated methods also exist. Still, these methods are based on a search in the space of the observable data attributes: the time of observation (MJD), the right ascension (RA) and declination (DEC). The apparent magnitude or brightness (MAG) is the third important observed quantity available for telescopic detections. Two observations made close together in time at two points in the sky very near to each other have a relatively high probability of belonging to the same object. Such a pair of observations is called a "tracklet." Today's most automated approaches to identifying new asteroids from telescopic data are based on performing a greedy search of the observed data to extend tracklets. Once a tracklet is identified, the algorithm attempts to extrapolate the path where future detections of this object might be. After enough detections are strung together, a fitting procedure is tried to determine the orbital elements.

This is a solid technique and I do not mean to cast aspersion on it. In this paper, however, I

[1] Wikipedia - Oumuamua

[2] NY Times - Astronomers Race to Study a Mystery Object from Outside the Solar System

propose a new method which I believe has some significant advantages. Rather than searching in the space of the data, i.e. (MJD, RA, REC, MAG), I propose to instead search over the six dimensional space of Keplerian orbital elements $(a, e, i, \Omega, \omega, f)$. Why should we complicate things by searching implicitly, as it were, on the space of possible orbits, rather than the simpler and more direct method currently used? The main reason is to avoid a combinatorial explosion.

If you limit your search to candidate tracklets where you detect the same object multiple times in a short span of time, you are going to miss out any object that you detect only once or twice on a given night of observations. But if this same object were seen on multiple nights, possibly separated over multiple days or longer, it becomes very costly to propose enough candidate tracklets to pick them up. Indeed you will soon face a combinatorial explosion in the number of possible tracklets. A simplified model of the number of tracklets might be that we have a data set containing observations with a uniform density ρ per day per degree squared of sky, and we set a threshold τ in time and d in angular distance for how close a second observation must be to mark it as a candidate tracklet.

Here is a simple model showing the quadratic cost of enumerating candidate tracklets.
If you extend this further to tracks with 3 observations, the scaling gets even worse (cubic).
Let ρ be the average density of detections per day per degree of sky.
Let T be the number of days of observations in our data set.
Let τ be the threshold in days for 2 observations to be considered close enough in time to form a candidate tracklet.
Let Δ be the threshold angular distance in degrees for 2 observations to be considered close in the sky to form a candidate tracklet.
Let $A = 41,253$ be the number of square degrees in the sky. [3]
Let $N = T \cdot A \cdot \rho$ be the total number of detections in the data set.
Let $m = \tau \cdot \pi\Delta^2 \cdot \rho$ be the average number of observations that will be close enough to each candidate starting point of a tracklet.
Let $NT_2 = \dfrac{N \cdot m}{2!} = \dfrac{\tau}{T} \cdot \dfrac{\pi d^2}{A} \cdot \dfrac{\rho^2}{2!}$ be the total number of candidate trackets of size 2.
Let $NT_k = \dfrac{N \cdot m^{k-1}}{k!} = \left(\dfrac{\tau}{T} \cdot \dfrac{\pi d^2}{A}\right)^{k-1} \cdot \dfrac{\rho^k}{k!}$ be the total number of candidate trackets of size k.

[3] Wikipedia - Square Degrees in the Sky

We can see the bad news right away. The number of tracklets of size k, NT_k, scales as ρ^k. And the factors in the denominator don't bail us out. The number of possible ways m to extend a tracklet is going to be a large number well in excess of 1.

This is the principal motivation for searching in the space of orbital elements. While it's a large 6 dimensional space, its size is fixed. The cost of the search algorithm presented below scales linearly in the observation density ρ for each candidate element analyzed. The cost of the entire algorithm is therefore on the order of N^2, with no explosion as you consider tracklets larger than 2. The second major reason for searching in the space of orbital elements is that it permits the search algorithm to string together observations made far apart in time. This is a capability that eludes searches based on tracklets.

I summarize now the key steps in the search algorithm. The first step is to generate a set of candidate orbital elements. This is done with a very simple approach, one which can almost certainly be imporved on later: random initialization. For four of the orbital elements, a, e, i, and Ω, one of the $\sim 780,000$ catalogued asteroids is selected at random. Its orbital elements are used to populate these four. It is worth emphasizing that each element is initialized with a *different* random asteroid; the four elements in this part will almost never match one of the known asteroids across all four elements. The remaining two orbital elements, M (mean anomaly) and ω (argument of periapsis), are modeled to be distributed uniformly at random on the circle $[0, 2\pi)$. These are then converted to the representation using $(a, e, i, \Omega, \omega, f)$ using the rebound numerical integrator.

Once the candidate elements have been initialized, they are integrated numerically using the `rebound` library. This is considered to be the gold standard of their true orbits. This initial integration is then used to filter the data set of ZTF observations to a subset that are relevant for searching for orbits. A routine computes the direction $\mathbf{u}_{\text{pred}}$ in the barycentric mean ecliptic (BME) reference frame that an observer at a given observatory site on earth would have seen light leaving an object with the candidate elements at a given observation time (MJD). This quantity is computed at each unique observation time in the ZTF data set. A separate computation is performed once on all of the ZTF observations converting the observed triplets (MJD, RA, DEC) into vectors $\mathbf{u}_{\text{obs}}$, the direction of the observation in the BME frame. The angular distance between the predicted and observed direction is computed. A threshold (2.0 degrees) is applied, and all

ZTF observations falling within this threshold are cached in memory of the search class.

During the main body of the search process, the elements will be adjusted by a small amount in each training round. These perturbed elements will have their orbits evaluated using the Kepler two body model. An implementation is performed on the GPU using TensorFlow that is fast and differentiable. The ground truth orbit is used to provide an adjustment term so that the predicted orbits will match the true orbits exactly when the perturbation is zero. The predicted orbit can therefore be considered to be a linearization of the true orbits based on the Kepler model.

The objective of the optimization function is based on the log likelihood of a statistical model for the distribution of distances between predicted and observed directions. A lemma will demonstrate that for directions uniformly distributed on the sphere, the squared distance over the threshold distance would be uniformly distributed on the interval $[0, 1]$. A mixture model is formulated, where the distance between every predicted and observed direction is modeled as a mixture of hits and misses. The misses are distributed uniformly on $[0, 1]$. The hits are distributed as a truncated exponential distribution. The decay parameter λ of this exponential process is associated with a resolution paramater R. This model is equivalent to assuming that some fraction h (for hits) of the detections are due to a real body with the candidate elements, and that the results of the detection will be normally distributed with a precision parameter equal to the resolution. During the search process, the threshold parameter is also updated. This dynamic threshold should not be confused with the original threshold of 2.0 degrees used to build the filtered training data.

The optimization process jointly optimizes the candidate orbital elements and three parameters in the mixture model: the assumed number of hits, the resolution R, and the threshold. Intuitively, we want the model to gradually tighten its focus, and adjust the orbital elements so they hit as many observations as closely as possible. But we *don't* want the model to get "faked out" by trying to get the elements closer to observations that belong to *other* asteroids. The model needs some way to update probabilities that each observation is a hit or a miss, which it does using the mixture model. Early on, the optimization will tries to get close to the central tendency of the data set. If the initialization was good, it will gradually tighten in the resolution and threshold parameters. The gradients will encourage the model to adjust the candidate orbital elements so that some of the observations, the ones it sees as highly probable hits, will be very close what is predicted by the candidate elements. The observations modeled as highly probable misses will hardly contribute to the gradients of the candidate elements.

In practice, the optimization is carried out in alternating stages. In odd numbered stages, only the resolution parameters are tuned at a higher learning rate; in even numbered stages, both the resolution and orbital elements are adjusted together at a slower learning rate. There are some additional subtleties where the actual optimization function during the training of the mixture parameters has a term to encourage the model to shrink the resolution and threshold parameters. These will be discussed at greater length below.

As much as possible, I have sought to validate individual components of these calculations in isolation. My numerical integration of the planets is validated against results from NASA JPL (Jet Propulsion Library) using the superb Horizons system [4] I separately validated the numerical integration of the first 20 asteroids against positions and velocities obtained from Horizons.

GET EXACT NUMBER

The notion of a direction in space from an observer on earth is typically reported in telescopic data using a right ascension and declination. While these are convenient and standard for reporting observed data, they are not well suited to the approach taken here. All directions are represented internally in this project as a unit vector $\mathbf{u} = (u_x, u_y, u_z)$ in the Barycentric Eliptic Plane. These calculations were validated in isolation by querying the Horizons system for both the positions of and directions to known asteroids. It is vital that this calculation takes into account the finite speed of light. Treating light travel as instantaneous leads to errors that are catastrophically large in this context, on scales in the arc minutes rather than arc seconds.

The end to end calculation of a direction from orbital elements was verified indirectly as follows. I integrated the trajectories of all the known asteroids using a collection of orbital elements downloaded from JPL. I then computed the nearest asteroid number to each ZTF asteroid, and the distance between the predicted direction and observed direction. I reviewed the statistical distribution of these distances. I observed 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

---

[4] NASA Horizons
I cannot say enough good things about Horizons. If you want an external "gold standard" of where an object in the solar system was or will be and a friendly user interface, Horizons is an excelent resource.

directions of known asteroids. I took this as overwhelming evidence that these calculations were accurate.

To put this degree of precision in context, 1.0 arc second is a back of the envelope estimate of the precision with which a modern telescope can determine direection of an observation under ideal observational conditions. [5] If you were to use an approximation that observations were made at Earth's geocenter (i.e. you did not account for location of the observatorory on Earth's surface) you would already be making errors on the order of 3 arc seconds. If you were to perform your calculations using the sun's location as your coordinate origin rather than the solar system barycenter, you would make errors larger than 1.0 arc second. I know because I made both of these errors in earlier iterations before squeezing them out!

I tested the capabilities of the search process with an increasingly demanding set of search tasks. The first three search tasks involved recovering the elements of known asteroids. I took a batch of 64 asteroids that appeared most frequently in the ZTF data set. These asteroids were represented between 160 and 200 times in the data, where hits here are counted at a threshold of 2.0 arc seconds as before. Here is a summary of the tests I ran:

- Initialize search with correct orbital elements, but resolution $R = 0.5°$ and threshold $\tau = 2.0°$. All 64 elements were recovered to **???**

- Initialize search with small perturbation applied to orbital elements; a by 1.0%, e by 0.25%, i by 0.05°, remaining angles f, Ω and ω by 0.25°. 37 of 64 elements were recovered to **???**.

- Initialize search with large perturbation applied to orbital elements; a by 5.0%, e by 1.0%, i by 0.25°, remaining angles f, Ω and ω by 1.0°. 11 of 64 elements were recoverd to **???**. In some cases, a different (but correct) set of orbital elements was obtained; the perturbation was so large the search found a different asteroid.

- Initialize a search with **randomly initialized** orbital elements. Search against the subset of ZTF observations within 2.0 arc seconds of a known asteroid. This search converged on one set of orbital elements matching a real asteroid.

[5] Discussion with Pavlos Protopapas

The last last test was significantly more demanding in that it did not rely on known orbital elements.

The work encombpassed in the first three tests above can be seen as a way to indpendently validate a subset of the known asteroid catalogue. It can efficiently associate a large number of telescope observations with known asteroids, which could in turn be used to further investigate those asteroids. Analysis might include refining their estimated orbital elements, fitting the $H - G$ model of brightness (magnitude), or identifying some of them for further investigation if they meet criteria of interest, e.g. orbits that will approach near to Earth in the future.

The main thrust of this work, however, is not on refining the existing asteroid catalog, it is finding new asteroids. The final search I ran was against the subset of ZTF observations that did not match any of the known asteroids. Random initializations for orbital elements were tried. Most of these initializations fail to converge on elements with enough hits to match real asteroids in the data, but a small number do successfully converge. So far I have identified 10 asteroids with 8 or more hits. I have verified that none of the orbital elements modeled for these asteroids appear in the catalogue of known asteroids I obtained from JPL. I have also done an ad-hoc review of the ZTF records to ensure that they are plausibly belonging to the same object. I believe that these represent new and unkown asteroids, and plan to submit them to the Minor Planet Center for possible classification. **UPDATE WITH REAL NUMBERS**

The ultimate goal of this project is not to simply perform a one time search of a dataset to identify some new asteroids. The goal is rather to create a tool that will be of enduring use to astronomy community for solving the problem of searching for new asteroids given large volumes of telescopic data. To that end, I plan to consult with Matt Holman and his colleagues at the Minor Planet Center to see what refinements and improvements would be required to upgrade this from a tool I can use to one that is of wider use to the astronomy community.

Chapter 1

Integrating the Solar System

1.1 Introduction

The calculation of planetary orbits is arguably the canonical problem in mathematical physics. Isaac Newton invented differential calculatus while working on this problem, and used his theory of gravitation to solve it. In the important special case that one body in the system is a dominant central mass, and all other bodies are viewed as massless "test particles", then a simple closed form solution is possible. This formulation of the gravitational problem is often called the Kepler Problem, named after Johannes Kepler. Kepler first studied this problem and published his famous three laws of planetary motion, the first of which states that the planets move in elliptical orbits with the sun at one focus. This is a surprisingly good approximation for the evolution of the solar system, and the basis for the efficient linearized search over orbital elements developed in this thesis.

The two body approximation is not, however, sufficiently accurate for a high precision model of the past and future positions of the known bodies in the solar system. While the mass of the sun is much larger than that of the heaviest planet, Jupiter, the planets are sufficiently massive (and often closer to each other and other bodies of interest) that gravity due to their mass must also be accounted for. The modern approach to determining orbits in the solar system is to use numerical integrators of the differential equations of motion.

1.2 The REBOUND Library for Gravitational Integration

REBOUND is an open source library for numerically integrating objects under the influence of gravity. It is available on GitHub. It is a first rate piece of software and I would like to thank Matt Holman and Matt Payne for recommending it to me last year. At the end of Applied Math 225, I wrote a research paper in which I learned to use this library, extensively tested it on the solar system, and used it to simulate the near approach of the asteroid Apophis to Earth that will take place in 2029. In this project, I use REBOUND as the "gold standard" of numerical integration. Because of its important role, I describe below how the IAS15 integrator I selected works. [1]

The IAS15 integrator, presented in a 2014 paper by Rein and Spiegel, is a an impressive achievement. It a fast, adaptive, 15th order integrator for the N-body problem that is (amazingly!) accurate to machine precision over a billion orbits. The explanation is remarkably simple in comparison to what this algorithm can do. Rein and Spiegel start by writing the equation of motion in the form

$$y'' = F[y', y, t]$$

Here y is the position of a particle; y' and y'' are its velocity and acceleration; and F is a function with the force acting on it over its mass. In the case of gravitational forces, the only dependence of F is on y; but one of the major advantages of this framework is its flexibility to support other forces, including non-conservative forces that may depend on velocity. Two practical examples are drag forces and radiation pressure.

This expression for y'' is expanded to 7th order in t,

$$y''[t] \approx y''_0 + a_0 t + a_1 t^2 + \cdots + a_6 t^7$$

They next change variables to dimensionless units $h = t/dt$ and coefficients $b_k = a_k dt^{k+1}$:

$$y''[t] \approx y''_0 + b_0 h + b_1 h^2 + \cdots + b_6 h^7$$

The coefficients h_i represent relative sample points in the interval $[0, 1]$ that subdivide a time step. Rein and Spiegel call them substeps. The formula is rearranged in terms of new coefficients g_k

[1] REBOUND provides a front end to use multiple integrators. In this project, I make exclusive use of the default IAS15 integrator.

with the property that g_k depends only on force evaluations at substeps h_i for $i \leq k$.

$$y''[t] \approx y''_0 + g_1 h + g_2 h(h - h_1) + g_3 h(h - h_1)(h - h_2) + \cdots + g_8 h(h - h_1) \cdots (h - h_7)$$

Taking the first two g_i as examples and using the notation $y''_n = y''[h_n]$,

$$g_1 = \frac{y''_1 - y''_0}{h_1} \qquad g_2 = \frac{y''_2 - y''_0 - g_1 h_2}{h_2(h_2 - h_1)}$$

This idea has a similar feeling to the Jacobi coordinates: a change of coordinates with a dependency structure to allow sequential computations.

Using the b_k coefficients, it is possible to write polynomial expressions for $y'[h]$ and $y''[h]$:

$$y'[h] \approx y'_0 + hdt\left(y''_0 + \frac{h}{2}\left(b_0 + \frac{2h}{3}\left(b_1 + \ \cdots\right)\right)\right)$$
$$y[h] \approx y_0 + y'_0 hdt + \frac{h^2 dt^2}{2}\left(y''_0 + \frac{h}{3}\left(b_0 + \frac{h}{2}\left(b_1 + \ \cdots\right)\right)\right)$$

The next idea is to use Gauss-Radau quadrature to approximate this integral with extremely high precision. Gauss-Radau quadrature is similar to standard Gauss quadrature for evaluating numerical integrals, but the first sample point is at the start of the integration window at $h = 0$. This is a strategic choice here because we already know y' and y'' at $h = 0$ from the previous time step. This setup now reduces calculation of a time step to finding good estimate of the coefficients b_k. Computing the b_k requires the forces during the time step at the sample points h_n, which in turn provide estimates for the g_k, and then feed back to a new estimate of b_k.

This is an implicit system that Rein and Spiegel solve efficiently using what they call a predictor-corrector scheme. At the cold start, they set all the $b_k = 0$, corresponding to constant acceleration over the time step. This leads to improved estimates of the forces at the substeps, and an improved estimates for the path on the step. This process is iterated until the positions and velocities have converged to machine precision. The first two time steps are solved from the cold start this way.

Afterwards, a much more efficient initial guess is made. They keep track of the change between the initial prediction of b_k and its value after convergence, calling this correction e_k. At each step, the initial guess is b_k at the last step plus e_k. An adaptive criterion is used to test whether the predictor-corrector loop has converged. The error is estimated as

$$\widetilde{\delta b_6} = \frac{\max_i |\delta b_{6,i}|}{\max_i |y_i''|}$$

The index i runs over all 3 components of each particle. The loop terminates when $\widetilde{\delta b_6} < \epsilon_{\delta b}$; they choose $\epsilon_{\delta b} = 10^{-16}$. It turns out that the b_k behave well enough for practical problems that this procedure will typically converge in just 2 iterations!

The stepsize is controlled adaptively with an analogous procedure. The tolerance is set with a dimensionless parameter ϵ_b, which they set to 10^{-9}. As long as the step size dt is "reasonable" in the sense that it can capture the physical phenomena in question, the error in y'' will be bounded by the last term evaluated at $h = 1$, i.e. the error will be bounded by b_6. The relative error in acceleration $\widetilde{b_6} = b_6 / y''$ is estimated as

$$\widetilde{b_6} = \frac{\max_i |b_{6,i}|}{\max_i |y_i''|}$$

These are similar to the error bounds for convergence of the predictor-corrector loop, but involve the magnitude of b_6 rather than its change δb_6. An immediate corollary is that changing the time step by a factor f will change b_6 by a factor of f^7.

An integration step is computed with a trial step size dt_{trial}. At the end of the calculation, we compute the error estimate $\widetilde{b_6}$. If it is below the error tolerance ϵ_b, the time step is accepted. Otherwise, it is rejected and a new attempt is made with a smaller time step. Once a time step is accepted, the next time step is tuned adaptively according to $dt_{\text{required}} = dt_{\text{trial}} \cdot \left(\epsilon_b / \widetilde{b_6}\right)^{1/7}$ Please note that while the relative error in y'' may be of order 7, the use of a 15th order integrator implies that shrinking the time steps by a factor α will improve the error by a factor of α^{16}.

1.3 A Brief Review of the Keplerian Orbital Elements

In his work on the two body problem and the orbits of the planets, Kepler defined six orbital elements that are still in use today. A set of orbital elements pertains to a body as of a particular instant in time, which is typically referred to as the "epoch" in this context. The data sources I've seen all describe the time as a floating point number in the Modified Julian Day (mjd) format. In particular, I obtained orbital elements for all the known asteroids from JPL small body orbital

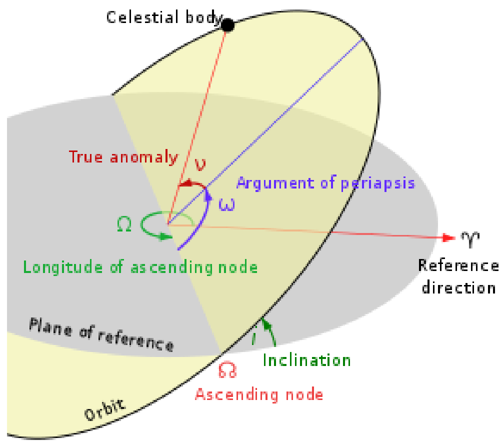

Figure 1.1: *Definition of the traditional Keplerian orbital elements orbital elements, courtesy of Wikipedia. Two parameters define the shape and size of the ellipse; two define the orientation of the orbital plane; and the last two orient the ellipse in its plane and the phase of the body on its ellipse.*

elements as of MJD 58600, corresponding to 27-Apr-2019 on the Gregorian calendar.

Here is a brief review of the definitions of these orbital elements

- a, the semi-major axis; named `a` in JPL and REBOUND
- e, the eccentricity; named `e` in both systems
- i, the inclination; named `i` in JPL and `inc` in REBOUND
- Ω, the longitude of the ascending node; named `node` in JPL and `Omega` in REBOUND
- ω, the argument of perihelion; named `peri` in JPL and `omega` in REBOUND
- f, the true anomaly; named `f` in REBOUND; not quoted directly by JPL
- M, the mean anomaly; named `M` in both systems
- mjd, the epoch as a Modified Julian Date

Distances are in A.U. in both JPL and REBOUND.

Angles are quoted in degrees in JPL and in radians in REBOUND.

These orbital elements have stood the test of time because they are useful and intuitive. They are ideal for computations, both theoretical and numerical, because in the case of the two body

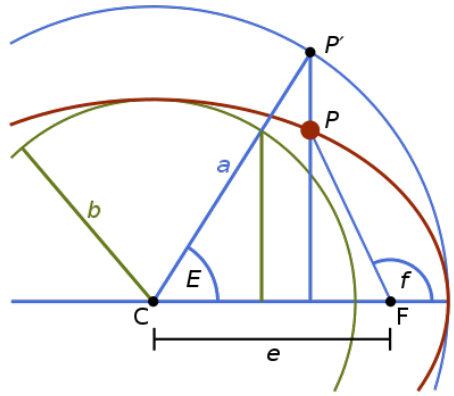

Figure 1.2: *Three Orbital Anomalies: Eccentric, Mean and True*

problem five of the six orbital elements remain constant. The careful reader will note that there are 8 entries in the table above, but I've described elements as coming six at a time. The epoch is considered to be the "seventh element" because in the Kepler two body problem, we can describe one body at different times, but it will have the same orbit. This point of view extends to the N-body problem, which is fully reversible; the same system can be described at at different moments in time. In practice, the orbital elements are often used to describe the initial conditions of all the bodies for an integration. The problem is then integrated numerically, possibly both forwards and backwards. Orbital elements can be reported for any body of interest.

A body orbiting the sun has six degrees of freedom. In Cartesian coordinates, there are three for the position and three for the velocity. In orbital elements, the first five are almost always $(a, e, i, \Omega, \omega)$. These five will remain constant for a body moving in the Kepler two body problem.

There is some variation in the choice of the sixth element, because different representations have different pros and cons. The true anomaly f is most convenient for transforming back and forth between orbital elements and Cartesian space. The mean anomaly M is most convenient for studying the time evolution of the system, because it changes linearly with time in the Kepler two body problem. The mean anomaly and true anomaly are related by the famous Kepler's Equation This relates the mean anomaly M to the eccentric anomaly E. The eccentric anomaly is yet another angle describing a body in orbit.

$$\tan\left(\frac{f}{2}\right) = \sqrt{\frac{1+e}{1-e}} \tan\left(\frac{E}{2}\right)$$

$$M = E - e\sin(E)$$

The linear evolution of the mean anomaly, along with Kepler's equation, allows us to efficiently compute orbits for the Kepler two body problem. The relationship between the eccentric anomaly E and true anomaly f is a one to one function that can be evaluated fast on a computer. The mapping from eccentric anomaly E to mean anomaly M is also fast. The inverse mapping from M to E does not have a known analytical form. But it can be evaluated raplidly using Newton's Method with a reasonable initial guess. This is the method that I use to compute the orbits under the Kepler approximation.

1.4 Numerical Integration of the Planets and Asteroids

I have described above a library `REBOUND` that can efficiently integrate the solar system, and a data source Horizons that can be used to obtain accurate initial conditions for solar bodies. In principle integrating the solar system is a straightfoward exercise. In practice, there are quite a few details that need to be worked out before you can obtain reliably correct answers. You need to carefully specify the bodies you submit to Horizons. Horizons has separate identifiers for e.g. the barycenter of the Earth-Moon system, the Earth, and the Moon.

The module `horizons.py` contains functions used to query the Horizons AP. It also maintains a local cache with the results of prior queries; this yields significant savings in time because a typical horizons query using the Horizons API in `REBOUND` takes about one second. The main function in this module is `make_sim_horizons`. Given a list of object names and an epoch, it queries Horizons for their positions and velocities as of that date. It uses this data to instantiate a `REBOUND Simulation` object.

The module `rebound_utils.py` contains functions used to work with `REBOUND` simulations. It includes functions to build a simulation (`make_sim`). This will seek to load a saved simulation on disk if it is available, otherwise it will query Horizons for the required initial conditions. The function `make_archive` builds a `REBOUND SimulationArchive`. As the name suggets, a `SimulationArchive` is a collection of simulation snapshots that have been integrated. This function also saves the integrated positions of the planets and test bodies as plain old `Numpy` arrays for use in downstream computations.

The module `planets.py` performs the numerical integration of the planets. To be more precise, it will integrate different collections of massive bodies in the solar system

- **Planets**: The Sun; The Earth and Moon as separate bodies; and the barycenters of the other seven IAU planets Mercury, Venus, Mars, Jupiter, Saturn, Uranus, and Neptune (10 objects)

- **Moons**: The 8 IAU planets, plus the following significant moons and Pluto (31 objects):
  Jupiter: Io, Europa, Ganymede, Callisto
  Saturn: Mimas, Enceladus, Tethus, Dione, Rhea, Titan, Iapetus, Phoebe
  Uranus: Ariel, Umbriel, Titania, Oberon, Miranda
  Neptune: Triton, Proteus
  Pluto: Charon

- **Dwarfs**: All objects in the solar system with a mass at least $1E-10$ Solar masses (31 objects):
  Planets: Earth, Moon, and barycenters of other seven planets
  Above 1E-9: Pluto Barycenter, Eris, Makemake, Haumea
  Above 1E-10: 2007 OR10, Quaoar, Hygiea, Ceres, Orcus, Salacia, Varuna, Varda, Vesta, Pallas

- **All**: All objects in the solar system with a mass at least $1E-10$ Solar masses (45 objects):
  All 8 planets (not barycenters)
  All the heavy moons above
  All the dwarf planets above

Each configuration above was integrated for a 40 year period spanning 2000-01-01 to 2030-12-31 and a time step of 16 days. I tested the integration by comparing the predicted positions of the 8 planets to the position quoted by Horizons at a series of test dates . The test dates are at 1 year intervals over the full 40 year span that is simulated. The best results were obtained by integrating smallest collection: Earth, Moon, and the barycenters of the other 7 planets. I was a bit surprised at this result and expected to do slightly better as the collection of objects became larger. Position errors are reported in AUs, with the root mean square (RMS) error over the 40 annual dates. I

Object Collection	Position Error	Angle Error
Planets	5.38E-6	0.79
Moons	1.35E-5	0.81
Dwarfs	5.38E-6	0.79
All	1.35E-5	0.81

Table 1.1: *Root Mean Square Error in Integration of Planets vs. Horizons*
Position Error: RMS error of 8 planets in AU.
Angle Error: RMS error in direction from planet to Earth geocenter, in Arc Seconds

also compute an angle error by comparing the instantaneous direction from each planet to Earth geocenter in the BME frame. I reported errors on this basis because on this problem, everything is done in terms of directions so precision eventually comes down to a tolerance in arc seconds.

While it might at first seem surprising that the results are worse for the more complex integrations including the moons, it's important to realize that this problem is intrinsically more difficult. Simulating the evolution of the barycenter of e.g. the Jupiter system is significantly easier than keeping track of the heavy moons and integrating them separately. Overall these results are excellent; over a span of 20 years in either direction, integrations are accurate on the order of 10^{-6} AU. The angular precision on the order of ~ 0.8 arc seconds is also excellent for such a long time span and well within the tolerance of this application.

After reviewing these results, I decided that the optimal strategy for the asteroid search problem was to treat the heavy bodies in the solar system as the smallest collection, shown on row 1. It is necessary to model the position of the Earth and Moon separately rather than the Earth-Moon barycenter, since our observatories are on the planet, not relative to the planetary system barycenter. However, the role of the other planets is only as a gravitational attractor that deflects the orbit of the Earth and the Asteroids. Speed is important in this application, so the smallest and fastest collection was the clear choice.

The second test of the integration of the planets was a "soup to nuts" test with the integration of the planets, plus ten test asteroids. I selected as the test asteroids the first 10 IAU numbered asteroids: Ceres, Pallas, Juno, Vesta, Iris, Hygiea, Egeria, Eunomia, Psyche, Fortuna. This test does not yet exercise the part of the code that insantiates asteroid orbits based on the bulk orbital elements files; that comes later. The asteroids here are initialized the same way as the planets, by querying the Horizons API in `REBOUND`. (This method would not scale up to integrating all the asteroids though, because it is far too slow at about 1 second per asteroid.) The test protocol here was the same as for the planets. I compared the positions of these asteroids in the barycentric mean ecliptic frame predicted by my integartion at annual dates to the positions quoted by JPL. I also compared the instananeous angle from Earth geocenter to the asteroid.

Here is a chart summarizing the results.

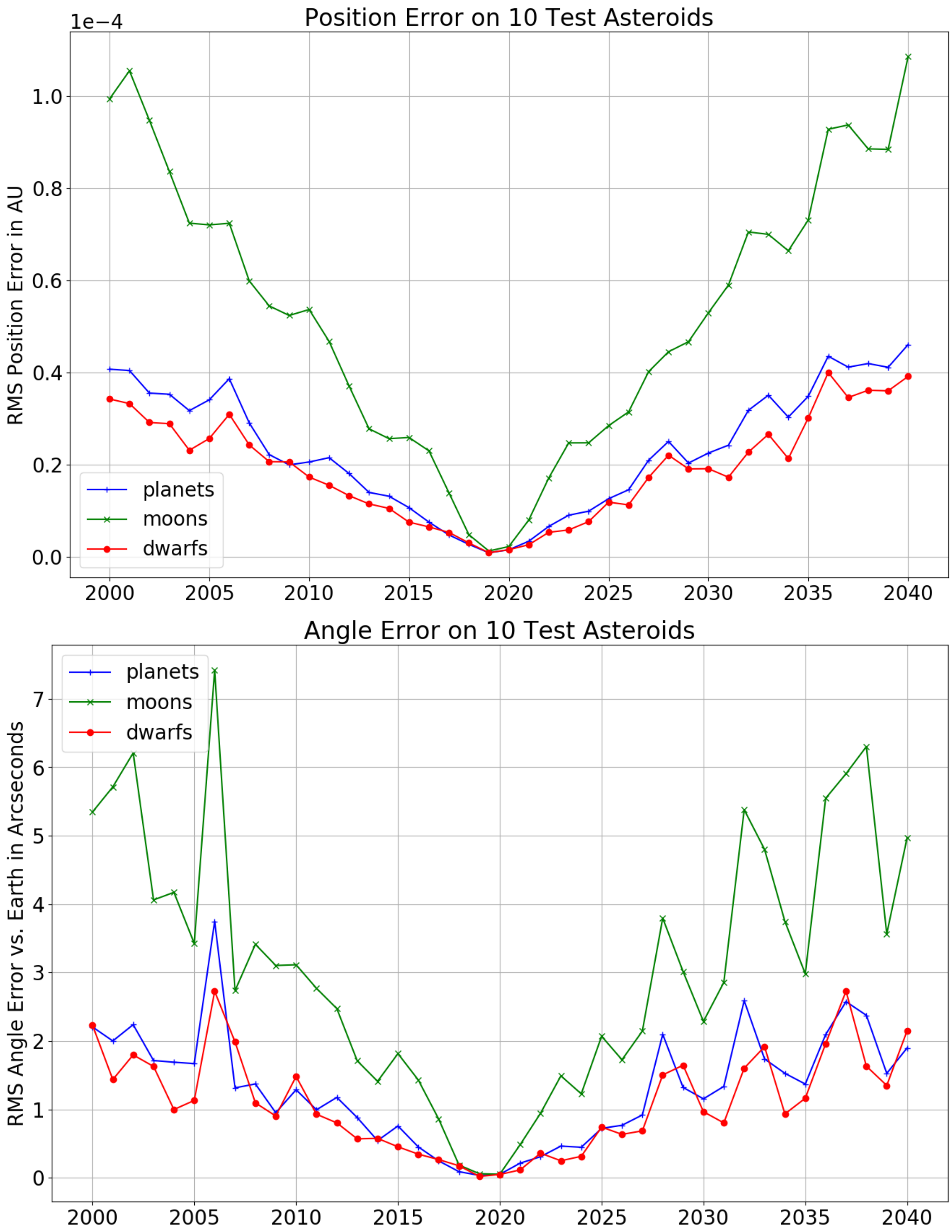

Figure 1.3: *Position and Angle Error of 10 Test Asteroids*

Chapter 2

Predicting Directions from Positions

2.1 Introduction

2.2 Potential outcomes framework

Chapter 3

Searching for Asteroids

3.1 Introduction

Some people just cite papers in introductions for no reason.

3.2 Setup

3.3 Conclusion

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