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 12 new, previously unknown asteroids with at least 8 hits within 10 arc seconds on ZTF asteroid detections.

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

Chapter 1

Asteroid Search Results

1.1 Comparing Candidate Elements to the Nearest Asteroid

Before we review the results of our asteroid search experiments, it will be helpful to have in hand a notion of how closely two sets of orbital elements match. In particular, we will test below whether or not we successfully recovered orbital elements when we started with them as an initial guess. Answering this question requires that we have a useful metric on the space of orbital elements.

I spent a fair amoutn of time trying to develop such a metric. While orbital elements are convenient for intuition and calculating orbits, there isn't an obvious distance metric we can put on them that makes a lot of sense. Eventually I decided that the canonical way to compare two orbits by comparing the predicted vector of positions on a set of representative dates. Logically this is hard to argue with, but it is somewhat computationally expensive compared to a computation that can be run directly on the elements.

The method `nearest_ast` searches the asteroid catalogue for the known asteroid whose orbit is closest to that predicted by the candidate elements. It delegates its work to the function `nearest_ast_elt_cart`, which is defined in `nearest_asteroid.py`. This function creates a set of 240 sample time points over 20 years spanning 2010 to 2030 sampled monthly. The resulting table of positions for the asteroid catalog is fairly large, with a size of $[733490, 240, 3]$ (5.28E8 elements and about 2.11 GB using 32 bit floats). Computing the nearest asteroid against 64 candidate elements by brute force in TensorFlow would necessitate creating a tensor with 3.38E10 elements or 135 GB of memory–two orders of magnitude too large for a high quality consumer

grade GPU with ~ 10 GB of memory. The nearest asteroid method is therefore forced to iterate through the elements one at a time, taking the norm of the difference against the table. In one important optimization, the tensor of known asteroid positions in loaded into memory once as a TensorFlow constant to avoid recomputing it every time the function is called.

I also sought to develop a sensible metric of the distance between a pair of arbitrary orbital elements. This is implemenented in the same module with the function `elt_q_norm` and `nearest_ast_elt_cov`. The idea is to transform the elements to a Cartesian representation where they have a well behaved covariance matrix. In particular, the goal is to find a deterministic transform of the elements that is distributed approximately as a multivariate normal. Then the Mahalanobis distance is a natural metric on the transformed elements. This process can be reviewed in the Jupyter notebook `11_nearest_asteroid.ipynb`. I initially tried working with the full empirical distribution to convert every orbital element to a percentile and then to a normally distributed z score, but the results didn't make any sense, so I dropped that approach.

Instead, I switched to a simpler approach and standardized variables so they would have mean zero and variance 1. I attempt to make them close to normal if possible, but without overfitting against the empirical distribution. I standardized the log of the semimajor axis a and directly standardized the eccentricity e. (Even though this admits mappings from z to eccentirities outside $[0, 1]$, the mapping is only used in the direction from a reported eccentricity e to a transformed e_z that is approximately normal). The quantity $\sin(inc)$ was also also standardized. Here is a summary of the mathematical transformations to create approximately normal variables from a, e and i:

$$
\begin{aligned}
a_z &= \frac{\log(a) - \mathrm{E}[\log(a)]}{\sqrt{\mathrm{Var}[\log(a)]}} \\
e_z &= \frac{e - \mathrm{E}[e]}{\sqrt{\mathrm{Var}[e]}} \\
i_z &= \frac{\sin(i) - \mathrm{E}[\sin(i)]}{\sqrt{\mathrm{Var}[\sin(i)]}}
\end{aligned}
$$

The expectation and variance here are estimated using the sample mean and sample variance respectively.

Here are visualizations of comparing the hypothetical and empirical distributions of a and e:

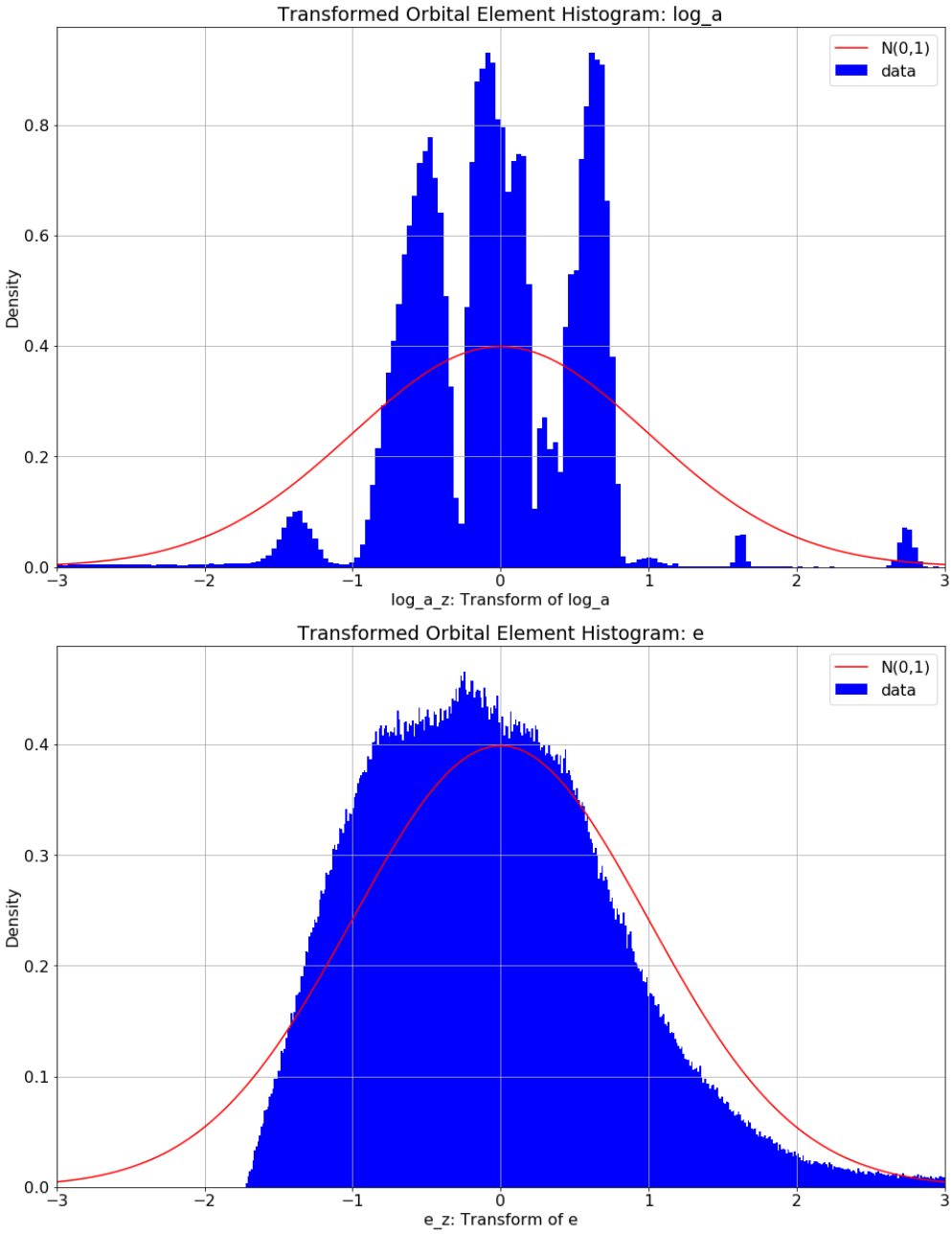


Figure 1.1: *Transformations of a and e to standardized (and "approximately" normal) variables a_z and e_z.*

The other three angular orbital elements Ω, ω and f are handled identically. We can inject i into Cartesian space with only its sine because it is constrained to $[-\pi, \pi]$. But the other three angles are unconstrained. I will take Ω as an example. I transform Ω into *two* variables, named `cos_Omega_z` and `sin_Omega_z`. These are not transformed empirically, but using a theoretical distribution. Let x be the sine or cosine of one of Ω, ω or f. x is mapped to a variable z that is distributed appproximately normal by applying the tranformation

$$u = \frac{1/2 + \arcsin(x)}{\pi}$$
$$z = \Phi^{-1}(u)$$

where Φ is normal CDF.

Below are visualizations of comparing the hypothetical and empirical distributions of $\sin(i)$ and $\sin(\Omega)$:

Better results for e and i could be obtained by using the Beta distributions noted above, for this purpose the simple standardization is adequate. The plot shown for the tranform of $\sin(\Omega)$ shows that it is very close to the theoretical distribution. I generated analogous plots for the sin and cos of Ω, ω and f which are in the Jupyter notebook. They are qualitatively similar to this one and all show excellent fits.

I have now given a recipe with which six orbital elements can be injected into R^9. Let X be the $Nx9$ matrix of transformed elements (N = 733,489 is the number of asteroids). The orbital elements are only very lightly correlated with each other, and so are the X_j except for the tightly correlated pairs with the sin and cos of the same angle. Next, using the Spectral Theorem, I find a $9x9$ matrix β such that the covariance matrix of $X\beta$ (which also has shape $Nx9$) is the $9x9$ identity matrix. The only wrinkle is that I assign importance weights to the 9 columns before building X and computing β. The importance weights are:

- 1.0 for a_z and e_z
- 0.5 for i_z
- 0.1 for the sin and cos of Ω, ω and f

These are admittedly qualitiative judgments on my part. I initially only compensated for the

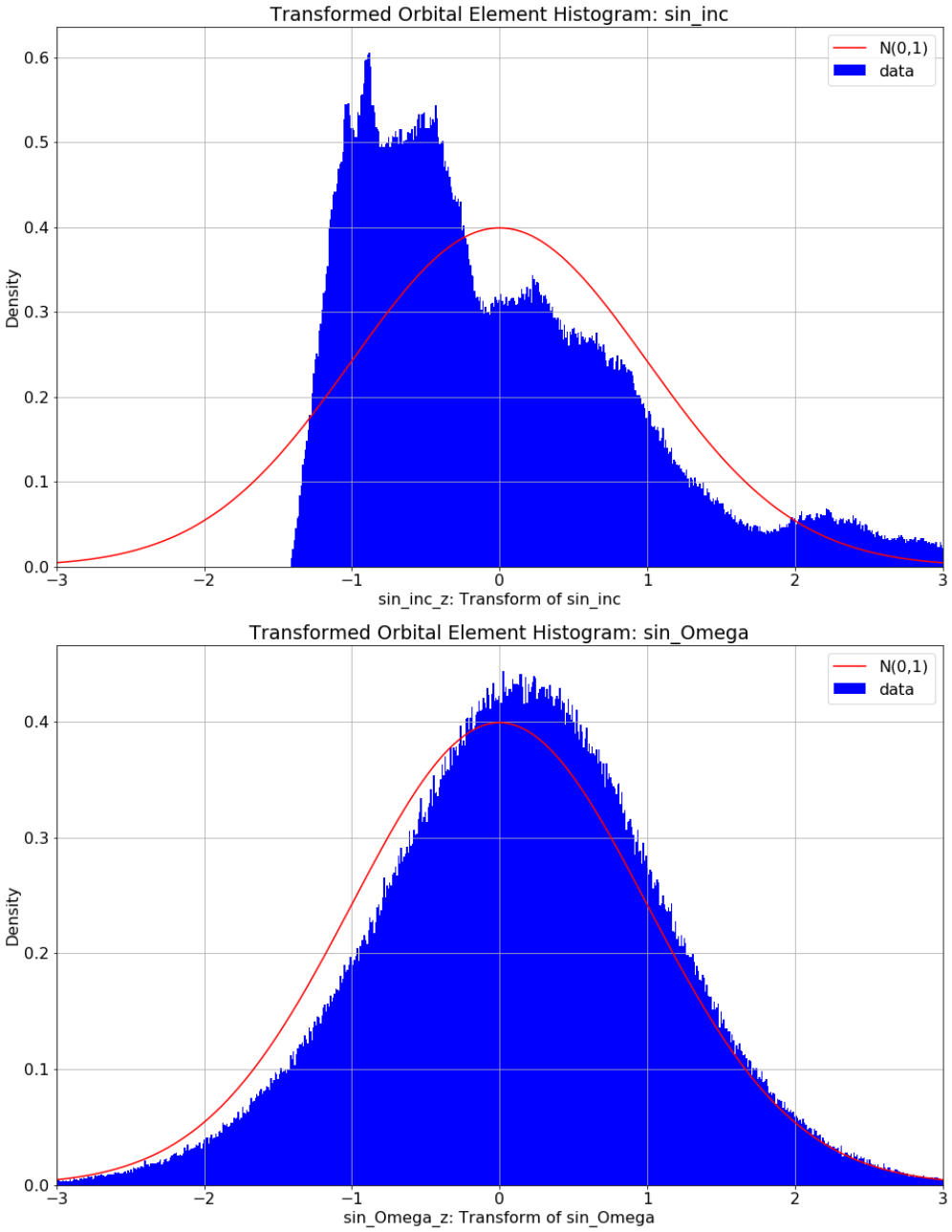


Figure 1.2: *Transformations of a and e to standardized (and "approximately" normal) variables a_z and e_z.*

double counting of Ω, ω and f, but I noticed that relatively small differences in e.g. ω on a near circular orbit that hardly effected the shape of an orbit were having a disproportionately large influence of covariance score.

The covariance metric between two sets of orbital elements is ϵ_1 and ϵ_2 is defined by

$$\|\epsilon_2 - \epsilon_1\|_{\text{cov}} = \|\epsilon_2\beta - \epsilon_1\beta\|$$

The importance weights are rescaled so the diagonal of the covariance matrix sums to 1 and a random pair of elements should have distance 1. These calculations are also in `nearest_asteroid.py` and done by the functions `elts_to_X_cov`, `calc_beta`, `elt_q_norm` and `nearest_ast_elt_cov`. Now that we know what it means for two orbital elements to be "close," we are ready for our first test: recovering unperturbed elements.

1.2 Recovering the Unperturbed Elements of Known Asteroids

The first and easiest proof of concept for the search process is to see if the mixture parameters will converge correctly when the search is initialized with correct orbital elements for asteroids that are well represented in the data, but with "neutral" or uninformative mixture parameters. I liken this test to a kid learning to swing a bat by trying to ball sitting on a tee. This test is demonstrated in the Jupyter notebook `14_asteroid_search_unperturbed.ipynb`.

It's worthwhile to follow through the steps to assemble the data to get familiar with how everything fits together. These two lines of codes load the ZTF data observations associated with the nearest asteroid, and count hits by asteroid number:

```
ast_elt = load_ztf_nearest_ast()
ast_num, hit_count = calc_hit_freq(ztf=atf_ast, thresh_Sec=2.0)
```

The next few lines sort the asteroids in descending order by number of hits, and assemble a data frame of the orbital elements belonging to the 64 "best" asteroids The function `asteroid_elts` in `candidate_elements` provides a batch of candidate orbital elements that exactly match known asteroids; it assigns an `element_id` matching the original asteroid number to make it easy to check later if the fitted elements match the original. The function `load_ztf_batch` assembles the batch of ZTF observations within a threshold, here 2.0 degrees, of these candidate elements

```
elts_ast = asteroid_elts(ast_nums=ast_num_best[0:64])
ztf_elt = load_ztf_batch(elts=elts_ast, thresh_deg=2.0)
```

Reviewing the `ztf_elt` on screen we can see that there are 322,914 rows which include 10,333

hits: an average of 161.5 hits per candidate element and 3.2% of the total rows of data. A call to `score_by_elt` computes the t-score described earlier based on the mean and standard deviation of $\log(v)$. This shows a mean t-score of +45.0, which is off the charts good. It's interesting to see that a set of observations with 3.2% hits and 96.8% noise achieves such a good score. This also puts into context the challenge of the search problem: we have an average of 5045 detections within 2.0 degrees of each set of candidate elements, of which 160 are hits and the remaining 4885 are random detections belonging to other asteroids. If we want a search process to detect asteroids with as few as 8 hits in the data, we will need a process selective enough to pick out just 0.16% of the observations.

To initiate the search, we also need to choose our initial mixture parameters. We set `num_hits` to 10 and the resolution to 0.5 degrees

```
elts_add_mixture_params(
        elts=elts, num_hits=num_hits,
        R_deg=R_deg, thresh_deg=thresh_deg)
```

Now that we have the candidate elements and the ZTF data frame, we are ready to instantiate the asteroid search model:

```
model = AsteroidSearchModel(
        elts=elts_ast, ztf_elt=ztf_elt,
        site_name='palomar', thresh_deg=2.0)
```

Before we start training the model, we can get a plain text report or a visualization of the starting point. I will omit these here. The report shows that at the start of training, all 64 elements are "good" with 5 or more hits, and the overall mean log likelihood is 3.13 and the mean number of hits is 162.7.

Here is an excerpt from the plain text model report at the end of training:

```
model.report()
Good elements (hits >= 5):  64.00
         \  log_like :  hits  :    R_sec : thresh_sec
Mean Good:  1096.95  : 162.55 :     3.01 :   165.95
Mean Bad :      nan  :    nan :      nan :      nan
Mean     :  1096.95  : 162.55 :     3.01 :   165.95
Median   :  1086.03  : 161.00 :     2.26 :   140.88
GeoMean  :  1092.15  : 162.03 :     2.79 :   157.15
Min      :   890.84  : 146.00 :     1.87 :   109.56
Max      :  1336.11  : 201.00 :     8.07 :   360.24
Trained for 7808 batches over 122 epochs and 49 episodes (elapsed time 312 seconds).
```

Figures 1.3 and 1.4 illustrate the training progress. We can see that the model is behaving as

hoped. It is gradually ratcheting the resolution parameter and scoring a high log likelihood as it does so. It does this without getting deked and polluting the originally correct orbital elements.

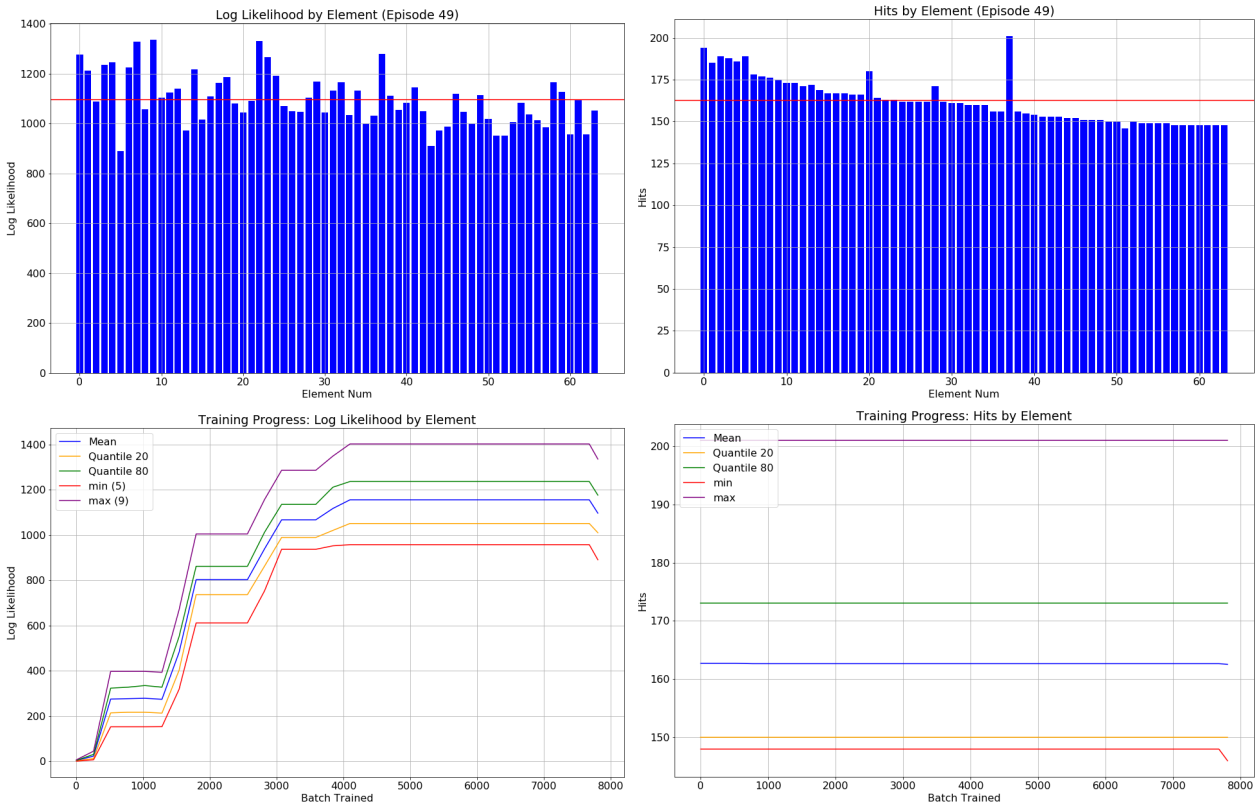


Figure 1.3: *Training progress on 64 unperturbed orbital elements.*

These are very happy diagnostics. The log likelihood averages over 1,000 and the hits average 160. An earlier iteration of the training protocol that did not roll back training when hits were lost in an episode did considerably less well.

The diagnostics presented above work equally well for any set of candidate orbital elements, whether or not they are ostenisbly associated with a known asteroid. In this case, we can further validate the results by comparing our fitted elements to the nearest asteroid using the two metrics described in the previous section. The simplest test is how many of 64 recovered elements have as their nearest asteroid the same asteroid used to initialize the elements. The answer is 64: the fitting process "tried" to converge back to the right asteroid every time. A more substantive question is how close did it come. Here are the summary statistics of two metrics:

- Mean distance in AU for 240 test points: 4.57E-8 (median over 64 elements)

Figure 1.4: *The resolution R decreases monotonically when training the unperturbed elements.*

- Covariance Norm of elements: 5.70E-6 (geometric mean)

This is an excellent level of agreement. The covariance norm is a good summary statistic, but may be hard to relate to astronomy. The mean absolute error in the recovered a is 4.3E-5 and in the recovered e is 1.0E-5. Thanks to the one way ratchet that prevents it from losing hits once they acquired, the model can hit a tee ball out of the infield. In the next section we will see how it fares against a moving target.

Figure 1.5 illustrates the distance in AU and covariance norm to the nearest (original) asteroid for all 64 candidate elements.

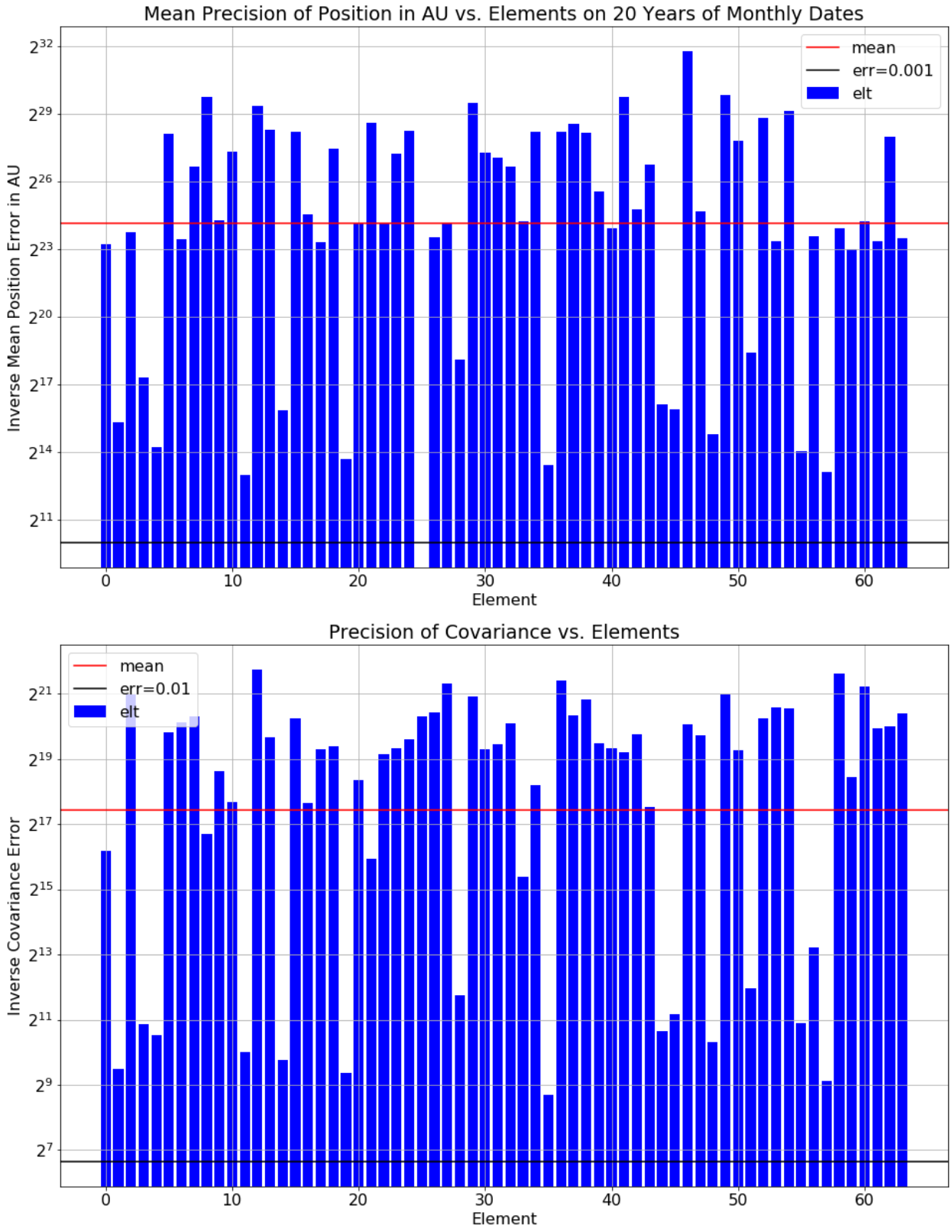


Figure 1.5: *Two metrics comparing the recovered orbital elements to the true elements of the asteroid in question. Both charts are plotted on a log scale with preicision (reciprocal of the error) on the y axis. The geometric mean error is shown in red: 6.61E-6 AU and 2.00E-3 on the covariance norm.*

1.3 Recovering the Perturbed Elements of Known Asteroids

1.3.1 Small Perturbation

The next experiment is similar to the previous one. This time we will apply a small perturbation to the orbital elements in our initial guess. If the last experiment was like hitting a tee ball, this one may be likened to hitting a ball gently pitched by your little league coach in batting practice. The elements are perturbed using the function `perturb_elts` in `candidate_elements.py`. The perturbation adds normally distributed random noise with the specified standard deviation to $\log(a)$, $\log(e)$, and the four angles i, Ω, ω and f. The small perturbation shifts $\log(a)$ by 0.01, $\log(e)$ by 0.0025, i by 0.05 degrees, and the the other angles by 0.25 degrees. A random seed is used for reproducible results. The code to do this is

```
elts_pert= perturb_elts(elts_ast, sigma_a=0.01, sigma_e=0.0025,
        sigma_inc_deg=0.05, sigma_f_deg=0.25,
        sigma_Omega_deg=0.25, sigma_omega_deg=0.25,
        random_seed=42)
```

Last time the pre-training summary statistic based on $\log(v)$ showed a very positive t score. This time the t score has dropped to $+3.71$, and the model has zero hits before it begins training. Even this small perturbation is enough that the model is going to have to work quite a bit to recover the elements.

Here is the text report after sieving:

```
Good elements (hits >= 5):  42.00
         \  log_like :  hits  :    R_sec : thresh_sec
Mean Good:   798.24  : 117.50 :    36.06 :   779.79
Mean Bad :    42.97  :   0.64 :   266.21 :  2374.53
GeoMean  :   218.79  :  20.01 :    42.97 :   934.96
Trained for 15552 batches over 243 epochs and 105 episodes (elapsed time 751 seconds
```

Here are summary statistics for the run on the small perturbation of real asteroid elements:

- Successfully converged for 42 out of 64 candidate elements (65.6%)
- Mean hits on converged elements: 117.50
- Resolution on converged elements: 18.2 arc seconds
- Distance in AU to nearest asteroid: 2.58E-4
- Covariance Norm to nearest asteroid: 1.22E-2

These results are not as strong as on the unperturbed elements, but the method is still clearly working. It's coverged on almost two thirds of the candidate orbital elements. The converged elements are fit well, averaging 117 hits at 18 arc seconds. The distance to the nearest asteroid is 2.58E-4 AU, which is still very close and an excellent description of the orbit. The nearest asteroid to the recovered elements matches the original asteroid on 48 out the 64 elements.

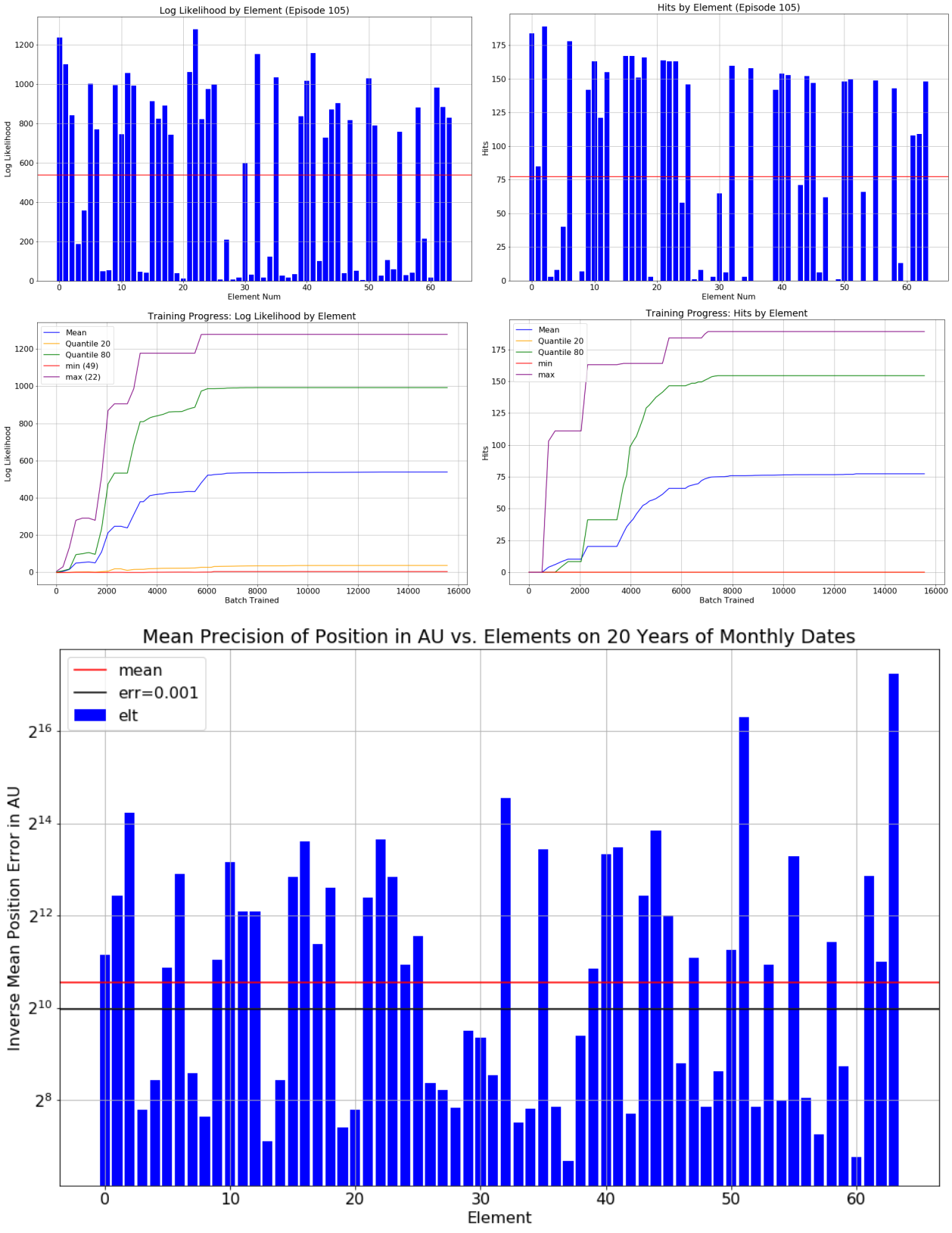


Figure 1.6: *Training progress on 64 orbital elements initialized with small perturbations from real asteroids. 42 of the 64 candidate elements converge, averaging 118 hits each.*

1.3.2 Large Perturbation

In our third test, we will again start with perturbed orbital elements. But this time, we will apply a larger perturbation, about five times larger. This is a much harder task. To continue with the baseball analogy, it might be likened to facing a high school pitcher. The perturbation size this time is 0.05 on $\log(a)$, 0.01 on $\log(e)$, 0.25 degrees on i, and 1.0 degree on the other three angles. While this might not sound like much at first, they are large perturbations. In fact, they are so large they led me down a painful rabbit hole. I repeatedly failed to recover the orbital elements of the original asteroids before I realized that the perturbations were large enough that in many cases, the nearest asteroid to the perturbed elements was no longer the original asteroid! The results started to make much more sense when I compared each fitted element to the nearest real asteroid, regardless of whether this matched the original source of the elements before perturbation.

Here is the text report after sieving:

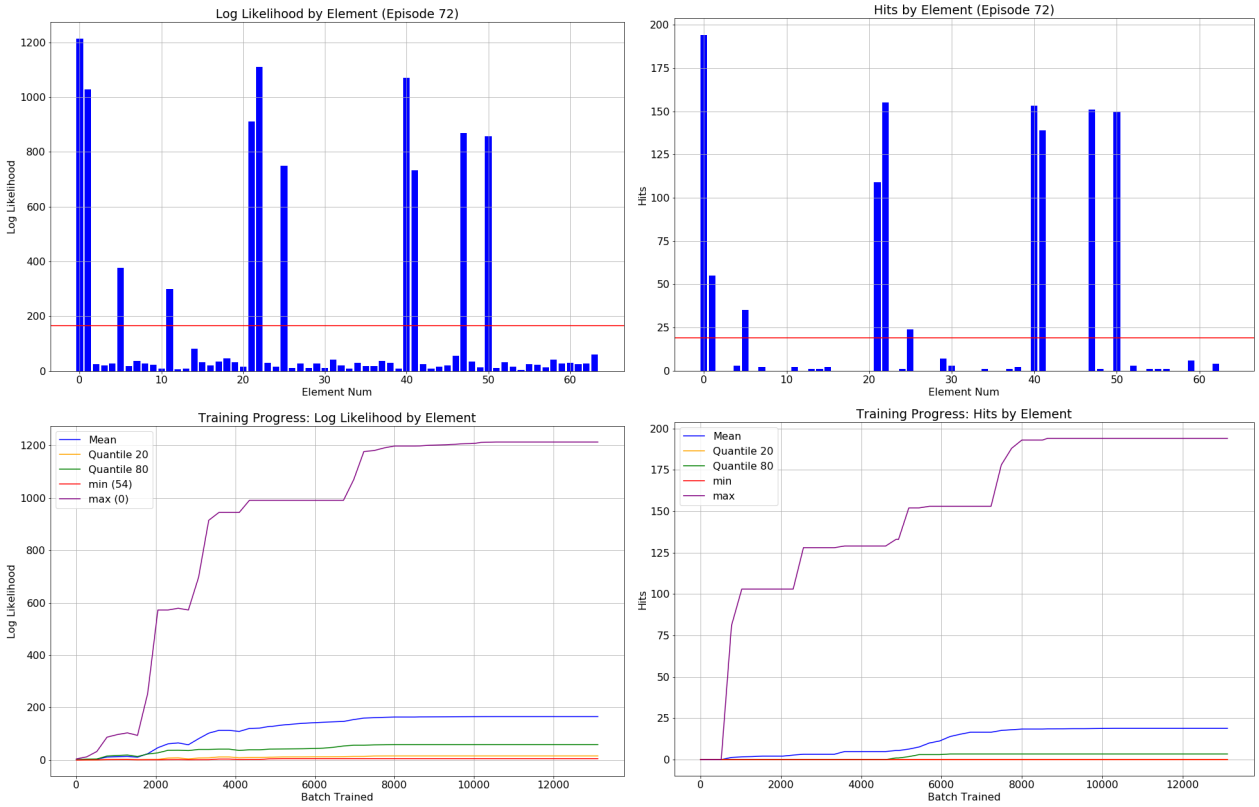
```
Good elements (hits >= 5):  12.00
         \  log_like :  hits  :    R_sec : thresh_sec
Mean Good:   748.07  :  98.17 :    61.36 :  1178.94
Mean Bad :    30.35  :   0.58 :   261.02 :  2296.18
Trained for 13120 batches over 205 epochs and 72 episodes (elapsed time 533 seconds).
```

Here are summary statistics for the run on the small perturbation of real asteroid elements:

- Successfully converged for 12 out of 64 candidate elements (18.8%)
- Mean hits on converged elements: 98.2
- Resolution on converged elements: 32.4 arc seconds
- Distance in AU to nearest asteroid: 4.45E-4
- Covariance Norm to nearest asteroid: 3.32E-2

This time we've only converged on 12 of the 64 orbital elements. But the encouraging news is that when we have converged, the fit is still adequately good. The average hits are 98 and the resolution is 32.0 arc seconds. The distance to the nearest asteroid is somewhat larger to the batch initialized with small perturbations, at 4.5E-4. This is telling us something important: when the model starts from a good enough guess that it has a path in search space to the local

maximum, it will converge to an adequate solution. Starting from a poor initialization will reduce the probability of successful convergence, but it doesn't dilute the quality of the results.



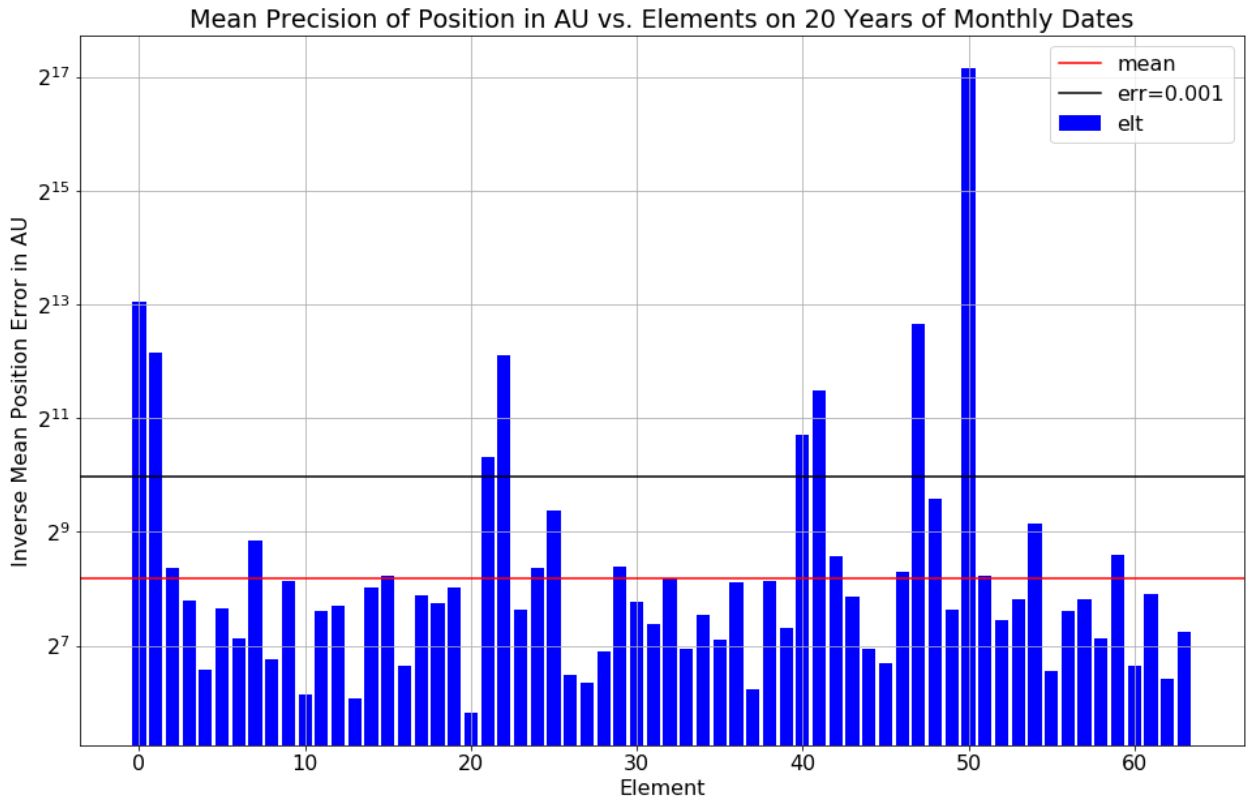


Figure 1.7: *Training progress on 64 orbital elements initialized with large perturbations from real asteroids. 9 of the 64 elements converge, averaging 87 hits.*

1.4 Searching for Known Asteroids with Random Initializations

Our final test case before search for new asteroids is to attempt to recover asteroids in the known catalog, but without peeking at the answers. I will now transition from small tests on a single batch 64 candidate elements to analyisis of the results of a large scale computational job. The program `asteroid_search.py` can be run from the command line. It searches against one of two subsets of the ZTF data set. When run in "known asteroids" mode, the ZTF observations are filtered to include only the 3.69 m rows that are within 2.0 arc seconds of a known asteroid. When run in "unknown asteroids" mode, it searches in the complement, those ZTF detections that are at least 2.0 arc seconds from a known asteroid. The other arguments include the range of random seeds `seed0` to `seed1` and a stride to support parallelization.

This program was run on approximately 4096 random seeds against the known asteroids over the better part of a week. Once a ZTF observation was associated with a set of candidate elements, it was subtracted from the data set so it would not be included in subsequent fits. I filtered the results to those with at least 8 hits and a resolution of at most 20 arc seconds. The results are reviewed in the Jupyter notebook `19_search_known.ipynb`. Since this was a search against known orbital elements, we can gauge the quality by measuring the distance of the recovered orbits to the nearest known asteroid. Here are the summary statistics for the resulting fitted elements:

- 125 fitted orbital elements were found
- they had 19.20 hits on average
- the geometric mean resolution was 9.3 arc seconds
- the geometric mean distance to the nearest asteroid was 2.66E-3 AU
- the geometric mean covariance norm to the nearest asteroid as 0.73

Figure **??** shows training results for the fitted elements. The first row has the internal metrics number of hits and log likelihood. The second row has the comparison to the known asteroids. I thought these results were respectable but not great. I will need to significantly improve the initializations to get a higher yield. The plots with the precision to the nearest asteroid are uneven.

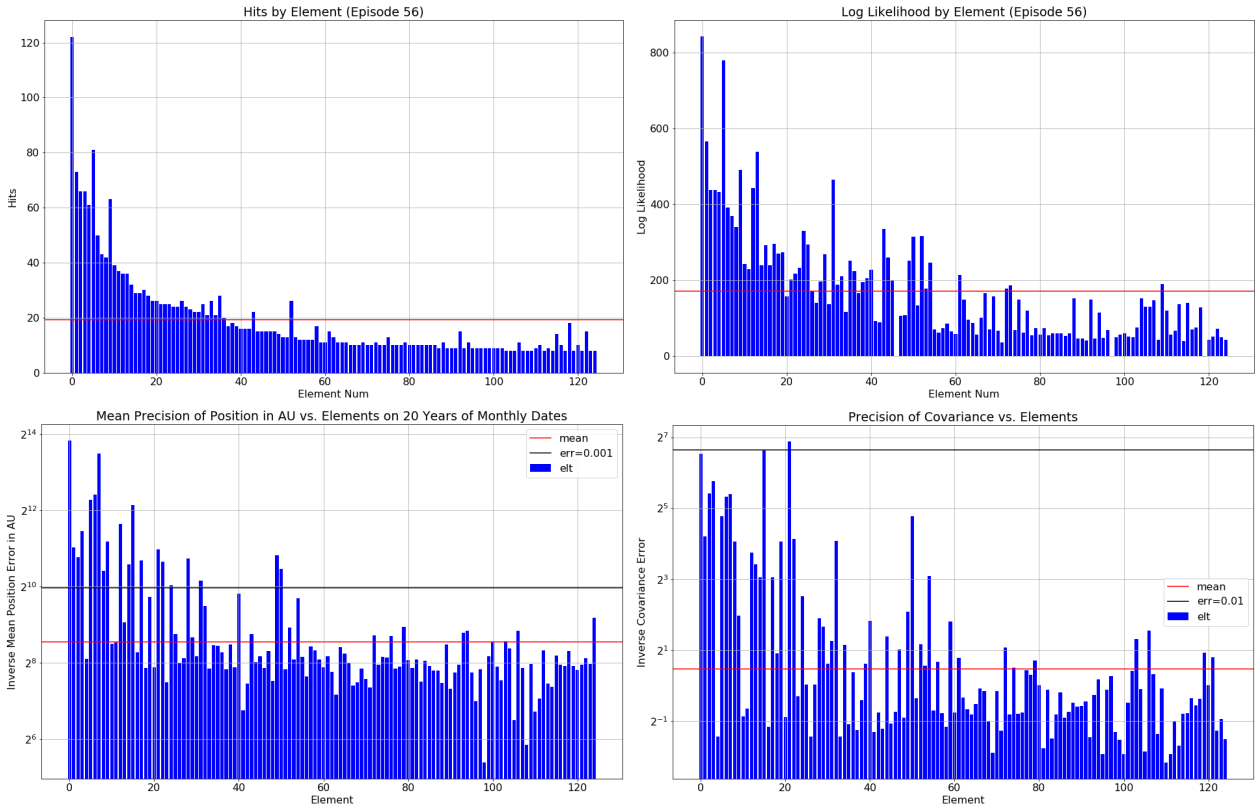


Figure 1.8: *125 orbital elements fitted to observations of known asteroids.*
The first row shows internal estimates of training quality: number of hits and log likelihood.
The second row shows distance to the nearest known asteroid in mean distance of orbit and covariance metric.

Some of these elements have clearly achieved excellent agreement with known asteroids, with positions accurate to 1E-3 AU or better and covariance norms bettern than 0.1 (this is very close in orbital element space). While there is room for improvement, I view this as a successful proof of concept that this technique can recover correct orbits of asteroids in the catalogue without peeking at the correct orbital elements.

1.5 Presenting 9 New Asteroid Candidates

The search for new undiscovered asteroids is very similar to the search discussed in the previous section. The only difference is that this time we limit the search to the subset of ZTF observations more than 2.0 arc seconds away from any known asteroid. A search was carried out starting from 4,096 random seeds, taking about 5 days running on 4 GPUs in parallel. This search identified

9 candidate orbital elements that achieved at least 8 hits within 10 arc seconds and a converged resolution less than 20 arc seconds. These are the same criteria used to select the ostensibly high quality recovered orbital elements of known asteroids presented in the last section.

element_id	a	e	inc	Omega	omega	f	epoch	num_hits	R_sec	thresh_sec	num_rows_close	log_like
178421	3.160327	0.089064	0.153620	2.668766	4.773995	-0.463213	58600.0	10.996569	3.773662	350.229950	15.0	68.716019
3308	3.026962	0.119945	0.129409	3.903006	4.525979	4.819063	58600.0	9.996422	3.821569	351.971649	14.0	63.716438
44117	2.935863	0.187419	0.124516	3.166528	1.230836	-3.122138	58600.0	9.994445	5.225097	357.466431	16.0	61.298466
170789	2.735335	0.152867	0.403704	6.038029	3.016815	-3.443415	58600.0	9.996418	7.083936	347.111053	13.0	60.860794
113970	2.897024	0.068932	0.209250	5.663728	3.868474	4.450756	58600.0	7.999145	3.958026	348.778046	9.0	60.584415
45801	2.754677	0.047293	0.118126	3.139070	5.767782	-1.949476	58600.0	8.996888	4.321019	326.387482	14.0	56.090981
50775	2.374712	0.100280	0.165483	4.280114	5.955170	2.995417	58600.0	9.948814	9.705722	809.804260	29.0	55.641151
96507	2.820351	0.068964	0.080233	2.222034	0.960931	-2.630966	58600.0	8.821539	3.905472	262.245087	12.0	52.416988
191915	2.315446	0.192885	0.057156	2.130249	2.865086	-4.122024	58600.0	7.982198	9.668445	388.469055	21.0	38.837540

Figure 1.9: *Orbital elements of 9 candidate asteroids.*

Figure 1.9 shows the candidate orbital elements. Here are the summary statistics for the new asteroid candidates:

- 9 fitted orbital elements were found
- they had 9.11 hits on average
- the geometric mean resolution was 5.3 arc seconds
- the geometric mean distance to the nearest asteroid was 4.10E-3 AU
- the geometric mean covariance norm to the nearest asteroid as 1.10

Figure 1.10 shows the fit quality and distance to the nearest asteroid, as with the known asteroid search. These results are encouraging but not quite as definitive as I had hoped. The convergence with 9.1 hits to a resolution of 5.3 arc seconds strikes me as excellent. A RA/Dec observation has 2 degrees of freedom, so a collection of 6 parameters is matching 18 degrees of freedom of data at a composite tolerance of about 5 arc seconds. Log likelihood scores in the range of 40-60 lead me to believe to me that this process has successfully identified collections of observations that are highly likely to belong to the same or perhaps related objects, and that it has fitted orbital elements to the observations to a good tolerance.

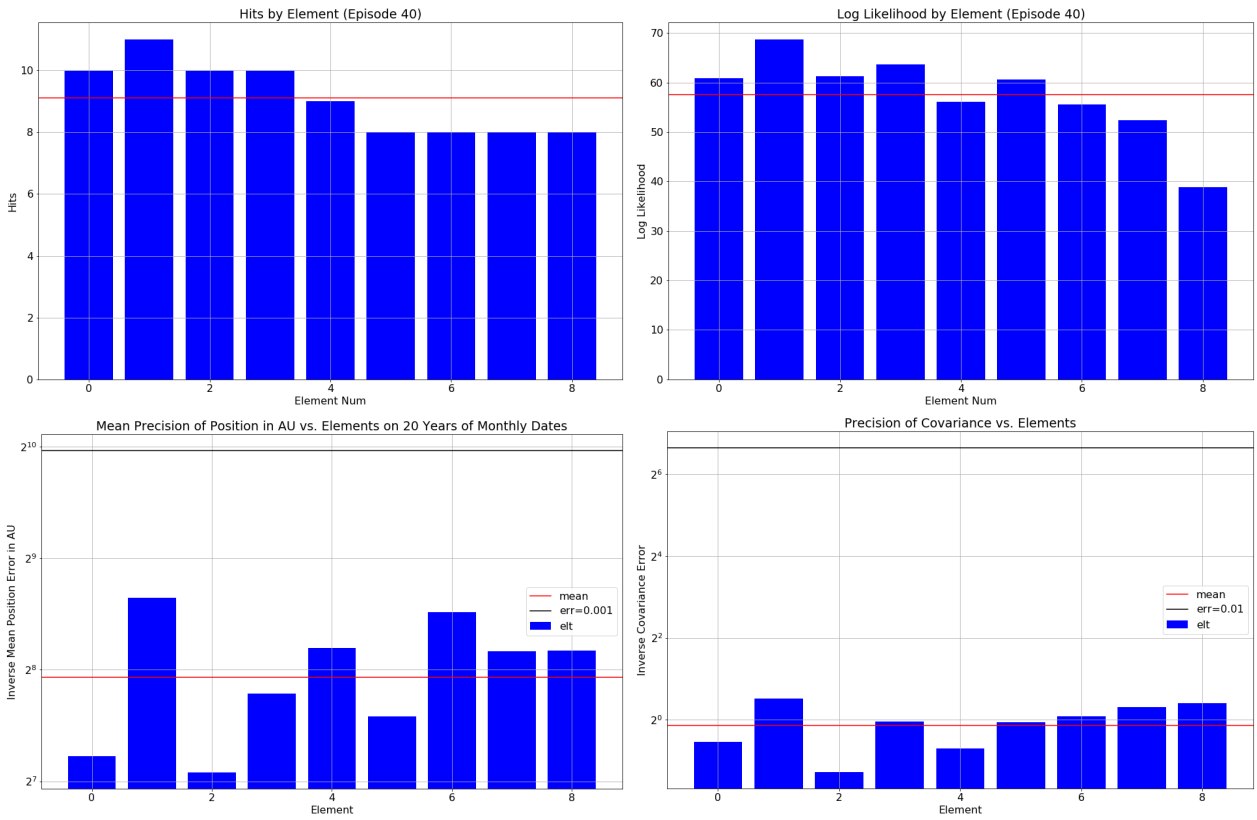


Figure 1.10: *9 orbital elements fitted to observations of unknown asteroids.*
The first row shows internal estimates of training quality: number of hits and log likelihood.
The second row shows distance to the nearest known asteroid in mean distance of orbit and covariance metric.

The news is not quite as a good when I look at the distance to the nearest asteroid in the catalogue. I had hoped to present compelling evidence that the recovered elements trained against known asteroid subset of the ZTF detections were much closer to the nearest known asteroid than these elements. While it is true that the these elements are further away by a factor of 1.54 in mean distance, and 1.51 on the covariance norm, that is not a big margin. I have to admit here the possibility that while these elements correspond to a real object, they might have converged to elements close to those of a known asteroid. When I set up the training to exclude detections near to known asteroids, I thought it was unlikely that it would converge to a known asteroid, and I still believe that to be true. There is a possibility that these candidates might be better interpreted as proposed revisions to the orbital elements of nearby asteroids than as new, undiscovered objects.

Figure **??** shows the ZTF hits the search process has associated with each set of candidate elements. We can gain understanding by paying close attention to the ObjectID column. These

are provisional assignments by ZTF indicating that multiple detections are likely to belong to the same object. Take four example candidate elements in turn.

Element `178421` is stringing together four detections of object ZTF18aboluox and 4 detections of object ZTF18acewaex. They are seen 433 days apart. The magnitudes are very different, so this is probably a spurious connection between two different objects with compatible orbits.

Element `3308` is piecing together 6 detections of ZTFV18abtpdzg and 2 detections of ZTF18abskzw. These detections were made 193 days apart. The magnitudes of these are compatible, making it far more plausible that this is a legitimate connection between two sets of detections that may not have been made previously.

Element 191915 is again stringing together 6 + 2 detections of different objects (ZTF18abtxgd and ZTF19abtsqmn) made 469 days apart. The magnitudes are all compatible. This too seems likely to be a legimate and novel connection between different sets of detections.

Element `170789` is an example of a fit made by stringing together just one set of detections made at similar times. All of the detections are of the same ZTF object ZTF17aaqwwg made on the same date, indeed made in a time span of only 55 minutes. The model has indepndently matched the conclusion of the ZTF classifier that all 8 detections belonged to the same object.

element_id	ObjectID	mjd	ra	dec	mag_app	s_sec
178421	b'ZTF18aboluox'	58430.166620	346.704046	-10.675142	15.787200	1.260372
178421	b'ZTF18aboluox'	58430.170313	346.704079	-10.675160	15.800000	0.982009
178421	b'ZTF18aboluox'	58430.166620	346.704001	-10.675028	15.587700	1.699850
178421	b'ZTF18aboluox'	58430.170313	346.704073	-10.675099	15.654600	1.175734
178421	b'ZTF18acewaex'	58863.138472	67.229666	17.295098	19.328100	6.219038
178421	b'ZTF18acewaex'	58863.138472	67.229789	17.295055	19.343500	5.789846
178421	b'ZTF18acewaex'	58863.152465	67.229835	17.295083	19.283501	2.389730
178421	b'ZTF18acewaex'	58863.152465	67.229783	17.295077	19.263100	2.392794

element_id	ObjectID	mjd	ra	dec	mag_app	s_sec
191915	b'ZTF18abtxtgd'	58430.170313	341.181109	-12.234676	19.693501	0.036218
191915	b'ZTF18abtxtgd'	58430.166620	341.181092	-12.234595	19.260099	2.621655
191915	b'ZTF19abtsqmn'	58899.139884	94.436593	22.583914	18.914301	6.521414
191915	b'ZTF19abtsqmn'	58899.140336	94.436579	22.583982	18.900700	6.623162
191915	b'ZTF19abtsqmn'	58899.192569	94.436720	22.583899	19.819700	0.946245
191915	b'ZTF19abtsqmn'	58899.222442	94.436672	22.583855	19.873501	4.829147
191915	b'ZTF19abtsqmn'	58899.220162	94.436545	22.583886	20.234699	4.129297
191915	b'ZTF19abtsqmn'	58899.220613	94.436587	22.583902	20.670900	4.263810

element_id	ObjectID	mjd	ra	dec	mag_app	s_sec
3308	b'ZTF18abtpdzg'	58670.439884	354.565806	-8.964600	17.605200	3.286346
3308	b'ZTF18abtpdzg'	58670.440336	354.565888	-8.964426	17.601101	3.888707
3308	b'ZTF18abtpdzg'	58670.462604	354.565787	-8.964429	16.858500	0.697156
3308	b'ZTF18abtpdzg'	58670.463056	354.565909	-8.964536	17.128700	1.329765
3308	b'ZTF18abtpdzg'	58670.462604	354.565824	-8.964445	17.092899	0.837276
3308	b'ZTF18abtpdzg'	58670.463056	354.565809	-8.964412	16.658600	0.769468
3308	b'ZTF18abspkzw'	58863.110058	346.642277	1.047679	16.541800	8.282505
3308	b'ZTF18abspkzw'	58863.109606	346.642261	1.047839	16.946899	7.604068

element_id	ObjectID	mjd	ra	dec	mag_app	s_sec
170789	b'ZTF17aaaqwwg'	58903.113588	84.725827	15.284655	19.437901	7.917067
170789	b'ZTF17aaaqwwg'	58903.116806	84.725856	15.284678	19.699699	7.043546
170789	b'ZTF17aaaqwwg'	58903.129097	84.725897	15.284685	20.146000	3.737848
170789	b'ZTF17aaaqwwg'	58903.126840	84.725899	15.284678	19.689501	4.333303
170789	b'ZTF17aaaqwwg'	58903.128194	84.725824	15.284668	19.916201	3.767049
170789	b'ZTF17aaaqwwg'	58903.149248	84.726023	15.284693	19.143600	3.802050
170789	b'ZTF17aaaqwwg'	58903.149699	84.725870	15.284728	19.707399	4.209481
170789	b'ZTF17aaaqwwg'	58903.151053	84.725889	15.284665	19.904301	4.672150

Figure 1.11: *ZTF hits associated with 4 of the new asteroid candidate elements.*

While I would like to marshall further evidence that these candidate elements belong to new objects, I find the detailed analysis of the claimed series of ZTF hits to be persuasive and

encouraging. This search process is clearlying identifying plausible connections between related detections by brute force analysis of their location in the sky.

1.6 Conclusion

In this thesis I have presented a novel technique for learning orbital elements of asteroids from telescopic data: searching in the space of orbital elements. This approach has significant theoretical advantages over methods that rely on stringing tracklets together, which suffer from a combinatorial explosion. I have demonstrated a working prototype and shown that it can converge well to correct orbital elements from a suitable initialization. This prototype also illustrates how to perform large scale and effient astrometric computations on GPUs using TensorFlow, which is a second potentially novel contribution. I have further demonstrated that even naive random initializations are sufficient to reidentify up to 125 known asteroids, albeit to varying degrees of quality. Finally, I have proposed candidate orbital elements for up to 9 previously unknown asteroids. By analyzing the details of the ZTF detections associated with each candidate element, I have provided anecdotal evidence that the search is identifying detections that plausibly belong to the same object.

1.7 Future Work

1.7.1 Intelligent Initialization of Candidate Elements

When I first conceived this project, I intended to devote substantial time to formulating intelligent initializations of the candidate orbital elements. The random orbital elements were intended as a quick and dirty placeholder to prove if the system works. Due to the looming deadline I have had to defer that to future work. The results presented for perturbed orbital elements were very encouraging, showing excellent convergence. If we can identify a set of candidate orbital elements that are close to real ones, this process should efficiently grab all the other detections in the data set that are consistent and combine them.

The ZTF data set includes an ObjectID column that represents a preliminary assessment of which detections are related. A straightforward method for initializing candidate orbital elements

is to take these classifications at face value. For each ObjectID, take the collection of k detections and use a traditional technique (e.g. least squares fitting) to find orbital elements consistent with them. In fact, the existing `AsteroidSearch` class provides most of the code required to do this. To adapt it to this problem, we would ignore the mixture parameters and minimize the least squares loss rather than maximizing the log likelihood.

Here is a more ambitious idea for intelligent initialization. By looking at the subset of ZTF detections within 2.0 arc seconds of a known asteroid, we can generate a large collection of training data of pairs of ZTF detections that we are highly confident belong to the same object. We can randomly sample other pairs of ZTF detections, and use the two data sets to train a binary classifier to predict the probability that a pair of ZTF detections belong to the same object. This classifier could be used to come up with tracklets that might not have the same ObjectID. An additional classifier might also be trained to try to extend a tracklet with a third detection. Each detection has 2 degrees of freedom, and an orbital element has 6. So a single tracklet allows us to sample a 2D space of candidate element consistent with it, while three or more detections should in theory give us a uniquely determined initialization point.

1.7.2 Incorporating Additonal Data

The method presented doesn't use anything special about the ZTF data set besides for the filtering of likely asteroid detections. While the ZTF data set is excellent, it only dates back effectively to the start of 2019. An obvious step to improve these results would be to bring in a second major data source. I would probably start with Pan-STARRS. Ideally I would like to identify a subset that has already been classified as likely to be near earth objects. Failing that, we could start with all Pan-STARRS data; remove bogus detections with a real-bogus classifier; and subtract out detections of stars and and galaxies by using the catalogue of their known positions in the sky. The code would need to be changed in some places, but the change would not be major.

1.7.3 Incorporate Magnitude

I developed code in the `AsteroidSearcch` model to predict the magnitude of a detection using the H-G model. I also had a version of the log likelihood that incorporated a joint probability

density on both direction and magnitude. The theoretical advantages of an estimation framework that incorporates magnitude are substantial. The model will be much less likely to make spurious connections of detections with compatible orbits that can't be the same object because they have different brightnesses. An added benefit is that the fitting procedure would give estimates for H and G. Unfortunately I was not able to get the fitting process to converge well when I included the magnitude in the joint probability. As time drew short, I was forced to remove it from the model presented here and punt it to future work. I am highly optimistic that given enough time, this can be made to work and will sharpen the results.

1.7.4 Rebuild the Known Asteroid Catalogue

If the three improvements above are made, I believe there is a good chance this model might be able to rebuild a large portion of the known asteroid catalogue. We saw that just seven months of ZTF data included over 100,000 asteroids with 20 or more hits. The random initialization was not remotely up to the task of recovering these elements, but with intelligent initialization alone we would be in the game. I think it would be a very powerful proof of concept if we could run a program that would crunch through tens or hundreds of millions of detections and generate an output with large overlap against the known asteroids. At a minimum, it would validate a single automated procedure that could also detect new objects. It might also allow us to gradually improve the quality of the data in the known asteroid catalog.

The ultimate goal of this body of work could be to build a single, fully automated computational pipeline for asteroid classification. Telescopic detections from multiple surveys would go in, optionally alongside snapshots of the known asteroid catalogue. The output would include an association of detections to known objects; revisions to the catalog where necessary; and proposed additions to the catalogue when supported by new detections.

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

[4] Joseph Blitzstein, Jessica Hwang
Introduction to Probability
CRC Press. 2019 (Second Edition)