An Orbit-fitting Optimization for Detecting Near-Earth Asteroids

AM205 Final Project

Paul Blankley, Vincent Casser, Ryan Janssen

December 2017

Abstract

This report summarizes a multi-step method for identifying full orbit trajectories of asteroids in our solar system. The input to this method is a list of partial "tracklets," a series of visible-light observations of an object in the night sky over the course of several hours. The output is a series of "orbital elements" that completely define the asteroid's trajectory through our solar system.

Some steps of this method have been completed as a prior project and are outside the scope of this paper; specifically, (i) the partitioning of the night sky into 768 spatial regions and monthly time windows, and (ii) the initial clustering of asteroids within those regions/windows into "asteroid slices."

The steps that are within the scope of this paper are (a) the optimization of the six "motion parameters" of each cluster from a variety of optimization solvers, (b) performing refinements on the earlier clusters using the optimized motion parameters, (c) converting the motion parameters into time independent "orbital elements," and (d) meta-clustering the asteroid slices from different time/space windows into final asteroid trajectories.

Ultimately, our algorithm correctly identifies 69.9% of just over 21,000 asteroids in a labeled dataset with an approx. 1% error rate of spurious asteroids. We hope to run this algorithm on a full, unsolved database of asteroids known as the Isolated Tracklet File (ITF).

Preamble: A note on the structure of this project/document

This AM205 project is an extension of a final project (also Fall 2017) for CS182: Artificial Intelligence. This project made good progress on an as-yet unsolved problem in astronomy: matching and identifying asteroids from a database of partial asteroid observations. As described in more detail below, thousands of previously unknown asteroids were identified by the clustering algorithm designed for the CS182 final project.

However, the clustering performed by the CS182 project was only the first step in identifying the missing asteroids. The next steps involve efficiently fitting asteroid orbits and performing meta-clustering across highly disparate time windows. As this is an optimization problem that is especially germane to AM205's course material, we have elected to complete the latter half for our AM205 final project.

Since this is a direct extension to the previous work, the Background and Problem Description sections of this report are substantially similar to those of the CS182 Final Project. This is done with permission of the third CS182 team member, Matthew Holman, who also happens to be a TF for AM205. The other two CS182 team members, Paul Blankley and Ryan Janssen, have joined with Vincent Casser to complete the optimization step of the algorithm for this AM205 final project.

1 Background: The Minor Planet Center

The Minor Planet Center (MPC), at the Harvard-Smithsonian Center for Astrophysics, is the world's clearinghouse for asteroid observations. The MPC collects, in real-time, data from hundreds to thousands of observing sites, roughly 10^5 individual observations per night.

The data are archived, processed, and served back to the public and researchers with a primary goal: facilitating the rapid detection and characterization of near-Earth objects (NEOs). The MPC is one of a number of NASA-funded facilities working toward fulfilling a US Congressional mandate of finding 90% of potentially hazardous NEOs that have estimated diameters greater than 140m.

The MPC archives currently contain over 170 million individual observations of $\sim 700K$ distinct objects. Observations reported to the MPC pass through a number of processing steps:

- Validation and archiving. Improperly formatted or physically implausible observations are returned. Valid observations are archived in their original form.
- Identification. Observations are checked to see if they match the predicted locations of known objects with well-determined orbits. Identified observations are set aside for later processing.
- Classification. Tracklets that display rapid motion are often near-Earth objects. These are prioritized for additional observations because they can be lost if not observed immediately.
- Linking. Separate tracklets of the same object are matched together. We do not know *a priori* which tracklets go together. Plausible matches are hypothesized and then tested with orbit fitting.

2 Problem description: The Isolated Tracklet File (ITF)

2.1 What is the ITF?

The observations reported to the MPC follow a fixed 80-character format that originated in the era of punch cards. As an example, here are the first few observations of A/2017 U1, the recently discovered interstellar object:

```
AK17U010  C2017 10 18.47298 01 59 57.442+02 06 04.30          19.8 wLEU183F51
AK17U010  C2017 10 18.49990 01 59 08.910+02 07 20.19                LEU183F51
AK17U010  C2017 10 19.39715 01 34 55.362+02 45 03.20          19.9 wLEU183F51
AK17U010  C2017 10 19.40837 01 34 38.745+02 45 28.24          19.9 wLEU183F51
AK17U010  C2017 10 19.41968 01 34 21.948+02 45 53.55          20.1 wLEU183F51
```

The first few characters are the unique designation of the object (in compact form). The 'C' indicates CCD digital observations. Next are the year, month, and fractional day. Following that are the right ascension and declination (longitude and latitude on sky), the magnitude (a measure of brightness), filter, and observatory code (the 'F51' at the end means the Pan-STARRS-1 observatory on Hale'akala, Maui). This information

specifies what object was observed, its observed location and brightness at each time, and where it was observed from.

These observations are grouped by the observers into 'tracklets' (Kubica et al. 2017). A tracklet is a collection of two or more observations taken over a short enough time span (minutes to hours) that the observer is confident that the observations are real rather than spurious and that they represent the same object. Tracklets normally exhibit linear, constant-rate motion within the span of a few hours.

Roughly 90% of the observations reported to the MPC can be immediately identified as known objects. And of the remaining 10%, most can be linked with other tracklets reported within the past few days. However, some tracklets cannot be identified or linked. These end up in the *Isolated Tracklet File* or ITF.

The ITF currently contains about fourteen million observations grouped into about three million tracklets. Most of those tracklets are real. And given the area of sky observed nightly by large NASA-funded surveys, we can estimate that most objects have been observed multiple times. It is not uncommon for nearly discovered objects with well-determined orbits to match several tracklets in the ITF, spread out over the course of several years.[1]

2.2 A challenging problem

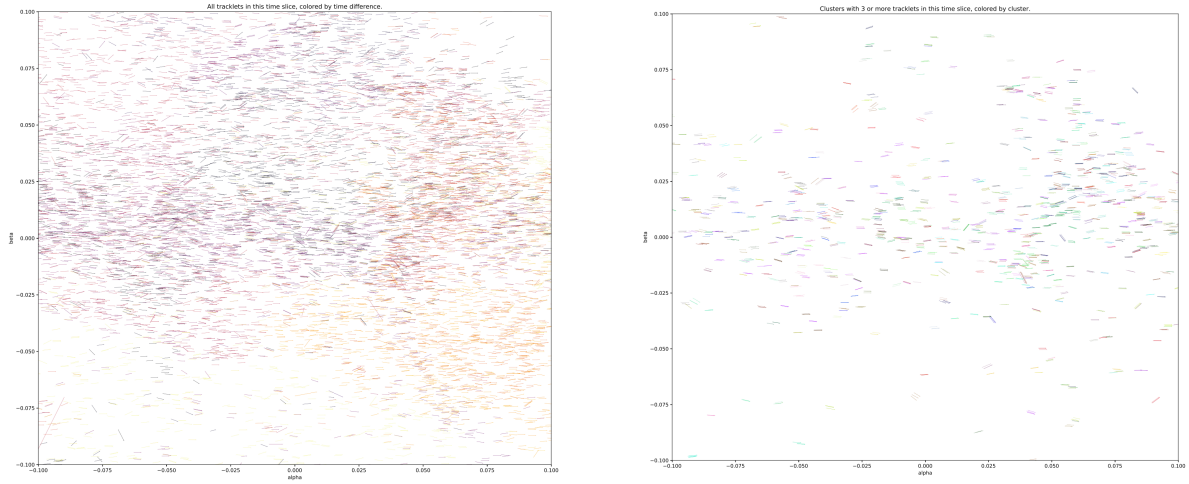
As telescope apertures increase, the number of tracklets stored in the ITF continues to grow. To satisfy the Congressional mandate, we will ultimately require a methodology to link ITF tracklets into full asteroid orbits.

This is a not a toy problem - the ITF is in fact a real dataset that is as of yet unsolved. The challenges with processing the ITF are numerous:

- **Large state space:** The brute force solution is to pair every tracklet in the ITF with every other tracklet, and run an orbit-fitting algorithm to test the match. Then, check the plausible matches with other tracklets to see if a third, fourth, fifth, etc., tracklet also matches (multiple matches are required for sufficient confidence levels). As we would need to iterate this process over 3 million tracklets, the brute force method is computationally difficult.
- **Partial data:** Each asteroid has six motion parameters (three position components and three velocity components). Observations of a single tracklet provide four of these (a starting location and two rates of motion on the sky plane), but the radial distance and velocity from the observer are not directly observed. Thus, asteroids cannot be matched simply by comparing parameters. The fundamental challenge in the linking problem, and orbit fitting in general, is inferring the distance to the observed object at the time of the observations.
- **High asteroid density:** In any given region of the night sky, many asteroids can be observed (several tens per square degree for the brightness limits of current surveys - see Figure 1 below). As a result, clustering algorithms must be carefully designed and tuned to avoid false positive clusters.

[1] Source: MPC background, ITF description completed by Matt Holman as part of original CS182 project

- **Large time separation:** In the ITF, observations are sparse in time; this makes it difficult to 'connect the dots' by just looking which tracklets are nearby.
- **Highly nonlinear:** When observed from the moving reference from of the earth, asteroids follow a highly non-linear trajectory across the sky. This necessitates complex curve-fitting algorithms, compounding the issues of computational tractability.



(a) Tracklet density of one region in time/space on night sky

(b) Clusters identified on same region as one asserted $\dot{(\gamma)}$

Figure 1: Full density of 1/768th window of the night sky, with associated "correct" clusters

2.3 Project objective

The asteroid tracklets are short term movements described in six motion parameters (α, $\dot{\alpha}$, β, $\dot{\beta}$, γ, and $\dot{\gamma}$, which resemble the x, y, and z position and velocity in the night sky, respectively). However, as we cannot see the z-distance well, γ, and $\dot{\gamma}$ are unknown, so the motion of each tracklet is only partially known. Furthermore, the each tracklet in the ITF is a short, isolated observation that cannot be matched to an asteroid trajectory.

Our objective is to design an algorithm that will take in the partial, disparate sets of tracklet motion parameters, and assemble them into full asteroid trajectories.

Rather than parsing the ITF for this project, which has no measure of confirming accuracy, we will parse a training file with known asteroids provided by the Minor Planet Center. Apart from its smaller size, the training file has the same challenges (partial data, etc.) as the ITF and we believe our solution will be extensible with a larger training time.

This training file contains 81,982 tracklets corresponding to 21,359 known asteroids. Our objective is to input the tracklet list and identify as many of the 21,359 asteroids as possible.

3 Previous and Related work

3.1 Asteroid Motion Model

There have been previous attempts to address this problem using a KD-tree clustering methodology, such as the Moving Object Processing System (MOPS) ([1]). However, MOPS has had challenges scaling to sufficiently large numbers of tracklets. It runs in $O(n^3)$ time, and therefore becomes computationally difficult for 3 million tracklets. Additionally, Veres and Chesley ([2]) note that MOPS has challenges at higher tracklet densities.

Methods described in this paper are based on the work of Bernstein et al. [3] on orbit fitting and uncertainty estimation for Kuiper Belt Objects (KBOs). We modified and extended their approach from a technical perspective, and applied it to asteroids located generally within our solar system as opposed to strictly Kuiper belt objects (the majority of asteroids are in fact located in a belt roughly between the orbits of Mars and Jupiter).

3.2 Initial work already completed by CS182 project team

Previous work completed on this project ("Stage 1") was to cluster the individual tracklets into suspected asteroid "slices," which is a set of partial asteroid parameters at a given point in time.

This "Stage 1" was achieved by:

1. Dividing the night sky into 768 geographic regions, and monthly time steps centered on the new moon
2. In each time/space window, assume a fixed radial distance $z = 2.5$ AU (represented as $\gamma = 1/z = 0.4(AU)^{-1}$ per the Bernstein et al. coordinate system), which corresponds to the center of the large asteroid belt in our solar system.
3. Iterate over a number of radial velocities $\dot{z}$
4. Perform a Heliocentric Transform - that is, for the assumed constant z and asserted $\dot{z}$, translate the parameters into the fixed reference frame of the Sun rather than the moving reference frame of the Earth.
5. For each window, perform a KD-tree nearest neighbours clustering algorithm over the remaining four motion parameters (lateral x and y position and velocity). The clusters identify groups of asteroid tracklets that are likely to be a single asteroid.

This was performed over the training set to tune two hyperparameters - (i) the cluster radius, and (ii) the weighting of velocity parameters relative to positional parameters. The algorithm identified 79,434 asteroid slices in the training set, many of which were redundant observations.

4 Approach

Stage 1 (the heliocentric clustering) was completed successfully, however it has several limitations:

- First, the two unknown asteroid parameters (γ and $\dot{\gamma}$) have been estimated in Step 2 above. We assumed the former to be a constant and iterated over the latter until a cluster was obtained.
- Second, several individual tracklets still exist that have not been correctly matched. It is likely that many of these "singletons" correspond to one of the discovered clusters. There are also several pairs of nearby slices that should have actually been joined as one slice.
- Finally, the motion parameters of each asteroid slice are only valid for a small time window. It is likely that several observed slices correspond to the same asteroid trajectories, months or years apart. Ideally, the slice motion parameters could be extrapolated to any time.

Stage 2 will overcome these limitations by optimizing orbit fits (including across the two missing dimensions), and enabling clustering of slices that are disparate in time.

The steps in Stage 2 are as follows:

6. Take the slices obtained in each time/space slice in Stage 1, and conduct an orbit fit to refine the two estimated trajectory dimensions. This is an optimization problem that will compute more reliable motion parameters. In particular, we will calculate solutions for the γ and $\dot{\gamma}$ radial parameters. Until now the former has been assumed constant, the latter has been iterated through trial-and-error at low resolution.
7. Now that optimal orbits have been determined, re-examine remaining unclustered tracklets to confirm if any belong to the fitted orbits.
8. Now that γ and $\dot{\gamma}$ are confirmed, cluster the clusters along all six degrees of motion.
9. At this stage, the slices are fully refined. Now, calculate "orbital elements" for each discovered slice using the six-dimensional motion parameters. This will map the slice trajectories independently of time.
10. Perform a final slice clustering on the orbital elements across all times, to consolidate into final asteroids.

Each of these steps is discussed in detail below:

4.1 Step 6: Fitting an orbit to each asteroid slice

When fitting the cluster orbits in the "motion parameter domain" (that is, in terms of α, $\dot{\alpha}$, β, $\dot{\beta}$, γ, and $\dot{\gamma}$, the three-dimensional axes position/velocity), we have θ_x and θ_y ground truth terms. We can use these ground truth terms to create a loss function to minimize error in our six motion parameters. Here θ_x and θ_y represent the (x, y) values of the asteroid when projected on the plane tangent to our chosen reference time, the new moon of each month, and are captured by tracklet observation.

We know from the form introduced in Bernstein (2000) that, when ignoring gravity, θ_x and θ_y can be expressed in terms of the motion parameters by the following expressions:

$$\theta_x = \frac{\alpha + \dot{\alpha}t - \gamma X_e(t)}{1 + \dot{\gamma}t - \gamma Z_e(t)}$$

$$\theta_y = \frac{\beta + \dot{\beta}t - \gamma Y_e(t)}{1 + \dot{\gamma}t - \gamma Z_e(t)}$$

Here we simplify the above equations from their original form by assuming that acceleration due to gravity is zero in all directions. This assumption loses precision, but on the short time scale we are considering for clustering (a few hours within the multi-year asteroid orbit), gravitational acceleration will not be meaningful and trajectories will be nearly-linear. This is supported by Bernstein et al, who assessed that angular displacement due to gravity is approximately 300 times smaller than other motion components.

The remaining $X_e(t)$, $Y_e(t)$, and $Z_e(t)$ terms are the (x, y, z) coordinates of the observatory from which the asteroid was observed. These parameters are known absolutely and can be treated as constants in our loss function.

We also know the observation time t absolutely. We have the Julian date for each individual observation, and we also know the time it takes light to travel from the γ values we asserted to reach earth. We subtract these two values to get the "real" time of observation at the asteroid. We set a reference time to be the date of the nearest new moon to the observation. We then subtract the Julian date of the new moon, our chosen reference date, to transform all time scales to center around our chosen reference time.

4.1.1 Orbit fitting: Optimizing motion parameters of each asteroid slice

Now that we have our ground truth in terms of the orbit parameters we care about and values we know, we can define our loss function.

We choose squared error as the value we will minimize, because of its mathematical convenience in calculating the Jacobian and Hessian. Because we are minimizing two independent terms, we will have to define a single loss in terms of both θ_x and θ_y. Since we treat error in θ_x and θ_y with equal priority, we will simply sum the two respective squared error terms.

Thus, we define the loss function as:

$$\mathcal{L}(\alpha, \dot{\alpha}, \beta, \dot{\beta}, \gamma, \dot{\gamma}) = \sum_{i \in obs} (\theta_{x_i} - \hat{\theta}_{x_i})^2 + (\theta_{y_i} - \hat{\theta}_{y_i})^2$$

where $\hat{\theta}_{x_i}$ and $\hat{\theta}_{y_i}$ are derived from the observed/asserted motion parameters of each respective tracklet i. For any given tracklet i:

$$\hat{\theta}_x = \frac{\alpha_{tracklet} + \dot{\alpha}_{tracklet}t - \gamma_{constant} X_e(t)}{1 + \dot{\gamma}_{asserted}t - \gamma_{constant} Z_e(t)}$$

$$\hat{\theta}_y = \frac{\beta_{tracklet} + \dot{\beta}_{tracklet} t - \gamma_{constant} Y_e(t)}{1 + \dot{\gamma}_{asserted} t - \gamma_{constant} Z_e(t)}$$

Specifically, $\alpha_{tracklet}$, $\beta_{tracklet}$, $\dot{\alpha}_{tracklet}$, and $\dot{\beta}_{tracklet}$ are the observed tracklet motion parameters. $\dot{\gamma}$ is the value in the $\dot{\gamma}$ iteration which the cluster satisfies the cluster radius of 0.00124 radians (this value was selected as part of Stage 1 of this project, to maximize the accuracy of slices). $\gamma_{constant}$ is the assumed constant value of 0.4 as discussed above.

We will apply this technique to find the parameters that minimize the loss in each distinct set of slice motion parameters. Each cluster is made up of tracklets that each have a few separate observations. An average cluster in our training data has between 3 and 4 tracklets a total of about 10 observations, so this is an underconstrained system.

The goal of our function is to minimize the error with respect to θ_x and θ_y for each observation in a cluster, treating each observation with equal weight. Now that we have a formalized loss function, we will look at several different optimization algorithms to find the parameters that minimize our loss.

4.1.2 Orbit fitting: selecting solver for efficiency and accuracy

As the number of isolated tracklets is large (there are 3 million unmatched asteroids tracklets in the full Isolated Tracklet File), efficiency is important when calculating an orbit fit. We have evaluated a number of different methods to apply the orbit fitting minimization discussed above.

The solvers we consider are the Nelder-Mead Simplex Algorithm [4], Powell's Method [5], Conjugate Gradient (*cg*) [6], BFGS [7], L-BFGS-B [8], Newton-Raphson Method (*newton-cg*), Truncated Newton/Newton Conjugate-Gradient Method (*TNC*) [9], Constrained optimization by linear approximation (*COBYLA*) [10], Sequential Least Squares Programming (*SLSQP*) [11], Dog-leg Trust Region [12], and Newton Conjugate Gradient Trust-region Algorithm (*trust-ncg*). All solver schema were implemented via scipy.optimize's *minimize()* method.

Certain methods can be further optimized by (or require) implementing an analytical Jacobian and Hessian. We have implemented these respectively in every schema that accepts the analytical solution, as follows:

Solver Method	Analytical Jacobian?	Analytical Hessian?
Nelder-Mead Simplex Algorithm		
Powell's Method		
Conjugate Gradient	Yes	
BFGS	Yes	
Memory-limited BFGS (L-BFGS)	Yes	
Newton-Raphson Method	Yes*	Yes*
Truncated Newton Method	Yes	
COBYLA		
SLSQP	Yes	
Dog-leg Trust Region	Yes*	Yes*
Newton Conjugate Gradient Trust Region	Yes*	Yes*

Table 1: Analytic solutions used to optimize each solver schema. The star denotes that this method must be given the respective information.

To calculate the Jacobian and Hessian, we ignore the gravity term - as discussed earlier in the Orbit Fitting section of this report, this has a very small impact on overall asteroid motion over our short time frame.

The analytical Jacobian of the loss functions can then be calculated as:

$$\frac{\partial L}{\partial \alpha} = \frac{2(\alpha + \dot{\alpha}t + \theta_x(\gamma z - \dot{\gamma}t - 1) - \gamma x)}{(-\gamma z + \dot{\gamma}t + 1)^2}; \quad \frac{\partial L}{\partial \dot{\alpha}} = -\frac{2t(-\alpha - \dot{\alpha}t - \gamma L z + \gamma x + \dot{\gamma}\theta_x t + \theta_x)}{(-\gamma z + \dot{\gamma}t + 1)^2} \tag{1}$$

$$\frac{\partial L}{\partial \beta} = \frac{2(\theta_y(\gamma z - \dot{\gamma}t - 1) - \gamma y + \dot{\beta}t + \beta)}{(-\gamma z + \dot{\gamma}t + 1)^2}; \quad \frac{\partial L}{\partial \dot{\beta}} = -\frac{2t(-\gamma M z + \gamma y - \dot{\beta}t + \dot{\gamma}\theta_y t + \theta_y - \beta)}{(-\gamma z + \dot{\gamma}t + 1)^2} \tag{2}$$

$$\begin{aligned}\frac{\partial L}{\partial \gamma} &= 2\left(\frac{x}{-fz + \dot{\gamma}t + 1} - \frac{z(\alpha + \dot{\alpha}t - \gamma x)}{(-\gamma z + \dot{\gamma}t + 1)^2}\right)\left(\theta_x - \frac{\alpha + \dot{\alpha}t - \gamma x}{-\gamma z + \dot{\gamma}t + 1}\right) \\ &+2\left(\frac{y}{-\gamma z + \dot{\gamma}t + 1} - \frac{z(-\gamma y + \dot{\beta}t + \beta)}{(-fz + \dot{\gamma}t + 1)^2}\right)\left(\theta_y - \frac{-\gamma y + \dot{\beta}t + \beta}{-\gamma z + \dot{\gamma}t + 1}\right)\end{aligned} \tag{3}$$

$$\frac{\partial L}{\partial \dot{\gamma}} = \frac{2t(\alpha + \dot{\alpha}t - \gamma x)\left(\theta_x - \frac{\alpha + \dot{\alpha}t - \gamma x}{-\gamma z + \dot{\gamma}t + 1}\right)}{(-\gamma z + \dot{\gamma}t + 1)^2} + \frac{2t(-\gamma y + \dot{\beta}t + \beta)\left(\theta_y - \frac{-\gamma y + \dot{\beta}t + \beta}{-\gamma z + \dot{\gamma}t + 1}\right)}{(-\gamma z + \dot{\gamma}t + 1)^2} \tag{4}$$

where

$$\boldsymbol{J}_L = \begin{bmatrix} \frac{\partial L}{\partial \alpha} & \frac{\partial L}{\partial \dot{\alpha}} & \frac{\partial L}{\partial \beta} & \frac{\partial L}{\partial \dot{\beta}} & \frac{\partial L}{\partial \gamma} & \frac{\partial L}{\partial \dot{\gamma}} \end{bmatrix}$$

Also, for the Hessian of the loss function, we have:

$$\frac{\partial^2 L}{\partial \alpha \partial \alpha} = \frac{\partial^2 L}{\partial \beta \partial \beta} = \frac{2}{(-\gamma z + \dot{\gamma}t + 1)^2}; \quad \frac{\partial^2 L}{\partial \dot{\alpha} \partial \alpha} = \frac{\partial^2 L}{\partial \dot{\beta} \partial \beta} = \frac{2t}{(-\gamma z + \dot{\gamma}t + 1)^2} \tag{5}$$

$$\frac{\partial^2 L}{\partial\dot{\alpha}\partial\dot{\alpha}} = \frac{\partial^2 L}{\partial\dot{\beta}\partial\dot{\beta}} = \frac{2t^2}{(-\gamma z + \dot{\gamma} t + 1)^2}; \quad \frac{\partial^2 L}{\partial\beta\partial\alpha} = \frac{\partial^2 L}{\partial\dot{\beta}\partial\alpha} = \frac{\partial^2 L}{\partial\beta\partial\dot{\alpha}} = \frac{\partial^2 L}{\partial\dot{\beta}\partial\dot{\alpha}} = 0 \tag{6}$$

$$\frac{\partial^2 L}{\partial\gamma\partial\alpha} = -\frac{2z\left(\theta_x - \frac{\alpha+\dot{\alpha}t-\gamma x}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2\left(\frac{x}{-\gamma z+\dot{\gamma}t+1} - \frac{z(\alpha+\dot{\alpha}t-\gamma x)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{-\gamma z + \dot{\gamma} t + 1} \tag{7}$$

$$\frac{\partial^2 L}{\partial\dot{\gamma}\partial\alpha} = \frac{2t(-2\alpha - 2\dot{\alpha}t - \gamma\theta_x z + 2\gamma x + \dot{\gamma}\theta_x t + \theta_x)}{(-\gamma z + \dot{\gamma} t + 1)^3} \tag{8}$$

$$\frac{\partial^2 L}{\partial\gamma\partial\dot{\alpha}} = -\frac{2tz\left(\theta_x - \frac{\alpha+\dot{\alpha}t-\gamma x}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2t\left(\frac{x}{-\gamma z+\dot{\gamma}t+1} - \frac{z(\alpha+\dot{\alpha}t-\gamma x)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{-\gamma z + \dot{\gamma} t + 1} \tag{9}$$

$$\frac{\partial^2 L}{\partial\dot{\gamma}\partial\dot{\alpha}} = \frac{2t^2(-2\alpha - 2\dot{\alpha}t - \gamma\theta_x z + 2\gamma x + \dot{\gamma}\theta_x t + \theta_x)}{(-\gamma z + \dot{\gamma} t + 1)^3} \tag{10}$$

$$\frac{\partial^2 L}{\partial\gamma\partial\beta} = -\frac{2z\left(\theta_y - \frac{-\gamma y+\dot{\beta}t+\beta}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2\left(\frac{y}{-\gamma z+\dot{\gamma}t+1} - \frac{z(-\gamma y+\dot{\beta}t+\beta)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{-\gamma z + \dot{\gamma} t + 1} \tag{11}$$

$$\frac{\partial^2 L}{\partial\dot{\gamma}\partial\beta} = \frac{2t(-\gamma\theta_y z + 2\gamma y - 2\dot{\beta}t + \dot{\gamma}\theta_y t + \theta_y - 2\beta)}{(-\gamma z + \dot{\gamma} t + 1)^3} \tag{12}$$

$$\frac{\partial^2 L}{\partial\gamma\partial\dot{\beta}} = -\frac{2tz\left(\theta_y - \frac{-\gamma y+\dot{\beta}t+\beta}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2t\left(\frac{y}{-\gamma z+\dot{\gamma}t+1} - \frac{z(-\gamma y+\dot{\beta}t+\beta)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{-\gamma z + \dot{\gamma} t + 1} \tag{13}$$

$$\frac{\partial^2 L}{\partial\dot{\gamma}\partial\dot{\beta}} = \frac{2t^2(-\gamma\theta_y z + 2\gamma y - 2\dot{\beta}t + \dot{\gamma}\theta_y t + \theta_y - 2\beta)}{(-\gamma z + \dot{\gamma} t + 1)^3} \tag{14}$$

$$\begin{aligned}\frac{\partial^2 L}{\partial\gamma\partial\gamma} = {} & 2\left(\frac{2xz}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2z^2(\alpha + \dot{\alpha}t - \gamma x)}{(-\gamma z + \dot{\gamma} t + 1)^3}\right)\left(\theta_x - \frac{\alpha + \dot{\alpha}t - \gamma x}{-\gamma z + \dot{\gamma} t + 1}\right) \\ & + 2\left(\frac{x}{-\gamma z + \dot{\gamma} t + 1} - \frac{z(\alpha + \dot{\alpha}t - \gamma x)}{(-\gamma z + \dot{\gamma} t + 1)^2}\right)^2 \\ & + 2\left(\frac{2yz}{(-\gamma z + \dot{\gamma} t + 1)^2} - \frac{2z^2(-\gamma y + \dot{\beta}t + \beta)}{(-\gamma z + \dot{\gamma} t + 1)^3}\right)\left(\theta_y - \frac{-\gamma y + \dot{\beta}t + \beta}{-\gamma z + \dot{\gamma} t + 1}\right) \\ & + 2\left(\frac{y}{-\gamma z + \dot{\gamma} t + 1} - \frac{z(-\gamma y + \dot{\beta}t + \beta)}{(-\gamma z + \dot{\gamma} t + 1)^2}\right)^2\end{aligned} \tag{15}$$

$$\begin{aligned}\frac{\partial^2 L}{\partial\dot{\gamma}\partial\gamma} = &-\frac{2tx\left(\theta_x - \frac{\alpha+\dot{\alpha}t-\gamma x}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^2} + \frac{4tz(\alpha+\dot{\alpha}t-\gamma x)\left(\theta_x - \frac{\alpha+\dot{\alpha}t-\gamma x}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^3} \\ &+\frac{2t(\alpha+\dot{\alpha}t-\gamma x)\left(\frac{x}{-\gamma z+\dot{\gamma}t+1} - \frac{z(\alpha+\dot{\alpha}t-\gamma x)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{(-\gamma z+\dot{\gamma}t+1)^2} - \frac{2ty\left(\theta_y - \frac{-\gamma y+\dot{\beta}t+\beta}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^2} \\ &+\frac{4tz(-\gamma y+\dot{\beta}t+\beta)\left(\theta_y - \frac{-\gamma y+\dot{\beta}t+\beta}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^3} \\ &+\frac{2t(-\gamma y+\dot{\beta}t+\beta)\left(\frac{y}{-\gamma z+\dot{\gamma}t+1} - \frac{z(-\gamma y+\dot{\beta}t+\beta)}{(-\gamma z+\dot{\gamma}t+1)^2}\right)}{(-\gamma z+\dot{\gamma}t+1)^2}\end{aligned} \tag{16}$$

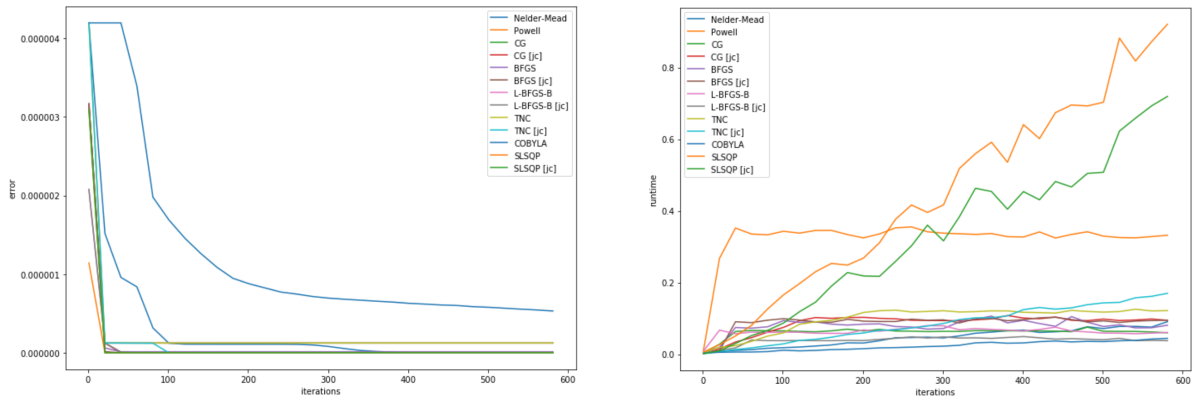
$$\begin{aligned}\frac{\partial^2 L}{\partial\dot{\gamma}\partial\dot{\gamma}} = &-\frac{4t^2(\alpha+\dot{\alpha}t-\gamma x)\left(\theta_x - \frac{\alpha+\dot{\alpha}t-\gamma x}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^3} + \frac{2t^2(\alpha+\dot{\alpha}t-\gamma x)^2}{(-\gamma z+\dot{\gamma}t+1)^4} \\ &-\frac{4t^2(-\gamma y+\dot{\beta}t+\beta)\left(\theta_y - \frac{-\gamma y+\dot{\beta}t+\beta}{-\gamma z+\dot{\gamma}t+1}\right)}{(-\gamma z+\dot{\gamma}t+1)^3} + \frac{2t^2(-\gamma y+\dot{\beta}t+\beta)^2}{(-\gamma z+\dot{\gamma}t+1)^4}\end{aligned} \tag{17}$$

where

$$\boldsymbol{H}_L = \begin{bmatrix} \frac{\partial^2 L}{\partial\alpha^2} & \frac{\partial^2 L}{\partial\alpha\partial\dot{\alpha}} & \frac{\partial^2 L}{\partial\alpha\partial\beta} & \frac{\partial^2 L}{\partial\alpha\partial\dot{\beta}} & \frac{\partial^2 L}{\partial\alpha\partial\gamma} & \frac{\partial^2 L}{\partial\alpha\partial\dot{\gamma}} \\ & \frac{\partial^2 L}{\partial\dot{\alpha}^2} & \frac{\partial^2 L}{\partial\dot{\alpha}\partial\beta} & \frac{\partial^2 L}{\partial\dot{\alpha}\partial\dot{\beta}} & \frac{\partial^2 L}{\partial\dot{\alpha}\partial\gamma} & \frac{\partial^2 L}{\partial\dot{\alpha}\partial\dot{\gamma}} \\ & & \frac{\partial^2 L}{\partial\beta^2} & \frac{\partial^2 L}{\partial\beta\partial\dot{\beta}} & \frac{\partial^2 L}{\partial\beta\partial\gamma} & \frac{\partial^2 L}{\partial\beta\partial\dot{\gamma}} \\ & & & \frac{\partial^2 L}{\partial\dot{\beta}^2} & \frac{\partial^2 L}{\partial\dot{\beta}\partial\gamma} & \frac{\partial^2 L}{\partial\dot{\beta}\partial\dot{\gamma}} \\ & & & & \frac{\partial^2 L}{\partial\gamma^2} & \frac{\partial^2 L}{\partial\gamma\partial\dot{\gamma}} \\ Sym & & & & & \frac{\partial^2 L}{\partial\dot{\gamma}^2} \end{bmatrix}$$

We investigated the performance of the different solvers both in terms of runtime and accuracy. We chose to investigate clock runtime rather than number of iterations, as iterations can have quite different computational costs across the different methods.

We tried to enforce maximum accuracy (that is, setting the tolerance to (almost) zero). While this increased comparability, note that in some implementations it is not possible to set a perfect accuracy target with iteration limit, which caused some solvers to stop early in the optimization process, thus underestimating runtime and overestimating achievable solution accuracy. Thus, we recommend considering both the achieved accuracy and runtime simultaneously when interpreting results. The relationship between runtime, error, and number of iterations is illustrated in Figure 2:



(a) Error versus number of iterations for selected orbit-fitting optimization solvers

(b) Runtime versus number of iterations for selected orbit-fitting optimization solvers

Figure 2: Illustration of runtime/accuracy tradeoff for selected solvers

In Figure 3 below, we show our results with respect to runtime and achieved accuracy at the natural stopping mechanism for each solver method (but setting the target accuracy equivalently where possible). We can see that the runtime varies quite significantly: passing the Jacobian/Hessian explicitly almost triples the performance of the Conjugate Gradient (*cg*) method and roughly doubles the performance of BFGS. For L-BFGS-B, TNC and SLSQP, these differences are not as significant. Powell, Conjugate Gradient without Jacobian and Nelder-Mead have the slowest average performance, with only Powell achieving a highly accurate solution. In terms of an accuracy-runtime trade-off, BFGS with Jacobian and Newton Conjugate Gradient Trust Region appear to be the most favorable.

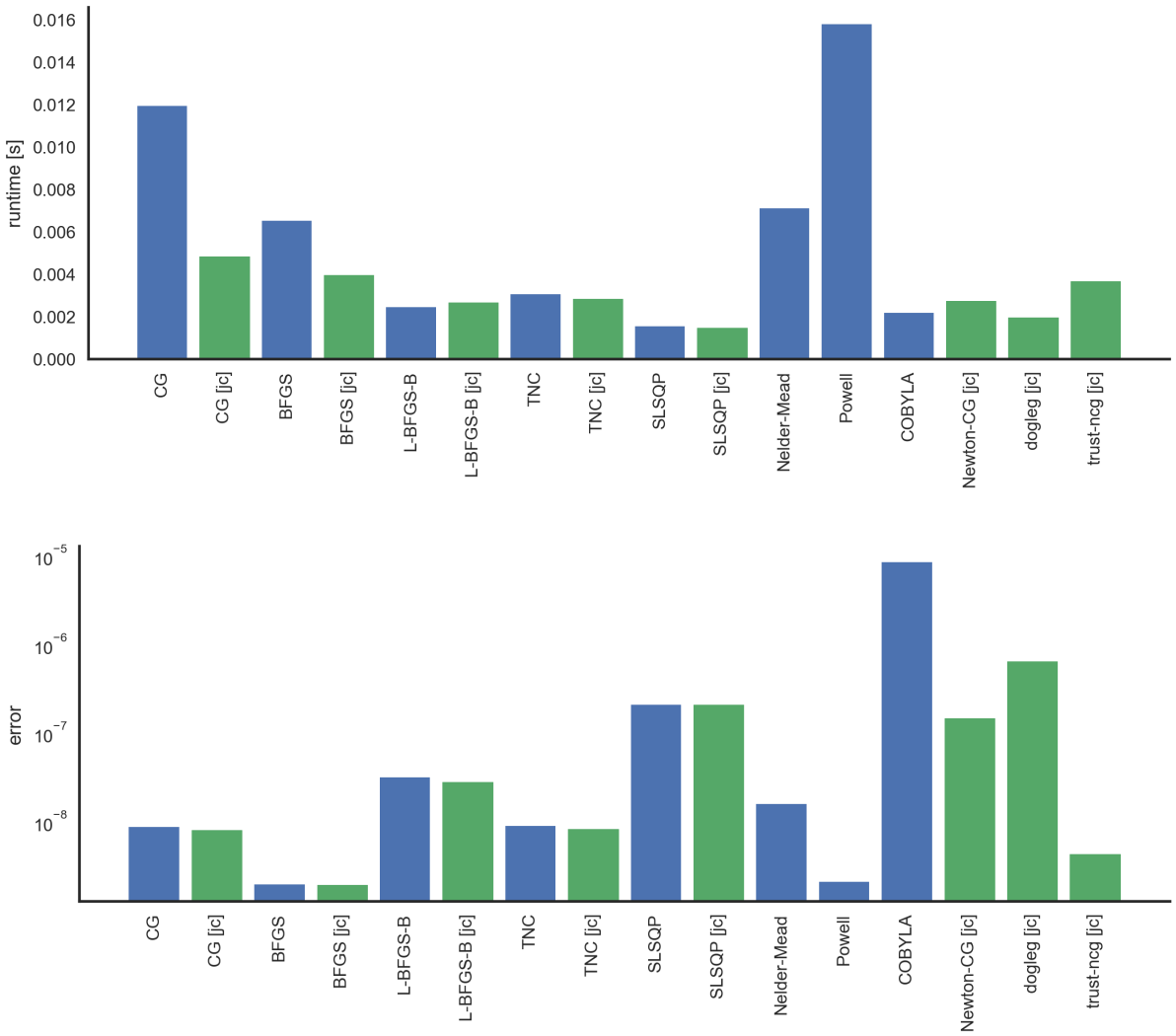


Figure 3: Comparison of runtime and accuracy between the different solvers we evaluated. Solvers that use an analytically derived Jacobian and/or Hessian are marked in green. As can be seen, many of those perform significantly better than their counterparts in terms of runtime, while the accuracy is equivalent.

For our final implementation, we chose to set BFGS with Jacobian as a default. This is because BFGS is relatively fast and highly accurate, and we think it is the best tradeoff between speed and accuracy. However, we provide support for all other solvers in our framework as well.

4.1.3 Orbit fitting: Optimizing tradeoff between ($\dot{\gamma}$) step size versus solver complexity

The efficiency of the orbit fitting is also relevant to the resolution of the $\dot{\gamma}$ we iterate through in our initial clustering.

Recall that to initially cluster the tracklet into slices, we would iterate over $\dot{\gamma}$ and perform a four-dimensional

clustering at each $\dot{\gamma}$ step. This process is like focusing a camera lens - there will be a value of $\dot{\gamma}$ that a cluster "comes together" into a slice. Since the slice cluster is tightest at those values of $\dot{\gamma}$, we use this value of $\dot{\gamma}$ as our initial estimate for the orbit fit. However, we use a relatively coarse grid (we iterate over only 5 steps of $\dot{\gamma}$), and $\dot{\gamma}$ could still be as far as our grid interval divided by two away from the optimal $\dot{\gamma}$.

Therefore, is it possible a finer $\dot{\gamma}$ grid will decrease overall runtime, because the orbit fit will converge in fewer iterations with a better initial condition?

We test this question below; however, our conclusion is that the increase in grid iterations does not justify the reduced time for orbit fitting. This is because the grid iterations are quite computationally expensive - for every $\dot{\gamma}$, we will have to perform our linear approximation of α, $\dot{\alpha}$, β, and $\dot{\beta}$ at the individual tracklet level, and then cluster. Since each step in this grid is not a trivial amount of computation, increasing the grid density leads to minimal gain with a significant amount of added cost. And in comparison, the runtime savings in the orbit fit are minimial (up to 39% faster) with larger grid steps:

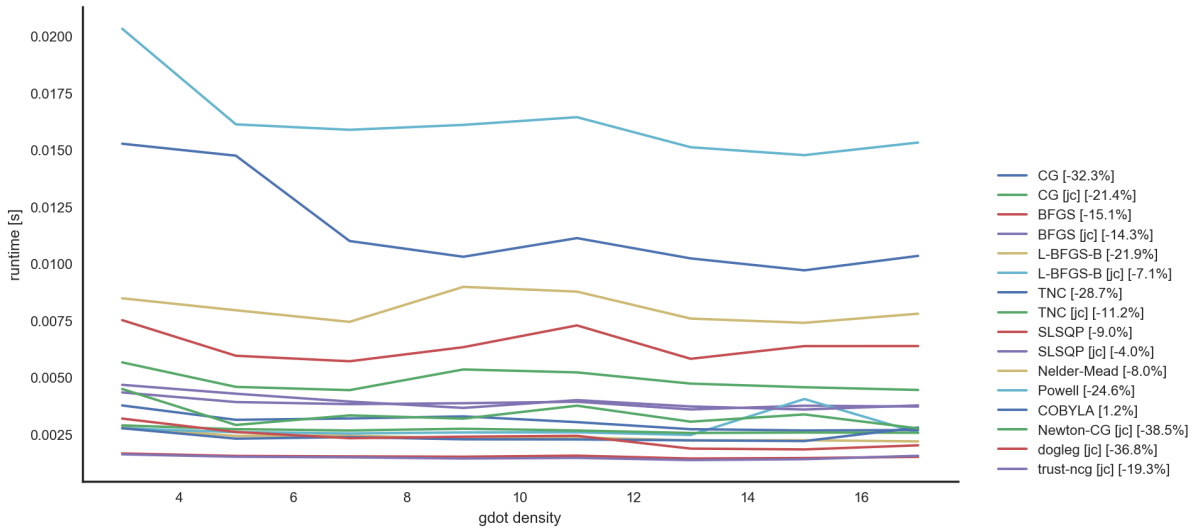


Figure 4: Runtime comparison of different solvers when changing $\dot{\gamma}$ resolution. As can be seen, some solvers converge up to 39% faster given higher density, while others perform equivalently.

In addition to runtime optimization, the orbit-fitting step is also necessary to correct errors in the other five motion parameters. There will be natural differences in the tracklets regardless of how fine the γ and $\dot{\gamma}$ grid is, due to perturbation from observational error, projection error (when we project to the tangent plane of our reference time), and approximation error when we calculate the α, $\dot{\alpha}$, β, and $\dot{\beta}$, using a linear approximation without regard to gravity.

Although all these errors will be small, they will still constitute a difference between individual tracklets even with a near-continuous grid of $\dot{\gamma}$. So, even with a very fine grid of $\dot{\gamma}$, we will still have differences in our clusters and will need to fit some sort of orbit to get all six orbital parameters.

Conversely, if we set the $\dot{\gamma}$ iteration too coarse, we either might not find clusters that really exist in the first place, or we will not have a good enough starting point for our orbit fit to converge to a result. The best $\dot{\gamma}$

spacing to choose is a grid that is just fine enough to capture the majority of the meaningful clusters, but not overly fine as to waste computational time. Our conclusion is that roughly five steps for $\dot{\gamma}$ is the correct balance, and that Matthew Holman's initial intuition for Stage 1 of this project is best.

4.2 Step 7: Extending slices to include singleton tracklets

When we did our initial run through the dataset, we wanted to keep our error rate low, even if it meant sacrificing the number of clusters identified. To accomplish this we tuned our hyperparameter for the radius of the KD-Tree search with our training data to find the maximum number of clusters while keeping the error rate below 0.1%.

This radius did a very good job of identifying "pure" asteroid slice clusters both in the training data and in the ITF data. However, this approach is very conservative and will often miss "singletons" that are just outside of the cluster radius, even if it is likely that these singletons are part of the cluster.

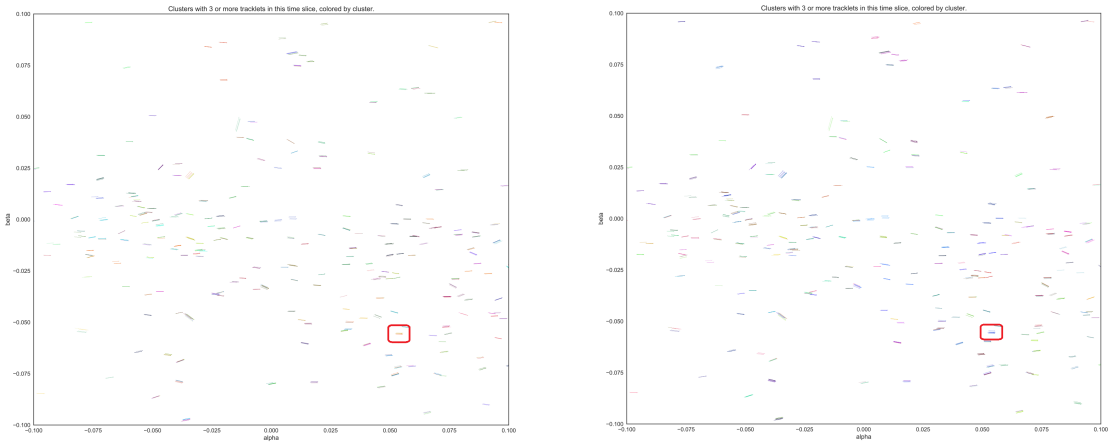
We want to be able to find the singletons related to a slice cluster so we can make that cluster's orbit more robust and cluster as many related tracklets as possible. To cluster these additional singletons, we could extend the radius of the cluster in our initial pass, which would certainly cluster more of the singletons, but that would also result in a higher false positive rate. Our orbit fit optimization step has enabled a better solution, though.

Once we initially cluster the tracklets and fit the orbits for each cluster, we have fitted orbital parameters α, $\dot{\alpha}$, β, $\dot{\beta}$, γ, and $\dot{\gamma}$ for every cluster. Recall when we initially clustered the tracklets we used a grid of $\dot{\gamma}$ values and a constant γ assumption to find the α, $\dot{\alpha}$, β, and $\dot{\beta}$ parameters. Now that we have performed the orbit-fitting optimization, we have more robust values for γ and $\dot{\gamma}$. We can use these values with our linear approximation of α, $\dot{\alpha}$, β, and $\dot{\beta}$ for each tracklet to find additional singletons.

In this method, we take the KD-Tree we previously built for that time window and look in an arbitrarily large neighborhood to find "candidates" to be added to the cluster. The radius of this neighborhood does not matter as long as it is sufficiently large to encompass all singletons with a reasonable chance of being clustered with the fitted orbital parameters of our cluster. In the interest of time, we choose a 20x multiplier of our initial radius (which is definitely large enough), and accept the performance hit to reduce the number of hyperparameters for us to tune.

Now that we have a "neighborhood" around our cluster, and the γ and $\dot{\gamma}$ values from the orbit parameters for that cluster, we can re-transform the neighborhood using our linear approximation of α, $\dot{\alpha}$, β, and $\dot{\beta}$, but this time with our fitted γ and $\dot{\gamma}$ values instead of the rough values from a discrete grid. If any singleton tracklets surrounding the cluster are really part of the cluster, using the fitted γ and $\dot{\gamma}$ values should bring them closer to the cluster because that γ and $\dot{\gamma}$ are closer to the "real" γ and $\dot{\gamma}$ for the asteroid. So we will be able to add any tracklets to the cluster that move within the original cluster radius of the α, $\dot{\alpha}$, β, and $\dot{\beta}$ parameters.

Some examples of the outcome of this process are illustrated below:



(a) Clusters before extension

(b) Clusters after extension - including new singletons added

Figure 5: Extension of existing clusters - clustered 'singleton' highlighted in red

A closer view of the extend operation on several tracklets is as follows:

(a) Asteroid slice cluster before extend - singleton in green

(b) Asteroid slice cluster after extend - all tracklets are in cluster

Figure 6: Zoomed-in example of cluster extend operation

We think the extension methodology is conceptually valid, and have found instances of it working in practice, but we decided not to include it in our final process due to the computational cost for relatively small gain. We find we add very few singletons to any cluster with this method, and the method takes about $\frac{2}{3}$ the processing time of our initial clustering run. In our opinion, the very small increase in singletons added does not justify the additional computational burden.

4.3 Step 8: Finding clusters of clusters

In addition to missing singletons, our initial clustering approach tends to form very tight sub-clusters that are actually all part of one larger cluster.

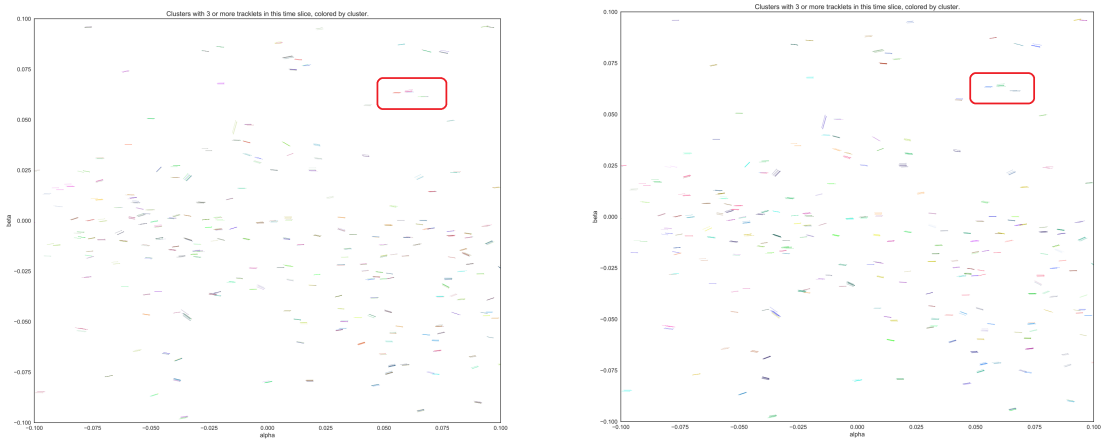
This happens because we want to keep our false positive rate very low, so we keep the search radius of our KD-Tree sufficiently small to maintain an error rate of 0.1% or less in our training data. This is an issue

because the small clusters can give results for the orbit fit, but those results will be less robust than the orbit fitting results of the (true) larger cluster. We want to be able to find as large, valid clusters as possible to increase the robustness of our orbit fit parameters.

There are a few ways to address this problem of clustering sub-clusters instead of finding the cluster as a whole. One way, like the singleton problem, is to just increase the radius on the initial pass. The issue here is the same as the issue with the singleton problem. The KD-Tree will tend to find the larger clusters when we make the radius larger, but it will also include unrelated tracklets due to the larger radius, raising our error rate.

The method we decided to use will again take advantage of the orbit parameters for each slice cluster. Since our orbit optimization increases our number of accurate dimensions from four to six, we can make a new six-dimensional KD-Tree and cluster based on a new radius in six dimensions. This requires a new hyperparameter - namely, the six-dimensional cluster radius. For this, we use the same value we used for the radius in the initial four-dimensional clustering (0.00124 radian).

We then aggregate the tracklet ids for each of the clusters we bring together and treat that result as a new cluster:



(a) Asteroid slices before six-dimensional cluster of clusters

(b) Asteroid slices after six-dimensional cluster of clusters

Figure 7: Effects of six-dimensional cluster-of-cluster operation - note highlighted slices are corrected

The KD-Tree nearest neighbour search is fast $O(log(n))$ and effectively clusters sub-clusters instead of one large, correct cluster. We will accept these as our fully refined "asteroid slices" and next we will assemble the slices into time-independent, full asteroid trajectories.

4.4 Step 9: Convert asteroid slices to time-independent "orbital elements" domain

Now that we have clustered the clusters in each window, we have a more robust estimate for our fitted orbit parameters of each asteroid "slice." But those slices are each valid in a small window of time (the 1-month period they have been windowed into), and often slices months or years apart will redundantly correspond to the same asteroid. To correct this, we will transform into the "orbital elements" domain, which defines the asteroid orbit at all times.

Using Kepler's equations, we can transform the motion parameters we have been using: α, $\dot{\alpha}$, β, $\dot{\beta}$, γ, and $\dot{\gamma}$ into the commonly used "orbital elements" domain which are given by the six parameters: a, E, i, Ω, ω, and m. In the orbital elements paradigm, a is the semi-major axis of the ellipse, E is the eccentricity, i is the inclination, Ω is the longitude of the ascending node, ω is the angle to the periapsis (point the asteroid is closest to the sun) from the line of nodes, and m is the mean anomaly, where the asteroid is located on its orbit. The line of nodes is the line created from the intersection of the orbital plane of the asteroid and the reference plane, in our case the equatorial plane of the earth.

This transform takes us back into the realm of well-established astronomy orbit parameters. Although the orbital elements are a well established convention in astronomy, we have a much better reason than convention for choosing this transform.

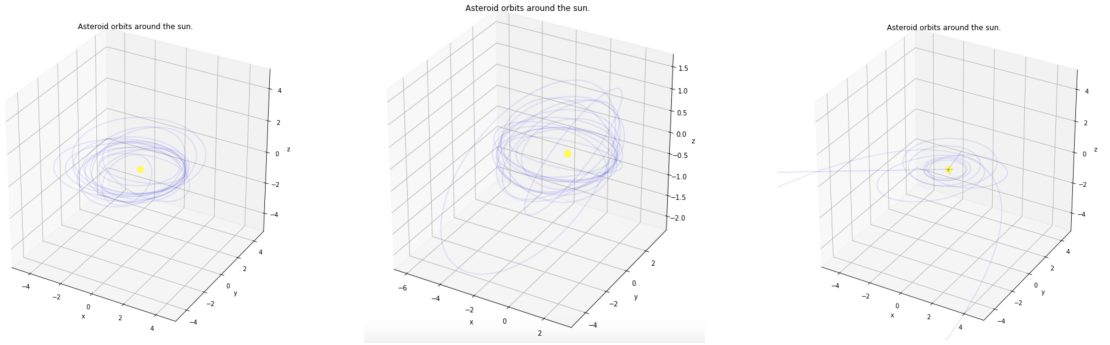
The beauty of this transform is that we go from several parameters dependent on time, to only one parameter dependent on time. The other five orbital elements are not strictly independent of time, but they are approximately independent, which is close enough for our purposes. This means that once we do this transform, we can compare the first five orbital element parameters of asteroid slices across *all* our time frames.

4.5 Step 10: Meta-cluster slices into full asteroid trajectories

Now that we are in a time independent domain, we can simply use a KD-Tree to spatially compare and cluster over much longer time horizons, just as we previously did on short time horizons. We can even compare clusters we made in different years and check to see if they are related.

This meta-cluster has one hyperparameter - namely, the cluster radius. We use the "elbow" method to set this to 0.05 across all five-dimensions in the clustering (this is discussed further in the Results section below with the cluster accuracies), and again perform a KD-Tree nearest neighbours clustering.

Some sample trajectories are included in figure 8. Each plot has several orbits that illustrate the best, average, and worst cluster variances, respectively:



(a) Example asteroid orbit - lowest cluster variance

(b) Example asteroid orbit - typical cluster variance

(c) Example asteroid orbit - highest cluster variance

Figure 8: Several illustrations of calculated asteroid trajectories around the sun - various cluster error levels

4.6 Meta-clustering: Allocating shared tracklets to asteroid trajectories

One treatment of note in the Step 10 Metaclustering is the treatment of numerous overlapping asteroid meta-clusters. This a limitation of the KD-Tree nearest neighbors method - while it is fast and accurate, there is the possibility that multiple asteroid paths "claim" the same tracklet, meaning that tracklet is within the acceptable radius in our KD-Tree for all the meta-clusters that claim the tracklet.

This means we must come up with a heuristic for allocating tracklets to meta-clusters in a manner that will maximize likelihood of getting high quality clusters. If cluster A and cluster B claim the same tracklet, which cluster should "get" the tracklet? We present four different heuristics for allocating tracklets to clusters:

- The first method is the "fit" method. In this method we prioritize clusters that have low orbital fit error. For each cluster we have a related orbital fit error measure, and we will simply assign the tracklets to the cluster with the lowest orbital fit error first, and then the next lowest and so on. The idea here is that the better clusters will have lower fit errors because they really form an orbit, and the clusters that do not really form an orbit will have higher fit errors.
- The next two heuristics are based on the size of the cluster. The "smaller" heuristic prioritizes smaller clusters and then moves to larger clusters, and the "larger" heuristic does the opposite.
- Finally, the most straightforward heuristic we present is to randomly select a cluster from the set of clusters that claim a tracklet when assigning the tracklet to a cluster. We call this method the "coin" method.

We apply this method to the errant tracklets and classify each asteroid as a "Success," "Potential," or "Failure." We define these terms specifically in the results section of this report, but for now, these can be thought of as decreasing quality for goodness of fit. The four methods performed as follows:

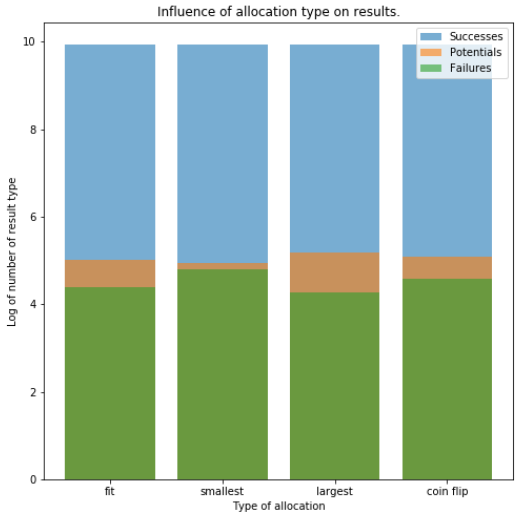


Figure 9: Classification accuracy with several de-duplication methods

Figure 9 shows the best options will be either the "fit" option or the "largest" option. We decided to opt for the fit option because it seems to perform well, and it is the most defensible from a mathematical standpoint. Meaning, the orbital fits with low fit errors are more likely to be real orbits than the fits with high fit errors. This makes more sense from a first principles standpoint than saying "larger clusters are usually better than smaller clusters."

5 Results

5.1 Results: Defining success

The Minor Planet Center, with thanks to Matt Payne, provided us with a labeled training set of known asteroids for tuning hyperparameters and testing efficacy of the algorithm.

This dataset includes a total of 81,982 tracklets which correspond to 21,359 known asteroids when properly matched. Our Stage 1 project from CS182 was able to identify 70,434 partial and overlapping clusters, which we have defined above as "asteroid slices."

Each of these slices has a partial set (four out of six) of motion parameters, and each of those sets of parameters is assumed valid for a small time window (one month). Note that the number of slices is much higher than the number of asteroids, because multiple slices in different time windows may correspond to a single asteroid.

The objective of this project was to take these slices and assign as many as possible to full asteroid trajectories. A full asteroid is superior to a slice in three ways:

- The asteroid's trajectory is a full set of five "orbital elements" that completely define its orbit (rather than just four motion parameters that partially define it)
- The asteroid trajectory will be defined as a continuous function of time, rather than a single "snapshot" in time
- Once the trajectory is defined for all times, observations of the same asteroid that are months or years apart can be grouped together into "meta-clusters"

Our goal was to assign orbital element parameters and identify as many of the 21,359 asteroids as possible. We define a success when every tracklet in the discovered asteroid perfectly matches our training label (i.e., a successful asteroid has zero false positive tracklets and zero false negative tracklets).

5.2 Results: 69.9% of clusters identified

The full evolution of the dataset is shown in Figure 10 below. We are pleased to report that the end-to-end process correctly identified 14,929 of the 21,359 asteroids in the labeled dataset. The overall progression of asteroid numbers was as follows:

1. Import the 81,982 tracklets (partial asteroid observations) from the training set
2. Perform nearest neighbours clustering on the four known dimensions for each tracklet (position and velocity in the two lateral dimensions). This results in 79,434 slices, including a number of overlapping or spurious clusters
3. Perform the constrained optimization described above on each slice to infer a least-squares fit of the two missing radial motion parameters, γ and $\dot{\gamma}$ (still 79,434 slices at this point)
4. Create a cluster-of-clusters of the asteroid slices in six dimensions. This greatly improves the quality and we go from 79,434 poor-quality slice-clusters to 35,387 high-quality clusters. After the aggregation, each slice cluster is more likely to be unique in its own time window.
5. Use Kepler's equations to translate into full time-independent orbital parameters, and cluster on those to identify individual asteroids. The clusters identified a total of 20,964 asteroids, and after dropping clusters representing the same asteroid, we correctly identified a total of 14,929 unique asteroids.

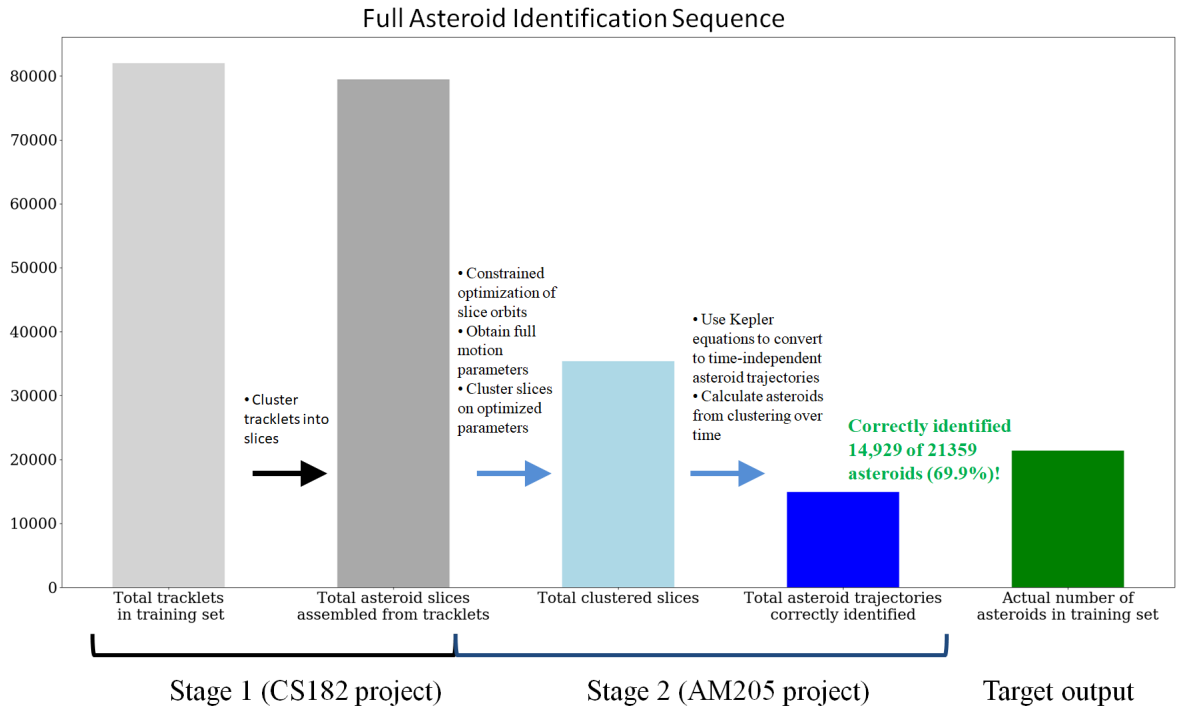


Figure 10: Full sequence of steps involved in identifying asteroids. Our procedure identified 69.9% of all asteroids correctly

Thus, we have correctly identified 69.9% of the asteroids in the dataset. The overall error rate was quite low, with a total of 1.1% of identified asteroids containing at least one incorrect tracklet.

5.3 Results: Classification of failures

We also examined the incorrectly classified asteroids and divide them into two categories.

If the spurious asteroid contains three or more tracklets from any labeled training asteroid, we categorize the asteroid as "Potential" as the actual orbit can likely be inferred by choosing the modal tracklets. If there are fewer than three tracklets from any single asteroid, the meta-cluster is classified as "Failure."

The meta-cluster error log distributions are as follows:

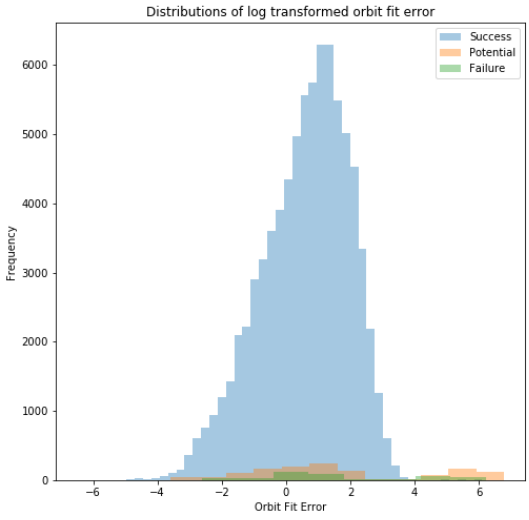


Figure 11: Log fit error of each classification type

We note that the Potential/Failure loss follows a bi-modal distribution, for which we can only speculate as to the drivers. However, we are happy to see a relatively lognormal distribution for correct asteroids, which implies our methodology for identifying correct asteroids is not subject to significant bias.

5.4 Results: Hyperparameter tuning and tracklet deduplication

Our Stage 2 algorithm has one hyperparameter to tune - specifically, the cluster radius when performing KD-Tree asteroid meta-clustering.

Our intent is to capture as many asteroids as possible without spuriously clustering two nearby slices. So our approach was to use the "elbow method" to select the largest cluster radius before error numbers begin to increase:

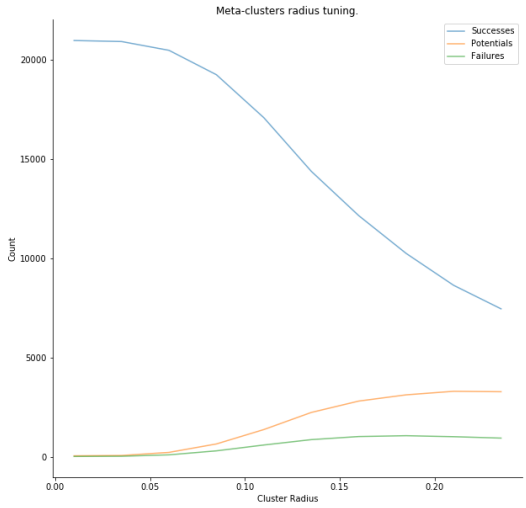


Figure 12: Tuning of meta-cluster radius hyperparameter

Using the results in 12, we selected a cluster radius at the error "elbow" of of 0.05. This radius is a synthesis of five different orbital element dimensions, and therefore the units are not particularly meaningful; however most of the orbital elements are measured in radians.

6 Conclusion

We are pleased with our results - not only in terms of accuracy, but also computational efficiency.

First, in terms of accuracy, our methodology has successfully identified asteroids from a series of noisy observations, which are highly disparate in both time and geography. More so, this has been in a state space that is both large and dense. We believe our 69.9% correct identification is an achievement, especially when considering that to the best of our knowledge, the problem was previously unsolved.

One of the key achievements of this methodology is its efficiency in dealing with the large dataset. Previous brute force solutions such as MOPS [1] run in $O(n^3)$ time, so the 3-million observation ITF is computationally difficult. In fact, the ongoing construction of the LSST telescope array in Chile includes an on-site supercomputer for solving calculations such as these. Our solution runs the approximately 80,000 tracklet labeled set end-to-end on a 2017 MacBook Pro in about ninety minutes, and we expect the full ITF will run in less than two days.

6.1 Limitations and Next Steps

We believe there are opportunities to improve this methodology further. Some of the key next steps on this project would be:

- *Accuracy improvement:* Account for the impact of gravitational effects in the orbit-fitting procedure to further reduce our loss functions.
- *Accuracy improvement:* Currently our minimization routine assumes that $\alpha/\dot{\alpha}$ and $\beta/\dot{\beta}$ are not constrained. We believe the wealth of training data compensates for this in the loss minimization scheme, but a more sophisticated model would only allow for the symbolic link between these pairs.
- *Accuracy improvement:* We note that the loss of the incorrect asteroids follows a bimodal distribution in Figure 11. We don't know the cause of this but we do believe it is a sign that further steps can be taken to isolate incorrect asteroids.
- *Implementability improvement:* the multi-stage nature of this project required us to build several partially overlapping data structures to process the tracklets to completion. A code rebuild would allow us to move all tracklets, slices, and asteroids into a single master dataframe. This would allow us to more easily link each asteroid to its original constituent tracklets.

Finally, the major next step is to run the full algorithm on the full Isolated Tracklet File. We hope to perform this with the MPC and confirm the discovery of hundreds of thousands of previously unknown asteroids.

6.2 Acknowledgements

We would like to first thank Matt Holman, AM205 TF and our teammate for Stage 1 of this project, for his patient support with our rudimentary understanding of astronomy.

Thanks also to the Minor Planet Center and Matt Payne for providing training data and testing support.

Finally, thanks to the entire AM205 teaching staff for their instruction with this class, and in particular Chris Rycroft for igniting a lifelong love of cool-looking asteroid plots in HW3 :)

6.3 Tools used

We implemented all methods in Python and made use of some popular libraries for astronomy (*novas* and *novas_de405* for positional astronomy, *healpy* for manipulating *healpix* maps). We also used methods from the Minor Planet Center's *kepcart* package for calculating orbital elements.

This code-base also builds on the previous code-base for completing stage one of the project, such as the asteroid slice fitting. The two projects also share common data structures.

Labeled training data was provided by the MPC, and the Isolated Tracklet File, while also managed by the MPC, is public domain.

References

[1] G. T. e. a. Kubica J., Denneau L., "Efficient intra- and inter-night linking of asteroid detections using kd-trees," *Icarus*, vol. 189, pp. 151–168, 2007.

[2] P. Veres and S. Chesley, "Near-earth object linking with the large synoptic survey telescope," *Astronomy J. (Submitted to)*, 2017.

[3] G. Bernstein and B. Khushalani, "Orbit fitting and uncertainties for kuiper belt objects," *The Astronomical J.*, vol. 120, no. 6, p. 3323, 2000.

[4] J. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[5] J. Nelder and R. Mead, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, no. 2, pp. 155–162, 1964.

[6] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Research of the National Bureau of Standards*, vol. 29, no. 6, pp. 409–436, 1952.

[7] D. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1970.

[8] C. Broyden, "The convergence of a class of double rank minimization algorithms," *The new algorithm, J.*, vol. 6, pp. 222–231, 1989.

[9] L. F. Grippo, L. and S. Lucidi, "A truncated newton method with nonmonotone line search for unconstrained optimization," *J. Optimization Theory and Applications*, vol. 60, no. 3, pp. 401–419, 1989.

[10] M. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," *Kluwer Academic, Dordrecht*, vol. 275, pp. 51–67, 1994.

[11] C. Lawson and R. Hanson, "Solving least squares problems," *DFVLR*, 1975.

[12] T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization," *Siam J. Numer. Anal.*, vol. 20, no. 3, pp. 626–637, 1983.