OPTIMIZATION TECHNIQUES FOR LONG TERM ACTIVE DEBRIS REMOVAL MISSION DESIGN APPLICATIONS

A Thesis

by

TYLER J. DOOGAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Manoranjan Majji
Co-Chair of Committee,	Robert E. Skelton
Committee Members,	Robert Brown
	John L. Junkins
Head of Department,	Rodney D. Bowersox

August 2019

Major Subject: Aerospace Engineering

ABSTRACT

Debris in Low Earth Orbit (LEO) poses a great risk to humanity's access to space in the coming years. Seemingly the only way to prevent a full scale catastrophe rendering LEO unusable is to begin a series of active debris removal (ADR) missions.

An approach for designing a series of active debris removal missions is presented in this thesis. This approach has multiple steps and begins with a large list of high risk, high mass debris objects. The first stage of the planning process entails the use of clustering algorithms to partition a large catalogue of orbits into groups of user defined sizes. It is shown that this operation is tantamount to histogram analysis in non-Euclidean spaces. The second stage of the approach then solves a dynamic traveling salesman problem to compute an order of visitation through each cluster. Lastly, a novel trajectory optimization technique is presented, using Chebyshev polynomials to approximate the dynamics of the system (rather than the states, which is typical). This technique is then used to further optimize the solution of the dynamic traveling salesman problem in order to arrive at a locally optimal solution to each multi-rendezvous problem.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Majji, and my committee members, Dr. Skelton, Dr. Brown, and Dr. Junkins for their guidance and support throughout my time here.

Thank you to all of my friends here at Texas A&M for making this possible by being so bad at smash and giving me moral victories when we played. Also for all the morning coffees, taco Tuesdays and the late night discussions on flutter.

Most importantly thank you to my Mother and Father, who made this possible with all of their love and support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

The topic of Active Debris Removal (ADR) has gained significant attention over the past few decades. As the number of objects in Low Earth Orbit (LEO) increases from year to year, we come closer and closer to the "Kessler Syndrome" first described by Kessler and Cour-Palais [1] in 1978, leading to a point of no return in which the number of debris objects created from collisions will exceed the number of objects decaying, eventually rendering LEO unusable. In order to prevent this from becoming a reality two things need to be done. Firstly, the number of new debris objects being put into LEO from launches needs to be reduced. However, this alone is not enough to prevent the problem; studies have indicated that at least 5 large LEO debris objects must be removed each year (along with a no further release scenario) in order to stabilize LEO for the future [2]. The challenge of finding the most accessible debris objects to remove is a complex problem in its own right. While more information about the large objects mus be obtained, a routing strategy to schedule the visitation of the objects that are orbiting around the Earth is a key challenge. The main questions tackled in this thesis are the following. If we have a list of objects to take down, how do we divide this list into tractable missions, and how do we optimize the path through each mission?

The problem of optimal multi-rendezvous mission design not only has applications to active debris removal but is also important for potential satellite refueling missions or repair missions.

The problem that will be investigated in this thesis can simply be put as taking a large list of high risk debris objects and planning successive missions to remove at least 5 objects per year using either a low thrust or impulsive thrust retrieval satellite. The list of debris objects will be a list of 640 objects retrieved from a collision analysis done by Brent Barbee [3].

1.2 Overview

The problem of long term ADR can be split into a few distinct problems. Firstly the list of high risk objects needs to separated into missions of a tractable size, throughout this thesis the process of separation will be known as clustering. Secondly, each mission path will need to be optimized for fuel efficiency; this optimization can be posed as a multiphase optimal control problem.

In the past, clustering has primarily been a tool of data scientists to group together similar data into clusters. Similar methods can be used for a large set of debris objects in various orbits. The difference in this clustering when compared to classical clustering is that each cluster will represent a mission, meaning that the clusters will have a temporal order to them, complicating the clustering problem itself. Additionally, each cluster will be restricted in size as there are only so many objects that can be removed in a given mission.

In Chapter 2 of this thesis, the k-medoids clustering algorithm and the generalized assignment problem (GAP), along with some GAP approximation algorithms, will be introduced. Knowledge of these two topics proves important when developing an algorithm for mission clustering. Next, a novel clustering algorithm, modified k-medoids (MKM), will be presented in order to solve this problem for an arbitrary number of objects and an arbitrary number of missions.

After clustering, many distinct missions will be created, and the only thing left to do will be to optimize the path and order for each mission. Firstly a Dynamic Traveling Salesman Problem (DTSP) will be solved to determine an optimal order for the rendezvous', the subsequent optimization of the path can be posed as a multiphase optimal control problem. As with most optimal control problems of this type (trajectory optimization problems) it is nearly impossible to solve analytically, instead it is solved using a numerical method. Numerical methods in optimal control are typically broken up into two types, direct and indirect, with the latter making use of the adjoint equations and the former not [4]. A more in-depth discussion on the differences between the two will be presented in Chapter 3.

In this thesis, a novel direct method called Chebyshev Dynamics Approximation Method, shortened to CDAM, will be used to solve the multiphase trajectory optimization problem for each

mission. As with most direct methods, this method boils down to a nonlinear programming (NLP) problem. The NLP solver SNOPT will be used to solve all NLP problems throughout this thesis [5]. In Chapter 3, CDAM will be outlined and used to solve a few simple optimal control problems including the Brachistochrone problem as well as the Earth Mars transfer problem, simply to give the reader a taste of how it works and prove its validity. Next, in Chapter 4, a long term mission using a list of 640 high risk satellites will be designed. Firstly, the clustering will be performed and the DTSP will be solved for this list. Next, the multiphase optimal control problem will be outlined and solved for a low thrust assumption.

Chapter 5 will discuss the conclusions and possible future work that can stem from this thesis.

1.3 Literature Review

Previous research into how to tackle the problem of ADR includes Missel et al. [6] who optimized, using a genetic algorithm, the path for a rendezvous satellite. This optimization was specific to a satellite that uses the capture and ejection of debris as a form a thrust.

Braun et al. [7] analyzed chemical vs. electric propulsion and each method's feasibility for single missions.

Barbee et al. [3] identified a group of high risk, high mass objects and used a series method algorithm (minimum next step) to solve the traveling salesman problem. The objects found through this collision analysis will be used in Chapter 4 of this thesis in order to demonstrate the techniques developed.

Alfriend et al. [8] developed a Geo-Synchronous rendezvous strategy by developing and solving a traveling salesman problem; this approach will be modified and expanded upon in order to account for geopotential perturbation in this thesis.

Cerf [9] found an optimal mission profile for a small set of debris objects using a simulated annealing algorithm. The approach used by Cerf is similar to what will be used in this thesis, specifically with regards to using a discretized cost matrix (which will be seen in Chapter 2); however, Cerf's optimization used a heuristic method and only involved 12 objects.

One glaring hole in all of this previous research is the lack of any mission design for a large

list of debris objects, when in reality many objects will have to be removed in order to prevent a Kessler Syndrome from becoming a reality.

2. CLUSTERING AND THE DYNAMIC TRAVELING SALESMAN PROBLEM

In the first part of this chapter, the orbital dynamics used throughout this thesis and within the clustering and dynamic traveling salesman problems will be derived. The generalized assignment problem (GAP) and a clustering algorithm will then be introduced. The GAP is an NP-hard problem that appears within the clustering algorithm and must be approximated as it can not be analytically solved for a problem of this size. Using this clustering algorithm based on the K-medoids clustering algorithm, the objects can be placed into distinct missions that occur within distinct time periods, with the assumption being that they do not occur simultaneously but rather one right after the other. Once the objects have been clustered into missions the visitation order within each mission must be determined. This will be done by solving a Dynamic Traveling Salesman Problem (DTSP) using a variation of the Miller Tucker Zemlin algorithm for the static traveling salesman problem [10].

2.1 Orbital Mechanics

The entirety of this section will deal not with the osculating orbital elements, which are the exact orbital elements at a given time, but instead will deal with the mean orbital elements, these are the orbital elements derived from a doubly averaged Hamiltonian function. These elements are much easier to work with, and due to the fact that only J_2 effects will be considered, the only variations that occur end up being a secular variation of the right ascension of the ascending node (RAAN).

2.1.1 Orbital Elements

For the purposes of this section, the classical Kepler elements will be introduced. In a future section a different, nonsingular set of orbital elements will be introduced that are easier to work with for that respective section. The classical Kepler elements are defined as follows, and can be defined geometrically in Figure 2.1 [11].

$$a := \text{Semi-Major Axis} \tag{2.1}$$

$$e := \text{Eccentricity} \tag{2.2}$$

$$i := \text{Inclination} \tag{2.3}$$

$$\Omega := \text{Right Ascension of the Ascending Node} \tag{2.4}$$

$$\omega := \text{Argument of Perigee} \tag{2.5}$$

$$\nu := \text{True Anomaly} \tag{2.6}$$

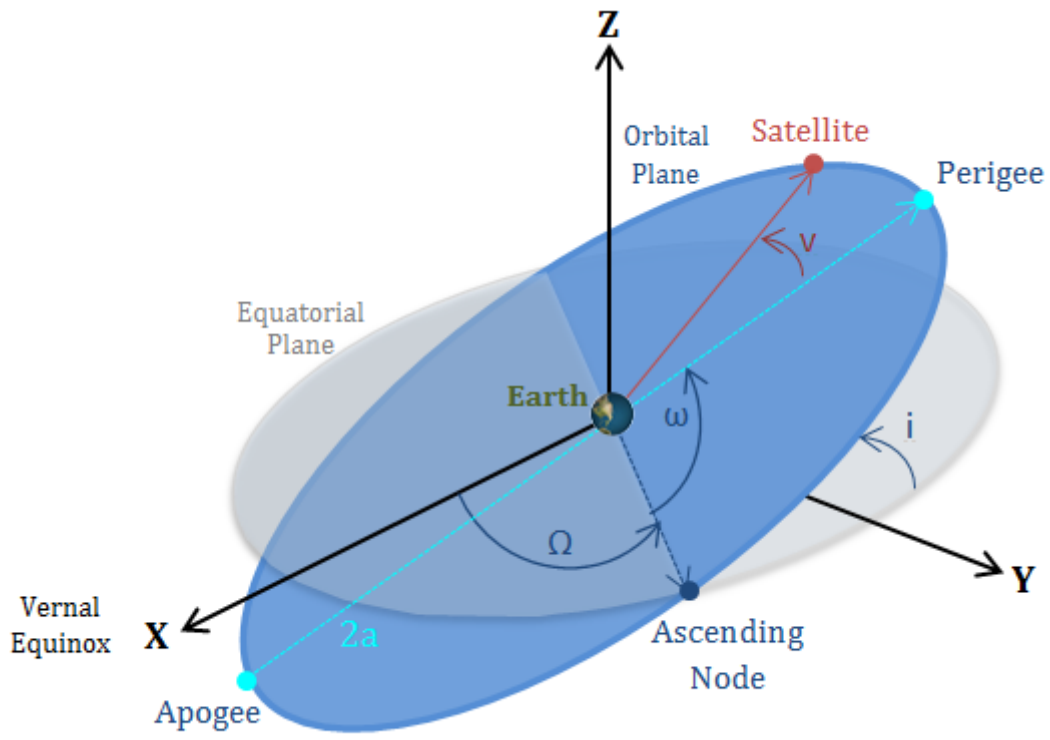

Figure 2.1: Definition of Orbital Elements

Figure 2.1 does not explicitly show the eccentricity. But it is defined the same as the eccentricity for any arbitrary ellipse.

2.1.2 Gravitational Potential

The gravitational potential of the Earth is well known and commonly used. It is shown in Equation 2.7 [11].

$$V_{Earth} = -\frac{\mu}{r}\left[1 - \sum_{n=2}^{\infty} J_n \left(\frac{R_e}{r}\right)^n P_n(\sin\theta) + \sum_{n=2}^{\infty}\sum_{m=1}^{n} P_{mn}(\sin\theta)\left(\frac{R_e}{r}\right)^n (C_{nm}\cos\lambda + S_{nm}\sin\lambda)\right] \tag{2.7}$$

However, for this problem we are only considering the J_2 term as it is the most dominant of all of the geopotential terms. Therefore, the gravitational potential of the Earth is shown below with

the substitution for the second Legendre polynomial already made in Equation 2.8.

$$V_{Earth} = -\frac{\mu}{r} + \frac{J_2}{2}\frac{\mu}{r}(\frac{R_e}{r})^2(3\sin^2\theta - 1) \tag{2.8}$$

With the assumption of circular orbit, which can be justified as the objects in this thesis all have eccentricities on the order of 10^{-2}, and knowledge of θ in terms of the orbital elements. The potential term is shown in Equation 2.9.

$$V_{Earth} = -\frac{\mu}{a} + \frac{J_2}{4}\frac{\mu R_e^2}{a^3}[(1 - 3\cos^2 i) - 3(1 - \cos^2 i)\cos 2(\nu + \omega)] \tag{2.9}$$

The average gravitation potential over a single orbit can be calculated by averaging out the fast variable, in this case the true anomaly, ν, from Equation 2.9. This average gravitational potential proves to be useful as the secular variations of the orbital elements can be extracted from it. The averaged gravitational potential is shown in Equation 2.10.

$$\bar{V}_{Earth} = -\frac{\mu}{a} + \frac{J_2}{4}\frac{\mu R_e^2}{a^3}(1 - 3\cos^2 i) \tag{2.10}$$

In order to calculate the variations of the orbital elements, it is beneficial to first convert to a set of canonical orbital elements, the Delaunay Variables, which are defined in Equations 2.11-2.16 [12].

$$L = \sqrt{\mu a} \tag{2.11}$$

$$G = L\sqrt{1 - e^2} \tag{2.12}$$

$$H = G\cos i \tag{2.13}$$

$$l = \nu \tag{2.14}$$

$$g = \omega \tag{2.15}$$

$$h = \Omega \tag{2.16}$$

Using this set of canonical elements will allow for the direct calculation of the orbital element variations by a simple partial derivative. As it turns out, the only element that varies secularly due to the J_2 perturbations is the RAAN, Ω. The equation for this comes from the following, where K is the Hamiltonian of the system.

$$\dot{h} = \dot{\Omega} = \frac{\partial K}{\partial H} \tag{2.17}$$

$$K = T + \bar{V}_{Earth} = \frac{\mu^2}{2L^2} + \frac{1}{4} J_2 \frac{\mu^4}{L^6} R_e^2 (1 - 3\frac{H^2}{G^2}) \tag{2.18}$$

$$\dot{\Omega} = -\frac{3}{2} J_2 (\frac{R_e}{a})^2 \sqrt{\frac{\mu}{a^3}} \cos i \tag{2.19}$$

2.2 Clustering Algorithm

2.2.1 Standard K-Medoids Algorithm

The clustering algorithm developed in this section is an extension of the classical K-medoids algorithm which is review in Table 2.1.

Standard K-Medoids Algorithm

Input: A set, S, of n objects along with their attributes. The number, k of clusters that are desired.

Output: k clusters that minimize the intra-cluster Euclidean distances.

1. Select k objects at random as the initial medoids.
2. Assign each object to its nearest (in terms of Euclidean distance) medoid.
3. Calculate the sum of all distances from each object in a cluster to its respective medoid.
4. Swap one medoid with one non-medoid at random.
5. Repeat steps 2 and 3.
6. If:
    a. The total cost decreased, keep the swap and reset flag to zero.
    b. The total cost increased, undo the swap and increment the flag.
7. Continue until flag reaches a designated number of maximum iterations.

Table 2.1: Standard K-medoids Algorithm

In order to apply the K-medoids algorithm to the problem at hand, it needs to be slightly altered. First off, the number of objects in each cluster needs to be fixed in order to ensure that each ADR mission formed removes a set number of objects. Additionally, the costs associated with each cluster are not Euclidean distances, they are $\Delta V's$ associated with orbital transfers that are calculated *a priori*.

In order to calculate the costs associated with the clusters we begin with a set of N objects in elliptical orbits of arbitrary inclination, right ascension of the ascending node (RAAN), semi-major axis, and eccentricity. Creating a matrix of these costs, where C_{ij} represents the ΔV associated

with going from object i to j, will prove useful for solving the DTSP. These costs (ΔVs) between orbits can be calculated in various ways. For the purposes of this thesis the cost matrix, C, will be calculated as a four impulse transfer, the four impulses represent a circularizing maneuver, an inclination change and an eccentricity matching maneuver shown in Equation 2.20. The four impulse transfer ΔV is calculated using the ad hoc rule below where r_p is periapsis and r_a is apoapsis.

$$\begin{aligned} C_{ij} = &\left|\sqrt{\tfrac{\mu}{r_{ai}}} - \sqrt{\mu(\tfrac{2}{r_{ai}} - \tfrac{1}{a_i})}\right| + 2\sqrt{\tfrac{\mu}{r_{ai}}}\sqrt{\tfrac{1-\cos(i_i)\cos(i_j)-\sin(i_i)\sin(i_j)\cos(\Omega_j-\Omega_i)}{2}} \\ &+\left|\sqrt{\mu(\tfrac{2}{r_{ai}} - \tfrac{2}{r_{ai}+r_{aj}})} - \sqrt{\tfrac{\mu}{r_{ai}}}\right| + \left|\sqrt{\mu(\tfrac{2}{r_{aj}} - \tfrac{2}{r_{ai}+r_{aj}})} - \sqrt{\mu(\tfrac{2}{r_{aj}} - \tfrac{1}{a_j})}\right| \end{aligned} \tag{2.20}$$

The phasing component of the rendezvous' will not be addressed, as the ΔV associated with phasing is typically far lower (as long as ample time is alloted for the maneuver) than for the other portions of the rendezvous, i.e. inclination change, Hohmann transfer.

2.2.2 Modified K-Medoids Algorithm

The dynamic nature of the problem also needs to be accounted for in the sense that each cluster represents a different mission occurring in a different timespan, and each timespan has different costs associated with it. Additionally, within each timespan the objects are not stationary, their orbital elements move according to Equation 2.19. To deal with this an average cost for each timespan is calculated using a set of discretized cost matrices. The time step used for discretization in this study is one day, but any reasonable time step can be used. Each cost matrix is calculated according to Equation 2.20, with the orbital elements of the objects calculated at the specified time. For a mission that spans n time steps, the average cost matrix is shown in Equation 2.22 using Simpson's rule.

$$C_{ij}^k = C_{ij}(t_k) \tag{2.21}$$

$\bar{C} :=$ Average C over one mission span

$$\bar{C} = \frac{3}{8}\hat{C}^1 + \frac{7}{6}\hat{C}^2 + \frac{23}{24}\hat{C}^3 + \sum_{j=4}^{n-3}\hat{C}^j + \frac{23}{24}\hat{C}^{n-2} + \frac{7}{6}\hat{C}^{n-1} + \frac{3}{8}\hat{C}^n \tag{2.22}$$

The coefficients in Equation 2.22 come directly from Simpson's rule.

Where the $\hat{C}$ matrix is the cost matrix discretized at each timestep (typically 1 day in this paper). This series of average cost matrices is then input into the modified K-Medoids algorithm for ADR mission design, shown in Table 2.2.

Modified K-Medoids Algorithm

Input: A set, S, of n debris objects along with their attributes. The number, k, of objects per cluster that is desired. The set of average cost matrices, $\bar{C_m}$, n / k matrices in total.

Output: n / k clusters with k objects each that minimize the intra-cluster costs. Each subsequent cluster represents a mission in subsequent time spans.

1. Select n / k objects at random as the initial medoids. The order of these medoids represents which mission timespans they are associated with and therefore which average cost matrix to use for each medoid.
2. Solve a Generalized Assignment Problem using the medoids as bins and the respective ΔVs as the cost. The Generalized Assignment Problem will be discussed in the next section.
3. Calculate the sum of all ΔVs from each object to its respective medoid.
4. Swap one medoid with one non-medoid at random.
5. Repeat steps 2 and 3.
6. If
    a. The total cost decreased, keep the swap and reset flag to zero.
    b. The total cost increased, undo the swap and increment the flag.
7. Continue until flag reaches a designated number of maximum iterations.

Table 2.2: Modified K-Medoids Algorithm

This algorithm depends on the solution of another problem (the Generalized Assignment Problem, GAP) which will be introduced in the next subsection. Like the classical K-Medoids algorithm, this modified algorithm does a good job of converging to a local optimum and is relatively

simple and quick. However, this local optimum may differ from the global optimum in a non-insignificant way. The only true way to find the global optimum is with a branch-and-bound algorithm, which for large data sets like this is not feasible. Additionally, the local optimum that is converged to depends nearly entirely on the choice of initial medoids (even more so for this modified algorithm due to each cluster representing a different timespan), with some choices leading to significantly better intra-cluster costs.

2.2.3 Generalized Assignment Problem

As mentioned before, the generalized assignment problem, GAP, is a problem that appears within the clustering problem, and solving it well and efficiently is an important part of clustering. GAP can be described as the following: Placing m objects into N bins, where each assignment of object i into bin j has a cost associated with it c_{ij} and each bin, j has a capacity p_j, in order to minimize the sum of the assignment costs. In the classic GAP the objects have weights, or sizes, associated with them such that some objects fill bins more than other. The GAP is described in the form of a linear program in Equation 2.23 [13].

$$p_j = \text{Capacity of Bins}$$

$$w_{ij} = \text{Weight of object } i \text{ when placed into bin } j$$

$$c_{ij} = \text{Cost of object } i \text{ when placed into bin } j$$

$$x_{ij} = \text{Assignment Matrix}$$

$$x_{ij} \in \{0, 1\}$$

$$\min \sum_{i=1}^{m} \sum_{j=1}^{N} c_{ij} x_{ij} \tag{2.23}$$

Within the scope of this problem the parameters defined above are the following: The bins themselves are the medoids chosen at a specific iteration. The capacity of the bins is the number of debris objects chosen to be in each mission. The weight of the objects will simply be set to one. The cost associated with placing object i into bin j is the ΔV requirement to transfer from object i to object j.

The GAP is known to be an NP-hard problem [14] and therefore cannot reasonably be solved for a sufficiently large problem. Instead, in this application an approximation to the problem will be used in order to ensure solvability. The approximation used for the purposes of this thesis is first defined in Martello and Toth's 1990 book on the multiple knapsack problem [15]. The algorithm is described in Table 2.3.

GAP Approximation Algorithm

Input: Cost matrix C_{ij}.

Output: Assignment matrix X_{ij}.

1. Calculate the difference between the minimum cost and second minimum cost of assignment for each object.
$f_i = \min(C[i,:]) - \min_2(C[i,:])$

2. While there is still room within a cluster, assign $\max(f)$ to the cluster with the minimum cost. If that cluster is full, assign it to the next open cluster with minimum cost.

Table 2.3: Generalized Assignment Problem Approximation

2.3 Dynamic Traveling Salesman Problem

Once the objects have been clustered into missions of a set size, N, the mission paths must be defined. Though the final optimization will be treated as an optimal control problem in the proceeding section, the order that the objects will be visited as well as a good initial guess for the optimization must be found. In order to do this each mission will be treated as a dynamic traveling salesman problem (DTSP) as the objects being visited are continuously being perturbed due to the

mass distribution of the Earth, for this problem this will be assumed to be caused completely by the J_2 term of Earth's gravitational potential and the RAAN is assumed to change based on Equation 2.19.

Similar to the clustering of the objects, the DTSP will be solved by discretizing the cost matrix with a time step of 1 day. Each transfer, of which there are $N-1$ is then placed into an evenly distributed timeslot based on the length of the mission. For example, for a 240 day mission containing 4 transfers each timeslot will be 60 days long. The minimums of all C_{ij}'s for each 60 day time slot is then taken to create a 3D cost tensor, where C_{ij}^k is the solution to the four impulse transfer from the previous section from orbit i to orbit j at time step k (i.e. the k^{th} transfer being performed). The algorithm used is a modification of the Miller-Tucker-Zemlin algorithm for the static traveling salesman problem. This algorithm, which transcribes the problem into a mixed integer convex program (MICP) is shown in Equations 2.24-2.30 [10].

$$X := \text{Matrix of transfers}$$

$$X_{ij} = \begin{matrix} 0, \text{ no transfer from i to j} \\ 1, \text{ a transfer occurs from i to j} \end{matrix}$$

$$k = \text{ Starting point of TSP}$$

$m = \max\ C_k$, kth row of C matrix. Represents the final rendezvous point

$$\sum_{i=1}^{N} X_{ij} = 1 \ \forall \ j \neq k \tag{2.24}$$

$$\sum_{i=1}^{N} X_{ik} = 0, \text{ Starting point has no entry} \tag{2.25}$$

$$\sum_{j=1}^{N} X_{ij} = 1 \ \forall \ i \neq m \tag{2.26}$$

$$\sum_{j=1}^{N} X_{mj} = 0, \text{ Final point has no exit} \tag{2.27}$$

$$X_{ii} = 0 \; \forall \; i \tag{2.28}$$

Additionally, in order to ensure that the solution is a single path, the following constraint must be met.

$$u_i - u_j + NX_{ij} \leq N - 1 \; \forall \; i \neq j \tag{2.29}$$

Where N is the number of objects being solved for, and u is a vector of arbitrary integers. With all of these constraints, the minimum ΔV solution can be found by minimizing the following

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} X_{ij} \tag{2.30}$$

This algorithm has been modified in order to take into account the dynamic effects present in the problem. The modified algorithm, which also transcribes the problem into a mixed integer convex program is shown in Equations 2.31-2.38. The algorithm was found to work reasonably well for DTSP's with 25 or fewer objects.

$$X^k := \text{Matrix for transfer number k}$$

$$\bar{X} := \sum_{k=1}^{N-1} X^k$$

$$q = \text{ Starting point of TSP}$$

$$m = \max \; C_q^k \text{ , qth row of C matrix. Represents the final rendezvous point}$$

$$\sum_{i=1}^{N-1} \sum_{j=1}^{N-1} X_{ij}^k = 1 \; \forall \; k = 1 : (N-1) \tag{2.31}$$

$$\sum_{i=1}^{N} \bar{X}_{ij} = 1 \; \forall \; j \neq q \tag{2.32}$$

$$\sum_{i=1}^{N} \bar{X}_{iq} = 0, \text{ Starting point has no entry} \tag{2.33}$$

$$\sum_{j=1}^{N} \bar{X}_{ij} = 1 \; \forall \; i \neq m \tag{2.34}$$

$$\sum_{j=1}^{N} \bar{X}_{mj} = 0, \text{ Final point has no exit} \tag{2.35}$$

$$\bar{X}_{ii} = 0 \; \forall \; i \tag{2.36}$$

To ensure that the X matrices are sequential (X^1 is the first transfer, etc.) we introduce the constraint.

$$X^k - (X^{(k+1)})^T = \underline{0} \tag{2.37}$$

With all of these constraints, the minimum ΔV solution can be found by minimizing the following in the program.

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N-1} C_{ij}^{k} X_{ij}^{k} \tag{2.38}$$

This is run for all possible start objects, q, and the minimum value is taken.

2.4 Departure Time Optimization

The method, described in the previous section, used to order the objects within a mission is clearly suboptimal. In order to simplify the problem each transfer is pigeonholed into four distinct time slots, for example, for a 240 day mission with 4 transfers the first transfer occurs in the first 60 days, the next transfer the next 60 days and so on. This means that two transfers cannot occur within the same timeslot, though the optimal solution may require this.

With the use of a few reasonable assumptions this problem can be transformed into a constrained nonlinear optimization problem solved by a sequential quadratic programming (SQP) algorithm available in a variety of commercial software. In this development Mathwork's™ Optimization Toolbox™ is used [16]. The most important assumptions made are the following: Firstly, the plane change is assumed to be the most costly portion of the maneuver; secondly, it is assumed that the order found using the dynamic traveling salesman algorithm is the optimal order.

Since the plane change is the most costly maneuver the sum of all of these plane change costs

can be minimized. The sine of half of the plane change angle is given by Equation 2.39. Additionally, the cost of an impulsive plane change transfer for a circular orbit is given in Equation 2.40

$$\sin\frac{\gamma}{2} = \sqrt{1 - \cos i_1 \cos i_2 - \sin i_1 \sin i_2 \cos\left(\Omega_2 - \Omega_1\right)} \tag{2.39}$$

$$\Delta V = 2\sqrt{\frac{\mu}{a}}\sin\left(\frac{\gamma}{2}\right) \tag{2.40}$$

By minimizing the sum of all plane change costs we can minimize the total cost of the path. Additionally, it is desirable to constrain the time between transfers to be above a certain value, δt, this value is the user-defined time needed to reasonably perform phasing, proximity operations and removal.

Using all of this information the nonlinear program shown in Equation 2.41 can be developed where m is the duration of the mission, N is the number of transfers and δt is the time allotted for proximity operations and deorbiting/refueling.

$$\begin{gathered}
\min \textstyle\sum_{i=1}^{N}\left[2\sqrt{\frac{\mu}{a_i}}\sin\left(\frac{\gamma_i}{2}\right)\right] \\
\sin\tfrac{\gamma_i}{2} = \sqrt{1 - \cos i_i \cos i_{i+1} - \sin i_i \sin i_{i+1}\cos\left(\Omega_{i+1} + \dot{\Omega}_{i+1}t_i - \Omega_i - \dot{\Omega}_i t_i\right)} \\
\textit{s.t. } t_1 \geq 0 \\
t_N \leq m \\
t_{i+1} \geq t_i + \delta t \;\forall\; i = 2 : N-1
\end{gathered} \tag{2.41}$$

Implementing this program with a reasonable initial guess (the departure times output from the dynamic TSP algorithm) has a significant impact on the ΔV of the mission. Table 2.4 shows the before and after fuel requirements for an example mission (5 objects, 240 days) is shown below.

Transfer Number	ΔV Before Optimization	Departure Time	ΔV After Optimization	Departure Time
1	886.4 m/s	1 days	882.2 m/s	0.000123 days
2	83.9 m/s	88 days	79.5 m/s	88.27 days
3	225.2 m/s	180 days	67.5 m/s	198.97 days
4	148 m/s	238 days	146.2 m/s	236.73 days

Table 2.4: Before and After Departure Time Optimization

2.5 Summary and Results

2.5.1 Clustering and Routing Example

In order to get a better understanding of the performance of the clustering algorithm and the dynamic traveling salesman problem, a small example will be shown using all of the techniques developed in this chapter. A small subset of 12 randomly generated LEO satellites, with a variety of orbital elements, are clustered using MKM and the paths within each respective cluster will be optimized. This clustering process will be compared to one done via a brute force approach, where the cost associated with every possible combination of clusters and paths is calculated using the same discretized cost matrices that the clustering algorithm and DTSP use. The minimum cost combination is then determined to be the optimal. Figure 2.2 shows the the solution using the techniques covered in this chapter, MKM followed by the solution of the dynamic traveling salesman problem and the departure time optimization compared to the solution using the brute force approach.

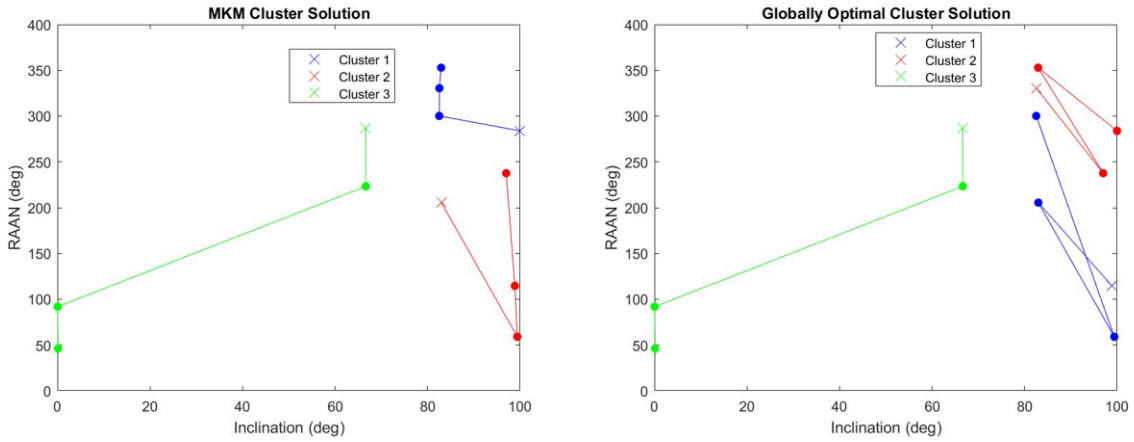

Figure 2.2: MKM vs. Optimal Clustering

Looking at the plots it is clear that they are very similar. Cluster 3 is identical between the two and the first two clusters only have one object switched between the two. The resulting costs of the two are 22.1 Km/s and 24.3 Km/s for the globally optimal solution and the MKM solution, respectively. This is only a 10 percent overall difference between the two. Unfortunately it is not certain how this difference will grow with the scale of the problem.

2.5.2 Summary

Three techniques that, when used in conjunction, can begin the process of designing a set of active debris removal missions have been outlined in this chapter. After a risk analysis is performed to create a list of high risk debris objects, these techniques can be used. Firstly the objects will be separated into missions based on the user's specifications, varying the number of objects in the missions and the mission lengths; this is done using the MKM clustering algorithm. Next, the dynamic traveling salesman problem is solved for each of the missions in order to determine the optimal path and create a reasonable initial guess on the departure times. Using this path and the initial guess on departure times, the departure time nonlinear program optimization can be performed. The resulting mission profiles are reasonable for a 4 impulse transfer. However, these mission profiles can be used as initial guesses for a superior trajectory optimization technique which will be introduced in the next chapter.

3. CHEBYSHEV DYNAMICS APPROXIMATION METHOD

3.1 The Optimal Control Problem

Optimal control is a rich field whose surface could not even be scratched in the pages of this thesis. Instead, the discussion will begin with a general form of an optimal control problem for a dynamic system, also known as Bolza form [17]. Consider, below, a dynamic system, written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{3.1}$$

with $\boldsymbol{x} \in R \rightarrow R^n$ being a vector function of the states and $\boldsymbol{u}$ being a vector function of control variables $\boldsymbol{u}(t) \in R \rightarrow R^m$, used to drive the system in order to achieve a desirable performance. The goal of driving the system in a certain direction is twofold: The first is to minimize a specific cost function, such as the general Bolza form of the cost function below

$$J = \Phi(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{3.2}$$

where $\mathcal{L}$ is the Lagrange term and Φ is the Mayer term [17]; The second is to meet a set of equality and/or inequality constraints, which can be encompassed by the following.

$$\boldsymbol{u}_{lowerbound} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{upperbound} \tag{3.3}$$

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \boldsymbol{0} \tag{3.4}$$

$$\boldsymbol{g}(\boldsymbol{x}(t_f), t_f) = 0 \tag{3.5}$$

Following the classical path of Calculus of Variations [18] the Hamiltonian function of the system can be written as.

$$\mathcal{H} = \mathcal{L} + \boldsymbol{\lambda}^T \boldsymbol{f} \tag{3.6}$$

Where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers known as the costates. The first order optimality conditions for finding an optimal control for a problem where the control is unconstrained are derived from the calculus of variations [17] and shown below.

$$\dot{\boldsymbol{x}} = \mathcal{H}_{\boldsymbol{\lambda}}^T \tag{3.7}$$

$$\dot{\boldsymbol{\lambda}} = \mathcal{H}_{\boldsymbol{x}}^T \tag{3.8}$$

$$0 = \mathcal{H}_u \tag{3.9}$$

While satisfying the boundary conditions.

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.10}$$

$$\Psi(\boldsymbol{x}_f, t_f) = 0 \tag{3.11}$$

In addition to these we have two equations for when the variation of the final state and the variation of the final time are non-zero (i.e. δx_f and δt_f).

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} + \nu^T \frac{\partial \Psi}{\partial \boldsymbol{x}(t_f)} \tag{3.12}$$

$$\mathcal{H}(t_f) + \frac{\partial \Phi}{\partial (t_f)} + \nu^T \frac{\partial \Psi}{\partial (t_f)} = 0 \tag{3.13}$$

Along with extra conditions based on interior point constraints and control constraints there are also second order conditions that need to be satisfied to ensure sufficiency of the candidate

minimizing trajectories. However these will not be covered here.

3.2 Chebyshev Dynamics Approximation Method (CDAM)

Many real-world optimal control problems are very difficult to solve analytically. The adjoint equations and necessary conditions become quite unwieldy and difficult to solve, owing to their sensitivity with respect to variations in initial conditions. Further, the open loop solutions typically need the recalculation of the initial conditions of the adjoint functions when the parameters and initial conditions of the dynamical system are changed. Therefore, many of these problems are solved numerically. The numerical solutions are especially important to provide a nominal path for control of nonlinear dynamical systems. Feedback control can then be invoked to stabilize the equilibrium point determined by such an open-loop solution. Direct computation of feedback control laws for nonlinear systems to optimize a performance function typically involves the solution of a hyperbolic partial differential equation (PDE) called the Hamilton Jacobi Bellman (HJB) equation [19]. The solution of the HJB PDE in high dimensions is also an involved process for most general problems of dynamic optimization. For computation of open-loop solutions to problems one does not need to solve the HJB PDE. In trajectory optimization, most numerical methods can be grouped into two categories: direct methods and indirect methods [4].

Direct methods often involve discretizing the optimal control problem such that the states and controls are piecewise interpolating polynomials or a single global polynomial. This discretized problem can be transformed into a nonlinear program with the dynamics being formed into constraints at the discretization points, the path constraints and boundary conditions can also be applied as constraints in the nonlinear optimization problem, this process is known as transcription [4]. The nonlinear program is set up in such a way as to minimize the discretized cost function. Indirect methods involve looking at the necessary conditions for optimality and attempting to meet them using a numerical scheme. This means that the adjoint equations must be derived, which depending on the problem may be unreasonably difficult [20]. Though indirect methods are often more accurate than direct methods they are sensitive to the choice of certain requisite parameters for optimization and require the derivation of the adjoint equations. For these reasons the method

being introduced and used during this thesis will be a direct method, however, it can be extended and used as an indirect method

The field of direct transcription for solving optimal control problems has flourished in the past 20 years or so, with numerous methods being developed. A survey of these methods would be a paper in its own right, so they will not all be reviewed here. But two of the most popular and powerful methods involve discretizing using Curtis-Clenshaw quadrature [21] and Gauss quadrature [22]. The solution to the states and controls are then found using a basis of Lagrange interpolating polynomials at these points.

The method being described in this chapter, however, differs from these two. Instead of estimating the states using a basis of polynomials, the derivatives of the states (i.e. the dynamics) are estimated using a set of basis polynomials. In the case of CDAM this is done with a set of Chebyshev polynomials at discretization points that are equal to the roots of these Chebyshev polynomials. The first step in using this method is to discretize the optimal control problem.

3.2.1 Discretization

The Chebyshev polynomials are useful as they are orthogonal on the interval $\tau \in [-1, 1]$ therefore the first step will be to scale time to fit this interval in order to preserve orthogonality. This is done with the relation

$$t = \frac{(t_f - t_0)\tau + (t_f + t_0)}{2} \tag{3.14}$$

The integral form of the dynamics (which is equivalent to the differential form in equation 3.1),

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau \tag{3.15}$$

Will be exploited and satisfied at all of the Chebyshev node points, p_k, where p_k can be defined as follows where N is the order of the largest polynomial being used for approximation.

$$p_k = \cos\left(\frac{2k-1}{2N}\pi\right) \tag{3.16}$$

The roots of the Chebyshev polynomials are useful for discretization for two main reasons. Firstly, the polynomials are discretely orthogonal at these points, a property useful for polynomial approximation, and secondly is the location of these discretization points. They are bunched up near the boundaries of the domain $[-1, 1]$, making it easier to approximate the boundary conditions more accurately.

The enforcement of the dynamics at each of the Chebyshev node points is done by first approximating rates of the states in the following way: The state rates function $\dot{\boldsymbol{x}}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ will be approximated as a set of Chebyshev polynomials.

$$f_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \approx \tilde{f}_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \tilde{\dot{x}}_i(t) = \frac{2}{(t_f - t_0)} \sum_{k=0}^{N} \alpha_k T_k \tag{3.17}$$

Here N is a chosen parameter for the number of nodes, which is equal to the number of polynomials, to be used. α_k represents the coefficients of the basis functions and the T_k represents the Chebyshev polynomials of order k. Now a very convenient property of Chebyshev polynomials can be used. This is the fact that the integral of a Chebyshev polynomials is the sum of Chebyshev polynomials itself [23].

$$\int T_k = \frac{1}{2}\left(\frac{T_{k+1}}{k+1} - \frac{T_{k-1}}{k-1}\right), \ \forall \ k \geq 2 \tag{3.18}$$

Integrals of these polynomials can then easily be computed by right multiplying by a matrix of easily computed values. For example.

$$\int_{-1}^{t} \begin{bmatrix} T_0 & \dots & T_k \end{bmatrix} = \Big(\begin{bmatrix} T_0(t) & \dots & T_{k+1}(t) \end{bmatrix} - \begin{bmatrix} T_0(-1) & \dots & T_{k+1}(-1) \end{bmatrix} \Big) \mathcal{J} \tag{3.19}$$

$$\mathcal{J} = \begin{bmatrix} 0 & \frac{1}{4} & 0 & \dots & 0 \\ 1 & 0 & -\frac{1}{2} & \dots & 0 \\ 0 & \frac{1}{4} & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{6} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{2N} \end{bmatrix} \tag{3.20}$$

Using equations 3.19 and 3.20 we can then determine an approximation of the states which will be exact in terms of the approximated dynamics, $\tilde{\dot{x}}_i$. This is done with the integration matrix, therefor both the states and the state dynamics are parameterized using the α optimization coefficients. The equations for state i at discretization point t_k is as follows.

$$x_i(t_k) \approx \tilde{x}_i(t_k) = \boldsymbol{\alpha}^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + x_{i0} \tag{3.21}$$

The controls for the problem can then be approximated the same way that the dynamics were approximated, as a series of Chebyshev polynomials.

$$u_i(t) \approx \tilde{u}_i(t) = \sum_{k=0}^{N} \beta_k T_k \tag{3.22}$$

Using these approximations for the states and the controls the dynamic constraints for the nonlinear programming problem can be formed by enforcing the approximated dynamics to be equal to the dynamics evaluated by plugging the approximated states into the dynamics function.

$$\tilde{\dot{\boldsymbol{x}}}_{\boldsymbol{k}} = \boldsymbol{f}(\tilde{\boldsymbol{x}}(t_k), \tilde{\boldsymbol{u}}(t_k), t_k) \tag{3.23}$$

Additionally, the cost function can be calculated using similar integration rules and Fejer's quadrature rule [24]. Fejer's quadrature rule is a methodology to calculate the integral of a function discretized at the roots of the Chebyshev Polynomials of the first kind, the exact situation that occurs here. The Mayer term, J_M, and the Lagrange term, J_L, of the cost function are calculated

as follows.

$$J_M = \Phi(\tilde{\boldsymbol{x}}(t_f), \tilde{\boldsymbol{u}}(t_f)) \tag{3.24}$$

$$\tilde{x}_i(t_f) = \boldsymbol{\alpha}^T((\boldsymbol{T}(1) - \boldsymbol{T}(-1))\mathcal{J}) + x_{i0} \tag{3.25}$$

$$\tilde{u}_i(t_f) = \sum_{k=1}^{N} \beta_k T_k(1) \tag{3.26}$$

$$J_L = \frac{t_f - t_0}{2} \int_{-1}^{1} \mathcal{L}(\tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{u}}(t), t)dt \tag{3.27}$$

$$J_L = \frac{t_f - t_0}{2} \boldsymbol{w}^T \mathcal{L}(\tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{u}}(t), t) \tag{3.28}$$

The constraints in equations 3.3, 3.4, and 3.5 can also be represented by the approximated states and controls as constraints in the nonlinear program used to solve this problem.

It is important to note the novelty of the proposed method when compared to the set of methods known as pseudospectral methods. Typically pseudospectral methods use a differentiation matrix in order to enforce the dynamics, rather than an integration matrix like the one shown. It is usually better, with respect to numerical procedures, to integrate rather than to differentiate. Additionally, the enforcement of the dynamics occurs at the state rate level, as shown in Equation 3.23 rather than the state level.

The process of solving a nonlinear programming problem involves the computation of the Jacobian matrix, the gradients of the objective function and the constraints with respect to the optimization variables. Some solvers give the option of numerically solving for the Jacobian, however this process is often much slower than supplying an analytic Jacobian. The partials of each of the states, state derivatives and controls with respect to the optimization parameters are shown in the proceeding equations.

$$\frac{\partial \tilde{x}(t_k)}{\partial \boldsymbol{\alpha}} = (\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) \tag{3.29}$$

$$\frac{\partial \tilde{\dot{x}}(t_k)}{\partial \boldsymbol{\alpha}} = \frac{2}{(t_f - t_0)}\boldsymbol{T}(t_k) \tag{3.30}$$

$$\frac{\partial \tilde{u}(t_k)}{\partial \boldsymbol{\beta}} = \frac{2}{(t_f - t_0)}\boldsymbol{T}(t_k) \tag{3.31}$$

These partial derivatives are necessary in the computation of the Jacobian. In conjunction with these, the partial derivatives of the dynamics and other constraints themselves (which are unique to the problem) will be needed in order to develop the Jacobian.

Many commercial solvers are readily available that can solve these nonlinear programs. The two used most frequently in this thesis are Matlab's fmincon and SNOPT developed by UC San Diego and Stanford University [5].

3.3 Example Problems

In this section, two example problems are solved using the CDAM method. The solutions will be compared to those attained using the Gauss Pseudospectral method, a method often used for trajectory optimization problems [22], in order to compare the methodologies. The first problem will be the Brachistochrone problem. This problem has an analytic optimal solution which allows for an easier measure of how good the solution truly is. The second problem will be the planar Earth to Mars orbital transfer problem with a free final orbit angle.

In both of these cases the problems are solved as direct trajectory optimization problems, largely ignoring the optimality conditions. Additionally, the nonlinear programming solver being used in all cases will be Stanford's SNOPT optimizer. And for both of these problems the analytic Jacobian will be provided to the optimizer for speed and consistency reasons.

3.3.1 Brachistocrone Problem

The brachistocrone problem is one of the earliest problems solved using the calculus of variations . The problem objective is simple; what is the optimal path that a frictionless object can take down a ramp in order to reach a given lateral position using only gravity in a minimum time, the problem can be seen in Figure 3.1.

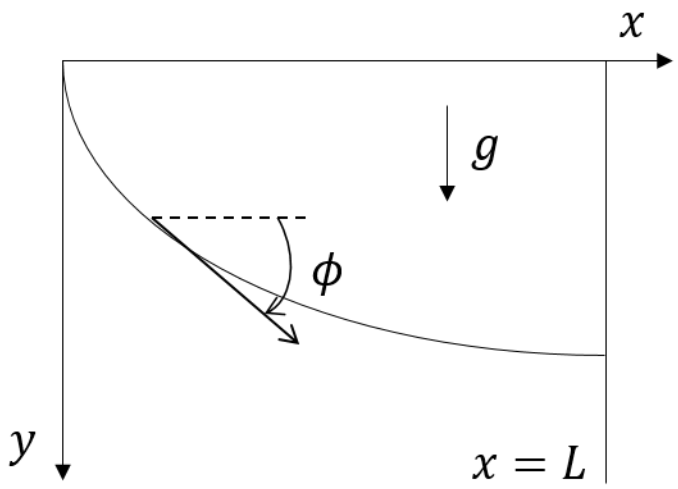

Figure 3.1: The Brachistochrone Problem

The equations of motion for the brachistochrone problem are [22]

$$\dot{x} = \sqrt{2gy}\cos\phi \tag{3.32}$$

$$\dot{y} = \sqrt{2gy}\sin\phi \tag{3.33}$$

Where g is the acceleration due to gravity. With initial conditions being

$$y(0) = 0,\ x(0) = 0 \tag{3.34}$$

and the terminal constraint

$$x(t_f) = L \tag{3.35}$$

In accordance with the previous section, the dynamics are approximated with a set of Chebyshev polynomials as

$$\tilde{y}(t_k) = \frac{2}{t_f} \boldsymbol{z}_1^T \boldsymbol{T}(t_k) \tag{3.36}$$

$$\tilde{x}(t_k) = \frac{2}{t_f} \boldsymbol{z}_2^T \boldsymbol{T}(t_k) \tag{3.37}$$

The $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ vectors are the coefficients to be optimized in the nonlinear program, in addition to the final time which is minimized. Just like the state dynamics, the control variable, ϕ is also represented in the same way.

$$\tilde{\phi}(t_k) = \boldsymbol{z}_3^T \boldsymbol{T}(t_k) \tag{3.38}$$

For the NLP in this problem the optimization parameters, $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ need to have an initial guess. In this implementation the guess for $\boldsymbol{z}_1$ is $[1\ 0\ ...\ 0]$, the guess for $\boldsymbol{z}_2$ is $[1\ 0\ ...\ 0]$ and the guess for $\boldsymbol{z}_3$ is a vector of zeros. The first two guesses can be interpreted as lines for the state variables.

The states themselves can be derived by directly integrating the dynamics equations.

$$\tilde{y}(t_k) = \boldsymbol{z}_1^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + y(0) \tag{3.39}$$

$$\tilde{x}(t_k) = \boldsymbol{z}_2^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + x(0) \tag{3.40}$$

The dynamics constraints for this problem are then simply written from these equations. Ad-

ditionally, there is a bound constraint at the final time.

$$\tilde{\dot{y}}(t_k) = \sqrt{2g\tilde{y}(t_k)}\cos\tilde{\phi}(t_k) \tag{3.41}$$

$$\tilde{\dot{x}}(t_k) = \sqrt{2g\tilde{y}(t_k)}\sin\tilde{\phi}(t_k) \tag{3.42}$$

$$\tilde{x}(t_f) = \boldsymbol{z}_2^T((\boldsymbol{T}(1) - \boldsymbol{T}(-1))\mathcal{J}) + x(0) = L \tag{3.43}$$

The number of constraints is dependent on the number of nodes used in the discretization process. If there are N nodes then each dynamics equation has N constraints associated with it, therefore for this problem there exists $2N + 1$ constraints. Additionally, there exists an analytical solution to this problem.

$$\phi(t) = \frac{\pi}{2} - wt \tag{3.44}$$

$$w = \sqrt{\frac{\pi g}{4L}} \tag{3.45}$$

$$x(t) = \frac{2L}{\pi}\left(wt - \frac{\sin(2wt)}{2}\right) \tag{3.46}$$

$$y(t) = \frac{2L}{\pi}(\sin(wt))^2 \tag{3.47}$$

$$t_f = \sqrt{\frac{\pi L}{g}} \tag{3.48}$$

The NLP solution found using CDAM, where $g = 1, L = 1$ is compared to this analytical solution for a variety of different nodes, between 10 and 50, in Figure 3.2.

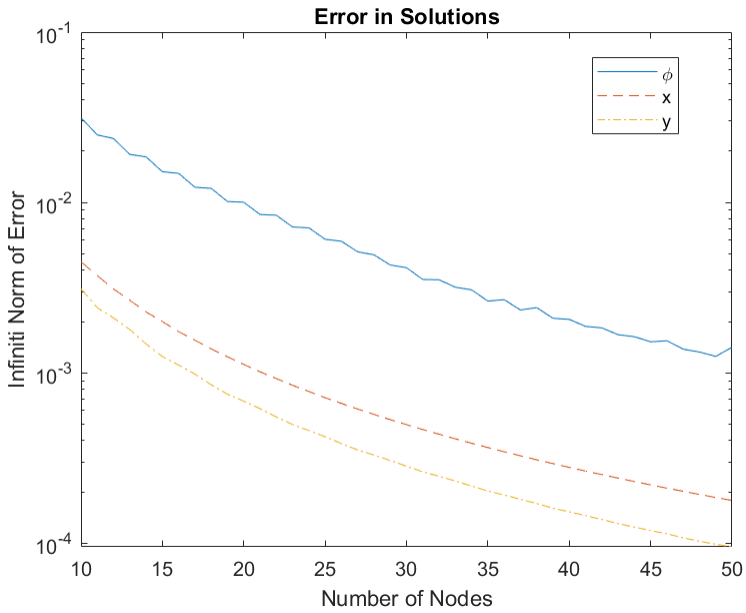

Figure 3.2: Error in Solutions: CDAM Method

Clearly the convergence properties of the states are better than those for the control. With overall convergence being relatively slow. Specifically, as the number of nodes increases the convergence begins to slow down. However, this problem is unique in the fact that the costate, λ_y, is infinite at initial time, a fact that separates this problem from other problems and can lead to convergence issues.

In order to assess the validity of this method it is to be compared to a solution derived from the Gauss Pseudospectral Method. All parameters are the same as were before. The Gauss Pseudospectral Method will not be covered here, but the reader may look at the Ph.D. dissertation of Benson [22] for more information. In Figure 3.3 the errors associated with the states and controls with respect to the analytic solution are shown.

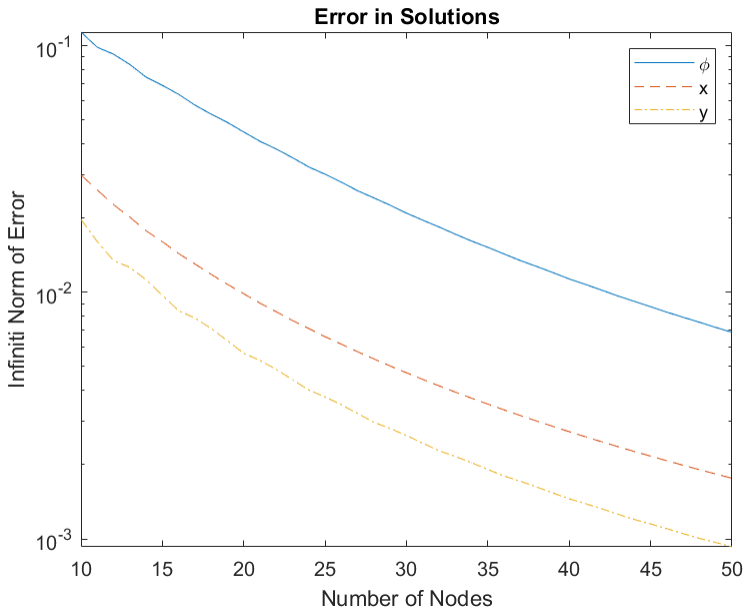

Figure 3.3: Error in Solutions: GPM Method

It is clear that for this particular problem CDAM has a better performance than the Gauss pseudospectral method. This will not always be the case; additionally, an NLP using GPM will typically be faster to solve as the jacobian matrix that results from GPM is sparser than the one in CDAM, leading to fewer operations, especially if the NLP solver is well optimized for speed.

Different problems will result in results of varying accuracies and convergence speeds. In the next subsection the planar Earth to Mars transfer problem will be analyzed using both CDAM and GPM.

3.3.2 Planar Earth to Mars Transfer Problem

The planar Earth to Mars transfer problem has many variations, the one solved in this section is the following: find the trajectory that matches Mars' orbital elements at a specific rendezvous time and minimizes the fuel used. The problem can be seen in Figure 3.4. The dynamics for this planar orbital problem are as follows.

$$\dot{r} = u \tag{3.49}$$

$$\dot{u} = \frac{v^2}{r} - \frac{\mu}{r^2} + u_r \tag{3.50}$$

$$\dot{v} = -\frac{uv}{r} + u_\theta \tag{3.51}$$

$$\dot{\theta} = \frac{v}{r} \tag{3.52}$$

Where r is the radius, u is the radial velocity, v is the tangential velocity, θ is the angle traversed, and u_θ and u_r are the thrust magnitudes in the tangential and radial directions, respectively. The initial conditions, based on an initial circular orbit, are

$$r(0) = 1,\ u(0) = 0,\ v(0) = 1,\ \theta(0) = 0 \tag{3.53}$$

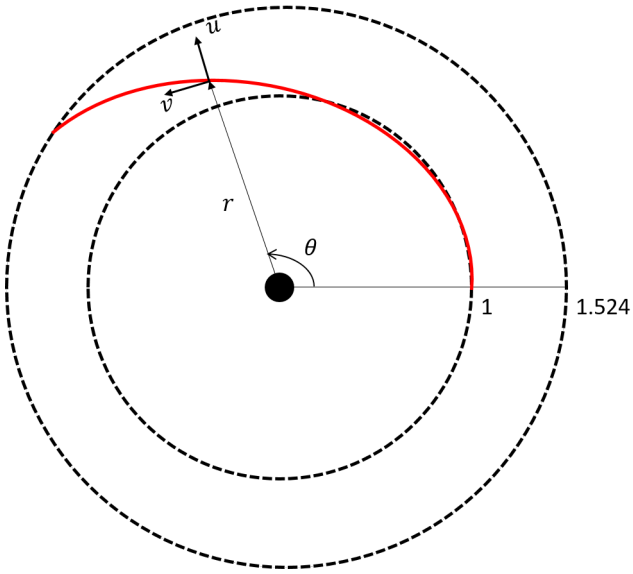

Figure 3.4: Earth to Mars Transfer Problem

The final conditions for this problem also correspond to a circular orbit, at Mars' radius.

$$r(t_f) = 1.524,\ u(t_f) = 0,\ v(t_f) = \sqrt{\frac{1}{1.524}} \tag{3.54}$$

$\theta(t_f)$ will be a free variable while t_f will be fixed to be 180 days. Since the problem is completely non-dimensional the final time is changed as follows.

$$t_f = \frac{180\ days}{58.13\ days} = 3.0965 \tag{3.55}$$

Unlike the Brachistochrone problem, the planar Earth to Mars transfer problem as described here has no analytical solution. In order to assess the accuracy of both the GPM and CDAM solutions, each will be compared to a solution obtained from an indirect shooting method. The costate dynamics, derived from the first order optimality conditions in equation 3.8 are

$$\dot{\lambda}_r = -\lambda_u(-\frac{v^2}{r^2} + \frac{2}{r^3}) \tag{3.56}$$

$$\dot{\lambda}_u = -\lambda_r + \frac{\lambda_v v}{r} \tag{3.57}$$

$$\dot{\lambda}_v = -\frac{2\lambda_u v}{r} + \frac{\lambda_v u}{r} - \frac{\lambda_\theta}{r} \tag{3.58}$$

$$\dot{\lambda}_\theta = 0; \tag{3.59}$$

$$u_r = -\lambda_u,\ u_\theta = -\lambda_v \tag{3.60}$$

Using CDAM, the state dynamics, controls and states are approximated in the same way as the Brachistochrone problem using the Chebyshev polynomials

$$\tilde{\dot{r}}(t_k) = \frac{2}{t_f}\boldsymbol{z}_1^T\boldsymbol{T}(t_k) \tag{3.61}$$

$$\tilde{\dot{u}}(t_k) = \frac{2}{t_f}\boldsymbol{z}_2^T\boldsymbol{T}(t_k) \tag{3.62}$$

$$\tilde{\dot{v}}(t_k) = \frac{2}{t_f}\boldsymbol{z}_3^T\boldsymbol{T}(t_k) \tag{3.63}$$

$$\tilde{\dot{\theta}}(t_k) = \frac{2}{t_f}\boldsymbol{z}_4^T\boldsymbol{T}(t_k) \tag{3.64}$$

$$\tilde{r}(t_k) = \boldsymbol{z}_1^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + r(0) \tag{3.65}$$

$$\tilde{u}(t_k) = \boldsymbol{z}_2^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + u(0) \tag{3.66}$$

$$\tilde{v}(t_k) = \boldsymbol{z}_3^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + v(0) \tag{3.67}$$

$$\tilde{\theta}(t_k) = \boldsymbol{z}_4^T((\boldsymbol{T}(t_k) - \boldsymbol{T}(-1))\mathcal{J}) + \theta(0) \tag{3.68}$$

$$\tilde{u_r}(t_k) = \boldsymbol{z}_5^T\boldsymbol{T}(t_k) \tag{3.69}$$

$$\tilde{u_\theta}(t_k) = \boldsymbol{z}_6^T\boldsymbol{T}(t_k) \tag{3.70}$$

The dynamics constraints can then be formed as in equation 3.23. The terminal conditions in equation 3.54 will also be incorporated into the NLP as constraints. Finally the minimization function can be formed using Fejer's quadrature rule [24].

$$J = \boldsymbol{w}^T(\tilde{\boldsymbol{u}}_r^2 + \tilde{\boldsymbol{u}}_\theta^2) \tag{3.71}$$

Figure 3.5 shows the maximum difference between the solution obtained from the indirect method (with optimality conditions satisfied to 10^{-11}) and the CDAM method.

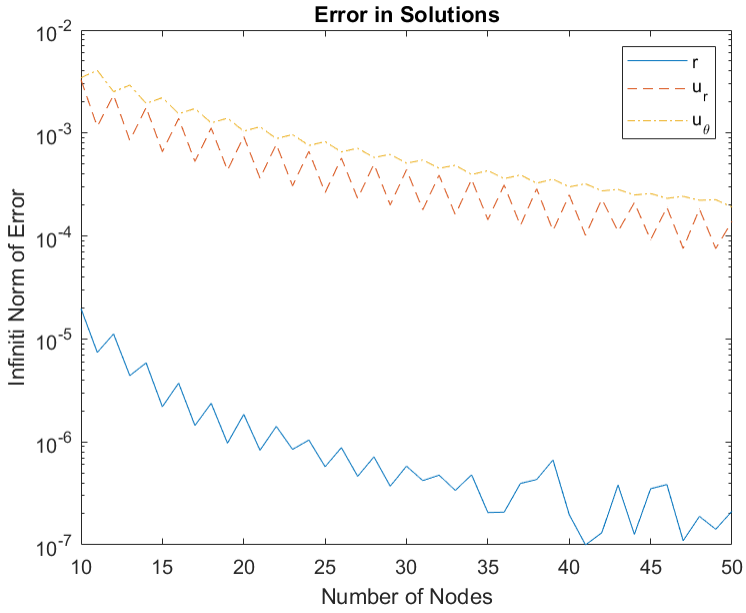

Figure 3.5: Errors in Solution: CDAM, Earth to Mars Transfer

Once again the CDAM solution increases in accuracy as the number of nodes, interestingly enough the solutions using an odd number of nodes are often better than using an even number. Also, as in the brachistochrone problem the accuracy of the optimal state is better than the accuracy of the optimal control profile. In addition to the errors in the CDAM solution the plot of errors in the GPM solution is shown in Figure 3.6.

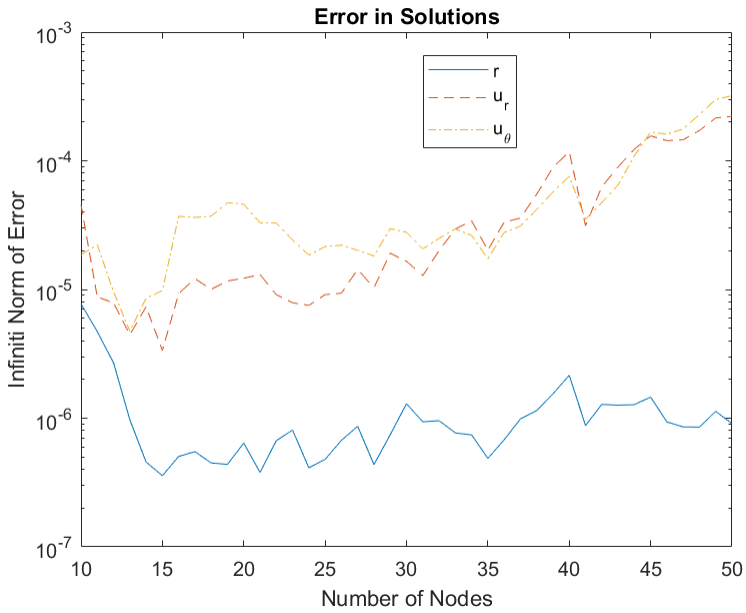

Figure 3.6: Errors in Solution: GPM, Earth to Mars Transfer

The convergence using the Gauss Pseudospectral method is quite troubling. The solutions for GPM for a small number of nodes is much more accurate than the solution found using CDAM. The accuracy found using the small number of nodes would lead one to believe that the implementation is correct. It is still not understood what is making this convergence so much worse than the convergence from the brachistochrone problem. One interesting thing to note, however, is that the solution does eventually begin to converge. After running the Gauss Pseudospectral method with 150 nodes, the error in radius comes down to 10^{-8} and the errors in controls come down to 10^{-5}.

3.4 Summary

A novel, Chebyshev polynomial based, direct trajectory optimization method has been outlined in the section. Its merit has been shown by solving two classic optimal control problems, with a comparison being made to the Gauss Pseudospectral Method. The method is shown to work well for the problems given, both in terms of accuracy and specifically in terms of convergence, where it performs more consistently for the two problems with respect to GPM. With its merits being proven, CDAM will be the method used to solve the multi-rendezvous problem outlined in the

next chapter in order to find the optimal path between debris objects.

4. LONG TERM ACTIVE DEBRIS REMOVAL MISSION DESIGN

Using the techniques developed in Chapters 2 and 3 an optimal design approach for a long term ADR mission can now be defined. For the purposes of this thesis, the two line element sets for a list of 640 debris objects located in Low Earth Orbit, LEO, are used; this list was developed through a collision analysis done in the following paper [3]. A heat map of these high risk objects is shown in Figure 4.1. First the clustering, DTSP and departure time optimization are performed. Then the Chebyshev Dynamics Approximation Method is developed for the multi-rendezvous problem, andis used to optimize each of the missions even further, producing a trajectory and control profile for each of the missions.

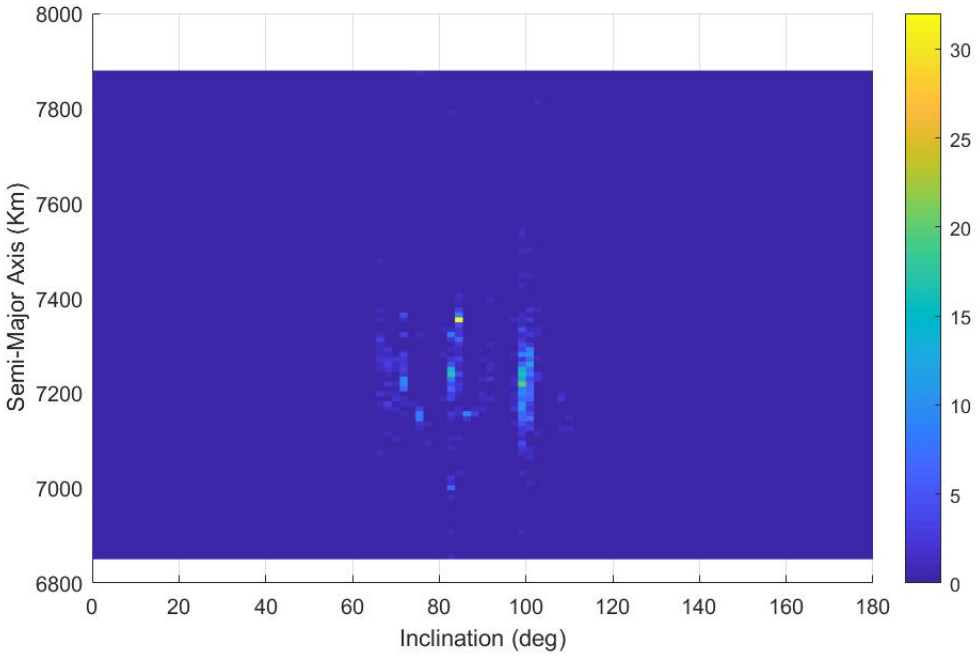

Figure 4.1: Heat Map of Debris Objects

4.1 Clustering and Path Optimization

The first task in this process is the clustering and optimization demonstrated in Chapter 2. The missions to be designed have the following characteristics: Each will be a maximum of 240 days, with each being launched immediately following the previous; Also each mission recovers 5 debris objects creating a total of 128 missions. An important fact here is that the capture and deorbit process has not been accounted for in these missions. After clustering the DTSP is solved for each mission. Figure 4.2 shows the resulting ΔV for each mission from two separate solutions using the MKM algorithm. Two solutions were created as there is some randomness when initializing the algorithm. This causes different local optima to be reached depending on the initialization.

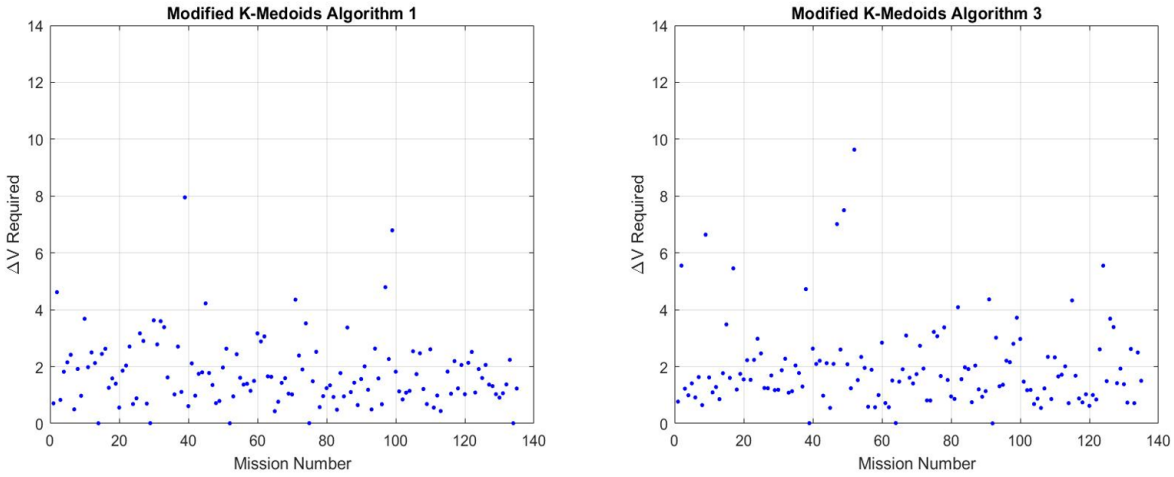

Figure 4.2: ΔV per Mission, Two Examples (Km/s)

The average ΔV values for the two sets are 1.965 Km/s and 1.991 Km/s, respectively. While there are some missions within each set that are unreasonable, the vast majority are, with 122 and 119 missions that require less than 4 Km/s, respectively. The ΔV values can be decreased, however, with the use of the departure time optimization.

Figure 4.3 shows the differences in ΔV for each mission from one of the clustering solutions in Figure 4.2 after the departure time optimization algorithm was used.

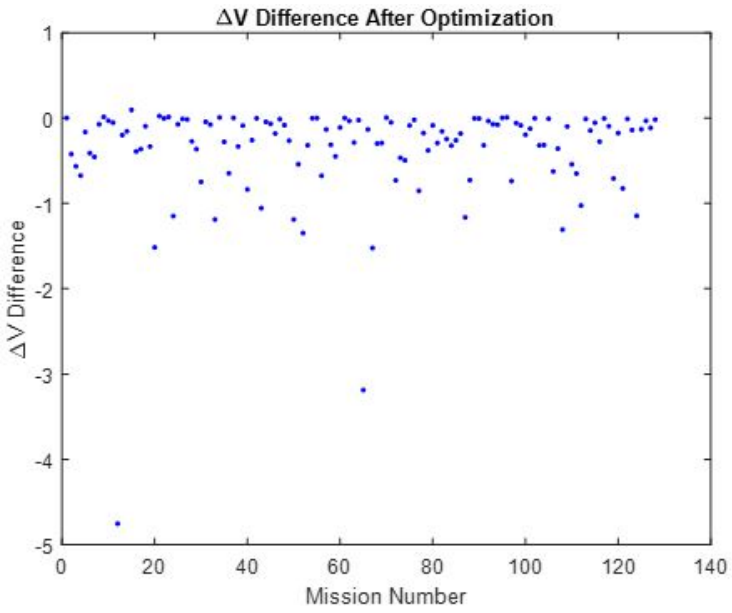

Figure 4.3: ΔV Difference After Optimization (Km/s)

Before the departure time optimization the average ΔV per mission was 1.991 Km/s, post optimization the ΔV per mission is now 1.52 Km/s, that indicates nearly a 25 percent decrease, which is significant. The procedure outlined thus far represents a reasonable solution to the impulsive multiple mission multi-rendezvous problem. However, this solution can also be used as an initial guess for a low thrust multi-rendezvous trajectory optimization algorithm which will be described in the next section.

4.2 Multi-Rendezvous Problem Using CDAM

In this section the optimal control problem describing the multi-rendezvous problem will be outlined with all constraints and minimization functions. Next, the optimal control problem will be transcribed into a nonlinear program using the CDAM approach outlined in Chapter 3. Using the initial guesses from the previous section this nonlinear program will be solved and all results will be shown for the missions.

4.2.1 State Definitions

The first step in defining this optimal control problem is to define the state dynamics. The states that were used for the clustering and DTSP problems suffered from a fatal flaw, they are undefined for an eccentricity of 0 and they are undefined for an inclination angle of zero. For these reasons it is useful to find a set of elements which are nonsingular. To do this, combinations of classical elements are found whose variational equations do not depend on a non circular or non-zero inclination orbit. The states that result from this search, and the ones used for this problem are known as the modified equinoctial elements. The element definitions are shown in Equations 4.1-4.6, where $a, i, e, \omega, \Omega, \nu$ are the classical Kepler elements. [25]

$$p = a(1 - e^2) \tag{4.1}$$

$$f = e\cos(\omega + \Omega) \tag{4.2}$$

$$g = e\sin(\omega + \Omega) \tag{4.3}$$

$$h = \tan(\frac{i}{2})\cos\Omega \tag{4.4}$$

$$k = \tan(\frac{i}{2})\sin\Omega \tag{4.5}$$

$$L = \nu + \omega + \Omega \tag{4.6}$$

Equations 4.7-4.12 show the dynamics of the modified equinoctial elements.

$$\dot{p} = 2\frac{p}{w}\sqrt{\frac{p}{\mu}}a_t \tag{4.7}$$

$$\dot{f} = \sqrt{\frac{p}{\mu}}(a_r \sin L + \frac{((w+1)\cos L + f)a_t}{w} - \frac{(h\sin L - k\cos L)ga_n}{w}) \tag{4.8}$$

$$\dot{g} = \sqrt{\frac{p}{\mu}}(-a_r \cos L + \frac{((w+1)\sin L + g)a_t}{w} + \frac{(h\sin L - k\cos L)fa_n}{w}) \tag{4.9}$$

$$\dot{h} = \sqrt{\frac{p}{\mu}}\frac{s^2 a_n \cos L}{2w} \tag{4.10}$$

$$\dot{k} = \sqrt{\frac{p}{\mu}}\frac{s^2 a_n \sin L}{2w} \tag{4.11}$$

$$\dot{L} = \sqrt{\mu p}(\frac{w}{p})^2 + \frac{1}{w}\sqrt{\frac{p}{\mu}}(h\sin L - k\cos L)a_n \tag{4.12}$$

w, s, r and the accelerations a_t, a_r, a_n are defined as follows.

$$w = 1 + f\cos L + g\sin L \tag{4.13}$$

$$s = \sqrt{1 + h^2 + k^2} \tag{4.14}$$

$$r = \frac{p}{w} \tag{4.15}$$

$$a_t = u_t - 12\mu J_2 \frac{Re^2}{r^4}\frac{(h\sin L - k\cos L)(h\cos L + k\sin L)}{s^4} \tag{4.16}$$

$$a_r = u_r - 3\mu J_2 \frac{Re^2}{2r^4}(1 - 12\frac{(h\sin L - k\cos L)^2}{s^4}) \tag{4.17}$$

$$a_n = u_n - 6\mu J_2 \frac{Re^2}{r^4}\frac{(1 - h^2 - k^2)(h\sin L - k\cos L)}{s^4} \tag{4.18}$$

In these acceleration equations the u portion refers to the acceleration due to spacecraft thruster whereas the rest is due to J_2 effects.

The dynamics shown here are completely nonsingular for any combination of inclination and eccentricity.

4.2.2 Cost Function

The cost function for the optimal control problem can now be defined in Equation 4.19.

$$J = \int_{t_0}^{t_f} \boldsymbol{u}^T \boldsymbol{u} \tag{4.19}$$

$$\boldsymbol{u} = [u_t \; u_r \; u_n]$$

4.2.3 Constraint Definitions

As this is a multi-rendezvous problem it is best described as a multiple stage optimal control problem. This multiple stage problem can best be described in Figure 4.4. For a problem involving 5 objects there are four distinct stages. Each stage is simply a transfer from one object to another, each transfer, i, begins at time t_i^- and ends at t_i^+. The time between t_0 and t_1^- is simply a coasting period as t_0 may not be the optimal time for departure. The same is true for the time between t_4^+ and t_f. Additionally, the time between t_i^+ and t_{i+1}^- is for phasing, deorbiting operations and coasting until the next optimal departure time is reached.

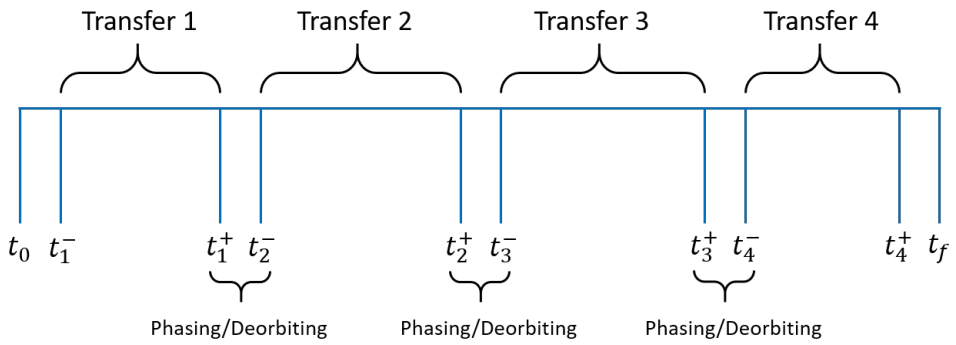

Figure 4.4: Multiple Stage Optimal Control Problem Layout

With the multiple stages of this optimal control problem being defined, a slew of constraints can now be defined. Equations 4.20 and 4.21 refer to the path constraints that will apply to all stages of the optimal control problem.

$$r(t) > Re \tag{4.20}$$

$$\sqrt{u_n^2 + u_r^2 + u_t^2} \leq u_{max} \tag{4.21}$$

This parameterization of the controls can prove problematic, however, as the Jacobian for this constraint and also the Jacobian for the performance index (which will both be evaluated at every step of the optimization) will be equal to Equation 4.22

$$\frac{\partial J}{\partial u_n} = \frac{u_n}{\sqrt{u_n^2 + u_r^2 + u_t^2}} \tag{4.22}$$

If at any point the value of the thrust becomes zero, this Jacobian will become undefined leading to poor performance of the optimization or even a failed optimization. For this reason the controls will be parameterized instead as a magnitude and a unit vector as in Equation 4.23 with the constraints in Equations 4.24 and 4.25.

$$u_r = u_m u_1;\ u_t = u_m u_2;\ u_n = u_m u_3 \tag{4.23}$$

$$u_m \leq u_{max} \tag{4.24}$$

$$u_1^2 + u_2^2 + u_3^2 = 1 \tag{4.25}$$

Bound constraints and interior constraints also apply to the various times associated with the problem. Equations 4.26 show these constraints.

$$\boldsymbol{x}(t_i^+) = \boldsymbol{x}_j(t_i^+) \tag{4.26}$$

In this equation the $\boldsymbol{x}(t_i^+)$ represents the first 5 states (not including the fast variable, which will not be matched at each transfer) at time t_i^+. $\boldsymbol{x}_j(t_i^+)$ represents the first 5 states of debris object j at time t_i^+. By enforcing these constraints at the end of each transfer window for each object the rendezvous' can be achieved.

Though all of the constraints and the minimization function have been defined there are still a few important decisions to be made in how this problem will be represented using the CDAM method. The simplest way seems to be the following: Each stage of the optimal control problem will be discretized, in accordance with Chapter 3, separately, essentially creating j optimal control problems for j transfers. This is done in order to increase the number of nodes in the areas where they are necessary, which is when the controls are active. In coast phases the controls are inactive, meaning that the states and controls are discontinuous throughout the entire mission, making the problem with only a single discretization throughout the entire mission difficult to solve. All of the "separate" optimal control problems are, however, solved simultaneously, along with the departure and arrival times which will also be optimized according to the constraints shown in Equations 4.27-4.30, where j is the number of transfers that occur.

$$t_0 \leq t_1^- \tag{4.27}$$

$$t_i^- \leq t_i^+ \tag{4.28}$$

$$t_i^+ \leq t_{i+1}^- - \epsilon \tag{4.29}$$

$$t_j^+ \leq t_f \tag{4.30}$$

Here ϵ is parameter representing the approximate amount of time required for phasing and servicing, this value should be a conservative value allowing ample time.

4.2.4 Design Choices

There is one more design choice that must be made that is crucial to many optimal control problems solved by a trajectory optimization scheme. This would be choosing coefficients in order to non-dimensionalize the problem. Scaling the problem in the proper way is important for the optimization process. Specifically, it is crucial since the values of some states have larger magnitudes than others. For example the state, p, in the modified equinoctial elements has a much larger magnitude than the other states, having a value in the thousands of Kilometers range, with all of the other states being on the order of 1. This means that when Jacobians are calculated for variations the constraints involving p, the values will be thousands of times larger than other Jacobians, leading the optimization to favor these constraints and enforce them earlier and to a higher degree, greatly degrading the optimization process. For these reasons the following two values will be used in order to non-dimensionalize distance as well as time [26].

$$Distance = R_e = 6378.137, \quad Time = 2\pi\sqrt{\frac{R_e^3}{\mu}}$$

After normalizing using these values μ becomes $4\pi^2$ and the radius of the Earth normalizes to 1.

4.2.5 Dynamics Constraint Expressions

With all of the boundary constraints, path constraints and design choices for the multi-rendezvous problem already being flushed out the only step left is to define the dynamics constraint expressions for the optimal control problem as they were defined in Chapter 3.

Assuming that N nodes are used for each of the j transfers there will a total of $10j(N+1)$ values to optimize. This number comes from the fact that there are 10 distinct states being approximated, those would be the 6 orbital element dynamics and the 4 control parameters, there are $N+1$ parameters to be optimized for each of these states as N nodes involved $N+1$ coefficients of Chebyshev Polynomials. All of the approximations can be seen in Equations 4.31-4.40.

$$\tilde{p}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{pi}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.31}$$

$$\tilde{\dot{f}}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{fi}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.32}$$

$$\tilde{\dot{g}}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{gi}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.33}$$

$$\tilde{\dot{h}}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{hi}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.34}$$

$$\tilde{\dot{k}}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{ki}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.35}$$

$$\tilde{\dot{L}}_i(t) = \frac{2}{t_f}\boldsymbol{z}_{Li}^T\boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.36}$$

$$\tilde{u_{mi}}(t) = \boldsymbol{z}_{umi}^T \boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.37}$$

$$\tilde{u_{1i}}(t) = \boldsymbol{z}_{u1i}^T \boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.38}$$

$$\tilde{u_{2i}}(t) = \boldsymbol{z}_{u2i}^T \boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.39}$$

$$\tilde{u_{3i}}(t) = \boldsymbol{z}_{u3i}^T \boldsymbol{T}(t);\ \forall\ i = 1 -> j \tag{4.40}$$

With these approximations being defined, the integration identity for Chebyshev polynomials shown in Equation 3.18 can now be used for Equations 4.31-4.36 in order to find the state approximations. With all of the states and controls defined properly for the CDAM method, the dynamics constraints themselves can now be detailed. These constraints are the same as the ones in Equation 3.23 and are shown once again in Equation 4.41 and are applied to all 6 states at each of the discretization nodes.

$$\tilde{\dot{\boldsymbol{x}}}_k = \boldsymbol{f}(\tilde{\boldsymbol{x}}(t_k), \tilde{\boldsymbol{u}}(t_k), t_k) \tag{4.41}$$

Lastly is the question of forming the cost function in a way that can be coded into the nonlinear program. This is done using Fejer's quadrature rule [24]. This cost is shown in Equation 4.42 as a sum of the costs associated with each transfer.

$$\sum_{i=1}^{j} \frac{t_i^+ - t_i^-}{2} \boldsymbol{w}^T \tilde{\boldsymbol{u_{mi}}}^2 \tag{4.42}$$

4.2.6 Results

With all of the constraint expressions defined for the multi-rendezvous problem, it can now be solved. Similar to Section 2, the missions that will be solved using this method are the via a colli-

sion analysis [3]. The missions will be those resulting from the clustering and dynamic traveling salesman problem analysis. Additionally, the times found using the departure time optimization will be used as the initial guesses for the departure times t_i^- in the trajectory optimization approach. The initial guesses for the arrival times t_i^+, however, will be estimated as 6 hours after the departure times; this will prove problematic later. To reiterate once again the missions will be 240 days long each, with a minimum 24 hour service/deorbit time ϵ between transfers. The maximum acceleration, chosen somewhat arbitrarily, used in the optimization is $0.05\frac{m}{s^2}$. This is equivalent to a 500 kg satellite with a 25 Newton thruster. The NLP solver for is SNOPT, a Fortran based solver with a Matlab interface [5].

As mentioned, the initial guess for the arrival times are simply guesses based on the departure times and based on experience of using the trajectory optimization method with various values of thrust. The optimization proves to be quite sensitive to this initial guess (and all initial guesses). Because of this fact some of the did not successfully converge during the optimization process. However, the ones that did converge led to a decrease in overall ΔV. Table 4.1 shows one of the successfully optimized missions.

Transfer Number	ΔV	Departure Time	Arrival Time
1	112.2 m/s	42.517 days	42.643 days
2	96.2 m/s	44.315 days	44.430 days
3	185.8 m/s	70.346 days	70.465 days
4	385.3 m/s	108.431 days	108.568 days

Table 4.1: Trajectory Optimization Results

Overall, only about 50 percent of the missions defined actually converged using the trajectory optimization method. This is with poor initial guesses on arrival times and poor initial guesses on the controls. The initial guesses for the controls were simply set to zero, however this can be improved. By knowing the location of the ascending and descending node of one orbit with respect

to the next, the control guess can be formed into a function that peaks at each of these nodes, as these are the most efficient times to change planes and the times that the successfully converged missions had the largest thrust magnitudes. It is anticipated that with better initial guesses for these values, the number of successfully converged missions would increase. For the missions that did converge, the average ΔV decreased slightly, which is nice to see (this will not always be true as the initial solution was an impulsive thrust assumption). The average ΔV previously was 1.3213 Km/s, while the ΔV from the trajectory optimization method was 1.2904 Km/s. Figures 4.5-4.10 shows a plot of the controls and states for one of the transfers of one of the mission designs.

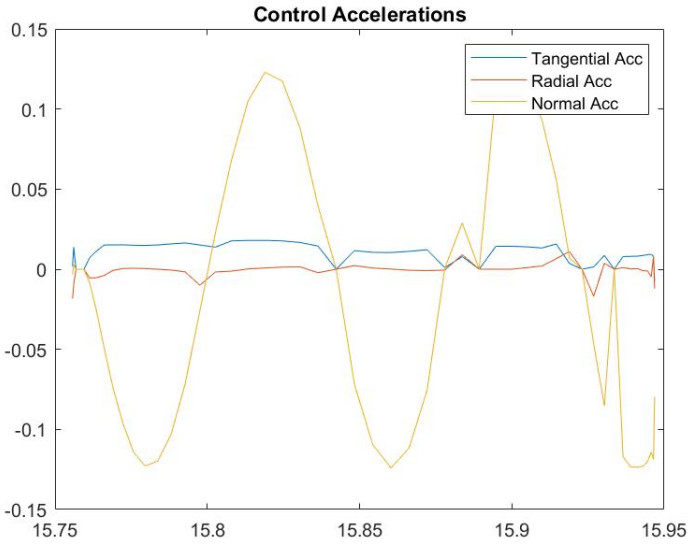

Figure 4.5: Optimization of the Controls

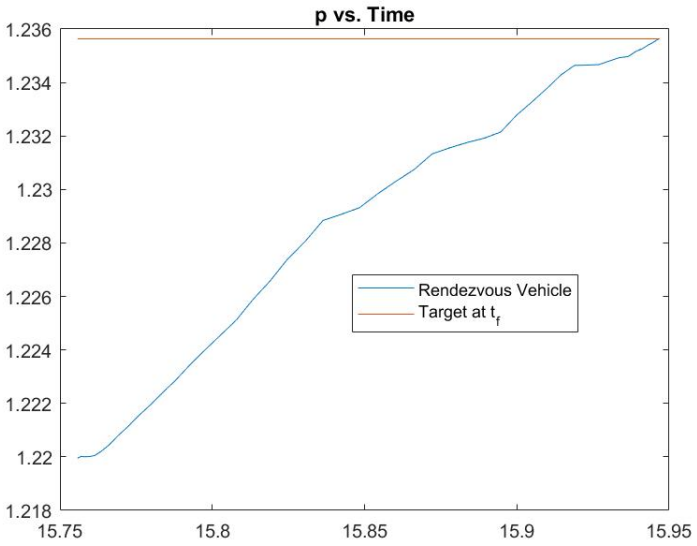

Figure 4.6: Optimization of the p Element

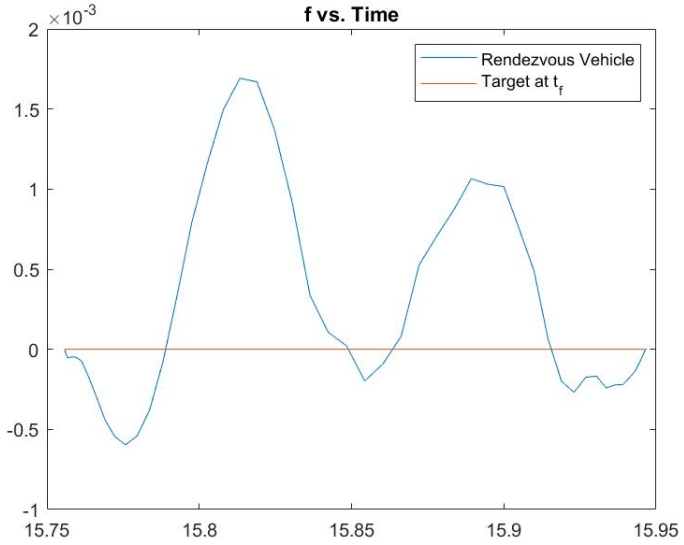

Figure 4.7: Optimization of the f Element

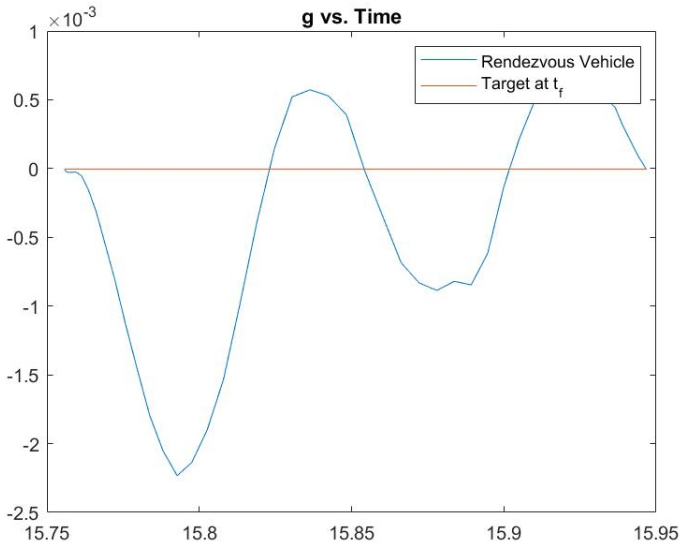

Figure 4.8: Optimization of the g Element

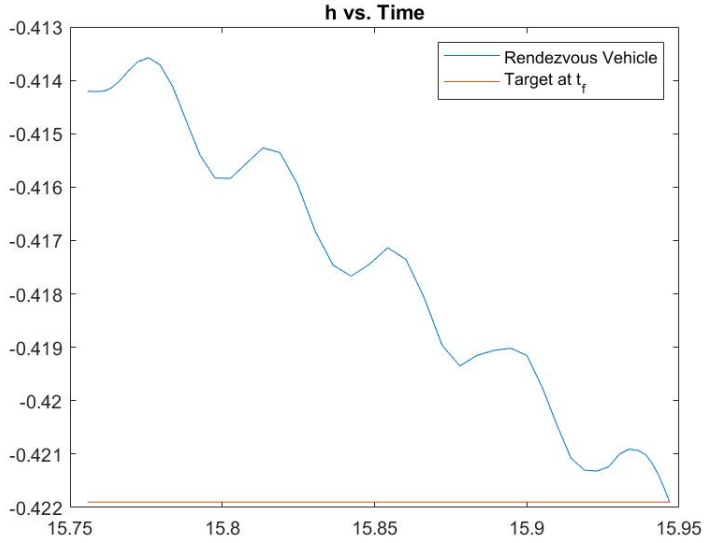

Figure 4.9: Optimization of the h Element

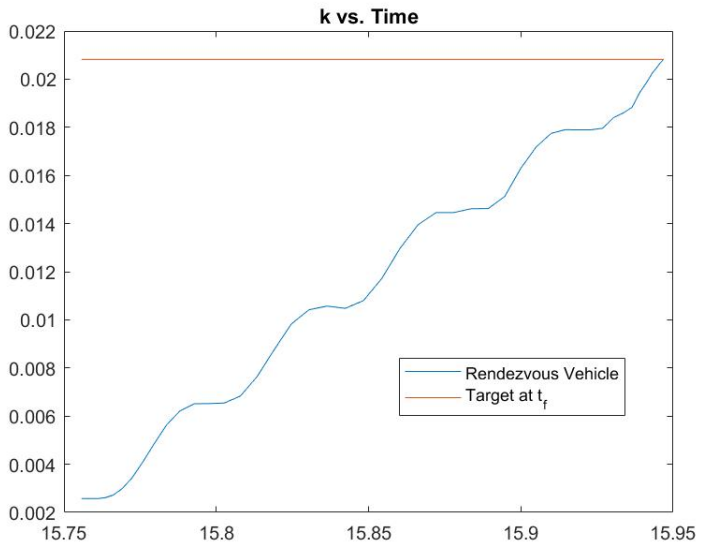

Figure 4.10: Optimization of the k Element

One interesting thing to note is when the vehicle decides to thrust. The thrust spikes that are occurring in the normal direction coincide with the ascending and descending nodes of the rendezvous satellite with respect to the target satellite.

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

A set of techniques that, when put together, can be used to develop a long-term active debris removal missions, have been developed. If given a large list of debris objects, they can be clustered using the MKM clustering algorithm, intracluster paths can be found by solving a dynamic traveling salesman problem and the overall trajectory can be optimized via a direct trajectory optimization method. And this is what has been done; using all of these techniques a long term mission was developed from a list of 640 debris objects in Low Earth Orbit. Additionally, through this work, a novel direct trajectory optimization technique has been born in the form of CDAM, which seemingly works well for a variety of different problems.

There are still a few unanswered questions, though. Firstly, can a superior GAP approximation algorithm be developed to assist with the clustering process. Secondly, and possibly more importantly, can the number of missions in which CDAM successfully converges to a locally optimal solution be increased. One promising path forward with respect to this problem will be to investigate better methods for determining an initial guess for the NLP, specifically for the controls.

5.2 Future Work

There is quite a bit of work and further research that still needs to be done regarding all of the techniques shown in this thesis. By no means is what is presented in this thesis the end all be all with respect to active debris removal.

The Chebyshev trajectory optimization method in particular is something that can and should be developed further. Like many other pseudospectral methods similar to it, there is most likely a way to estimate costates during the nonlinear programming process. The Gauss Pseudospectral Method, the method directly compared to the Chebyshev method here, has the ability to do this with reasonable success for certain problems. The ability to estimate costates would lead to the ability to use these solutions as initial guesses for a more accurate indirect method. In addition to

costate estimation, it would certainly be beneficial to compare the method to more than just the GPM, methods like the Legendre Pseudospectral Method and others. Not only should this method be compared to more methods, it should be compared using a variety of different problems. At the moment the only problems that have been compared are either trivial or extremely difficult; the trivial problems being the brachistochrone and the Earth Mars transfer, whereas the difficult one is the multi-rendezvous problem.

Additionally, an important thing that should be considered in future work is uncertainties, particularly with respect to orbital dynamics. The clustering algorithms, DTSP and the trajectory optimization method all made use of idealized versions of the dynamics. Each made the assumption that J_2 was the only form of perturbations, when in reality there are many more perturbations; in fact, the dynamics of an object in space is not completely known and there are always uncertainties involved. For this reason it would be beneficial to analyze how these uncertainties affected the results of each of the optimization techniques.

REFERENCES

[1] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research*, vol. 83, pp. 2637–2646, Jun 1978.

[2] N. R. Council *et al.*, *Continuing Kepler's Quest: Assessing Air Force Space Command's Astrodynamics Standards*. National Academies Press, 2012.

[3] B. W. Barbee, S. Alfano, E. Pinon, K. Gold, and D. Gaylor, "Design of spacecraft missions to remove multiple orbital debris objects," *2011 Aerospace Conference*, 2011.

[4] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[5] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[6] J. Missel and D. Mortari, "Path optimization for space sweeper with sling-sat: A method of active space debris removal," *Advances in Space Research*, vol. 52, pp. 1339–1348, Jul 2013.

[7] V. Braun, A. Lupken, S. Flegel, J. Gelhaus, M. Mockel, C. Kebschull, C. Wiedemann, and P. Vorsmann, "Active debris removal of multiple priority targets," *Advances in Space Research*, vol. 51, no. 9, pp. 1638–1648, 2013.

[8] K. Alfriend, D.-J. Lee, and G. Creamer, "Optimal servicing of geosynchronous satellites," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, vol. 29, pp. 203–206, May 2002.

[9] M. Cerf, "Multiple space debris collecting mission: Optimal mission planning," *Journal of Optimization Theory and Applications*, vol. 167, pp. 195–218, Jan 2015.

[10] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[11] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics, revised edition*. American Institute of Aeronautics and Astronautics, 1999.

[12] D. Brouwer and G. M. Clemence, *Methods of celestial mechanics*. Elsevier, 2013.

[13] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1-3, pp. 461–474, 1993.

[14] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management Science*, vol. 32, no. 9, pp. 1095–1103, 1986.

[15] M. Silvano and T. Paolo, "Knapsack problems: algorithms and computer implementations," 1990.

[16] "Matlab optimization toolbox," 2018. The MathWorks, Natick, MA, USA.

[17] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. Routledge, 2018.

[18] I. M. Gelfand, R. A. Silverman, *et al.*, *Calculus of variations*. Courier Corporation, 2000.

[19] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[20] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Society for Industrial and Applied Mathematics, 2010.

[21] C. Clenshaw, "The numerical solution of linear differential equations in chebyshev series," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 53, pp. 134–149, Cambridge University Press, 1957.

[22] D. Benson, *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[23] M. Hernández, "Chebyshev's approximation algorithms and applications," *Computers & Mathematics with Applications*, vol. 41, no. 3-4, pp. 433–445, 2001.

[24] J. Waldvogel, "Fast construction of the fejér and clenshaw–curtis quadrature rules," *BIT Numerical Mathematics*, vol. 46, no. 1, pp. 195–202, 2006.

[25] M. Walker, B. Ireland, and J. Owens, "A set modified equinoctial orbit elements," *Celestial mechanics*, vol. 36, no. 4, pp. 409–419, 1985.

[26] S. A. Stanton, "Optimal orbital transfer using a legendre pseudospectral method," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF AERONAUTICS AND ASTRONAUTICS, 2003.