# An Algorithm for Generating Feasible Low Thrust Interplanetary Trajectories

**IEPC-2009-219**

*Presented at the 31$^{st}$ International Electric Propulsion Conference,*
*University of Michigan, Ann Arbor, Michigan, USA*
*September 20–24, 2009*

Prashant R. Patel*
*Institute for Defense Analysis, Alexandria, VA, 22311, USA*

Alec D. Gallimore† and Thomas H. Zurbuchen‡
*University of Michigan, Ann Arbor, MI, 48109, USA*

*and*

Daniel J. Scheeres§
*University of Colorado, Boulder, CO, 80309, USA*

**Electric propulsion has offered the promise of more efficient transportation throughout the solar system. Capitalizing on EP's promise requires a new set of tools that can rapidly conduct trade studies to help scientists and mission designers understand the advantages and costs of various EP systems. This paper presents a proof of concept program and the underlying algorithm that allow for the computation of feasible EP trajectories. The tools described in this paper are designed to be simple to use and require minimal user input. The goal of these tools is to demonstrate that the feasible EP trajectories can be computed quickly and easily. When coupled with optimization algorithms, these methods allow for the optimization of trajectories without requiring the user to specify control inputs.**

*Research Staff Member, Cost Analyses and Research Division, ppatel@ida.org
†Arthur F. Thurnau Professor, Aerospace Engineering, alec.gallimore@umich.edu
‡Professor, Aerospace and Atmospheric, Oceanic and Space Science, thomasz@umich.edu
§Professor, A. Richard Seebass Chair, Department of Aerospace Engineering Sciences, scheeres@colorado.edu

## Nomenclature

$\boldsymbol{A}$ = acceleration vector/profile
$C_3$ = launch vehicle/orbital energy
$\boldsymbol{C}(\boldsymbol{U})$ = constraint vector that defines a feasible trajectory
$c_w$ = Chebyshev coefficient for coordinate $w$
$\boldsymbol{\delta U}$ = the control update found at iteration of the feasible trajectory solver
$F(C_3)$ = launch vehicle payload mass as a function of $C_3$
$J$ = cost function used to optimize the trajectory
$M$ = number of control segments
$m$ = number of heliocentric revolutions
$m_i$ = mass of spacecraft at node $i$
$\dot{m}$ = mass flow rate
$\mu_{sun}$ = the gravitational parameter of the sun
$P_x$ = orbital period of planetary body x
$P_{max}$ = maximum power allowed by power system and dynamics
$\phi$ = elevation angle as defined in spherical coordinates
$\phi_i$ = the throttle setting of the thruster on segment $i$. $\phi$ must be between $\pm$
$q_i$ = thrust constraint on segment $i$
$r$ = heliocentric distance of spacecraft in spherical coordinates
$S_i$ = thruster state constraint on segment $i$
$s_i$ = thruster state, on or off, on segment $i$
$\boldsymbol{T}_j$ = thrust vector on segment $j$
$T_j$ = Chebyshev polynomial or order $j$
$T_T(\phi_i)$ = thrust provided by thruster at throttle setting $phi_i$, on segment $i$
$\theta$ = the polar angle as defined in spherical coordinates
$t_0$ = launch date of spacecraft
$t_f$ = arrival date of spacecraft
$\boldsymbol{U}$ = the control set that defines the feasible trajectory problem
$z_i$ = the power system constraint on the segment $i$

## I. Introduction

Electric propulsion (EP) offers the potential for significant mass savings over chemical propulsion when the proper thruster, trajectory, and power system are selected. Selecting the appropriate technology and finding the optimal trajectory is a difficult process due to the large number of possible combinations and the different disciplines involved in the process. In order to make EP more attractive to scientists and mission designers, new tools are required that are easy to use and enable trade studies over a wide range of conditions. Ideally, a user should be able to describe the mission of interest and the software should identify the best possible mission times and the key systems that should be used. While the ideal situation does not currently exist, it is possible to work toward that goal and create tools that enable informed decisions about EP mission and systems.

The use of impulsive transfers (chemical propulsion) allows for the decoupling of the propulsion system and trajectory. The decoupling enables Lambert Solvers[1,2] to search over a wide range of launch and arrival dates and identify the cost of each trajectory. Typically, this information can be displayed in a "pork chop plot" as shown in Fig. (1). The advantage of this type of presentation is that it allows for the selection of the launch and arrival dates and informs the user of the cost, all without knowing anything about the propulsion system. Furthermore, because a trajectoryis computed for each data point, the user can have confidence in the fact that the costs are reasonable and accurate.

Obtaining that type of data quickly is advantagous because it informs decisions early in the design process and allows designers to focus on the more detailed issues. Unlike chemical propulsion, EP couples

the propulsion, power system, and trajectory. Solving for an EP based trajectory is much more difficult due to the large number of system possibilities and non linear dynamics. For example, the shape of an impulsive trajectory is a conic section, Fig. (2), while the shape of an EP trajectory greatly varies over time, as shown in Fig. (3). Furthermore, EP trajectories do not have the equivalent of a Lambert Solver which makes it more difficult to find trajectories. For instance, in the case of an Interstellar Probe Study[3] it took over 20 attempts before a single viable trajectory was found.

Significant effort has been focused on optimizing EP trajectories.[4–10] However, most direct optimizers require an initial guess, which can be difficult to generate due to the large number of choices, from thrust directions, to thruster selection, and power source size. In some cases optimizers may require trajectories that are feasible in order to ensure adequate performance.[10] In order to avoid this problem and reduce computational times it would be beneficial to autonomously generate feasible trajectories. Directly using an optimizer to generate an EP pork chop plot would result in the process shown in Fig. (4), which could be tedious and error prone because a user would be required to supply an initial guess for each point in the search space.

One attempt at simplifying the process of generating EP trajectories was the use of the shape based base approach.[11–15] In particular, Anastassios, successfully used the exponential sinusoid,[11,14–17] $r(\theta) = k_1 e^{k_2 \sin(k_3\theta + k_4)}$, to model gravity assist trajectories. He created a program called STOUR-LTGA[11,14,15,17] which generates candidate EP trajectories over a wide range of launch and arrival dates. The solutions from the STOUR-LTGA program could then be used as initial guesses for GALLOP,[16,18–20] a trajectory optimization program. The coupling of STOUR-LTGA to GALLOP changed the trajectory generation process from Fig. (4) to Fig. (5). The coupling of the two programs demonstrated that it is practical to use auto-generated trajectories as an initial guess for optimizers.

While successful, the shape based approach is a holonomic constraint; therefore, it is best suited for problems where the path is known and the forces and timing along the path are unknown. However, in orbital rendezvous and intercept problems the timing is usually known, and the forces and path are unknown. This disconnect in the problem type complicates the implementation and utility of the shape based approach. Also, the trajectories generated from STOUR-LTGA did not necessarily satisfy the boundary conditions, and they were not feasible because they did not take into account launch vehicle, thruster, or power constraints.

In this paper we expand on previous work and develop a process and methodology that allows for the generation of feasible interplanetary trajectories. We do not optimize the trajectories here because there are a plethora of trajectory optimization programs.[4–10,16,18–21] Instead, we focus on generating feasible trajectories that can be used as initial guesses for optimization. This creates a pipeline that allows for the autonomous generation of trajectories. We first modify the trajectory generation process by introducing an additional step, as shown in Fig. (6). The major difference in our process is that we add an additional step that converts the "simple problem" into a feasible solution before the optimization step. This ensures that in the worst case the problem and solution at every stage is well defined. Furthermore, we replace the shape based approach, instead modeling the trajectory in time using a polynomial and then searching for the polynomial coefficients that minimize the cost function. Finally, we introduce a method for converting the polynomial trajectory into a feasible trajectory by incorporating thrust, power, and launch vehicle constraints. Combining our broad search method with the feasible trajectory solver creates a pipeline that can autonomously generate feasible EP trajectories.

## II. Phase 1: Automated Broad Search Algorithm

In the first phase, the program searches over a wide range of launch times, $t_0$ and flight times, $\Delta t$, to find good candidate mission times. The user specifies a range of launch dates, range of flight times, and the departure and destination body. For each flight time the computer computes the range of possible heliocentric revolutions, $m$, using Eq. (1). With the search space fully defined by the launch dates, flight times, and heliocentric revolutions the program can begin a search for good trajectories.

$$\left\lfloor \frac{\Delta t}{max\left(P_{departure}, P_{destination}\right)} \right\rfloor \leq m \leq \left\lfloor 0.7 * \frac{\Delta t}{min\left(P_{departure}, P_{destination}\right)} \right\rfloor + 1 \tag{1}$$

The trajectory is parameterized by a set of coefficients and Chebyshev polynomials. The coefficients are the free parameters that are varied to generate different trajectories. The Chebyshev polynomials are the underlying functions that represent the trajectories. Each position coordinate is modeled separately; in

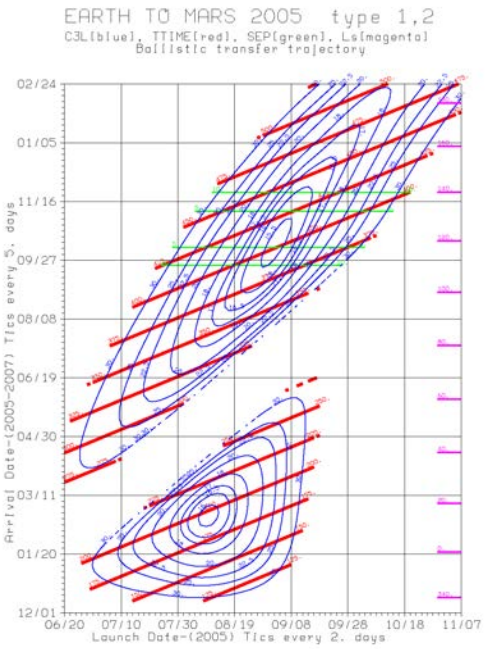



**Figure 1. A pork chop plot showing the $C_3$ cost of Earth to Mars trajectories over a wide range of launch and arrival dates. This chart allows for the identification of launch and arrival dates that will satisfy a particular flight time and the associated $C_3$ cost.**

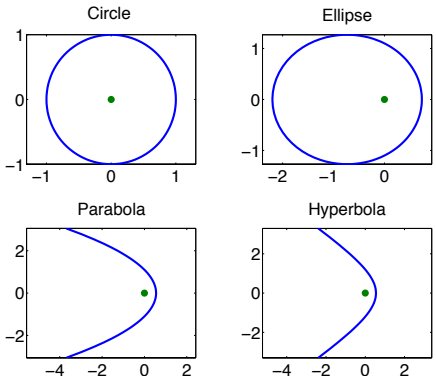



**Figure 2. Chemical trajectories can be modeled as an impulsive burn followed by a coast arc, which is a conic section.**

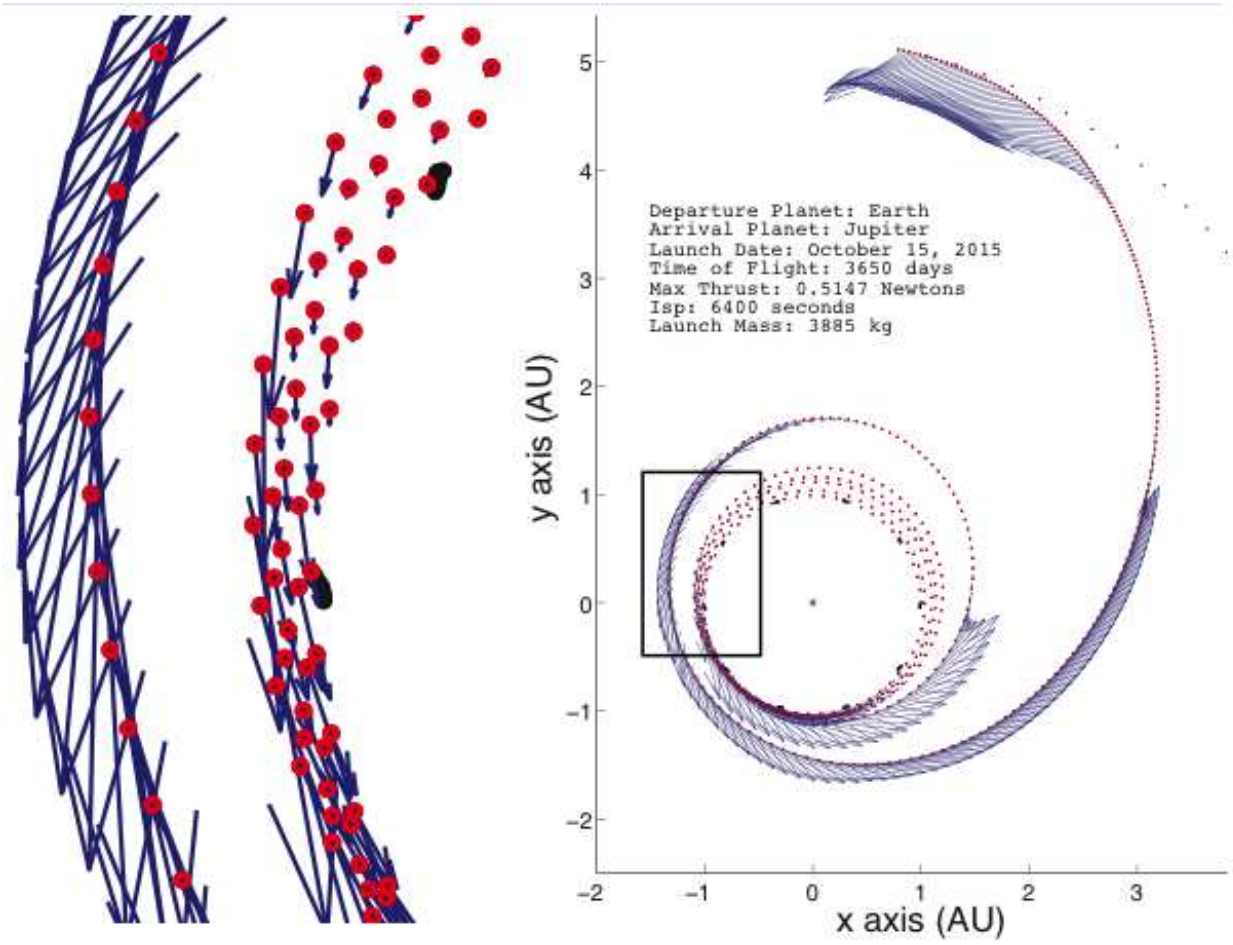



**Figure 3. The right hand side shows an Earth to Jupiter trajectory over 10 years. The trajectory makes several heliospheric revolutions without much change in the trajectory then it quickly spirals out to Jupiter. The left hand portion is a zoom in of the trajectory showing the numerous changes in the thrust direction and magnitude.**

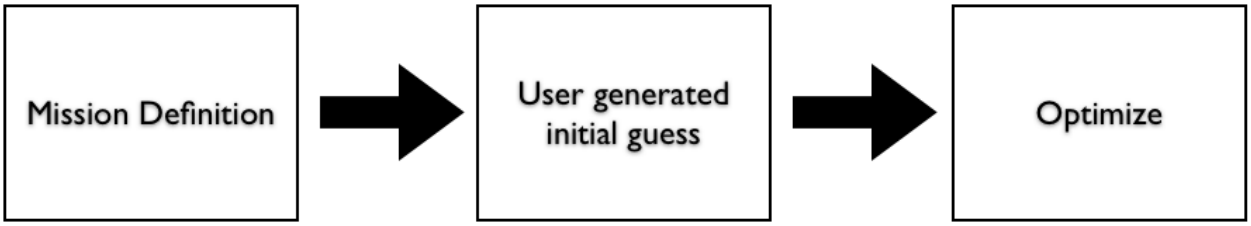



**Figure 4. The process for directly using an trajectory optimizer to conduct trade studies. The process relies on a user defined initial guess which is a road block due the potential difficulty in generating a good initial guess.**

cartesian coordinates the position coordinates would be $x, y, z$, while in spherical coordinates the position coordinates are $r, \theta, \phi$. Chebyshev polynomials were chosen because they are flexible and have been used in to model trajectories.[22] To demonstrate the flexibility and realism of the Chebyshev polynomials in representing EP trajectories, we show two different Earth to Jupiter missions. The first mission, Fig. (7(a)), is a short timescale ( 5 years) mission with a low number of heliocentric revolutions. The second mission, Fig. (7(b)), is a longer timescale ( 10 years) mission with a larger number of heliocentric revolutions. Both figures show that the polynomials have the flexibility to model EP trajectories over a wide range of timescales and heliocentric revolutions. Because the Chebyshev polynomials can handle different timescales and large changes in the state variables, they are a good basis function for modeling EP trajectories.

For a generic position coordinate, $w$, the time evolution of the coordinate is

$$w(t) = \sum_{j=0}^{N-1} c_{w,j} T_j(t) \tag{2}$$

and the time rate of change of the coordinate is

$$\dot{w}(t) = \sum_{j=0}^{N-1} c_{w,j} \dot{T}_j(t) \tag{3}$$

where $c_{w,j}$ is a coefficient that parameterizes the coordinate $w$ and $T_j$ is the $j^{th}$ order Chebyshev polynomial.[23] The order of the polynomial, $N$, is also the number of degrees of freedom for that coordinate. In this formulation, only the positions are parameterized. The model here differs from the shape based approach in two ways. First, the Chebyshev polynomials can be directly differentiated to compute the velocity and accelerations on the trajectory. Secondly, the timing is known a priori in this model, where as the timing for the shape based approach is undetermined until the time history of the control is specified.

Chebyshev polynomials can be computed recursively with Eq. (4).

$$T_j(\tau) = \begin{cases} 1, & \text{if } j = 0 \\ \tau, & \text{if } j = 1 \\ 2\tau T_{j-1}(\tau) - T_{j-2}(\tau), & \text{if } j \geq 2 \end{cases} \tag{4}$$

and

$$\dot{T}_j(\tau) = \begin{cases} 0, & \text{if } j = 0 \\ \dot{\tau}, & \text{if } j = 1 \\ 2\dot{\tau} T_{j-1}(\tau) + 2\tau \dot{T}_{j-1}(\tau) - \dot{T}_{j-2}(\tau), & \text{if } j \geq 2 \end{cases} \tag{5}$$

where $\tau$ is computed by

$$\tau = 2\frac{t - t_0}{t_f - t_0} - 1 \tag{6}$$

$$\dot{\tau} = \frac{2}{t_f - t_0} \tag{7}$$

and $t$ is bounded by

$$t_0 \leq t \leq t_f \tag{8}$$

Here $t_0$ is the time when the spacecraft begins it's interplanetary journey, and $t_f = t_0 + \Delta t$ is the time when the spacecraft reaches its destination. If there are enough degrees of freedom in the trajectory model, the parameterization can satisfy the boundary (departure and arrival) constraints. For a rendezvous problem, the parameterization requires a minimum of four degrees of freedom per coordinate. Two degrees of freedom are needed to ensure that the spacecraft leaves the departure body and arrives at the destination body. Another two degrees of freedom are needed so the spacecraft's departure and arrival velocity matches the departure and destination body.

For the rendezvous problem, the trajectory leaves the departure body at a particular time, $t_0$ and rendezvous with the destination body at a specified time, $t_f$. The position and velocity of the departure and destination body can be obtained from an ephemeris. With the states of the departure and arrival body

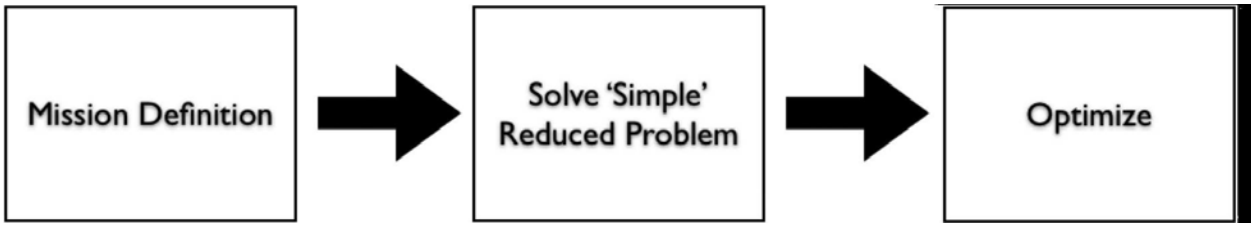



**Figure 5. The process for directly using an trajectory optimizer to conduct trade studies. The process relies on a user defined initial guess which is a road block due the potential difficulty in generating a good initial guess.**

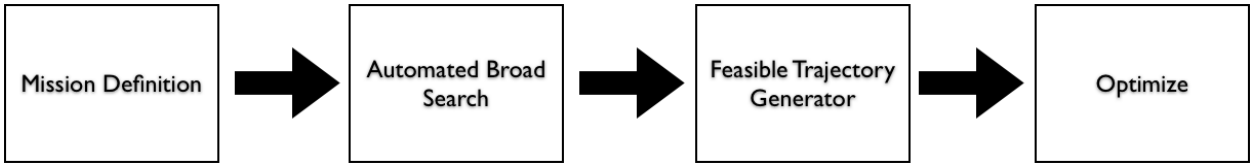
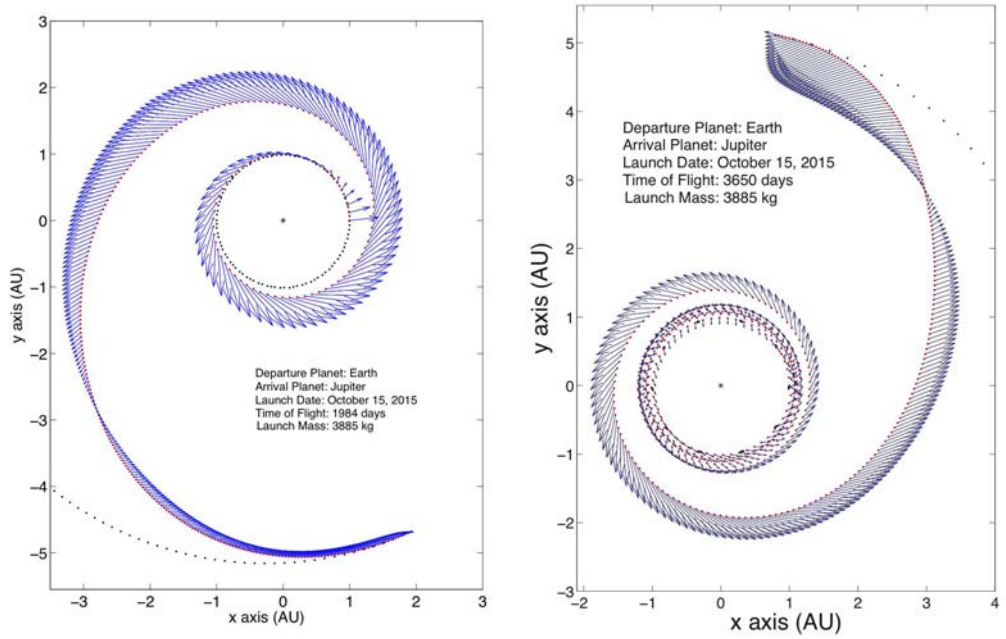



**Figure 6. The process for directly using an trajectory optimizer to conduct trade studies. The process relies on a user defined initial guess which is a road block due the potential difficulty in generating a good initial guess.**




(a) A 5.4 year Earth to Jupiter trajectory represented by a $10^{th}$ order polynomial. This trajectory makes slightly more than one full revolution before rendezvousing with Jupiter, indicating that the polynomials can model a large change in the states over a short time scale. The arrows indicate direction and magnitude of the thrust vector.

(b) A 10 year Earth to Jupiter trajectory represented by a $10^{th}$ order polynomial. This trajectory makes multiple heliocentric revolutions near 1 AU before rendezvousing at Jupiter. This indicates that the polynomials can model a large change in the states over a wide time scale. The arrows indicate direction and magnitude of the thrust vector.

**Figure 7. A Chebyshev representation of two different Earth to Jupiter trajectories.**

known, a constraint problem can be set up that restricts the coefficients such that the trajectory will satisfy boundary conditions.

The states of the departure body at $t_0$ in polar coordinates are $r_0, \theta_0, \phi_0, \dot{r}_0, \dot{\theta}_0, \dot{\phi}_0$, and the states of the arrival body are $t_f$ in polar coordinates are $r_f, \theta_f, \phi_f, \dot{r}_f, \dot{\theta}_f, \dot{\phi}_f$. The constraint for the departure body is given by Eq. (10), and the constraint for the arrival body is given in Eq. (11).

$$B(\tau) = \begin{bmatrix} T_0(\tau) & \dots & T_{N-1}(\tau) \\ \dot{T}_0(\tau) & \dots & \dot{T}_{N-1}(\tau) \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} r_0 \\ \dot{r}_0 \\ \theta_0 \\ \dot{\theta}_0 \\ \phi_0 \\ \dot{\phi}_0 \end{bmatrix} = \begin{bmatrix} B(-1) & 0 & 0 \\ 0 & B(-1) & 0 \\ 0 & 0 & B(-1) \end{bmatrix} \begin{bmatrix} c_{r,0} \\ \vdots \\ c_{r,N-1} \\ c_{\theta,0} \\ \vdots \\ c_{\theta,N-1} \\ c_{\phi,0} \\ \vdots \\ c_{\phi,N-1} \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} r_f \\ \dot{r}_f \\ \theta_f \\ \dot{\theta}_f \\ \phi_f \\ \dot{\phi}_f \end{bmatrix} = \begin{bmatrix} B(1) & 0 & 0 \\ 0 & B(1) & 0 \\ 0 & 0 & B(1) \end{bmatrix} \begin{bmatrix} c_{r,0} \\ \vdots \\ c_{r,N-1} \\ c_{\theta,0} \\ \vdots \\ c_{\theta,N-1} \\ c_{\phi,0} \\ \vdots \\ c_{\phi,N-1} \end{bmatrix} \tag{11}$$

Any solution that satisfies the linear constraints given by Eq. (10) and Eq. (11) will meet the boundary constraints. In order for a solution to Eq. (10) and Eq. (11) to exist, the order of the polynmials,$N$, must be greater than or equal to 4. Because Eq. (10) and Eq. (11) are linear, they can be solved with a variety of numerical methods. This formulation easily satisfies linear state boundary conditions, which ensures that the appropriate boundary conditions are met. With a method for parameterizing the path defined two critical elements still remain, a cost function for identifying good trajectories out of the infinite solutions that satisfy Eq. (10) and Eq. (11) and an algorithm for conducting the search.

When the order of the polynomials are greater then four, there exists an infinite number of trajectories that satisfy the rendezvous problem. In order to select a good trajectory, a cost function is needed that can measure the value of a particular trajectory. The cost function is

$$J = \int_{t_0}^{t_f} |\boldsymbol{A}|^2 \, dt \tag{12}$$

Eq. (12) is used because it is smooth and quadratic, which usually implies that the optimization problem is numerically simple to solve. Although the cost function used here does not minimize the propellant cost, this is not a major concern at this stage because the trajectories are only being approximated.

The dynamics for a point mass in a central gravity field (spacecraft orbiting the sun) are given as

$$\ddot{\boldsymbol{r}} = -\frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} + \boldsymbol{A} \tag{13}$$

where $\boldsymbol{A}$ is the acceleration required to maintain the path, and $\boldsymbol{r}$ is the position of the spacecraft. Solving for the acceleration, $\boldsymbol{A}$, is

$$\boldsymbol{A} = \ddot{\boldsymbol{r}} + \frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} \tag{14}$$

Eq. (14) shows that the acceleration is now a function of the position of the spacecraft, $\boldsymbol{r}$, and its time derivatives. This is useful because the position of the spacecraft is parameterized by Eq. (3), which means that the derivates can be computed simply by differentiating $T$, the basis function. Substituting in the trajectory function, the acceleration is

$$\boldsymbol{A} = \begin{bmatrix} \sum_{i=0}^{N-1} c_{x,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3}\sum_{i=0}^{N-1} c_{x,i}T_i \\ \sum_{i=0}^{N-1} c_{y,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3}\sum_{i=0}^{N-1} c_{y,i}T_i \\ \sum_{i=0}^{N-1} c_{z,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3}\sum_{i=0}^{N-1} c_{z,i}T_i \end{bmatrix} \tag{15}$$

$$|\boldsymbol{r}| = \sqrt{\left(\sum_{i=0}^{N-1} c_{x,i}T_i\right)^2 + \left(\sum_{i=0}^{N-1} c_{y,i}T_i\right)^2 + \left(\sum_{i=0}^{N-1} c_{z,i}T_i\right)^2} \tag{16}$$

The acceleration is now a function of the path coefficients, $c_{x,i}$, $c_{y,i}$, and $c_{z,i}$. Since the cost function is a function of the trajectory coefficients, this implies that the cost function can be minimized. With the cost function and constraints defined, the path coefficients can be found such that the constraints are satisfied and the cost is minimized. Next, an algorithm is needed that can compute trajectories over the search space.

Now we need to implement a method for optimizing the Chebyshev trajectory without requiring a user specified initial guess. The optimization algorithm used is a Sequential Quadratic Programming[8,9,24] algorithm. The Sequential Quadratic Programming method approximates the constraints to the first order and the cost function to the second order. Because the constraints are linear, the Sequential Quadratic Programming method will explicitly satisfy the constraints during every iteration.

The method used here to generate an initial guess that satisfies the boundary conditions, Eq. (10) and Eq. (11), is to initially set the Chebyshev polynomial order to $N = 4$. Because the number of coefficients is equal to the number of free parameters, there exists only one solution to Eq. (10) and Eq. (11), which define the rendezvous problems. Next, N is increased by 1 to 5, and the new higher order coefficients, $c_{r,N-1}$, $c_{\theta,N-1}$, $c_{\phi,N-1}$ are set equal to 0. Now the $N = 5$ trajectory is equivalent to the $N = 4$ trajectory, but the $N = 5$ trajectory has additional degrees of freedom that can be optimized. The $N = 5$ trajectory is then used as an initial guess to the sequential quadratic programming method, which returns the optimal solution. This process can be generalized to generate higher order trajectory approximations. The generalized algorithm is Alg. (1). The major benefit of the approach outlined here is that the user only has to specify the search space and the algorithm will return an optimized trajectory.

Alg. (1) represents a self contained method that can generate an initial guess and minimize the cost function. Alg. (1) is self contained because it begins with a unique solution to the rendezvous problem. This unique trajectory is then used as an initial guess to find subsequent optimal trajectories represented by higher order polynomials. A tool to generate feasible trajectories is coded up in C++ and Objective C on an Apple computer. The program uses the CSPICE libraries for ephemeris and time calculations. Because Alg. (1) does not require a user supplied initial guess, each point in the search space can be solved for independently. We take advantage of this by running the calculations concurrently, which reduces the wall time required to span the search space.

The user interface for the proof of concept broad search program is shown in Fig. (8). The program requires the departure body, arrival body, launch dates, and times of flight. The program then searches over the entire space and stores all potentially valid solutions. The limited input requirement makes broad searches easy to conduct. The program interface, shown in Fig. (8), allows for the use of multiple processors. Because each subproblem generates its own initial guess, the subproblems can be solved independent of the each other, allowing the methods to be used in a parallel or distributed environment, which reduces the computational time. The data from the program can then be used to generate plots like Fig. (9). This type of plot allows the user to easily identify low cost launch dates and flight times, similar to the Fig. (1), that can be utilized in the next phase to generate feasible trajectories.

## III. Phase II: Converting Chebyshev Approximations to Feasible Trajectories

While Chebyshev approximated trajectories are useful for identifying launch dates and flight times, as shown in Fig. (9), they do not incorporate the thruster, power system, and launch vehicle limitations, which strongly influences the trajectory and sensitivity of the optimal solution space. However, because

**Algorithm 1**: High level overview of Chebyshev trajectory generation algorithm

```
Select departure body
Select destination body
Select launch dates
Select flight times
Pd = [Orbital period of departure body, Orbital period of destination body]
switch Problem Type do
    case Rendezvous to Rendezvous problem
        N_min=4
    end
    case Rendezvous to Intercept problem or Intercept to Rendezvous problem
        N_min=3
    end
    case Intercept to Intercept problem
        N_min=2
    end
end
for t_0 ∈ Launch Dates do
    for Δt ∈ flight times do
        m_min = ⌊Δt / max(Pd)⌋
        m_max = ⌊0.7 * Δt / min(Pd)⌋ + 1
        for m = m_min to m_max do
            N = N_min
            Find coefficients that solve rendezvous problem (only one solution)
            for N = N_min + 1 to N_max do
                Set the 0 to the (N − 2)^th coefficients to the previously set of coefficients
                Set the (N − 1)^th coefficients to 0
                Minimize ∫_{t_0}^{t_0+Δt} |A|^2 dt s.t. boundary conditions are satisfied
            end
            Store trajectory solution
        end
    end
end
```

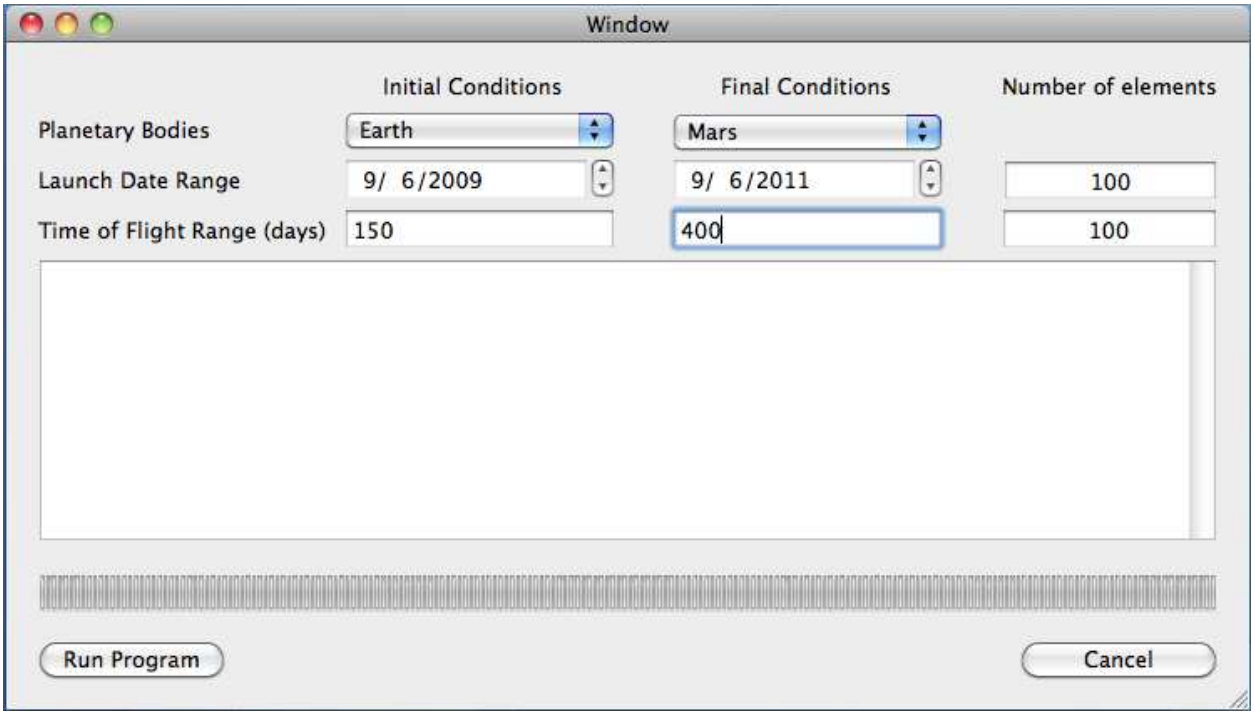



**Figure 8. The user interface for the proof of concept broad search tool. The program has a simple interface and only requires only a few user inputs. It then computes a trajectory for each point in the search space.**

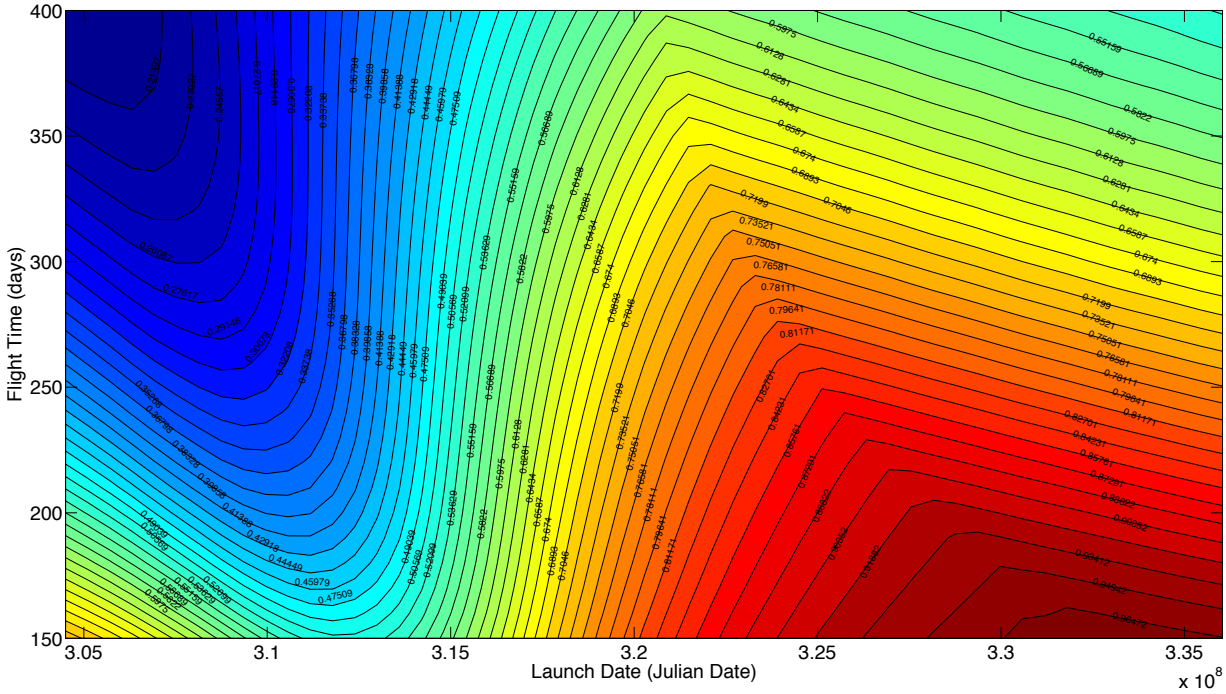



**Figure 9. A contour plot of the propellant cost for and Earth to Mars transfer assuming an $I_{sp} = 3000$ sec. This type of plot can be used to narrow down the search space by identifying low cost launch dates and flight times. Over 2000 trajectories were computed to create this plot with an average wall time of $0.1$ seconds per trajectory using only 2 cores. Because we can employ parallel computing, the wall time is less than the total CPU time.**

the Chebyshev polynomials have been optimized and do replicate the shape of EP trajectories, they can be converted easily into feasible trajectories, which could then be used as an initial guess for optimizers.

The process outlined here for converting the Chebyshev trajectories into feasible trajectories relies on numerically simple methods, so it can be implemented quickly and easily. In order to generate a feasible trajectory, the acceleration profile of the Chebyshev trajectory has to be converted into a thrust profile that can be used as an initial guess for the feasible trajectory solver. In order to convert the acceleration profile into a thrust profile the user has to select the launch vehicle(s), thruster(s), power level(s), and the number of control segments, $M$. From the thruster, the maximum exhaust velocity, $u_e$, is found, and from the launch vehicle, the initial mass is set. Using the exhaust velocity, initial mass, and the acceleration profile a thrust profile can be generated using Alg. (2). The initial thrust profile, $\boldsymbol{T}_i$, will not satisfy the

**Algorithm 2**: Algorithm for converting the acceleration profile of the Chebyshev trajectory into a thrust profile based on thruster and launch vehicle specifications. $u_e$ is the maximum exhaust velocity of the specified thruster and $d_c$ is the duty cycle of the thruster.

$m_0 = F(0)$;
**for** $i = 0$ *to* $M-1$ **do**
  $t_i = \frac{i}{M}(t_f - t_0) + t_0$;
  $\Delta V_i = \int_{t_i}^{t_{i+1}} |\boldsymbol{A}|\, dt$;
  $m_{i+1} = m_i e^{-\Delta V_i/u_e}$;
  $\boldsymbol{T}_i = \frac{(m_i - m_{i+1})u_e}{d_c(t_{i+1}-t_i)} \frac{\boldsymbol{A}(t_i)}{|\boldsymbol{A}(t_i)|}$;
**end**

thruster limitations or the rendezvous problem however the miss distance should be small enough such that the feasible trajectory solver should converge.

With the thrust profile defined, we need to implement a solver that will satisfy the dynamics, thrust constraints, power system constraints, and launch vehicle constraints. The equations of motion are

$$\ddot{\boldsymbol{r}} = -\frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} + s\boldsymbol{T}/m \tag{17}$$

$$\dot{m} = -\dot{m}(\phi)\, s \tag{18}$$

where $\phi$ is the throttle, $s$ is the engine state (on= 1 or off= 0), and $\boldsymbol{T}$ is the thrust. The control constraints are defined as

$$q_i = 0 = 0.5\left[T_T(\phi_i)\right]^2 - 0.5\left|\boldsymbol{T}_i\right|^2 \tag{19}$$

and

$$S_i = 0 = 0.5 s_i (s_i - 1) \tag{20}$$

In Eq. (19) $T_T(\phi_i)$ is the thrust that the propulsions system provides as a function of the throttle. The control inequality constraint is

$$-1 \leq \phi_i \leq 1 \tag{21}$$

The launch vehicle constraint is

$$0 \geq m_0 - F(C_3) \tag{22}$$

and the power system constraint is

$$z_i = 0 \geq P(\phi_i) - P_{max}(\boldsymbol{r}, t) \tag{23}$$

The equality constraints are converted into equality constraints using a slack variable method.[24] The full

constraint vector, $\boldsymbol{C}$, and control vectore, $\boldsymbol{U}$ are

$$\boldsymbol{C} = \begin{bmatrix} q_0 \\ \vdots \\ q_{M-1} \\ S_0 \\ \vdots \\ S_{M-1} \\ -1 \leq \phi_0 \leq 1 \\ \vdots \\ -1 \leq \phi_{M-1} \leq 1 \\ 0 \geq m_0 - F\left(C_3\right) \\ \vdots \\ 0 \geq P\left(\phi_i\right) - P_{max}\left(\boldsymbol{r}, t\right) \\ x_f - x_M \\ \dot{x}_f - \dot{x}_M \end{bmatrix} \tag{24}$$

and

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{T}_0 \\ \vdots \\ \boldsymbol{T}_{M-1} \\ \phi_0 \\ \vdots \\ \phi_{M-1} \\ s_0 \\ s_{M-1} \end{bmatrix} \tag{25}$$

The constraints represent the state boundary conditions, the launch vehicle limitations, the thruster limitations, the power system limitations. The control represent the directional thrust profile, the thruster throttle settings, and engine state.

### A. Linear Subproblem

For a trajectory to be feasible it must satisfy

$$\boldsymbol{C}\left(\boldsymbol{U}\right) = 0 \tag{26}$$

Generally, the controls, $\boldsymbol{U}$, will not satisfy Eq. (26), so we need to define a method that can modify $\boldsymbol{U}$ such that Eq. (26) will be true. The standard approach to a problem of this type is to take a first order Taylor series expansion of the constraint equations, $\boldsymbol{C}$, with respect to the controls, $\boldsymbol{U}$, and set that equation equal to zero. Taking the first order expansion gives

$$0 = \boldsymbol{C}\left(\boldsymbol{U}\right) + \frac{\partial \boldsymbol{C}\left(\boldsymbol{U}\right)}{\partial \boldsymbol{U}} \boldsymbol{\delta U} \tag{27}$$

The control update is

$$\boldsymbol{U}_{update} = \boldsymbol{U} + \boldsymbol{\delta U} \tag{28}$$

If $\boldsymbol{\delta U}$ satisfies Eq. (27), then it linearly satisfies the constraint equations, and if the initial control vector, $\boldsymbol{U}$ is close to a feasible solution, then the linear method will converge to a feasible solution. With the constraints and control vector defined, Alg. (3) is used to satisfy the constraints. The trajectory that is generated will depend on how Eq. (27) is solved.

**Algorithm 3**: Algorithm for satisfying the constraints. $\epsilon$ is a small positive number

**while** $|\boldsymbol{C}|^2 > \epsilon$ **do**
  solve $\boldsymbol{C} + \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{U}} \boldsymbol{\delta U}$;
  update $\boldsymbol{U} = \boldsymbol{U} + \boldsymbol{\delta U}$;
**end**

### B. Pseudoinverse Solution to the Linear Subproblem

The easiest and perhaps simplest method for solving Eq. (27) is to use the pseudoinverse[25] to construct $\boldsymbol{\delta U}$. The psuedoinverse minimizes the $\boldsymbol{\delta U}^T \boldsymbol{\delta U}$ while satisfying the constraint equation, Eq. (27). Defining $\boldsymbol{C_U} = \frac{\partial \boldsymbol{C}(\boldsymbol{U})}{\partial \boldsymbol{U}}$, the pseudoinverse, $\boldsymbol{C_U}^+$, of $\boldsymbol{C_U}$ is

$$\boldsymbol{C_U}^+ = \boldsymbol{C_U}^T \left( \boldsymbol{C_U} \boldsymbol{C_U}^T \right)^{-1} \tag{29}$$

Using the pseudoinverse, the control update is

$$\boldsymbol{\delta U} = -\boldsymbol{C_U}^+ \boldsymbol{C}\left(\boldsymbol{U}\right) \tag{30}$$

The pseudoinverse selects the "smallest" change in control such the constraint equation is satisfied. Because of the way the pseudoinverse generates the control update, it tends to converge quickly.

An Earth to Mars mission is considered to demonstrate the utility of the pseudoinverse method, with a flight time of 500 days. The initial trajectories are provided by the Chebyshev approximation method. For the 500 day example, the launch date is July 23, 2009, and the launch vehicle is the Atlas V 501 (Table (1)). For solar electric power sources the power output scales as

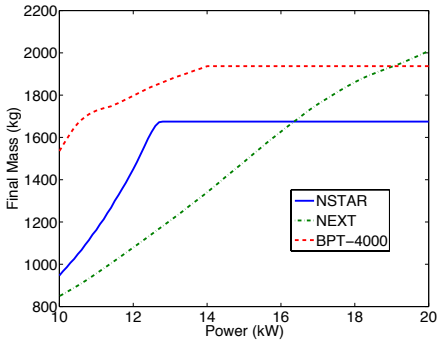
$$\frac{P\left(t\right)}{P\left(t_0\right)} = \left( \frac{r_0}{r\left(t\right)} \right)^2 \tag{31}$$

The results are shown in Fig. (10(a)), Fig. (10(b)), Fig. (10(c)), and Fig. (10(d)). The results in the figures are not optimized, and solar arrays provide power that vary with $|\boldsymbol{r}|^{-2}$. Although the trajectories are not optimized, the figures provide some interesting results. The BPT-4000 thruster performs the best when it is fully powered over the entire trajectory. Because of the NEXT thruster's high power requirements and higher specific impulse, the final mass increases as power increases, but for low powers the BPT-4000 delivers a larger mass to Mars. The final mass delivered to Mars by the NSTAR thruster rapidly decreases as power decreases because the thruster is fully utilized, resulting in a trajectory that is marginally feasible.

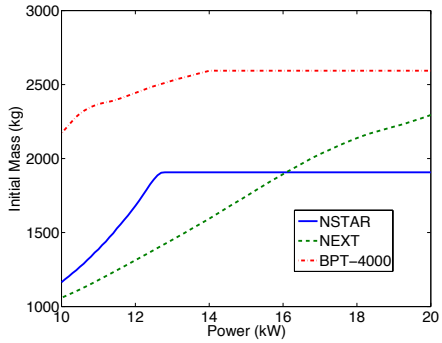
**Table 1. Table of launch vehicle properties. The table shows the launch vehicle name, $C_3 = 0$ launch mass and maximum $C_3$**

| Launch Vehicle | $C_3 = 0$ mass (kg) | Maximum $C_3$ (km/s) |
|---|---|---|
| Atlas V 401 | 3445 | 90 |
| Atlas V 501 | 2680 | 70 |
| Atlas V 511 | 3765 | 90 |
| Atlas V 521 | 4545 | 90 |
| Atlas V 531 | 5210 | 110 |
| Atlas V 541 | 5820 | 120 |
| Atlas V 551 | 6330 | 120 |
| Delta IV 4040-12 | 2735 | 30 |
| Delta IV 4050-19 | 9305 | 60 |
| Delta IV D4450-14 | 4580 | 25 |

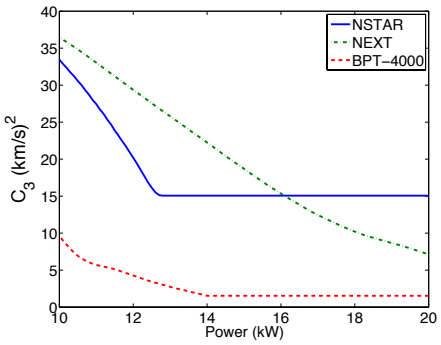
The examples demonstrate the solver can reuse a single Chebyshev trajectory to generate feasible trajectories with different thrusters and power constraints.



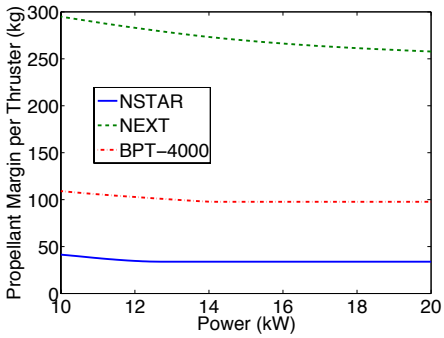

(a) The final mass for a series of feasible trajectories for a 500 Earth to Mars mission. The x axis is the initial power at 1 AU. Solar arrays are used as the power source. Two thrusters are used.




(b) Initial mass for a 500 day Earth to Mars trajectories. The x axis is the initial power at 1 AU. Solar arrays are the power source. The example utilizes 2 thrusters.




(c) The $C_3$ as a function of the initial power for an Earth to Mars 500 day mission. The example utilizes 2 thrusters.




(d) Propellant mass throughput margin remaining per thruster. Negative mass values would indicate that the throughput limit on the thruster has been violated.

**Figure 10. A series of feasible solutions over a range of thrusters and power levels.**

### C. Alternative Solutions to the Linear Subproblem

While the pseudoinverse approach does converge quickly and is simple to use, it is not the only approach to finding feasible trajectories. An alternative approach to the psuedoinverse method demonstrates that the solution to Eq. (27) can be customized to different needs. In the following examples, Eq. (27) is solved such that an alternative cost function is minimized for several iterations. The cost function used is

$$\sum_{i=0}^{M-1} |\boldsymbol{T}_i| + 2\pi \boldsymbol{\delta r}_i^T \boldsymbol{\delta r}_i \tag{32}$$

Here, $\boldsymbol{\delta r}_i$ is the deviation in the position of the spacecraft. $\boldsymbol{\delta r}_i$ is included in the cost function to stabilize the algorithm and prevent the control update from causing large changes to the control. In addition to modifying the cost function, a multiple shooting method[9,24] is also employed. This does not fundamentally change the underlying problem. It is used here to demonstrate that there exist multiple a variety of methods for obtaining feasible trajectories depending. The multiple shooting method breaks the trajectory into $M-1$ segments. The states are now treated as control variables, and the continuity conditions are added to the constraint set. The continuity conditions require that the solution does not have any discontinuities in the mass, position, and velocity of the spacecraft.

The initial Chebyshev trajectory is provided by Fig. (7(a)) and Fig. (7(b)). In these two figures, the initial thrust varies over the entire trajectory, and the thruster is always on. For this example, the specific impulse is set to 6400 seconds, the maximum thrust is 514.7 mN. The launch mass is 3885 kg, and the initial $C_3$ is constrained to 0. The power system is assumed to provide a constant power over all time; the mass of the power system is not considered in this example. The goal of these examples is to demonstrate that custom approaches to solving Eq. (27) can be developed depending on the need.

Applying the Eq. (32) and Alg. (3) to the trajectories in Fig. (11(a)) and Fig. (11(b)) result inFig. (7(a)) and Fig. (7(b)) respectively. The major difference between the initial guess and the final feasible trajectory is the appearance of coast-arcs.

### D. A Method for Handling Over Constrained Problems

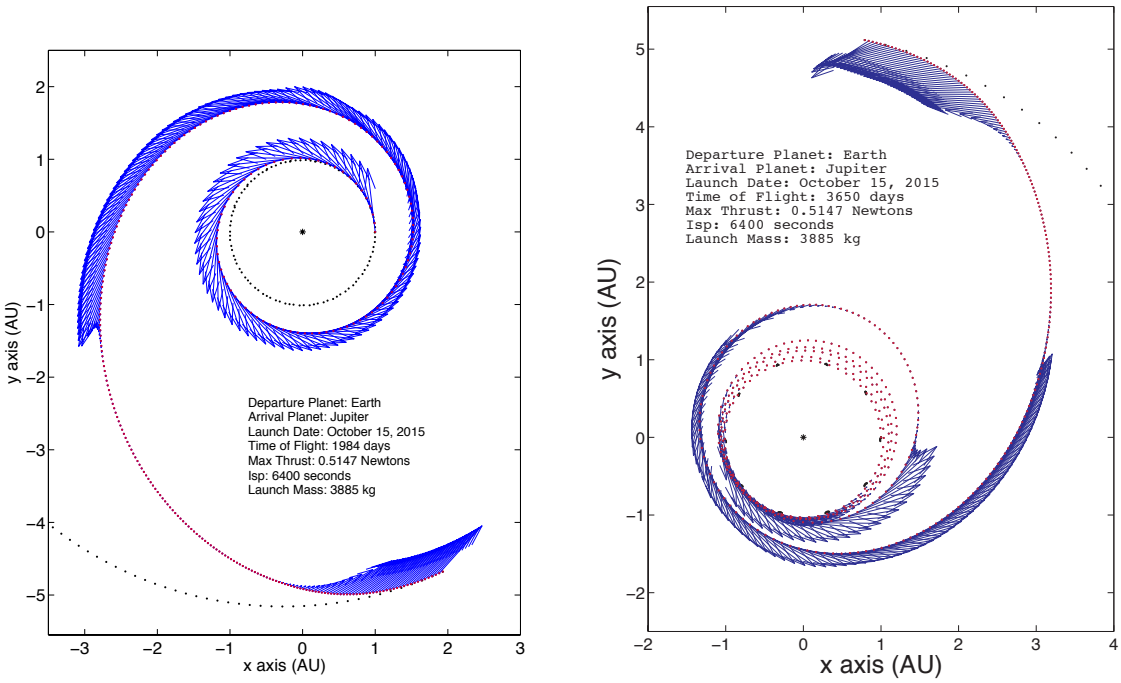
Assuming that $M$ is large enough, Eq. (27) is a linear equation and will always have a solution, however $\boldsymbol{C}$ may actually be over constrained such that no solution exists. For the over constrained case, it is beneficial to know how infeasible the trajectory is relative to an engineering metric. For example, a useful metric might be the thrust or power required to make the trajectory feasible. Normally, the infeasibility is quantified as $\boldsymbol{C}^T\boldsymbol{C}$.[8,19] Because $\boldsymbol{C}$ is a mixture of constraints, this does not provide any real insight as to how far the infeasible solution is from being feasible.

Instead of returning the norm of the constraint violations,[8,19] $\boldsymbol{C}$, when the solver fails, it attempts to answer the question, "How much thrust or power is required for a successful mission?" The solver uses a homotopy method to decrease the maximum thrust/power level until an infeasible solution found. When an infeasible solution is found, the last feasible solution is the minimum thrust/power required for the mission.

In this example, a solar power source is utilized and the specific impulse specified is 3100 seconds. The maximum initial jet power requested is 1.14 kW and scales with Eq. (31). The time of flight for the Earth to Mars trip is 600 days, the $C_3$ is zero, and the launch mass is 500 kg. The $C_3$ and launch mass are fixed to prevent the solver from reducing the launch mass or increasing the $C_3$ to satisfy the constraints. The requested thrust level is 75 mN. The solver fails to find a feasible trajectory with a thrust level of 75 mN and returns Fig. (12). This shows that the thrust level for a feasible trajectory is about 85 mN, so the thruster needs to provide 10 mN of extra thrust to make the mission feasible. In this case, the solver starts with a maximum thrust level of 119 mN then decreases the thrust in 1 mN increments. The solver fails at 84 mN so the 85 mN trajectory is the minimum thrust solution.

## IV. Conclusions

In this paper, we have demonstrated that it is possible and relatively simple to generate feasible EP trajectories without requiring a user to supply an initial guess. The algorithms described bootstrap themselves up, generate their own initial guess, and only require the user to supply the mission or system constraints, which defines the trade space. This reduces the burden on the user and allows non trajectory specialists




(a) Fully integrated 5.5 year Earth to Jupiter trajectory. Initial guess is provided from Fig. (7(a)). The jet power is 32.28 kW. The power system is assumed to provide a constant source of power.

(b) Fully integrated 10 year multi revolution Earth to Jupiter trajectory. Chebyshev approximation is provided Fig. (7(b)). The jet power is 32.28 kW. The power system is assumed to provide a constant source of power.

**Figure 11. The final fully integrated trajectories using the Chebyshev trajectories as initial guesses**

to easily find EP trajectories. Furthermore, the algorithms use simple numerical methods, so they can be implemented on almost any platform without requiring specially licensed software. Also, in the case where a feasible solution does not exist, we demonstrate that the solver returns a feasible solution and measures the infeasibility in terms of engineering parameters like the power/thrust required. This provides a useful metric that can easily be used to iterate the design. While the trajectories generated in this section are feasible, they are not optimal. In order to complete the design look, the feasible trajectories would need to be fed into an optimizer.

## Acknowledgments

This research was initially conducted a the University of Michigan and was supported by a NASA GSRP grant. Prashant Patel would like to thank Richard Hofer and Celeste Satter for their advice and help with this work.

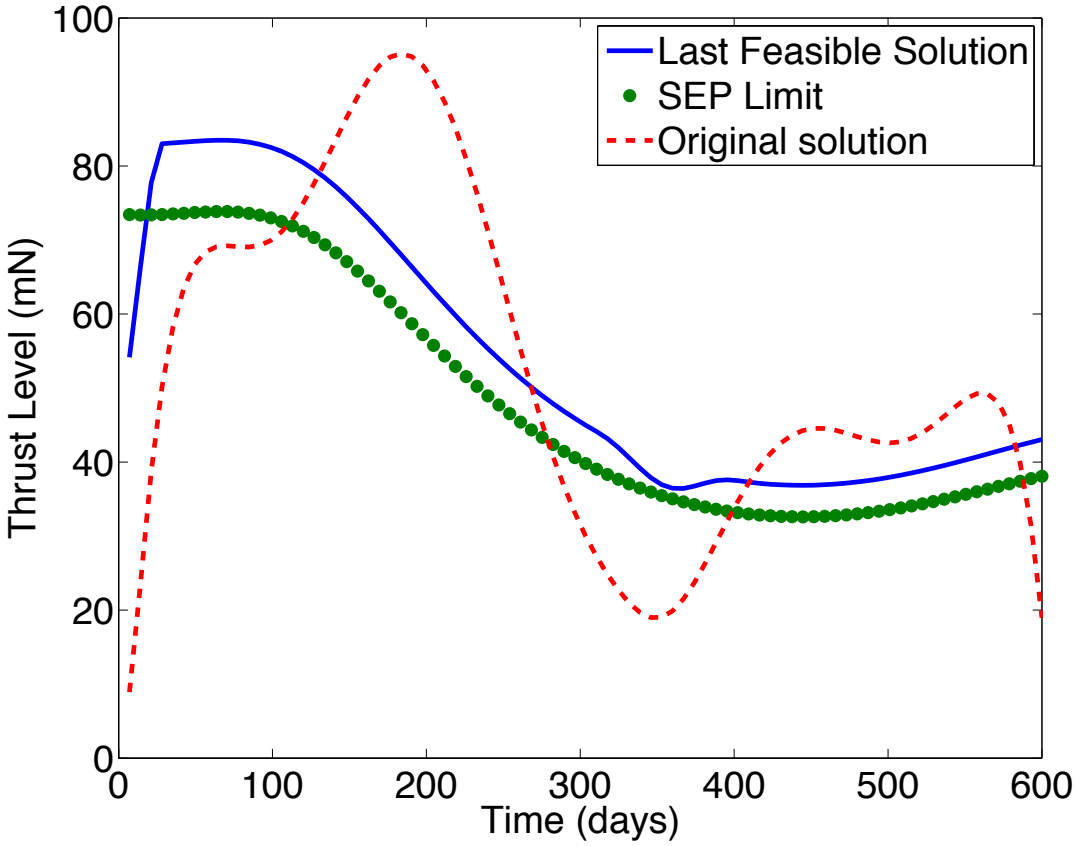



**Figure 12. The thrust profile for a 600 day Earth to Mars mission. The thrust and $C_3$ are fixed at 500 kg and zero respectively. The power is supplied by solar arrays. Using the psuedoinverse approach the solver attempts to converge on the closest thrust profile that satisfies the constraints. Because the constraint cannot be satisfied the solver returns the last feasible solution which indicates that the power system needs to provide an additional 152 W of initial jet power and the thruster needs to provide 10 mN of additional thrust.**

## References


[1]Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, Microcosm, Inc, 2nd ed., May 2001.

[2]Danby, J. M. A., *Fundamentals of Celestial Mechanics*, Willmann-Bell, 1988.

[3]Zurbuchen, T., Patel, P., Fisk, L., Zank, G., Malhotra, R., Funsten, H., Mewaldt, R. A., and Team, N. I. P. V. M., *NASA Space Science Vision Missions*, Vol. 224 of *Progress in Astronautics and Aeronautics*, chap. 5, AIAA, 2008, pp. 155–190.

[4]Whiffen, G. J., "Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity, Low-Thrust Trajectory Design," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, No. 2006-6741, AIAA, August 2006.

[5]Whiffen, G. and Sims, J., "Application of a Novel Optimal Control Algorithm to Low-Thrust Trajectory Optimization," No. AAS 01-209, February 2001.

[6]Whiffen, G. and Sims, J., "Application of the SDC Optimal Control Algorithm To Low-Thrust Escape and Capture Trajectory Optimization," No. AAS 02-208, January 2002.

[7]Liao, L. and Shoemaker, C. A., "Advantages of Differential Dynamic Programming Over Newton"s Method for Discrete-time Optimal Control Problems," Tech. rep., Ithaca, NY, USA, 1992.

[8]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131.

[9]Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[10]Lantoine, G. and Russell, R., "A Hybrid Differential Dynamic Programming Algorithm for Robust Low-Thrust Optimization," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, AIAA/AAS, August 2008.

[11]Petropoulos, A. and Longuski, J., "Shape-Based Algorithm for the Automated Design of Low-Thrust, Gravity Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, 2004, pp. 787–796.

[12]Izzo, D., "Lambert's Problem for Exponential Sinusoids," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 5, 2006, pp. 1242–1245.

[13]Patel, P., Scheeres, D., and Zurbuchen, T., "Spacecraft Trajectories a Shape Based Approach: Analysis and Optimization," No. AAS 05-130, January 2005.

[14]Petropoulos, A. E., *A Shape-Based Approach to Automated, Low-Thrust Gravity-Assist Trajectory Design*, Ph.D. thesis, Purdue University, 2001.

[15]Petropoulos, A. E. and Sims, J. A., "A Reveiw of Some Exact Solutions to the Planer Equations of Motion of a Thrusting Spacecraft," 2002.

[16]McConaghy, T. T., Debban, T. J., Petropoulos, A. E., and Longuski, J. M., "Design and Optimization of Low-Thrust Trajectories with Gravity Assists," *Journal of Spacecraft and Rockets*, Vol. 40, No. 3, 2003, pp. 380–387.

[17]Petropoulos, A., Kowalkowski, T., Parcher, D., Finlayson, P., Rinderle, E., Vavrina, M., Sims, J., Russell, R., Lam, T., Williams, P., Whiffen, G., Strange, N., Johannsen, J., Yen, C.-W., Sauer, C., Lee, S., and Williams, S., "Response to the First ACT Competition on Global Trajectory Optimisation," Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2006, February 2006.

[18]McConaghy, T. T. and Longuski, J. M., "Parameterization Effects on Convergence when Optimizing a Low-Thrust Trajectory with Gravity Assists," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2004.

[19]McConaghy, T. T., *GALLOP Version 4.5 User's Guide*, Purdue University, West Lafayette, Indiana, September 2005.

[20]Yam, C., McConaghy, T., Chen, K., and Longuski, J., "Preliminary Design of Nuclear Electric Propulsion Missions to the Outer Planets," No. AIAA Paper 2004-5393, August 2004.

[21]Sims, J. A., Finlayson, P. A., Rinderle, E. A., Vavrina, M. A., and Kowalkowski, T. D., "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design," No. AIAA 2006-6746, August 2006.

[22]Scheeres, D., "Ephemeris Design, Generation and Retrieval for On-Board Autonomous Navigation Applications," *JPL Interoffice Memorandum*, , No. 312.3-95-003.

[23]Weisstein, E. W., "Chebyshev Polynomial of the First Kind," May 2008.

[24]Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer Science+Business Media, Inc. New York, 1999.

[25]Strang, G., *Linear Algebra and its Applications*, Harcourt College Publishing, Orlando FL, 1998.