Optimal Low-Thrust GTO to GEO Pareto Front Generation Using Continuation Techniques

Galen Savidge*
University of Colorado at Boulder, Boulder, CO 80309

In this report we derive an indirect method to solve for fixed-time low-thrust transfers and apply it to the Pareto front generation problem. First, a low-thrust transfer problem is introduced and then solved analytically via an equinoctial element optimal control formulation. Some numerical solution algorithms are then explored, both for solving individual transfer problems and for generating time-ΔV Pareto fronts. The latter include a "brute force" method and a continuation algorithm where an initial optimal trajectory solution is used to generate solutions to nearby problems. Solutions to a multi-revolution GTO to GEO low-thrust transfer are presented. The numerical algorithms used exhibit good convergence properties, but turn out to often produce trajectories that are not fuel optimal.

I. Introduction

LOW-THRUST propulsion has increased in popularity in recent years for long orbit transfers due to its high specific impulse and thus mass-to-thrust efficiency. This has allowed small, cheap spacecraft to perform transfers previously only possible for large satellites. One example of such a mission type is the geosynchronous transfer orbit (GTO) to geostationary orbit (GEO) transfer, a maneuver which provides a cost-effective way to place dedicated satellites in geostationary Earth orbits. When designing such a mission, transfer time and expended fuel are typically both of interest, as the goal is often to maximize the operational lifetime of the satellite. It is then necessary to solve a large set of transfer trajectories to properly assess the trade-off between these two objectives. This time-ΔV Pareto front generation problem, then, tends to be very computationally expensive. In this report we will derive an indirect for solving the fixed time minimum fuel problem, then examine some algorithms for generating Pareto fronts and test them in the context of a low-thrust GTO to GEO transfer.

II. Problem Statement

We will construct and analyze a example mission in which we wish to move an ESPA-class satellite from GTO to GEO. The spacecraft design will be based on SMART-1,[1] with launch mass of 367 kg after separation, 82.5 kg of which is xenon propellant. It will carry a PPS-1350 electronic thruster on board, capable of producing 68mN of thrust with a specific impulse of 1640s at beginning of life. These thruster characteristics are assumed to remain constant over the mission since the spacecraft will remain in the vicinity of Earth and thus available power will not change. The initial GTO orbit is based on the capabilities of a Falcon 9, as this is a common launch vehicle for small commercial satellites. Perigee altitude is 185 km nominally, while apogee will be somewhere above the geosynchronous radius of 42,164 km. We will assume it is roughly 150 km above GEO at 42,300 km. This orbit will have a 28.5 degree inclination if the spacecraft is launched from the Kennedy Space Center.

We will use the modified equinoctial element set (MEE) to represent the orbital state of the spacecraft. This set

*Project report for ASEN 6020: Optimal Trajectories

can be defined in terms of the Keplerian element set $(a,\ e,\ i,\ \omega,\ \Omega,\ \Theta)$ as[2]

$$\begin{aligned} p &= a(1-e^2) \\ f &= e\ \cos(\omega+\Omega) \\ g &= e\ \sin(\omega+\Omega) \\ h &= \tan\left(\frac{i}{2}\right)\cos(\Omega) \\ k &= \tan\left(\frac{i}{2}\right)\sin(\Omega) \\ L &= \Omega+\omega+\Theta \end{aligned} \tag{1}$$

where p is the orbit parameter, f and g are components of the eccentricity vector, h and k define the orientation of the orbit plane, and L is true longitude. We chose this set because, unlike the classical orbital element set, it is non-singular near circular, non-inclined orbits. It is also numerically propagates faster and exhibits better convergence properties when used in numerical optimization solvers than the Cartesian element set.[3] The orbit state vector is defined as

$$\vec{x} = \begin{bmatrix} p & f & g & h & k & L \end{bmatrix}^T \tag{2}$$

The dynamics of the MEE set can be written as a set of first order nonlinear differential equations in terms of the instantaneous applied thrust.[3] For some acceleration vector $\vec{u} = [u_r,\ u_t,\ u_n]$ defined in the LVLH frame, the dynamics are given by

$$\dot{\vec{x}} = \boldsymbol{A} + \boldsymbol{B}\vec{u} \tag{3}$$

where the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices can be calculated as a function of the state $\vec{x}$:

$$\boldsymbol{A} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \sqrt{\mu p}(\frac{w}{p})^2 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & \frac{2p}{w}\sqrt{\frac{p}{\mu}} & 0 \\ \sqrt{\frac{p}{\mu}}\sin(L) & \sqrt{\frac{p}{\mu}}\frac{1}{w}[(w+1)\cos(L)+f] & -\sqrt{\frac{p}{\mu}}\frac{g}{w}[h\sin(L)-k\cos(L)] \\ -\sqrt{\frac{p}{\mu}}\cos(L) & \sqrt{\frac{p}{\mu}}\frac{1}{w}[(w+1)\sin(L)+g] & \sqrt{\frac{p}{\mu}}\frac{f}{w}[h\sin(L)-k\cos(L)] \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{s^2}{2w}\cos(L) \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{s^2}{2w}\sin(L) \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{1}{w}[h\sin(L)-k\cos(L)] \end{bmatrix} \tag{4}$$

The following intermediate quantities are used:

$$\begin{aligned} w &= 1+f\cos(L)+g\sin(L) \\ s^2 &= 1+h^2+k^2 \end{aligned} \tag{5}$$

We will also consider a simple model of the change in mass due to thruster firing. Rearranging the definition of the specific impulse (I_sp) we can derive an equation for change in mass due to the applied thrust magnitude $T(t)$:

$$\dot{m} = -\frac{1}{g_0 I_{sp}}T(t) \tag{6}$$

Each transfer problem will be set up by constraining the initial orbit, final orbit, and transfer time. We then would like to solve for the optimal control solution such that the amount of propellant used is minimized. The initial orbit

	r_a	r_p	i	ω	Ω	Θ
Initial GTO	$42,300.000$ km	$6,563.136$ km	$28.5°$	0	0	$90°$
Final GEO	$42,164.000$ km	$42,164.000$ km	0	N/A	N/A	Θ_f
	p	f	g	h	k	L
Initial GTO	$11,363.195$ km	0.73136656	0	0.25396765	0	1.5707963 rad
Final GEO	$42,164.000$ km	0	0	0	0	L_f

Table 1: Initial & Final Orbit States

is fully constrained, as it must match the post-separation GTO previously described. The final orbit will also be fully constrained, with the exception of the true longitude L. The transfer will then place the satellite in GEO over some unspecified location on Earth, and a reorientation maneuver would be required to reach the desired Earth longitude. The desired final state is a circular orbit with radius $a = p = r_{GEO}$ and some free true longitude L_f. Both the initial and final orbits are summarized in Table 1.

III. Primer Vector Theory

The idea of the primer vector in spacecraft trajectory optimization was developed by D. F. Lauden and its derivation was published in his book on the subject in 1963.[4] The advantage of this solution technique to the optimal control problem is that the form of the control itself can be solved for analytically in terms of a set of adjoint state variables. As the time history of the adjoints is dependent only on the solution of an initial value problem (IVP), the optimal control law can then be substituted back into the state dynamics and numerically propagated. In the case where the initial state is fully constrained this reduces the continuous optimization problem to a two-point boundary value problem (2PBVP) which can be solved using traditional root-finding techniques.

We will set up our direct solution by following Lauden's primer vector derivation in a similar manner to Junkins & Taheri.[3] The control acceleration is defined by the instantaneous applied thrust $T(t)$:

$$\vec{u} = \frac{T(t)}{m}\hat{u} \tag{7}$$

As mentioned previously we will assume a constant max thrust T_{max} since the spacecraft will remain in the vicinity of Earth. The orbit and mass states can be combined into a single state vector, and the dynamics of this state can be written in matrix-vector form:

$$\begin{bmatrix}\dot{\vec{x}}\\ \dot{m}\end{bmatrix} = F(\vec{x}, m, T, \hat{u}) = \begin{bmatrix}\boldsymbol{A}\\ 0\end{bmatrix} + \begin{bmatrix}\frac{1}{m}\boldsymbol{B}\hat{u}\\ -\frac{1}{g_0 I_{sp}}\end{bmatrix} T(t) \tag{8}$$

We would like to minimize the expended propellant mass, which is equivalent to minimizing total applied thrust for this problem. Since initial mass is constant, we can define the objective function to minimize the negative of the final mass, thereby maximizing the final mass and minimizing the difference in mass.

$$J = -m_f \tag{9}$$

Writing the objective function as a problem of Bolza,

$$J = K + \int L\,dt \quad \Longrightarrow \quad K = -m_f, \quad L = 0 \tag{10}$$

The first step in the primer vector derivation is to assemble the pre-Hamiltonian. First we must introducing an adjoint vector $\vec{p} = [\vec{p_x}^T \quad p_m]^T$, where $\vec{p}_x$ is of equal dimension to the orbital state $\vec{x}$. We can then write the pre-Hamiltonian

$$H = L + \vec{p}\cdot F(\vec{x}, m, T(t), \hat{u}) = \vec{p}_x \cdot \left[\boldsymbol{A} + \boldsymbol{B}\frac{T(t)}{m}\hat{u}\right] + p_m\left(-\frac{1}{g_0 I_{sp}}T(t)\right) \tag{11}$$

Note that the control has two parts, the LVLH thrust unit direction vector $\hat{u}$ and the thrust magnitude $T(t)$. From Pontryagin's Minimum Principle, the optimal control can be derived by choosing $\hat{u}$ and $T(t)$ to minimize the pre-Hamiltonian. Regardless of the choice of thrust magnitude, this occurs when $\hat{u}$ is opposite the direction of the *primer vector*, defined as $\boldsymbol{B}^T\vec{p}_x$:[3]

$$\hat{u}^* = -\frac{\boldsymbol{B}^T\vec{p}_x}{||\boldsymbol{B}^T\vec{p}_x||} \tag{12}$$

Substituting this result back into the pre-Hamiltonian, we find the Hamiltonian H^*:

$$H^* = \vec{p}_x \cdot \boldsymbol{A} - \frac{1}{m}||\boldsymbol{B}^T\vec{p}_x||T(t) - p_m\frac{1}{g_0 I_{sp}}T(t) \tag{13}$$

Recall that the choice of $T(t)$ must also minimize the resulting Hamiltonian. In other words, $T(t)$ must maximize

$$\left[\frac{1}{m}||\boldsymbol{B}^T\vec{p}_x|| + p_m\frac{1}{g_0 I_{sp}}\right]T(t) \tag{14}$$

Rewriting the condition in terms of a switching function S,

$$T(t) = \text{argmax}\Big[S(\vec{x}, m, \vec{p}_x, p_m)T(t)\Big], \quad S(\vec{x}, m, \vec{p}_x, p_m) = \frac{1}{m}||\boldsymbol{B}^T\vec{p}_x|| + p_m\frac{1}{g_0 I_{sp}} \tag{15}$$

By inspection, optimal thrust magnitude can be defined piece-wise as a function of S.

$$T(S) = \begin{cases} 0 & S < 0 \\ T_{max} & S > 0 \end{cases} \tag{16}$$

We will implement smoothing of this function to help solutions converge. Again following Junkins & Taheri,[3] the piece-wise control can be approximated as

$$T(S) = \frac{1}{2}T_{max}\left[1 + \tanh\left(\frac{S}{\rho}\right)\right] \tag{17}$$

where $\rho \in (0, \infty)$ is a smoothing parameter. As ρ approaches zero the returned thrust will more closely resemble a step function, the optimal control law. However, lower ρ values produce problems that are more difficult to solve numerically.

The dynamics of the state and adjoints can now be solved for by taking partial derivatives of the Hamiltonian as shown in Eq. (18). The form of the Hamiltonian makes these calculations difficult to do by hand, so they are instead found using Matlab symbolics. The resulting functions can be inserted directly into a numerical propagator.

$$\begin{aligned} \dot{\vec{x}} &= \frac{\partial H^*}{\partial \vec{p}_x} \\ \dot{m} &= \frac{\partial H^*}{\partial p_m} \\ \dot{\vec{p}}_x &= -\frac{\partial H^*}{\partial \vec{x}} \\ \dot{p}_m &= -\frac{\partial H^*}{\partial m} \end{aligned} \tag{18}$$

A solution to these equations must satisfy the constraints put on the initial and final values of the state, $\vec{x}_0$ and $\vec{x}_f$, as specified in Table 1. The adjoints also must satisfy constraints imposed by the transversality conditions, which are determined from boundary constraints and terminal cost K. Two useful transversality conditions emerge from the fact that L is unconstrained at t_f, and $K = -m_f$.

$$\begin{aligned} p_{x,6}(t_f) &= 0 \\ p_m(t_f) &= \frac{\partial K}{\partial m_f} = -1 \end{aligned} \tag{19}$$

There are seven total terminal constraints and seven unknowns (the values of the initial adjoints). Thus, the optimal control problem is reduced to a 2PBVP, the solution to which can be integrated forwards in time using Eq. (18) to determine the optimal state and adjoint trajectories. The optimal control law can then be calculated at any point in time from Eqs. (12) & (16). Finally, we can map the value of m_f to ΔV using the rocket equation:

$$\Delta V = g_0 I_{sp} \ln\left(\frac{m_0}{m_f}\right) \tag{20}$$

IV. Solution Methods

A. Indirect Single Shooting

Single shooting methods solve the 2PBVP posed by the optimal control solution by iteratively propagating through the entire transfer time and updating the initial conditions until the boundary conditions are met. In our case, the only unknowns are the seven initial adjoints. With a guess for these the dynamics in Eq. (18) can be propagated to the fixed transfer time t_f and the final states can be checked against the desired constraints on $\vec{x}_f$, $p_{x,6}(t_f)$, and $p_m(t_f)$. Solutions are found using a numerical root solver, in our case Matlab's fsolve, which provides a trust-region-dogleg algorithm that we found to be very robust. All propagations use Matlab's built-in 8th order variable step Runge-Kutta solver with tolerances set to 1×10^{-12}. The finite differencing gradient calculations that fsolve employs appear to be very sensitive to the step size setting, with values outside of a certain range lowering the maximum precision of the solver's solutions. We found that central finite differencing with the scaling factor parameter set to 5×10^{-5} led to more consistent and accurate convergence.

Recall that we introduced a smoothing parameter ρ which can be tuned to make the optimal control problem easier to solve at the cost of solution accuracy. Junkins & Taheri introduced a continuation technique that uses a high value of ρ to find a guess for the initial state, then iteratively decreases ρ and re-solves using the previously generated guesses in a number of sub-problems.[3] We adopted this technique with some algorithmic modifications to increase the odds of convergence, and thus increase the applicability to automated problem solving. In particular, if a sub-problem iteration fails to converge, ρ is increased and the initial adjoints are perturbed by a small amount which is scaled by the current ρ value. This has proved effective in increasing the number of converged cases and preventing the algorithm from becoming "stuck."

Algorithm 1 Continuation method for two-point boundary value problem solutions

$\rho \leftarrow 1$
while $e > e_{tol}$ **do**
  $\vec{p}_{0,guess} \leftarrow$ random values $\in [0, 0.1]$
  $\vec{p}_0,\ e \leftarrow$ solve_2pbvp($\vec{p}_{0,guess}$, ρ) ▷ Matlab's fsolve is used
end while
$\rho \leftarrow 0.1$
while $\rho > \rho_{min}$ **do**
  $\vec{p}_{0,soln},\ e \leftarrow$ solve_2pbvp($\vec{p}_0$, ρ)
  if $e < e_{tol}$ **then**
    $\rho \leftarrow \rho \times 0.1$
    $\vec{p}_0 \leftarrow \vec{p}_{0,soln}$
  else
    $\rho \leftarrow$ min(1, $\rho \times 10$)
    $\vec{p}_0 \leftarrow \vec{p}_0 +$ random values $\in [-\frac{\rho}{200}, \frac{\rho}{200}]$ ▷ Perturb the $\vec{p}_0$ value
  end if
end while
return $\vec{p}_{0,soln}$

B. Pareto Front Generation

Plotting a time-ΔV Pareto front requires solving the previously posed minimum-fuel optimization for a set of fixed transfer times within some time window of interest. Because each solution involves many sub-problems, each of which requires many precise numerical propagations, this process is naturally very computationally expensive. We developed two algorithmic approaches to automate the solution procedure for this process. The first is a "brute force" method where the procedure in Algorithm 1 is repeated for each transfer time in the set. The second solves some initial problem and then uses a continuation technique to solve for adjacent problems with similar transfer times. This way the initial adjoints must only be guessed once, and each subsequent solution will be seeded with a guess that is close to its solution. Note that the scaling of the smoothing parameter ρ works in a similar manner to Algorithm 1, but is kept low when moving to the next adjacent transfer time value.

C. Problem Normalization

Numerical solvers tend to have difficulty with space trajectory problems due to the large difference in magnitude between state components, thrusts, and other parameters expressed in canonical units. A common way to address this

Algorithm 2 Brute force method for Pareto front generation

$t_f \leftarrow t_{f,min}$
while $t_f \leq t_{f,max}$ **do**
  $\Delta V_i \leftarrow$ solve_transfer_problem(t_f) ▷ See Algorithm 1
  $t_f \leftarrow t_f + \Delta t_f$
end while
return $\{t_{f,i}, \Delta V_i\}$

Algorithm 3 Continuation method for Pareto front generation

$t_f \leftarrow t_{f,min}$
$\rho_{init},\ \vec{p}_0 \leftarrow$ solve_transfer_problem(t_f) ▷ See Algorithm 1
$N_{failed} \leftarrow 0$
$\rho \leftarrow \rho_{init} \times 10$
while $t_f \leq t_{f,max}$ **do**
  $t_f \leftarrow t_f + \Delta t_f$
  while $\rho > \rho_{min}$ **do**
    $\vec{p}_{0,soln},\ e \leftarrow$ solve_2pbvp($\vec{p}_0$, ρ)
    if $e < e_{tol}$ **then**
      $\rho \leftarrow \rho \times 0.1$
      $\vec{p}_0 \leftarrow \vec{p}_{0,soln}$
    else
      $\rho \leftarrow$ min(1, $\rho \times 10$)
      $\vec{p}_0 \leftarrow \vec{p}_0 +$ random values $\in [-\frac{\rho}{200}, \frac{\rho}{200}]$ ▷ Perturb the $\vec{p}_0$ value
    end if
  end while
  $\Delta V_i \leftarrow \Delta V(\vec{p}_0)$ ▷ From Eq. (20)
end while
return $\{t_{f,i}, \Delta V_i\}$

issue is to translate the problem into scaled units. We will set up scaling for the low-thrust transfer problem such that the states fall in the range $[0, 1]$ and $\mu = 1$. The problem-scaled units of distance, time, and mass are denoted LU, TU, and MU, respectively, and are defined as

$$\begin{aligned} LU &= p_f \\ TU &= \sqrt{\frac{LU^3}{\mu}} \\ MU &= m_0 \end{aligned} \tag{21}$$

The rest of the problem parameters can then be translated into the scaled units (note that p is the orbit parameter in this case, not an adjoint):

$$\begin{aligned} p_0' &= p_0 \times \frac{1}{LU} \\ p_f' &= p_f \times \frac{1}{LU} \\ m_0' &= m_0 \times \frac{1}{MU} \\ g_0' &= g_0 \times \frac{TU^2}{LU} \\ I_{sp}' &= I_{sp} \times \frac{1}{TU} \\ T_{max}' &= T_{max} \times \frac{TU^2}{MU \times LU} \end{aligned} \tag{22}$$

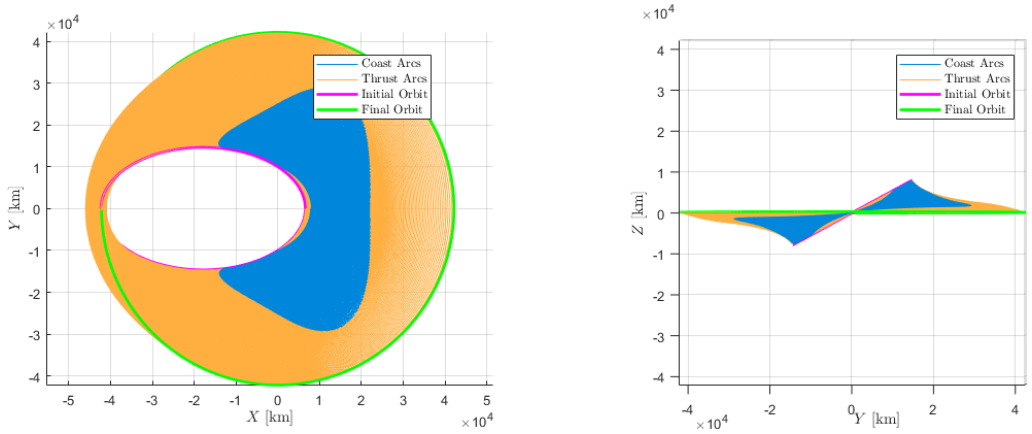

Figure 1: Optimal Six Month Transfer Trajectory

	p_f	f_f	g_f	h_f	k_f	$p_{x,6}(t_f)$	$p_m(t_f)$
State Error	1.1808 km	4.46×10^{-4}	2.85×10^{-4}	-1.67×10^{-4}	-7.48×10^{-4}	-1.00×10^{-4}	9.66×10^{-5}

Table 2: Final State and Adjoint Errors for the Six Month Transfer Solution

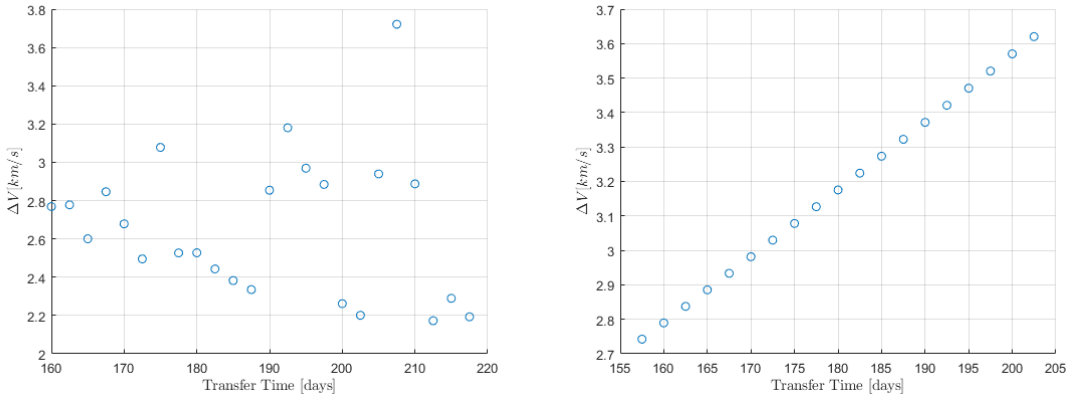

Figure 2: Results From Brute Force Pareto Front Generation Method (Left) and Continuation Method (Right)

V. Results

Using Algorithm 1 with the normalized problem to solve for individual fixed-time transfers works well, with some caveats. The error tolerance must be increased to 5×10^{-6} to see consistent convergence, which leaves some error in the final states and may lead to suboptimal solutions for the adjoint dynamics. The solution procedure also has trouble converging for transfer times outside of a fairly constricted window. The minimum transfer time appears to be about 4.7 months, below which no solutions converge as the maximum thrust available to the spacecraft is too low to complete the transfer. Above about 7.5 months the numerical solver also has trouble converging to solutions that meet the set tolerances, likely because of cumulative errors in long propagations and limitations of the finite differencing method used to calculating gradients. This occurs even when the initial guess is quite close to the optimal solution.

Neither Pareto front generation algorithm proved to perform very well in practice. Results from Algorithm 2 show reduction in minimum fuel use as transfer time increases, but also include many solution points which converged on another local minimum, or perhaps even the maximum thrust trajectory. Algorithm 3 did not provide any better results, actually appearing to converge to the maximum fuel trajectory in every case. The lack of a sufficient condition for optimality in our formulation may have allowed these solutions to be seen as valid, or another issue may exist in our numerical solution techniques.

VI. Conclusion

In this report we derived and tested indirect methods for the solution of the fuel-optimal GTO to GEO low-thrust transfer problem. The problem was set up as a fixed-time transfer with a spacecraft modelled off of the SMART-1 mission. We then derived a primer vector formulation in the two body problem using the modified equinoctial element set (MEE) and showed how it can be used to solve for optimal trajectories via single shooting. The results obtained from this formulation only hold for an unperturbed two-body system; integrating disturbance models into the analytical derivation would increase the applicability of this method to real-world missions. Many spacecraft are also unable to produce thrust while in eclipse, a possibility that we ignored in our derivation.

The numerical methods and algorithms described were shown to exhibit good convergence properties, but not always converge to optimal solutions. In particular, the Pareto front generators both failed to produce sets of optimal points, as is clear from Figure 2. Closer examination of the analytical formulation and convergence properties of the solution method used would be needed to determine why this is the case. The addition of either another constraint or further iteration (using some check for optimality) seem like plausible solutions. Additionally, we observed that the problem became more difficult to solve as transfer time increased, until the numerical solver was eventually unable to meet the tolerances we defined. Using a more precise numerical propagator or switching to a multiple shooting method are both possible solutions to this issue, though the latter would likely be less computationally intensive. A third option would be to experiment with other gradient calculation techniques, such as the STM method described by Taheri.[5]

Appendix: Unused Numerical Solution Techniques

A. Newton Iteration[6]

This is an iterative method to find roots of $f(\vec{x}) = \vec{0}$, $f(\vec{x}) \in \mathbb{R}^N$, $\vec{x} \in \mathbb{R}^N$, with each iteration given by the following equation:

$$\vec{x}_{n+1} = \vec{x}_n - \alpha \frac{\partial f(\vec{x})}{\partial \vec{x}}\bigg|_{x_n} f(\vec{x}_n) \tag{23}$$

To stabilize iteration, we can choose α such that $||\vec{x}_{n+1}|| < ||\vec{x}_n||$. There are multiple ways to generate an initial guess for this algorithm. One is to use random sampling, i.e. by taking a large set of random guesses and picking the lowest norm or least squares of $f(\vec{x}_0)$.

B. Constant Thrust Segments

This method can be set up as a "pseudo-collocation" direct multiple shooting problem by choosing N constraint points and assuming thrust over each segment will be constant. Then $N-1$ constraints can be constructed by integrating dynamics between points, either using a multi-step integrator or in one step, e.g. with a Runge-Kutta method. This transforms the optimal control problem into a nonlinear programming problem which can be solved numerically. The resulting control trajectory can then be propagated in a more accurate dynamical simulation to assess the amount of error in the solution.

C. Hermite-Simpson Collocation[7]

The state and control trajectories are approximated as a finite series of N collocation points P_k, each containing a snapshot of the state and control at that point in time. These points can then be interpolated between to solve for the full trajectory.

$$P_k = \left\{ t_k,\ \vec{x}_k,\ ,\vec{u}_k \right\} \tag{24}$$

Dynamic constraints are calculated from Eq. 25, for $\dot{\vec{x}} = f(\vec{x}, \vec{u})$ with $\Delta t_k = t_{k+1} - t_k$. The two equations can be combined, leading to $N-1$ constraint equations at the k points, which is known as compressed form. Alternatively, the $k + \frac{1}{2}$ points can be left as independent decision variables, resulting in $2N-2$ dynamics constraint equations. This is known as separated form. Generally, compressed form tends to become preferable as the number of segments increases.[7]

$$\begin{aligned} \vec{x}_{k+1} - \vec{x}_k &= \frac{1}{6}\Delta t_k \left[f(\vec{x}_k, \vec{u}_k) + 4f(\vec{x}_{k+\frac{1}{2}}, \vec{u}_{k+\frac{1}{2}}) + f(\vec{x}_{k+1}, \vec{u}_{k+1}) \right] \\ \vec{x}_{k+\frac{1}{2}} &= \frac{1}{2}(\vec{x}_k + \vec{x}_{k+1}) + \frac{1}{8}\Delta t_k (f(\vec{x}_k, \vec{u}_k) - f(\vec{x}_{k+1}, \vec{u}_{k+1})) \end{aligned} \tag{25}$$

There are $2N-1$ independent control variables $\vec{u}_i$, including both the k and $k+\frac{1}{2}$ collocation points. Path constraints are applied at all points, while boundary constraints are applied at the initial and final points.

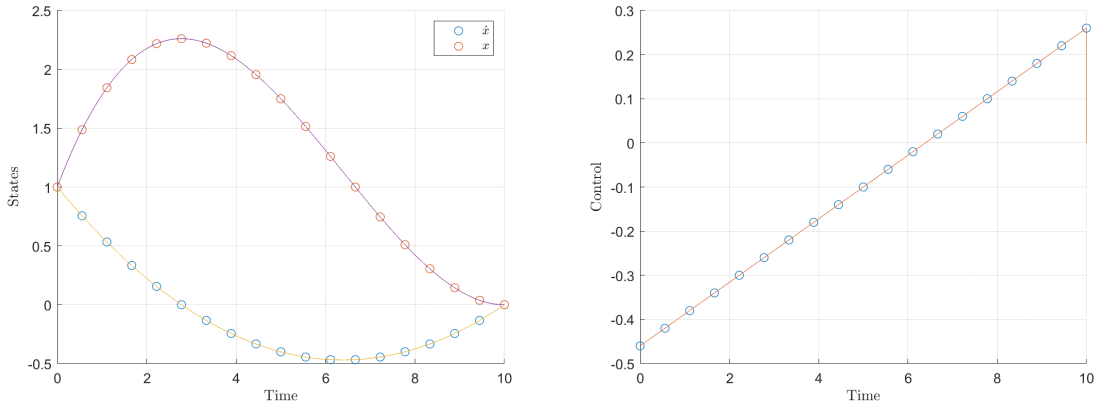

Figure 3: Example Solution by Collocation, States (Left) & Control (Right)

Once all the P_k have been solved, the control and state trajectories can be interpolated using the following spline functions for $\tau = t - t_k$:

$$\begin{aligned}
\vec{u}(t) &= \frac{2}{\Delta t_k^2}\Big(\tau - \frac{1}{2}\Delta t_k\Big)\Big(\tau - \Delta t_k\Big)\vec{u}_k - \frac{4}{\Delta t_k^2}\Big(\tau\Big)\Big(\tau - \Delta t_k\Big)\vec{u}_{k+\frac{1}{2}} \\
&\quad + \frac{2}{\Delta t_k^2}\Big(\tau\Big)\Big(\tau - \frac{1}{2}\Delta t_k\Big)\vec{u}_{k+1} \\
\vec{x}(t) &= \vec{x}_k + \Bigg[f_k\Big(\frac{\tau}{\Delta t_k}\Big) + \frac{1}{2}\Big(-3f_k + 4f_{k+\frac{1}{2}} - f_{k+1}\Big)\Big(\frac{\tau}{\Delta t_k}\Big)^2 \\
&\quad + \frac{1}{3}\Big(2f_k - 4f_{k+\frac{1}{2}} + 2f_{k+1}\Big)\Big(\frac{\tau}{\Delta t_k}\Big)^3\Bigg]\Delta t_k
\end{aligned} \tag{26}$$

We obtained good results when testing this method with a sample problem, but were unsuccessful in applying it to the orbit transfer problem. The sample problem is a fixed-time minimum-energy point mass motion problem, as defined below:

$$\begin{aligned}
\vec{X} &= \begin{bmatrix} \dot{x} \\ x \end{bmatrix} \\
\dot{x} &= u \\
J &= \int_{t_0}^{t_f} u^2 \, dt
\end{aligned} \tag{27}$$

The resulting interpolated state and control for $t_f = 10$, $N = 10$, $\vec{X}_0 = [1 \quad 1]^T$, and $\vec{X}_f = [0 \quad 0]^T$ are shown in Figure 3.

References

[1]Rathsman, P., Kugelberg, J., Bodin, P., Racca, G. D., Foing, B., and Stagnaro, L., "SMART-1: Development and lessons learnt," *Acta Astronautica*, Vol. 57, No. 2, 2005, pp. 455–468, Infinite Possibilities Global Realities, Selected Proceedings of the 55th International Astronautical Federation Congress, Vancouver, Canada, 4-8 October 2004.

[2]Walker, M. J. H., Ireland, B., and Owens, J., "A set of modified equinoctial orbit elements," *Celestial Mechanics*, Vol. 36, 1985, pp. 409–419.

[3]Junkins, J. L. and Taheri, E., "Exploration of Alternative State Vector Choices for Low-Thrust Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2019, pp. 47–64.

[4]Lauden, D. F., *Optimal Trajectories for Space Navigation*, Butterworths, 1963.

[5]Ehsan Taheri, I. K. and Atkins, E., "Enhanced Smoothing Technique for Indirect Optimization of Minimum-Fuel Low-Thrust Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, 2016, pp. 2500–2511.

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

[7]Kelly, M., "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, Vol. 59, No. 4, 2017, pp. 849–904.