Solar Sailing Between Earth & Mars

Victor Minden
vminden@stanford.edu
Stanford Institute for Computational and Mathematical Engineering

CME304: Numerical Optimization – Winter 2013

Abstract

The purpose of this project is to assess the optimal control and minimum time necessary to bring a spacecraft employing simple solar sail technologies from the Earth's orbit around the sun to the orbit of Mars and back again. The problem is modified as a two-dimensional system in polar coordinates about the sun and Newton's laws are assumed to be sufficient to model the required kinematics.

To format the minimum-time kinematics problem as a standard optimization problem, we solve a sequence of sub-problems where the final time is held fixed and a feasible trajectory is calculated. We consider the objective function to be the squared difference between the final point of the trajectory and the desired final state, and use a penalty function to account for the nonlinear constraints given by Newton's laws. The optimal control is subject to box constraints.

Solving this optimization problem is done with an active-set BFGS method employing a line-search satisfying the strong Wolfe conditions.

Contents

I. Introduction

We look in this project at the feasibility of using a solar-sail to bring a space craft from the orbit of the Earth around the sun to Mars's orbit. For a full background of the project, we refer to the project assignment sheet (or to [2], a standard reference on solar sail technologies). In essence however, we wish simply to determine as a function of time the control angle between the normal vector to the solar sail and the radial vector between the spacecraft and Sun. Furthermore, we wish to find the control which brings the spacecraft to the orbit of Mars in minimum time for a given solar sail technology (specified by its *characteristic acceleration*). We are also interested in solving the reverse problem, where the spacecraft starts at Mars and must make it to the orbit of Earth.

A. Notation

In this write-up we use the following conventions (Table I and II):

TABLE I. Variables used in this write-up

Name	Description
$\mathbf{r}$	position vector of the spacecraft with respect to the sun
r	radial distance from the spacecraft to the sun (i.e., $r = \|\|\mathbf{r}\|\|_2$)
θ	angle of the spacecraft relative to the sun as measure from the x-axis
m	mass of the spacecraft (unimportant)
a	characteristic acceleration of solar sail technology
α	angle of sail relative to the unit vector $\hat{\mathbf{r}}$
$\hat{\mathbf{n}}$	unit outward normal vector of sail (i.e., at $\alpha = 0$, we have $\hat{\mathbf{n}} = \hat{\mathbf{r}}$)

TABLE II. Constants used in this write-up

Name	Description	Value ($\bar{m} = 1 \times 10^{11} m$, $\bar{s} = 1 \times 10^{7} s$)
r_E	radius of Earth's solar orbit	1.496×10^{11} m = $1.496\bar{m}$
ω_E	angular velocity of Earth's solar orbit	1.991×10^{-7} rad/s = 1.991 rad$/\bar{s}$
r_M	radius of Mars's solar orbit	2.279×10^{11}m = $2.279\bar{m}$
ω_M	angular velocity of Mars's solar orbit	1.059×10^{-7} rad/s = 1.059 rad$/\bar{s}$
r_0	astronomical unit distance	1.496×10^{11} m = $1.496\bar{m}$
μ	heliocentric gravitational constant	1.327×10^{20} m^3/s^2 = $13.27\bar{m}^3/\bar{s}^2$

We are given that the characteristic acceleration a is bounded as $a \in [1, 2]$ mm/s^2, and with this, the transformation of scale ($\bar{m} = 1 \times 10^{11}$ m and $\bar{s} = 1 \times 10^{7}$ s) gives $a \in [1, 2]$ $\bar{m}/\bar{s}^2$. We employ this scaling throughout the problem to keep all the constants and state variables within an order of magnitude or so of each other, as it is well-known that vastly different scalings of variables can lead to an ill-conditioned Hessian.

B. Equations of Motion

In modeling the kinematics of the solar sail spacecraft, we make a few assumptions that are perhaps unrealistic. Enumerated, these are:

- orbits are heliocentric, circular and co-planar,
- gravitational effects due to celestial bodies other than the Sun are negligible,
- the Sun and spacecraft are point masses, and,
- the Sun has spherically symmetric gravitational and radiation fields.

Since we are ignoring gravitational effects other than those of the Sun (and those we assume to be spherically symmetric), the equation of motion we are given is simply Newton's law:

$$m\ddot{\mathbf{r}} = -\frac{m\mu}{r^2}\hat{\mathbf{r}} + ma\left(\frac{r_0}{r}\right)^2 \cos^2(\alpha)\hat{\mathbf{n}}.$$

The first term on the right-hand side corresponds to the gravitational force due to the sun, and the second term corresponds to the force on the spacecraft due to photons hitting the solar sail (which is oriented at angle α). We notice immediately that we can cancel the factor of m, and with this we obtain

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\hat{\mathbf{r}} + a\frac{r_0^2}{r^2} \cos^2(\alpha)\hat{\mathbf{n}}.$$

We will now formulate this as a time-optimal control problem using the state-space method [1]. Because we assume orbits are coplanar, we can look at this as a 2D problem in the plane as opposed to a 3D problem in all of space.

Thus, we will model the state of the spacecraft-Sun system using just four state variables: the radial distance between the spacecraft and the Sun, r, the angle of the spacecraft relative to the horizontal axis in the plane, θ, and their time derivatives, v and ω. Thus, we define our state vector as

$$x \equiv \begin{bmatrix} r \\ \theta \\ v \\ \omega \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

Newton's laws of physics in a polar frame of reference give the force in the radial direction to be $F_r = ma_r = m(\ddot{r} - r\omega^2)$ and the force in the transverse direction to be $F_\theta = ma_\theta = m(r\dot{\omega} + 2\dot{r}\omega)$ (see, e.g., [4]). Thus, we can frame this as a standard dynamical system by decomposing the forces into their $\hat{r}$ and $\hat{\theta}$ components, giving the following relationships:

$$\begin{aligned}
\frac{d}{dt}r &= v \equiv f_1(x,\alpha) \\
\frac{d}{dt}\theta &= \omega \equiv f_2(x,\alpha) \\
\frac{d}{dt}v &= \omega^2 r - \frac{\mu}{r^2} + a\frac{r_0^2}{r^2}\cos^3(\alpha) \equiv f_3(x,\alpha), \\
\frac{d}{dt}\omega &= -\frac{2\omega v}{r} + a\frac{r_0^2}{r^3}\cos^2(\alpha)\sin(\alpha) \equiv f_4(x,\alpha).
\end{aligned}$$

With these relationships, we can write out the system dynamics as

$$\dot{x} = F(x,\alpha) \equiv \begin{bmatrix} f_1(x,\alpha) \\ f_2(x,\alpha) \\ f_3(x,\alpha) \\ f_4(x,\alpha) \end{bmatrix}.$$

II. Discrete Approximation

In the last section, we derived the continuous-time equations of motion for the spacecraft trajectory. Here, we form the discrete approximation we will use to determine feasible trajectories using optimization over a discrete number of variables (i.e., a finite-dimensional space).

Suppose T, the final time is given. We will discretize time into N steps and use a forward-Euler finite-difference approach to approximate the derivatives of all variables involved, i.e.,

$$x_{k+1} \approx x_k + hF(x_k, \alpha_{k+1}),$$

with $h \equiv \frac{T}{N}$ being our finite-difference step size[1].

Assuming h is sufficiently small, we can optimize over the variables at each time-step as opposed to in the continuous-time domain. We are left with an optimization problem over the discrete variables,

$$\min_{\substack{x_1,\ldots,x_N \\ \alpha_1,\ldots,\alpha_N}} \ \frac{1}{2}\left(||r_N - r_f||^2 + ||v_N - v_f||^2 + ||\omega_N - \omega_f||^2\right) \tag{1}$$

subject to

$$x_{k+1} = x_k + hF(x_k, \alpha_{k+1}) \quad k = 0, \ldots, N-1] \tag{2}$$

$$\alpha_k \in [-\pi/2, \pi/2] \quad k = 1, \ldots, N \tag{3}$$

$$x_0 = x_i \equiv [r_0, 0, v_0, \omega_0]^T, \tag{4}$$

where we note we can take θ_0 to be 0 without loss of generality.

[1]The unfortunate subscript of α is due to the fact that our α vector is offset by one index for convenience in MATLAB

A. *Penalty Function and Gradient*

The equality constraint, Eq. (2), is unfortunately nonlinear, so we choose to treat it with a penalty function. As such, we will add a term $\rho/2E$ to our objective function, where ρ is the penalty parameter. Here we derive E. Once again, we define $h = T/N$. Using forward Euler approximations for all derivatives, we define state estimates:

$$\hat{r}_{k+1} = r_k + hv_k, \tag{5}$$

$$\hat{\theta}_{k+1} = \theta_k + h\omega_k, \tag{6}$$

$$\hat{v}_{k+1} = v_k + h\left(\omega_k^2 r_k + \frac{ar_0^2\cos^3(\alpha_{k+1}) - \mu}{r_k^2}\right), \tag{7}$$

$$\hat{\omega}_{k+1} = \omega_k + h\left(\frac{-2\omega_k v_k}{r_k} + \frac{ar_0^2\cos^2(\alpha_{k+1})\sin(\alpha_{k+1})}{r_k^3}\right), \tag{8}$$

where the hatted variables are predictions of what the variable value should be if it were actually given by the discrete state evolution equations. We then define our penalty function (or the *state error*) to be

$$E \equiv \frac{1}{2}\left(\sum(r_{k+1} - \hat{r}_{k+1})^2 + \sum(\theta_{k+1} - \hat{\theta}_{k+1})^2 + \sum(v_{k+1} - \hat{v}_{k+1})^2 + \sum(\omega_{k+1} - \hat{\omega}_{k+1})^2\right). \tag{9}$$

To use a quasi-Newton approximation routine with this penalty function E, the gradient of E with respect to each of the state variables at each time (as well as the control variable at each time) is necessary. This is obtained through judicious use of the chain rule in differentiation, though the results are long so we relegate them to Appendix A.

B. *Final Discrete Formulation*

Employing our penalty function, we rewrite the equation we want to solve, Eq. (1), in the form we will actually solve it:

$$\min_{\substack{x_1,\ldots,x_N \\ \alpha_1,\ldots,\alpha_N}} \frac{1}{2}\left(||r_N - r_f||^2 + ||v_N - v_f||^2 + ||\omega_N - \omega_f||^2\right) + \rho_t E \tag{10}$$

$$\begin{aligned}&\text{subject to}\\ &\alpha_k \in [-\pi/2, \pi/2] \quad k = 1, \ldots, N\\ &x_0 = x_i \equiv [r_0, 0, v_0, \omega_0]^T.\end{aligned}$$

For each choice of final time, T, we will solve this for a sequence of increasing ρ_t until the state evolution constraint is satisfied to sufficient precision.

III. Numerical Solution Method

To solve Eq. (10), we will use an active-set method on the box constraints on α_k and solve each equality-constrained subproblem with a BFGS quasi-Newton method. There are three main parts to the numerical solution scheme, as explained below. Note that each subsection uses notation standard to the field, which in some cases overloads notation used outside those subsections.

A. *Line Search*

The line search implemented in this project is based off the one described in [5]. The basic idea is, given a function to minimize f, its gradient g, an initial point x, and a descent direction p, find a step size $\alpha \leq \alpha_{max}$ to satisfy the strong Wolfe conditions[2]:

$$f(x + \alpha p) \leq f(x) + c_1\alpha g(x)^T p,$$

$$|g(x + \alpha p)^T p| \leq c_2|g(x)^T p|.$$

The first of these conditions is the sufficient decrease condition and the second is the curvature condition. We use what Nocedal and Wright in [5] describe as standard choices: $c_1 = 1 \times 10^{-4}$ and $c_2 = 0.9$.

To find a point that satisfies these conditions, we first check $\alpha = 1$ (if $1 \leq \alpha_{max}$, else we check α_{max}). If this point satisfies both conditions, we use it. Else, we use a bracketing scheme to determine an interval that contains an α which satisfies both conditions and then perform bisection in that interval. The scheme is not complicated and we point the reader that desires more exposition to [5].

[2]Note that this α is the step length, not to be confused with the solar sail angle from before.

Fig. 1. Error plot for an example test of the line search implementation on the random linear system.

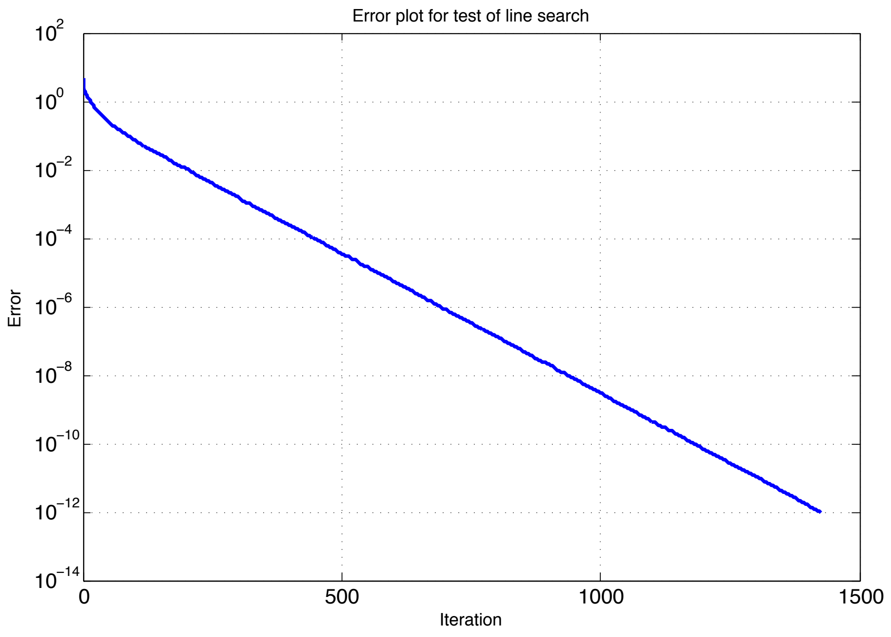
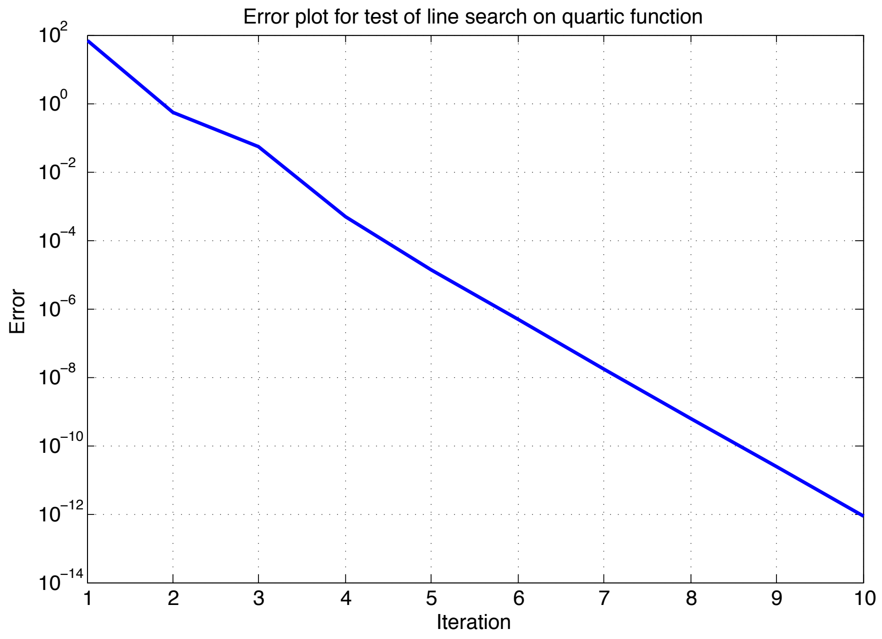


Fig. 2. Error plot for an example test of the line search implementation on the quartic function.



1) Verification: To ensure that our line search implementation works, we test on some randomly generated example problems using gradient descent. To do this, we generate random matrices of the form $A = I + G$ where G is a Gaussian random matrix, as well as random vectors b. Then, a simple gradient descent method was used with the line search to find the minimizer of $||Ax - b||$. In all tests, a linear convergence rate was observed as expected. An example result can be seen in Fig. 1 using vectors of size 20.

The above checks for the case of a quadratic function, which is not a difficult case because the curvature condition should not really come into play. As such, we also offer one more toy test – minimizing the function $f(x) = x^4 - x$ over $\mathbb{R}$. An error plot of the gradient descent solution with the linesearch versus the true solution is given in Fig. 2 where we again see linear convergence.

B. Active-Set Quasi-Newton

The next level of the solution scheme is the quasi-Newton method, which uses the line search as a subroutine. The basic loop for this is as follows:

```
while true do
    if the function is sufficiently small then
        Terminate algorithm, it has converged
    else if the norm of the reduced gradient is sufficiently small then
        Calculate the Lagrange multiplier estimates $\hat{\lambda}$ for the active constraints
        if any $\hat{\lambda}$ is negative then
            Remove the constraint corresponding to the most negative multiplier from the active set
        else
            Terminate algorithm, it has converged
        end if
    else
        Calculate step direction $p$ that is in the null space of the active set
        Calculate maximum feasible step $\alpha_{max}$ in direction $p$
        Use line search to find step size $\alpha$
        $x^{k+1} \leftarrow x^k + \alpha p$
        if $s = \alpha p$ is not too close to being orthogonal to $y = g^{k+1} - g^k$ then
            Update Hessian approximation $B$ with BFGS update
        end if
        if $\alpha = \alpha_{max}$ then
            Add any constraints that were hit to the working set
        end if
    end if
end while
```

The BFGS update procedure is standard and can be found in [5]. In fact, this entire active-set quasi-Newton method is a pretty standard formulation – what is interesting about this case is the specific details in some of the pseudo-code above, which we describe now.

In general for a quasi-Newton method combined with an active-set strategy, it is necessary to compute the reduced gradient and the reduced Hessian, that is, the gradient and Hessian restricted to the subspace orthogonal to the matrix of active constraints. This can be arduous, however, with box constraints on one-dimensional variables such as our control angle, we can note two things[3]. First, each active constraint is of the form $x_i = d$ for d being either the upper or lower bound on x_i. This means that the corresponding row of the constraint matrix is $e_i^T x = d$, where e_i is the i-th coordinate vector. Second, only one constraint can be active for any given x_i, since x_i can't be at its upper and lower bound simultaneously. Therefore, vectors that are in the null space of the active constraint matrix A are simply vectors which have a zero in the i-th component for each x_i at one of its bounds.

With these facts, forming the reduced Hessian and reduced gradient is simple. If B is the full Hessian and `free` is an indexing vector corresponding to the variables not at their constraints, then the reduced Hessian is simply $B(\texttt{free}, \texttt{free})$ (using MATLAB notation) and similarly with the reduced gradient.

Additionally, computing the Lagrange multiplier estimates is simple as well with these box constraints, since if A is the active constraint matrix then the solution of

$$\hat{\lambda} = \arg\min_{\lambda} ||A^T \lambda - g(\texttt{fixed})||_2$$

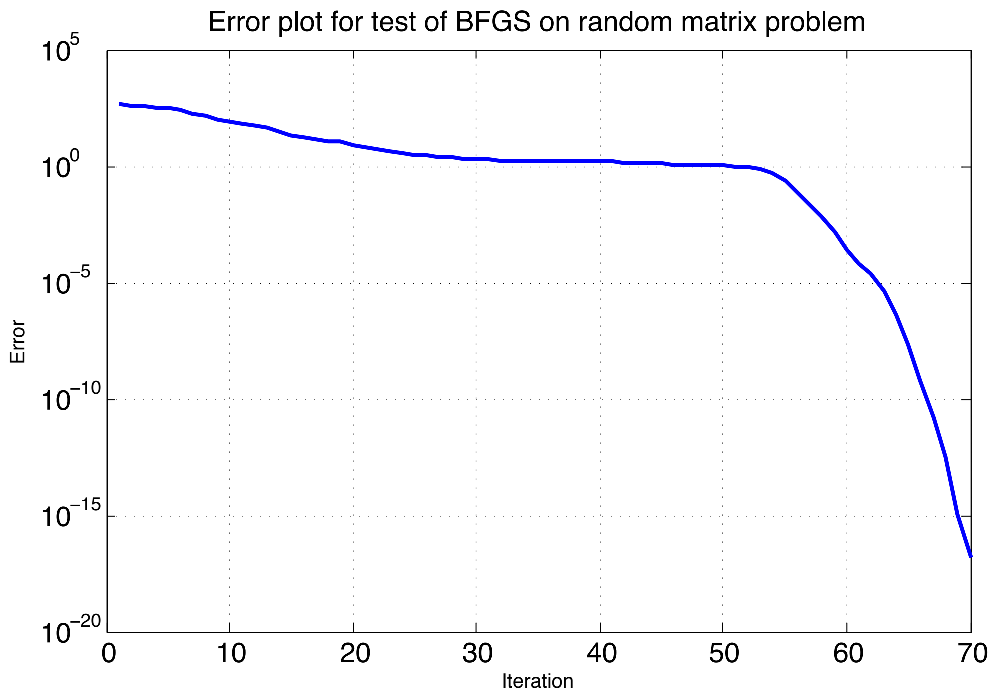
is given by $\hat{\lambda}_i = -g_i$ for x_i at its upper limit and $\hat{\lambda}_i = g_i$ for x_i at its lower limit[4].

1) Verification: To verify that the BFGS method is working correctly, we pose a similar problem to that used to test the line search method. As before, we generate a random matrix of the form $A = I + G$ where G is Gaussian as well as a Gaussian right-hand side, b. Then, we use the BFGS method to minimize $||Ax - b||_2^2$ with the lower and upper boundaries of the box constraints set to sufficeintly large values.

[3]These points were brought up by Tomas at office hours

[4]Here, `fixed` is simply the complementary set of `free` – the variables which are at their constraints

Fig. 3. Error plot for an example verification test of the BFGS method with box constraints on a random linear system of size 50×50.



An example convergence plot of this test for a matrix of size 50×50 can be seen in Fig. 3. We see in this plot that the convergence is not initially very good, but asymptotically the the log plot does seem to show superlinear convergence, in the sense that the curvature of the plot is concave on the log scale. This is good evidence that the quasi-Newton method is doing what it should and outperforming gradient descent (which would show linear convergence).

Similar tests to this were run with box constraints that actually restricted the solution of the problem to be different from the unconstrained solution. These also converged quickly and exhibited superlinear convergence after a certain number of iterations, as we would expect.

C. Outer Control Loop

With the working BFGS, we can minimize nonlinear functions subject to box constraints on the optimization variables. Solving the optimal control problem for solar sails formulated in this paper, it is necessary to solve a sequence of minimization problems as we vary parameters. In this section, we describe the logic used to determine the sequence of problems solved and thus to give the final solution (minimum time necessary to bring the spacecraft between orbits for a given characteristic acceleration).

Assuming we're given a final time T, we can generate a simulated trajectory by fixing the control angle α to some value for all time and then simply using the discretized state evolution equations directly. This will have a penalty function value that is numerically zero (by construction) but will probably be far from the desired final state. So, we use this as our initial guess to BFGS to get a better solution. However, this will probably cause the penalty function value to increase, so we must then loop and continually increase the penalty parameter until that function value is sufficiently small (i.e., the Newton's laws are satisfied to within a suitable small tolerance). The loop to describe this is:

```
$x_0 \leftarrow$ trajectory known to follow the discrete Newton's laws with final time $T$ given
$\rho \leftarrow 1$
**while** objective function is not sufficiently small and penalty function is not sufficiently small **do**
    $x \leftarrow$ result of running BFGS on $\frac{1}{2}\left(||r_N - r_f||^2 + ||v_N - v_f||^2 + ||\omega_N - \omega_f||^2\right) + \rho E$ with initial guess $x_0$
    $x_0 \leftarrow x$
    $\rho \leftarrow 5\rho$
**end while**
```

If a solution exists for the given T to within suitable accuracy, we can try decreasing T and running the above loop again. If not, then we must increase T until we find a solution that does work. By using a binary search on T, we can narrow down the minimum T needed to get a solution that both reaches the final state and follows Newton's laws to within sufficient accuracy.

IV. Simulation Results

In this section we give the final results for different values of the characteristic acceleration a. For all simulations, we used $N = 150$ timesteps and a finite difference steplength of T/N, which in most cases corresponds to about 3 to 4 days. In each simulation, we look for the minimum time of trajectory for which we can get the objective function
$\frac{1}{2}\left(||r_N - r_f||^2 + ||v_N - v_f||^2 + ||\omega_N - \omega_f||^2\right)$
below 1×10^{-5} and the penalty function E below 1×10^{-7}. We note that the tolerances should greatly influence the estimate of the minimum time, so different tolerances could give different results. The project assignment sheet did not recommend specific tolerances but we believe these to be reasonable.

We look at three different choices of characteristic acceleration below: $a = 2$, $a = 1$, and $a = 1.5$. We know that $a = 2$ should take the least amount of time since the speed of controlled motion of the spacecraft is limited by the acceleration possible with the solar sail. Thus, it doesn't make much sense to try to pick an "optimal" value of a or anything like that. Instead, we simply look qualitatively at the solutions below.

The main result here is the final minimum times for each a, so we describe only the case $a = 2$ in detail and comment lightly in the other cases in differences between those cases and $a = 2$.

A. *Characteristic Acceleration* $a = 2$

1) To Mars: For a characteristic acceleration of $a = 2$ in the scenario where we start at Earth's orbit and end at Mars's orbit, the minimum time that we were able to find to get below the tolerances mentioned before was $T = 2.65\bar{s} \approx 307$ days. In Fig. 4, we see that all the state variables exhibit smooth trajectories, as we'd hope for a physical system.

It's interesting to see that the optimal control involves keeping the control angle at or near $\pi/2$ for a while, which corresponds to no force from the solar sail. Thus, the best control seems to be "disrupt the current orbit, let the spacecraft drift towards the new orbit, then ease in with some corrective force". This would seem to make some sense from an aerodynamics perspective.

It's somwhat odd that the angular velocity in the optimal trajectory first increasese slightly and then overshoots the desired angular velocity. It would seem that it would be more optimal to come in more smoothly such as the trajectory of the radius. This may be a feature of the finite differencing, or perhaps from not solving to a high enough tolerance. In fact, this may actually be the behavior of an optimal solution – assessing this requires domain-specific knowledge.

2) To Earth: Fig. 5 shows the reverse case, where we start at Mars and head to Earth. The minimum time in this case was $T = 2.5\bar{s} \approx 289$ days. It is interesting to note that this is less than the time needed to go from Earth to Mars, but this makes some sense – when moving closer to the sun, gravity is assisting the motion of the spacecraft, whereas when moving further from the sun we must fight gravity.

As would be expected, the optimal control angle stays negative when moving back to Earth. A feature of the control for this direction that we didn't see before, however, is the inflection at roughly 260 days. This may be because we started with an initial guess that was closer to a local minimum with different characteristics than in the Earth-to-Mars case, or it may simply reflect that the problems are not symmetric as mentioned before – going from Mars to Earth is fundamentally different.

B. *Characteristic Acceleration* $a = 1$

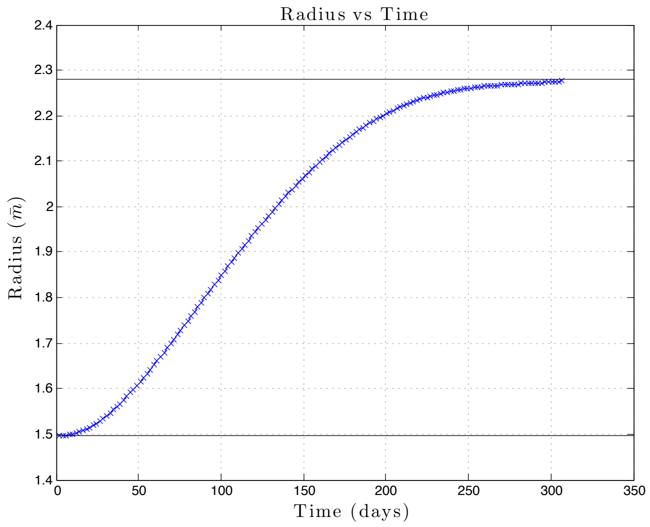
1) To Mars: Fig. 6 shows the case with the minimum characteristic acceleration tried, $a = 1$ For this, the minimum time was $T = 3.4\bar{s} \approx 394$ days. The plots look very similar to the case $a = 2$, but we note that the optimal control deviates less from its average value (α "wiggles" at the same places, but the amplitude is not as great), which is interesting.

2) To Earth: Going back to Earth with $a = 1$ can be seen in Fig. 7. This took $T = 2.8\bar{s} \approx 324$ days and looks largely the same as the $a = 2$ case.

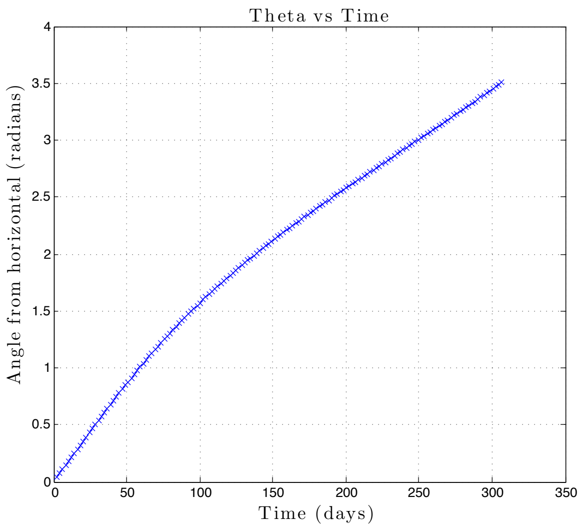
C. *Characteristic Acceleration* $a = 1.5$

1) To Mars: Once again, $a = 1.5$ looks pretty much the same as $a = 2$ and took $T = 2.8\bar{s} \approx 324$ days. Results can be seen in Fig. 8. Whereas we saw that $a = 1$ was not as extreme in its control as $a = 2$, $a = 1.5$ is once again more pronounced.

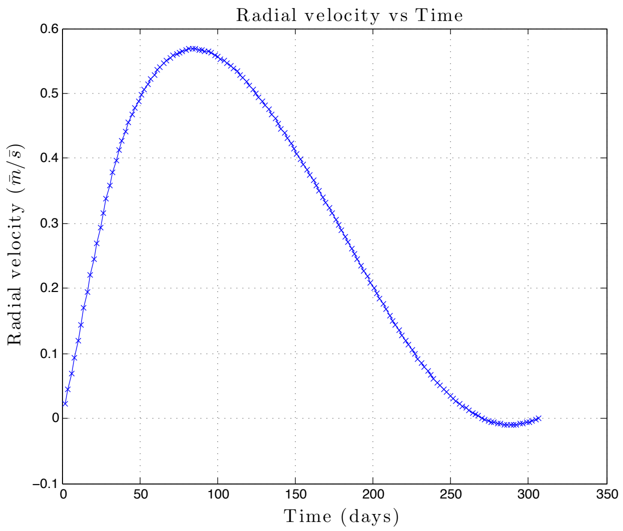
Fig. 4. Trajectory to Mars for $a = 2$



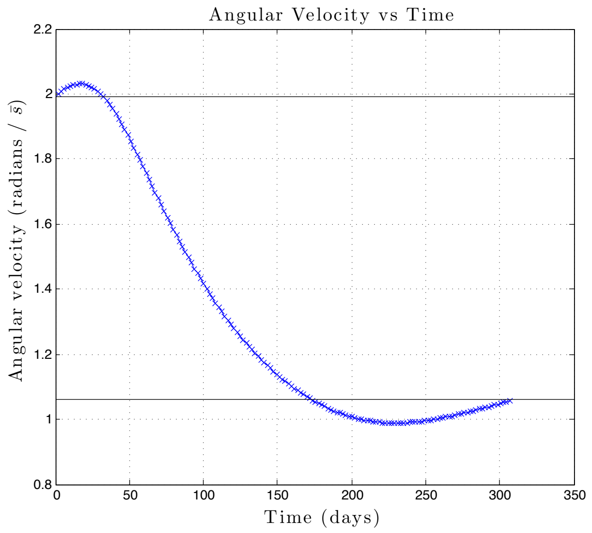
(a) Radius versus time



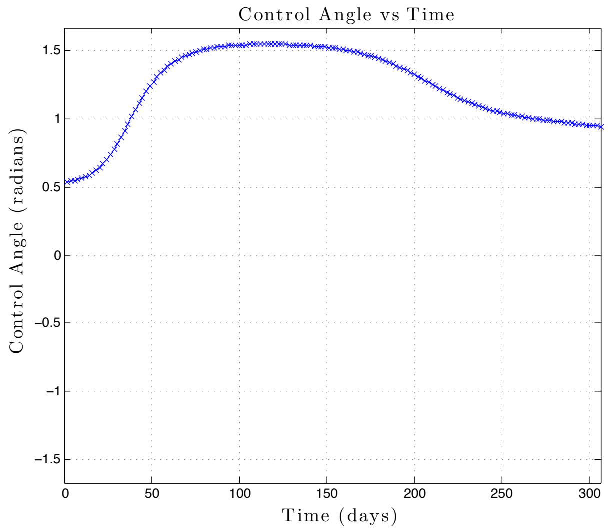
(b) Angle versus time



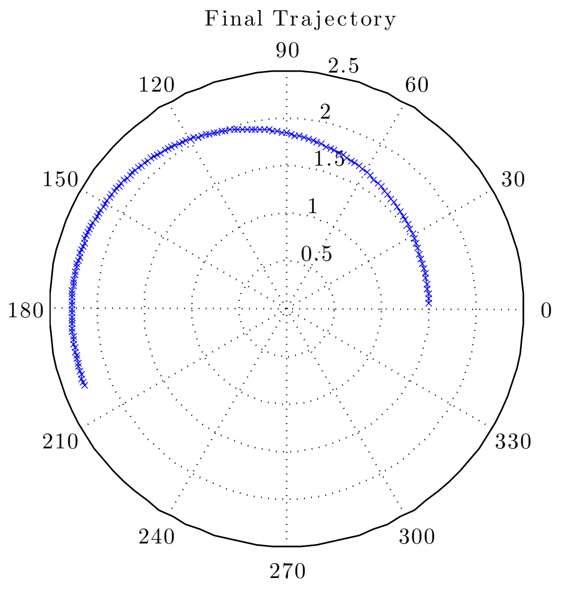
(c) Velocity versus time



(d) Angular velocity versus time

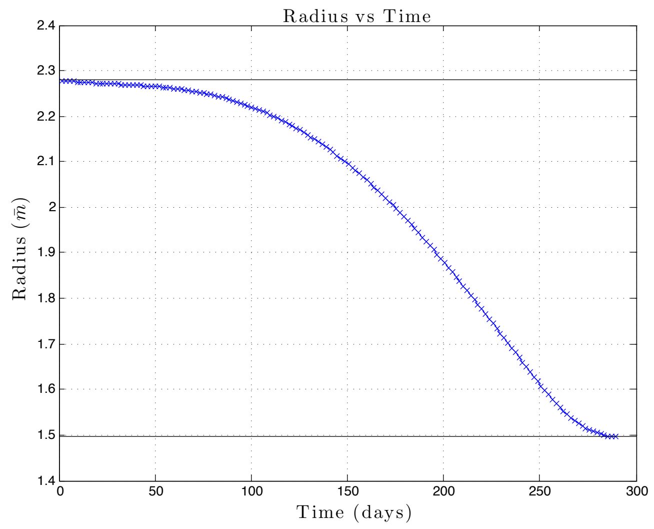


(e) Control angle versus time

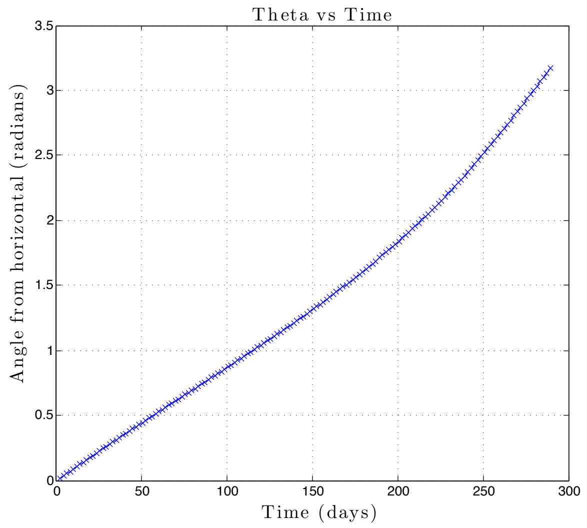


(f) Final trajectory

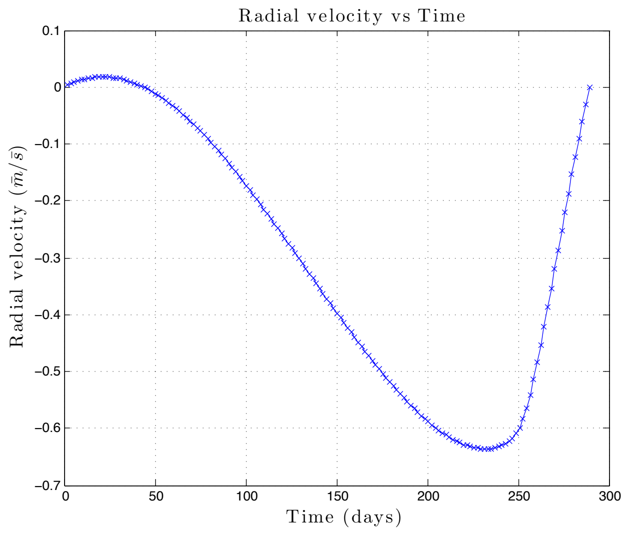
Fig. 5. Trajectory to Earth for $a = 2$



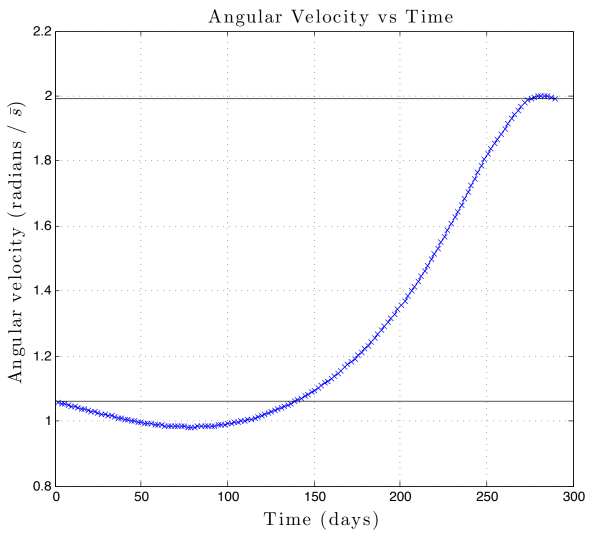
(a) Radius versus time



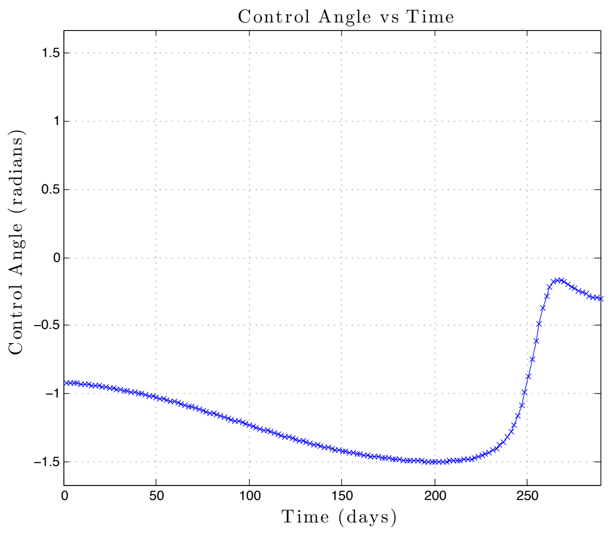
(b) Angle versus time



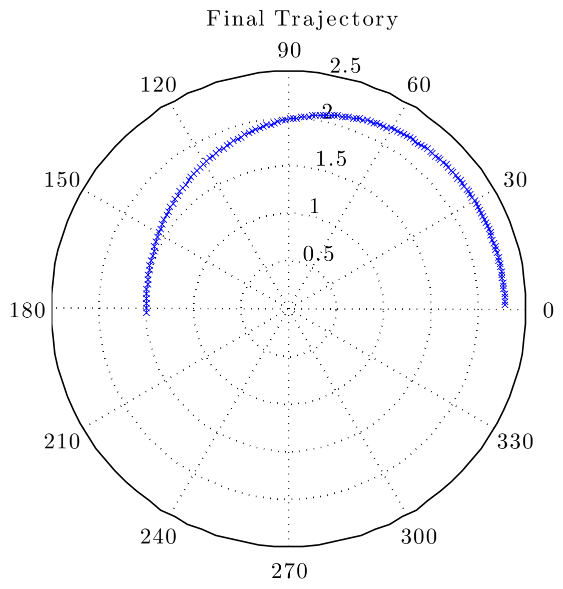
(c) Velocity versus time



(d) Angular velocity versus time

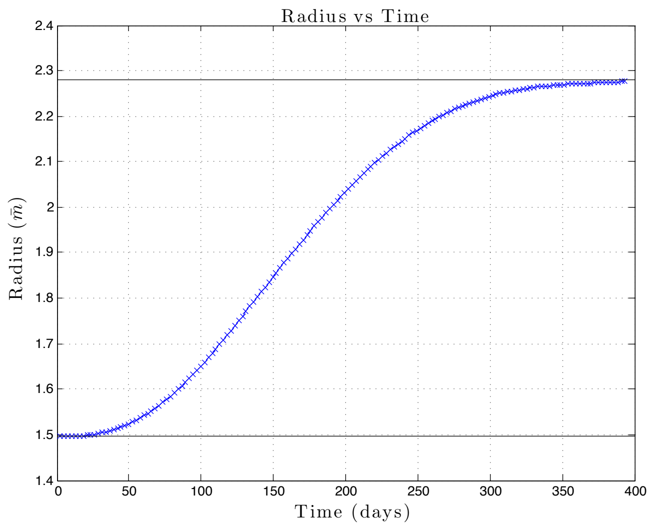


(e) Control angle versus time

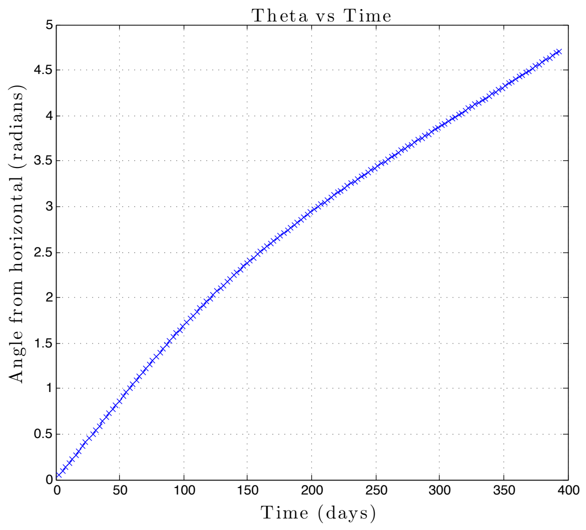


(f) Final trajectory

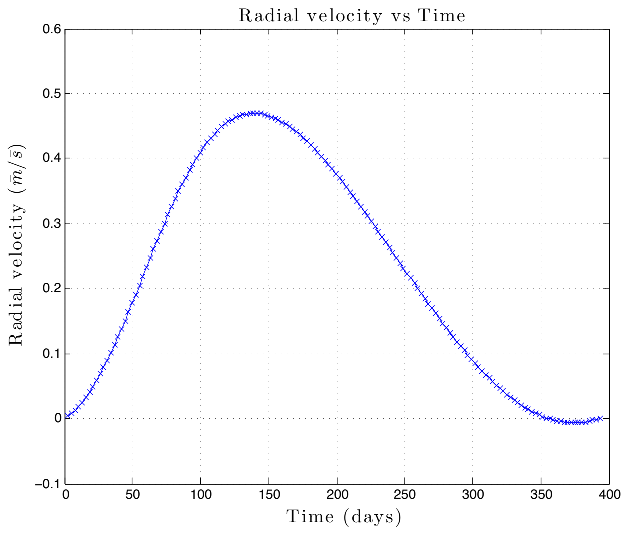
Fig. 6. Trajectory to Mars for $a = 1$



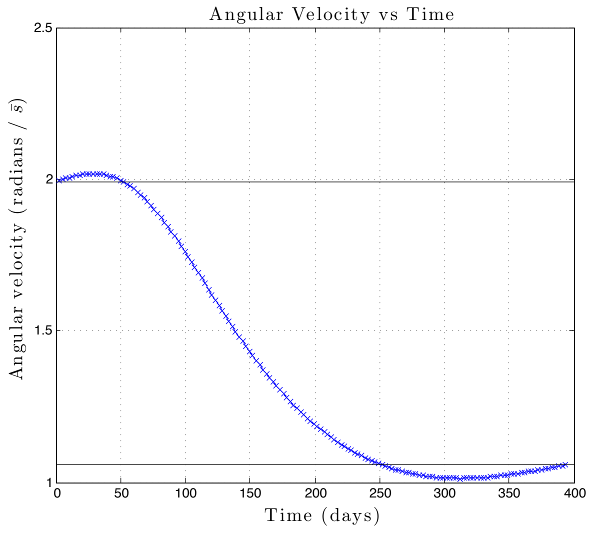
(a) Radius versus time



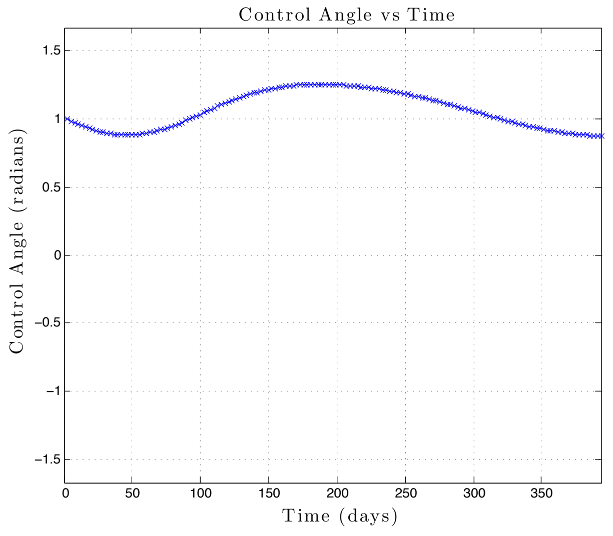
(b) Angle versus time



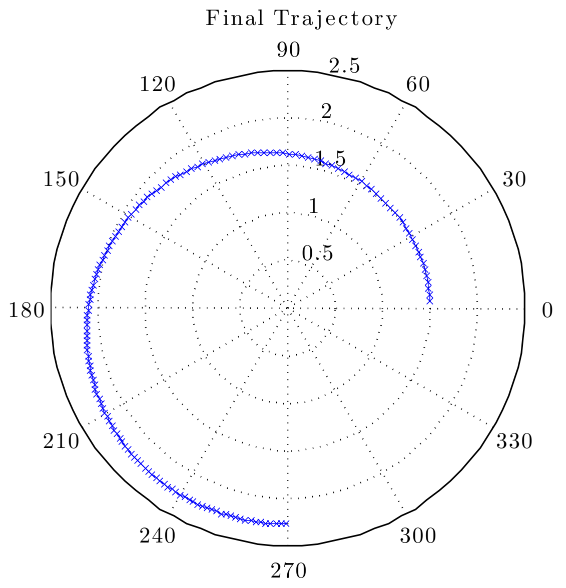
(c) Velocity versus time



(d) Angular velocity versus time

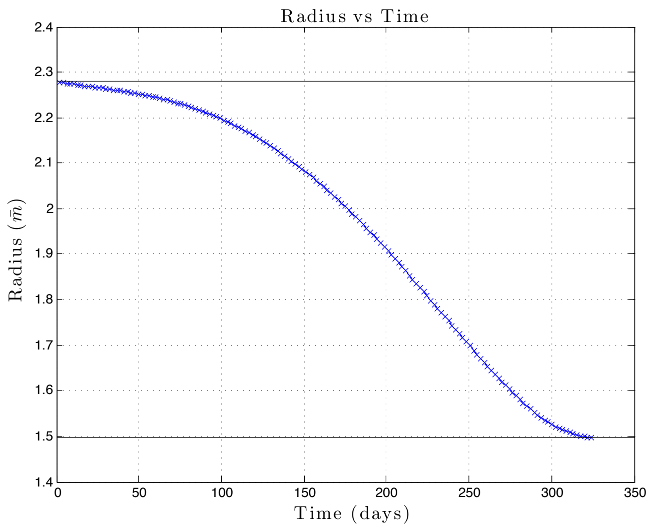


(e) Control angle versus time

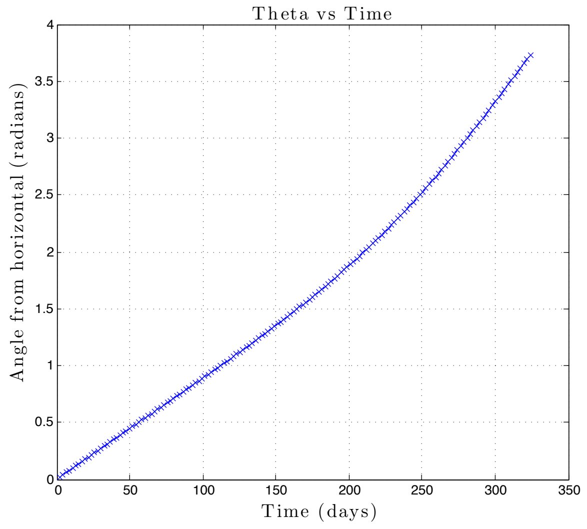


(f) Final trajectory

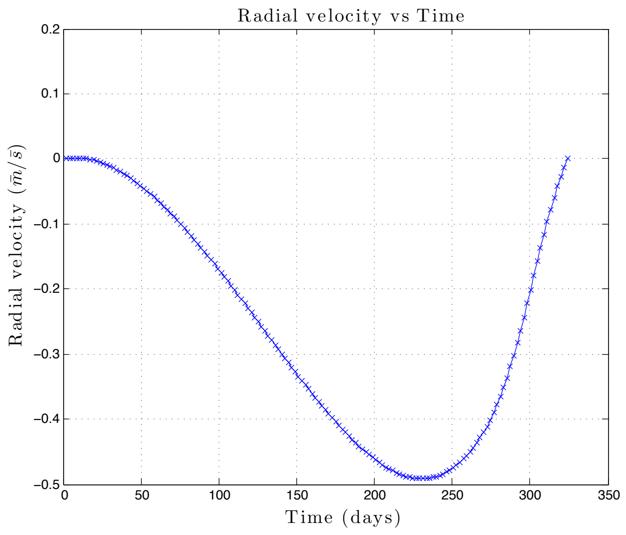
Fig. 7. Trajectory to Earth for $a = 1$



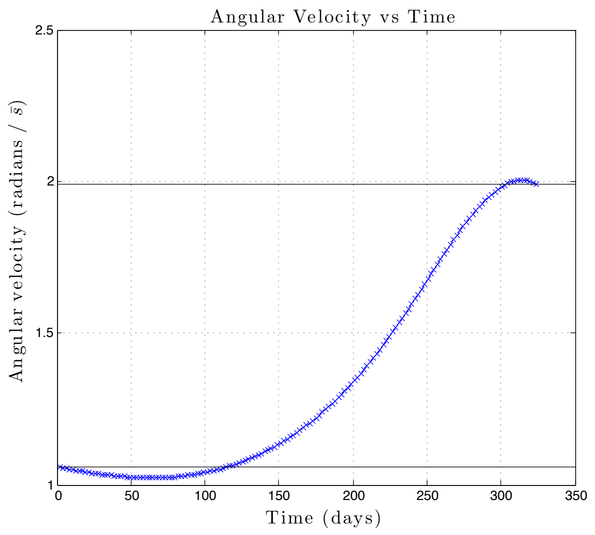
(a) Radius versus time



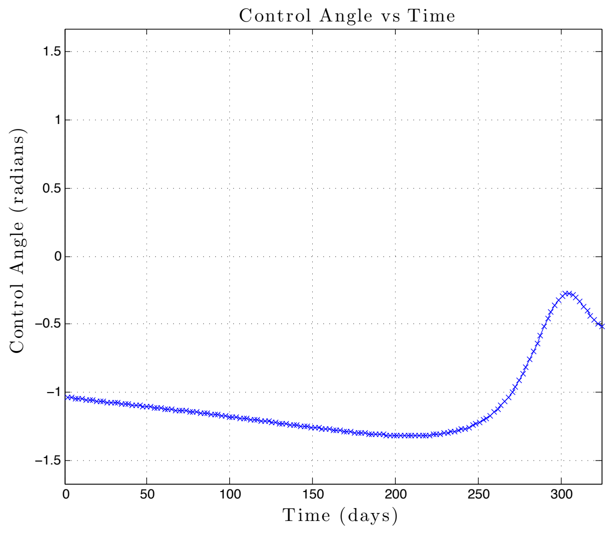
(b) Angle versus time



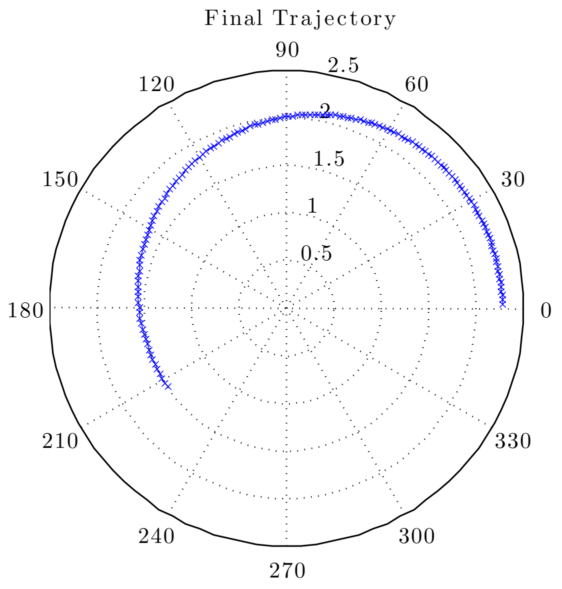
(c) Velocity versus time



(d) Angular velocity versus time

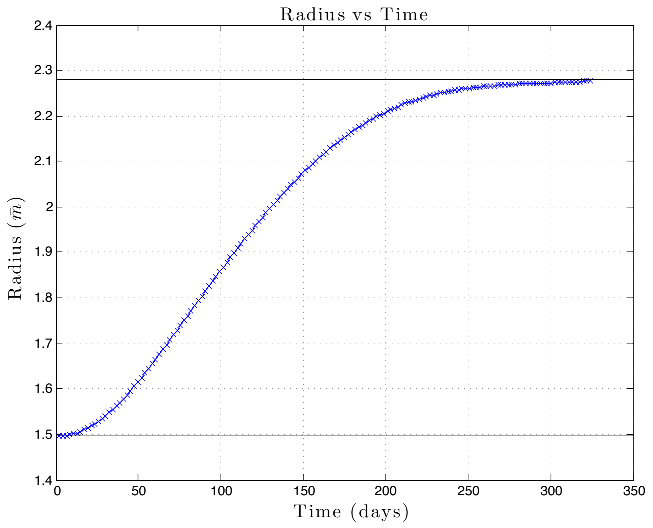


(e) Control angle versus time

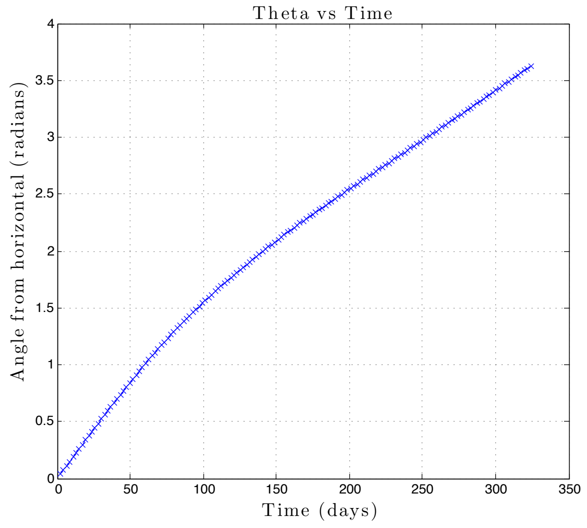


(f) Final trajectory

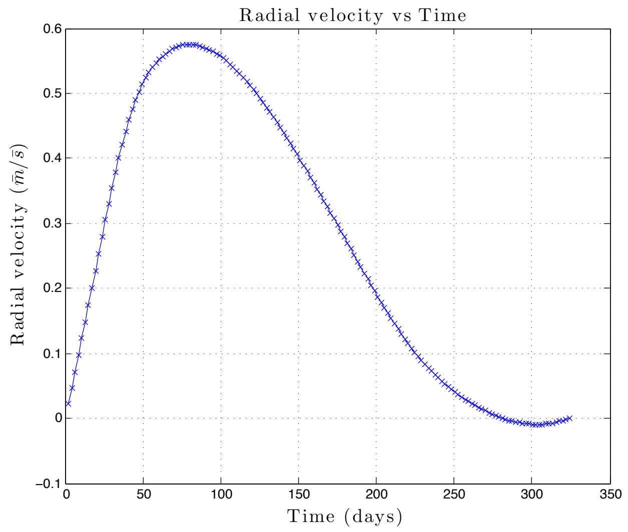
Fig. 8. Trajectory to Mars for $a = 1.5$



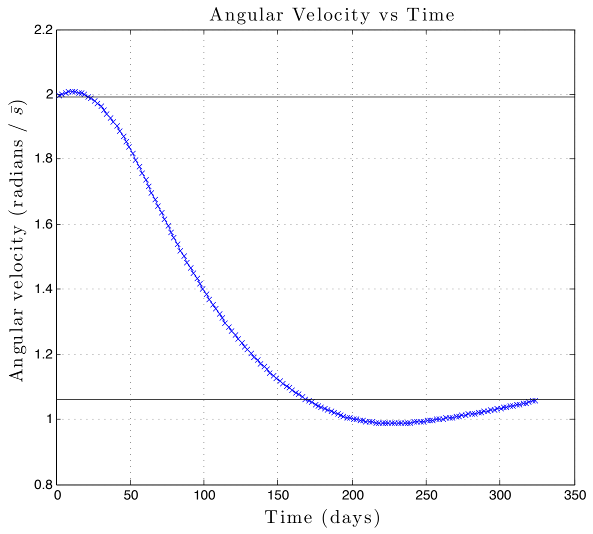
(a) Radius versus time



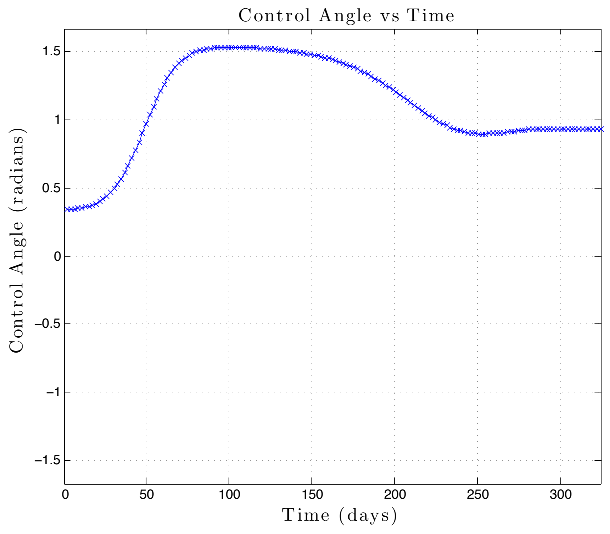
(b) Angle versus time



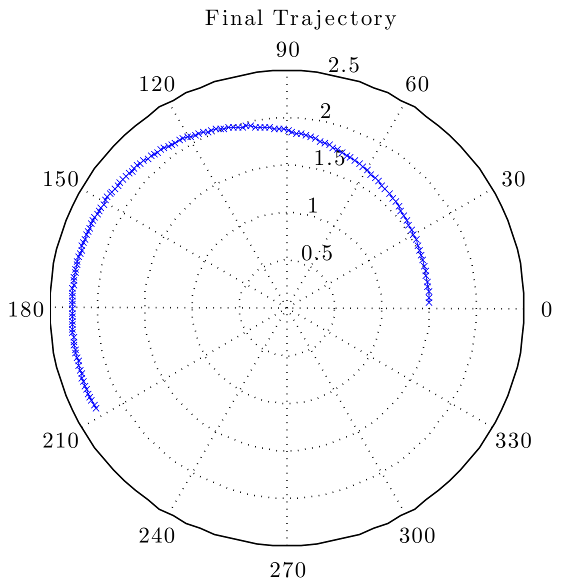
(c) Velocity versus time



(d) Angular velocity versus time

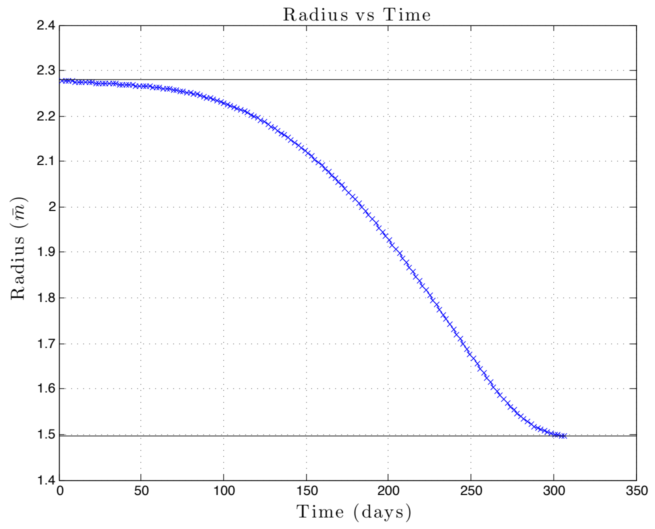


(e) Control angle versus time

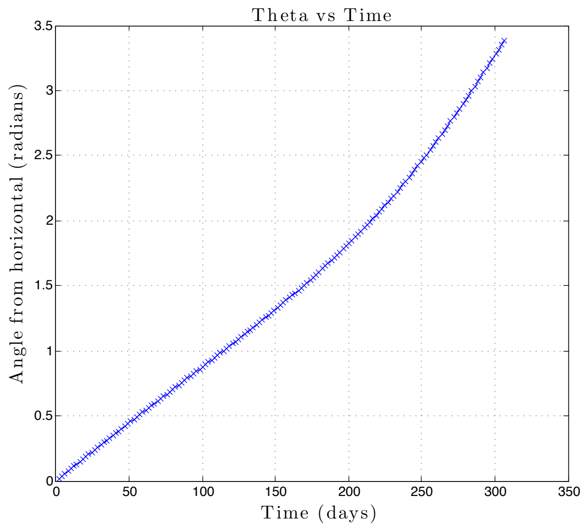


(f) Final trajectory

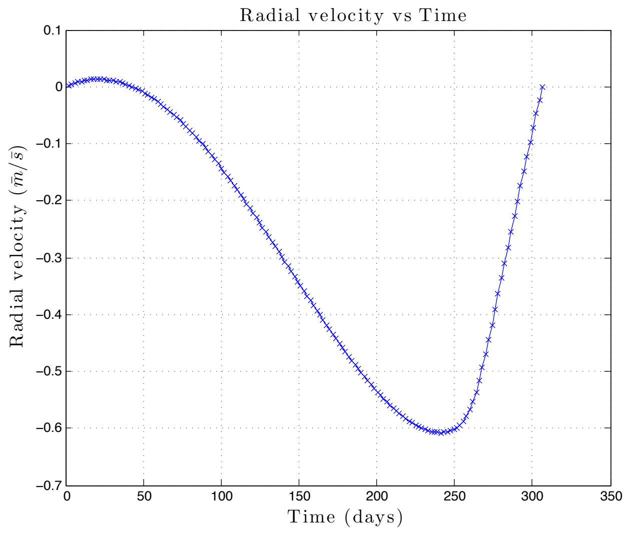
Fig. 9. Trajectory to Earth for $a = 1.5$



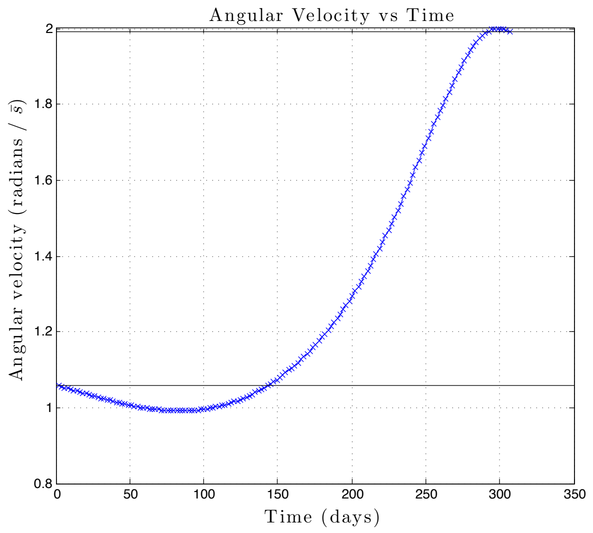
(a) Radius versus time



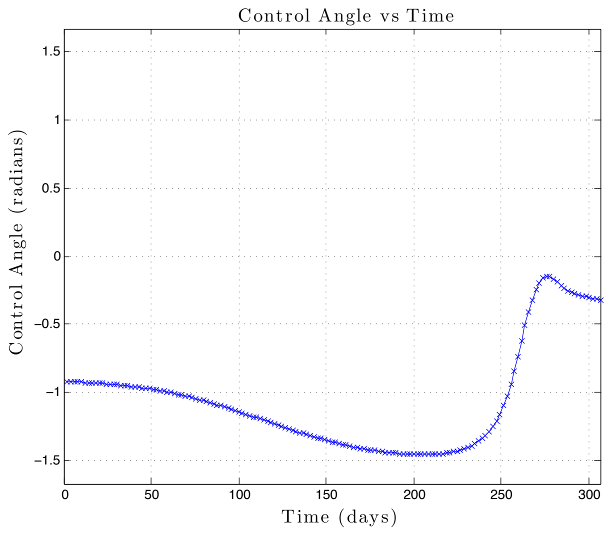
(b) Angle versus time



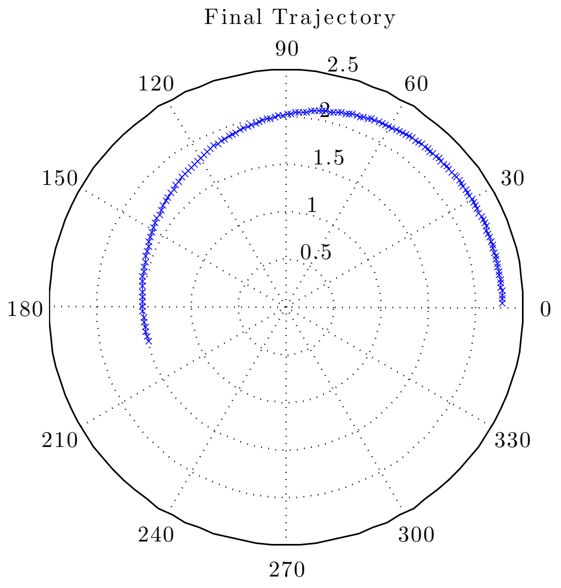
(c) Velocity versus time



(d) Angular velocity versus time



(e) Control angle versus time



(f) Final trajectory

2) To Earth: Fig. 9 shows the results of getting back to Earth with $a = 1.5$, which took about $T = 2.65\bar{s} \approx 307$ days. There is nothing fundamentally different about these figures.

V. Conclusion

It is interesting to see that in many of the state variable plots, it does not appear that the final position is necessarily "stable" in the sense that, whereas the final velocity may be 0, for example, extrapolation would seem to imply that the spacecraft is simply "passing through" a velocity of zero. However, it is important to note that we can only expect the state variables to continue their smooth trajectories if the control variable α continues its smooth trajectory, and clearly once we reach the desired final state we will set $\alpha = \pi/2$ immediately and that may be a discontinuous change (as it is in many of our simulations).

We note that there did seem to be issues with the numerical solution scheme when the penalty parameter was increased too drastically or the final time was grossly insufficient to reach the final state. In particular, we observed that for poor parameter choices convergence was very slow and could lead to problems in the line search subroutine. It is known that increasing the penalty parameter impacts the conditioning of the Hessian matrix, which could be responsible for some part of the observed problems. The main problem, however, seemed to be that with insufficient time to reach the final state the optimization problem becomes in some sense "harder", and we frequently saw very flat gradients in the line search method which led to numerical errors in calculating the step length. To remedy this, it would be necessary to use computational techniques that are more robust to numerical error than simply directly computing the expressions in the Wolfe conditions.

References

[1] J. Arthur E. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Halsted Press, Taylor & Francis, 1975.

[2] C. McInnes, *Solar Sailing: Technology, Dynamics and Mission Applications*, Springer Praxis Books / Astronomy and Planetary Sciences, Springer, 2004.

[3] D. Naidu, *Optimal Control Systems*, Electrical Engineering Textbook Series, CRC PressINC, 2003.

[4] I. Newton, *Philosophiae naturalis principia mathematica*, J. Societatis Regiae ac Typis J. Streater, 1687.

[5] J. Nocedal and S. Wright, *Numerical Optimization*, Springer Series in Operations Research Series, Springer-Verlag GmbH, 1999.

[6] R. Powers, *Solar Sail Optimal Orbital Transfers to Earth Synchronous Stable Orbits*, University of Illinois at Urbana-Champaign, 1999.

Appendix A
Gradient of the Penalty Function

The chain rule gives the derivative of the penalty function Eq. (9) with respect to each optimization variable at each time, and thus gives the full gradient once we vectorize all these derivatives. For completeness, we include these equations here. First, by the chain rule, we have

$$\begin{aligned}
\frac{\partial E}{\partial r_k} &= (r_k - \hat{r}_k) - (r_{k+1} - \hat{r}_{k+1})\frac{\partial \hat{r}_{k+1}}{\partial r_k} - (\theta_{k+1} - \hat{\theta}_{k+1})\frac{\partial \hat{\theta}_{k+1}}{\partial r_k} - (v_{k+1} - \hat{v}_{k+1})\frac{\partial \hat{v}_{k+1}}{\partial r_k} - (\omega_{k+1} - \hat{\omega}_{k+1})\frac{\partial \hat{\omega}_{k+1}}{\partial r_k},\\
\frac{\partial E}{\partial \theta_k} &= (\theta_k - \hat{\theta}_k) - (r_{k+1} - \hat{r}_{k+1})\frac{\partial \hat{r}_{k+1}}{\partial \theta_k} - (\theta_{k+1} - \hat{\theta}_{k+1})\frac{\partial \hat{\theta}_{k+1}}{\partial \theta_k} - (v_{k+1} - \hat{v}_{k+1})\frac{\partial \hat{v}_{k+1}}{\partial \theta_k} - (\omega_{k+1} - \hat{\omega}_{k+1})\frac{\partial \hat{\omega}_{k+1}}{\partial \theta_k},\\
\frac{\partial E}{\partial v_k} &= (v_k - \hat{v}_k) - (r_{k+1} - \hat{r}_{k+1})\frac{\partial \hat{r}_{k+1}}{\partial v_k} - (\theta_{k+1} - \hat{\theta}_{k+1})\frac{\partial \hat{\theta}_{k+1}}{\partial v_k} - (v_{k+1} - \hat{v}_{k+1})\frac{\partial \hat{v}_{k+1}}{\partial v_k} - (\omega_{k+1} - \hat{\omega}_{k+1})\frac{\partial \hat{\omega}_{k+1}}{\partial v_k},\\
\frac{\partial E}{\partial \omega_k} &= (\omega_k - \hat{\omega}_k) - (r_{k+1} - \hat{r}_{k+1})\frac{\partial \hat{r}_{k+1}}{\partial \omega_k} - (\theta_{k+1} - \hat{\theta}_{k+1})\frac{\partial \hat{\theta}_{k+1}}{\partial \omega_k} - (v_{k+1} - \hat{v}_{k+1})\frac{\partial \hat{v}_{k+1}}{\partial \omega_k} - (\omega_{k+1} - \hat{\omega}_{k+1})\frac{\partial \hat{\omega}_{k+1}}{\partial \omega_k},\\
\frac{\partial E}{\partial \alpha_{k+1}} &= -(r_{k+1} - \hat{r}_{k+1})\frac{\partial \hat{r}_{k+1}}{\partial \alpha_{k+1}} - (\theta_{k+1} - \hat{\theta}_{k+1})\frac{\partial \hat{\theta}_{k+1}}{\partial \alpha_{k+1}} - (v_{k+1} - \hat{v}_{k+1})\frac{\partial \hat{v}_{k+1}}{\partial \alpha_{k+1}} - (\omega_{k+1} - \hat{\omega}_{k+1})\frac{\partial \hat{\omega}_{k+1}}{\partial \alpha_{k+1}}.
\end{aligned}$$

So it simply remains to find the derivatives of the state estimates with respect to the optimization variables. By differentiating Eq. (5) through Eq. (8), we obtain the following unfortunate set of equations:

$$\frac{\partial \hat{r}_{k+1}}{\partial r_k} = 1$$
$$\frac{\partial \hat{\theta}_{k+1}}{\partial r_k} = 0$$
$$\frac{\partial \hat{v}_{k+1}}{\partial r_k} = h\left(\omega_k^2 r_k + 2\frac{-ar_0^2\cos^3(\alpha_{k+1}) + \mu}{r_k^3}\right)$$
$$\frac{\partial \hat{\omega}_{k+1}}{\partial r_k} = h\left(\frac{2\omega_k v_k}{r_k^2} - \frac{3ar_0^2\cos^2(\alpha_{k+1})\sin(\alpha_{k+1})}{r_k^4}\right)$$

$$\frac{\partial \hat{r}_{k+1}}{\partial \theta_k} = 0$$
$$\frac{\partial \hat{\theta}_{k+1}}{\partial \theta_k} = 1$$
$$\frac{\partial \hat{v}_{k+1}}{\partial \theta_k} = 0$$
$$\frac{\partial \hat{\omega}_{k+1}}{\partial \theta_k} = 0$$

$$\frac{\partial \hat{r}_{k+1}}{\partial \omega_k} = 0$$
$$\frac{\partial \hat{\theta}_{k+1}}{\partial \omega_k} = h$$
$$\frac{\partial \hat{v}_{k+1}}{\partial \omega_k} = 2h\omega_k r_k$$
$$\frac{\partial \hat{\omega}_{k+1}}{\partial \omega_k} = \left(1 - \frac{2hv_k}{r_k}\right)$$

$$\frac{\partial \hat{r}_{k+1}}{\partial v_k} = h$$
$$\frac{\partial \hat{\theta}_{k+1}}{\partial v_k} = 0$$
$$\frac{\partial \hat{v}_{k+1}}{\partial v_k} = 1$$
$$\frac{\partial \hat{\omega}_{k+1}}{\partial v_k} = -\frac{2h\omega_k}{r_k}$$

$$\frac{\partial \hat{r}_{k+1}}{\partial \alpha_{k+1}} = 0$$
$$\frac{\partial \hat{\theta}_{k+1}}{\partial \alpha_{k+1}} = 0$$
$$\frac{\partial \hat{v}_{k+1}}{\partial \alpha_{k+1}} = \frac{-3har_0^2\cos^2(\alpha_{k+1})\sin(\alpha_{k+1})}{r_k^2}$$
$$\frac{\partial \hat{\omega}_{k+1}}{\partial \alpha_{k+1}} = \frac{har_0^2}{r_k^3}\left[\cos^3(\alpha_{k+1}) - 2\sin^2(\alpha_{k+1})\cos(\alpha_{k+1})\right]$$

Summing the derivatives corresponding to differentiation with respect to the same optimization variables and vectorizing the result, we obtain our gradient.