IAA-AAS-DyCoSS1-07-08

HYBRID METHODS FOR DETERMINING TIME-OPTIMAL, CONSTRAINED SPACECRAFT REORIENTATION MANEUVERS

Robert G. Melton*

Time-optimal spacecraft slewing maneuvers with path constraints are difficult to compute even with direct methods. This paper examines the use of a hybrid, two-stage approach, in which a heuristic method provides a rough estimate of the solution, and that serves as the input to a pseudospectral optimizer. Three heuristic methods are examined for the first stage: particle swarm optimization (PSO), differential evolution (DE), and bacteria foraging optimization (BFO). In this two-stage method, the PSO-pseudospectral combination is approximately three times faster than the pseudospectral method alone, and the BFO-pseudospectral combination is approximately four times faster; however, the DE does not produce an initial estimate that reduces total computation time.

INTRODUCTION

The problem of reorienting a spacecraft in minimum time, often through large angles (so-called *slew maneuvers*) and subject to various constraints, can take a number of forms. For example, the axis normal to the solar panels may be required to lie always within some specified minimum angular distance from the sun-line. In cases where the control authority is low and the slew requires a relatively long time, certain faces of the vehicle may benefit from being kept as far as possible from the sun-line to avoid excessive solar heating. For scientific missions, observational instruments frequently must be kept beyond a specified minimum angular distance from high-intensity light sources to prevent damage.

Before addressing the time-optimal, constrained problem, it is useful to review what is known about the unconstrained problem. In a seminal paper, Bilimoria and Wie[1] considered time-optimal slews for a rigid spacecraft whose mass distribution is spherically symmetric, and which has equal control-torque authority for all three axes. Despite the symmetry of the system, the intuitively obvious time-optimal solution is not a λ-rotation (rotation through a specified angle about an axis fixed in both the body and an external reference frame) about the eigenaxis. Indeed, the fallacy here is that one easily confuses the minimum-angle of rotation problem (i.e. the angle about the eigenaxis) with the minimum-time problem, forgetting the constraints imposed by Euler's equations of rigid-body motion. Bilimoria and Wie found that the time-optimal solution includes precessional motion to achieve a lower time (approximately 10% less) than that obtained

* Professor, Department of Aerospace Engineering, Pennsylvania State University, 229 Hammond Bldg., University Park, PA, 16801.

with an eigenaxis maneuver. Further, they found that the control history is bang-bang, with a switching structure that changes depending upon the magnitude of the angular maneuver (referring here to the final orientation in terms of a fictitious λ-rotation, with associated angle θ): for values of θ less than 72 degrees, the control history was found to contain seven switches between directions of the control torque components; larger values of θ require only five switches.

Several subsequent papers revisited the unconstrained problem, including such modifications as axisymmetric mass distribution and only two-axis control,[2] asymmetric mass distribution,[3] small reorientation angles,[4] and combined time and fuel optimization.[5] Recently, Bai and Junkins[6] reconsidered the original problem (spherically symmetric mass distribution, three equal control torques) and found that at least two locally optimum solutions exist for reorientations of less than 72 deg. (one of which requires only six switches) if the controls are independently limited. Further, they proved that if the total control vector is constrained to have a maximum magnitude (i.e., with the orthogonal control components not necessarily independent), then the time-optimal solution is the eigenaxis maneuver.

Early work on constrained maneuvers focused upon generating feasible solutions[7, 8] but did not attempt to find optimal solutions. The advent of pseudospectral methods and their application to various trajectory optimization problems has made it possible to study problems with challenging constraints.[9] These methods are not fully automatic and may require an intelligent initial guess. The method proposed in this paper is to use a two-step hybrid approach to achieve overall faster performance.

This work uses the Swift spacecraft[10] as an example of a vehicle that must be rapidly reoriented to align two telescopes at a desired astronomical target, namely, a gamma-ray burst. The satellite's Burst Alert Telescope (wide field of view) first detects the gamma-ray burst and the spacecraft then must reorient to allow the x-ray and UV/optical instruments to capture the rapidly fading after-glow of the event. To prevent damage to the these instruments, the slewing motion is constrained to prevent the telescopes' axis from entering established "keep-out" zones, defined as cones with central axes pointing to the Sun, Earth and Moon, with specified half-angles.

PROBLEM STATEMENT

The problem is formulated as a Mayer optimal control problem, with performance index

$$J = t_f \tag{1}$$

where t_f is the final time to be minimized. Euler's equations of rigid-body motion describe the system dynamics

$$\begin{aligned} \dot{\omega}_1 &= \left[M_1 - \omega_2\omega_3\left(I_3 - I_2\right)\right]/I_1 \\ \dot{\omega}_2 &= \left[M_2 - \omega_3\omega_1\left(I_1 - I_3\right)\right]/I_2 \\ \dot{\omega}_3 &= \left[M_3 - \omega_1\omega_2\left(I_2 - I_1\right)\right]/I_3 \end{aligned} \tag{2}$$

These must be augmented with kinematic relationships in order to determine the orientation of the body over time. In this work, a formulation that uses Euler parameters is employed, with the relationship between the Euler parameters ε_i and the angular velocity components ω_j given by

$$\begin{bmatrix} \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \\ \dot{\varepsilon}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 & \varepsilon_1 \\ \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 & \varepsilon_3 \\ -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 & \varepsilon_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{3}$$

The problem to be considered is a rest-to-rest maneuver, consisting of a rotation about the z-axis through an angle of $3\pi/4$ radians

$$\begin{aligned} &\omega_1(0) = \omega_2(0) = \omega_3(0) = 0 \\ &\varepsilon_1(0) = \varepsilon_2(0) = \varepsilon_3(0) = 0,\ \varepsilon_4(0) = 1 \end{aligned} \tag{4a}$$

$$\begin{aligned} &\omega_1(t_f) = \omega_2(t_f) = \omega_3(t_f) = 0 \\ &\varepsilon_1(t_f) = \varepsilon_2(t_f) = 0,\ \varepsilon_3(t_f) = \sin(\frac{3\pi}{8}),\ \varepsilon_4(t_f) = \cos(\frac{3\pi}{8}) \end{aligned} \tag{4b}$$

In the numerical calculations, time is scaled by the factor $\sqrt{I / M_{\max}}$ and the control torques are scaled by the factor M_{max}.

For the optimal control problem formulated using Eqs. (1)-(3), the Hamiltonian is

$$H = \lambda_{\omega_1}\dot{\omega}_1 + \lambda_{\omega_2}\dot{\omega}_2 + \lambda_{\omega_3}\dot{\omega}_3 + \lambda_{\varepsilon_1}\dot{\varepsilon}_1 + \lambda_{\varepsilon_2}\dot{\varepsilon}_2 + \lambda_{\varepsilon_3}\dot{\varepsilon}_3 + \lambda_{\varepsilon_4}\dot{\varepsilon}_4 \tag{5}$$

For spacecraft of the type being modeled here (Swift, or other astronomical missions), one or more sensors is fixed to the spacecraft bus; these sensors all have the same central axis for their fields of view and this axis is designated here with the unit vector $\hat{\sigma}$ and referred to as the sensor axis. This axis must be kept at least a minimum angular distance α_x from each of several high-intensity light sources. Denoting the directions to these sources as $\hat{\sigma}_x$, where the subscript x can be S (Sun), E (Earth), or M (Moon), the so-called *keep-out* constraints are then written as

$$C_x = (\hat{\sigma} \cdot \hat{\sigma}_x) - \cos(\alpha_x) \le 0 \tag{6}$$

Without loss of generality, $\hat{\sigma}$ is assumed to lie along the body-fixed x-axis and its orientation with respect to the inertial frame is then determined from Eq. (A-5), with A as the inertial frame and B as the body-fixed frame

$$\begin{aligned} \sigma_1 &= 1 - 2(\varepsilon_1^2 + \varepsilon_3^2) \\ \sigma_2 &= 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\varepsilon_4) \\ \sigma_3 &= 2(\varepsilon_3\varepsilon_1 - \varepsilon_2\varepsilon_4) \end{aligned} \tag{7}$$

It is further assumed that the reorientation maneuver can occur quickly enough that the spacecraft's orbital position remains essentially unchanged, and that therefore, the inertial directions to the high-intensity sources also remain constant during the slew maneuver.

Inclusion of the path constraints, Eq. (6), requires augmenting the Hamiltonian with terms of the form $\mu_x C_x$, with the multiplier $\mu_x \ge 0$ if the constraint is active and $\mu_x = 0$ otherwise.

While this problem could, in principle, be solved using an indirect method, the path constraints, Eq. (6), create especially difficult challenges. A direct approach, such as a pseudospectral method, offers the potential advantage of easier problem formulation; however, for this problem, an optimal solution, or even a feasible solution, is not automatically guaranteed.

To understand this challenge, one must recognize that the direct methods all employ an implicit integration of the governing system equations. Such solutions require initial estimates (sometimes these are just outright guesses) of the states and controls at the discrete times (nodes) for which the overall solution is computed. Lacking any user-provided initial estimate of the states and controls at these nodes, the logical automated estimate for the states is simply a linear interpolation between the states at the initial and final times; the control is also assumed to be a linear function between the minimum and maximum specified values, mapped over the specified range of times. Such a linear relationship violates various dynamic and kinematic constraints, viz. Eqs. (2) and (3). Consequently, the direct-method solver must then expend some computational effort to find a feasible solution. As determined in previous efforts, this is not always successful or can take considerable time (on the order of 1 to 12 hours).

A good initial guess for the states and controls can be obtained via explicit numerical integration of the equations of motion, using a global stochastic optimizer to determine the maneuver time and a number of parameters that define the control torques. The best known of these optimizers are genetic algorithms, particle swarm algorithms, differential evolution algorithms, and bacteria foraging algorithms. All of these have been applied by themselves or in several combinations, to solve optimal control problems, with varying degrees of success. In the problem at hand, only an approximate solution is required. Therefore, a method that generates a *feasible* solution with reasonable time and effort is valued over a method that generates a high-fidelity solution (i.e., one that meets, or nearly meets, the necessary conditions of optimality). In descending order of coding complexity, the best-known stochastic methods are: genetic algorithms, bacteria foraging optimizers, particle swarm optimizers, and differential evolution optimizers. This paper considers only the last three, all of which are heuristic in nature.

PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO), a stochastic optimization method, belongs to a class of algorithms characterized as *biomimetic*. Each vector of decision parameters is called a particle, and the optimization algorithm employs rules based upon observed behaviors of swarms of birds or insects as they search for food. First proposed by Kennedy and Eberhart,[11] PSO has gained widespread use in a variety of disciplines, most recently in trajectory optimization.[12] The basic PSO algorithm is as follows (this assumes a minimization problem):

Initialization

For a problem with D decision variables, one forms an $N \times D$ array of N possible solutions

$$\mathbf{P} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \vdots \\ \boldsymbol{p_N} \end{bmatrix} \tag{8}$$

with each row $\boldsymbol{p_i}$, also known as a *particle*, representing one potential solution. The population (or swarm) is initialized with random values, uniformly distributed between lower and upper bounds for the corresponding elements of a particle. That is, for lower bounds

$$b_{Lp} = [b_{Lp,1}, b_{Lp,2}, \ldots, b_{Lp,D}] \tag{9}$$

and upper bounds

$$b_{Up} = [b_{Up,1}, b_{Up,2}, \ldots, b_{Up,D}] \tag{10}$$

the initial population is set to

$$P_{i,j} = b_{Lp,j} + r_{i,j} \cdot (b_{Up,j} - b_{Lp,j}),\ i=1\ldots N;\ j=1\ldots D \tag{11}$$

where $r_{i,j}$ is a random number, selected for each i,j pair, uniformly distributed on [0,1].

Iteration

For $k = 1\ldots N_{iterations}$

Evaluate each particle $\boldsymbol{p_i}$ to determine the associated cost J_i.

The value of J_i is compared with the best value of J obtained for that particle thus far in the iteration history, denoted $J_{Best,i}$. If $J_i < J_{Best,I}$, then set $J_{Best,I} = J_i$, and assign the corresponding particle $\boldsymbol{p_i}$ to the i-th row of the array

$$P_{Best} = \begin{bmatrix} p_{Best,1} \\ \vdots \\ p_{Best,N} \end{bmatrix} \tag{12}$$

If $J_i < G$ (the globally best value of J obtained by any particle thus far in the iteration history), then assign $G = J_i$ and assign $\boldsymbol{G_{Best} = p_i}$.

Update the so-called velocity of each particle:

$$\boldsymbol{v}_i = c_I \boldsymbol{v}_i + c_C(\boldsymbol{p_{Best,i} - p_i}) + c_S(\boldsymbol{G_{Best} - p_i}) \tag{13}$$

The accelerator coefficients c_I, c_C, and c_S have the following significances: c_I is the inertia coefficient, which determines how much the velocity for the particle remains unchanged; c_C is the cognitive coefficient, which determines how much the velocity changes based upon the particle's best "position" (i.e., having lowest J_i) thus far; and c_S is the social coefficient, which determines how much the velocity changes based upon the particle's position relative to the best position found for the entire population (swarm) thus far. The standard schemes for computing these is

$$c_I = \frac{1+r_1}{2} \quad c_C = 1.49445 \cdot r_2 \quad c_S = 1.49445 \cdot r_3 \tag{14}$$

where r_1, r_2, and r_3 are random numbers uniformly distributed on [0,1]. Other schemes are possible, including one that reduces the magnitude of c_I linearly as the iteration proceeds, giving greater importance to the inertia effect in earlier iterations; this has the effect of keeping the swarm somewhat more uniformly distributed in the search space for a longer time and can help prevent premature convergence (also known as stagnation). Several different schemes have been considered in this work; however, because of the relatively small number of iterations used (in order to produce an approximate solution very quickly), none of the schemes showed a significant advantage.

Determine if the velocity has exceeded the allowable velocity bounds

$$\boldsymbol{b_{Uv} = b_{Up} - b_{Lp}} \qquad \boldsymbol{b_{Lv} = -b_{Uv}} \tag{15}$$

(thus, a particle located at one bound is unlikely to exceed the opposite bound in the next iteration).

If $v_{i,j} > b_{Uv,j}$, then $v_{i,j} = b_{Uv,j}$, and if $v_{i,j} < b_{Lv,j}$, then $v_{i,j} = b_{Lv,j}$

Update the position of each particle and prevent it from exceeding the allowable bounds:

$$\boldsymbol{p_i} = \boldsymbol{p_i} + \boldsymbol{v_i} \tag{16}$$

If $p_{i,j} > b_{Up,ij}$ then $p_{i,j} = b_{Up,ij}$

and if $p_{i,j} < b_{Lp,ij}$ then $p_{i,j} = b_{Lp,ij}$

end

DIFFERENTIAL EVOLUTION

Differential evolution (DE), first proposed by Storn and Price,[13] employs the same array of potential solutions (called solution vectors) and uses the same initialization procedure as described above for PSOs to generate the initial population array $\mathbf{P_1}$. A second array of solution vectors $\mathbf{P_2}$ is used to store improved solution vectors during each iteration.

Iteration

For $k = 1 \ldots N_{iterations}$

Evaluate each solution vector $\boldsymbol{p_k}$ and compare its cost to that of a trial vector $\boldsymbol{q}$ (described below). If $J(\boldsymbol{q}) < J(\boldsymbol{p_i})$, then assign $\boldsymbol{q}$ to $\mathbf{P_2}(i)$.

Determining $\boldsymbol{q}$:

Randomly select three different solution vectors from $\mathbf{P_1}$: $\mathbf{p_a}$, $\mathbf{p_b}$, $\mathbf{p_c}$.

Set $\mathbf{q} = \mathbf{p_a}$.

Randomly select from 1 to D elements (i_1, i_2, …) of $\mathbf{q}$ and replace each with

$$q(i) = \begin{cases} p_a(i) + F\big(p_b(i) - p_c(i)\big), & \text{if } r < CR \\ p_a(i), & \text{otherwise} \end{cases} \tag{17}$$

where r = random variable uniformly distributed on [0,1],

F = amplification factor in the range [0,2], typically $F = 0.5$

and CR = crossover criterion, in the range [0,1] (this value determines the rate at which information is exchanged between solution vectors).

Set $\mathbf{P_1} = \mathbf{P_2}$

end

BACTERIA FORAGING OPTIMIZATION

Also a biomimetic algorithm, bacteria foraging optimization mimics the behavior of bacteria as they search for food, reproduce, disperse and die. The algorithm employed here is a modification of one described by Chen et al.[14] Here, the vector of parameters to be determined is called a bacterium $\boldsymbol{B}$. The algorithm is as follows:

Initialize the population of N_B bacteria, using Eqs. (9)-(11), and set the number of swim steps N_S during chemotaxis (the process in which bacteria direct their movements in order to find nutrients).

For $k = 1, \ldots, N_{generations}$

Tumble and Swim (chemotaxis)

For each bacterium $\boldsymbol{B_i}$, form a trial bacterium $\boldsymbol{B_i}' = \boldsymbol{B_i} + C(k)u$

(The search direction $u \in R^D$ is randomly generated and has unit Euclidean norm; it remains unchanged during the chemotaxis loop for each bacterium. The step size $C(k)$ varies with each generation according to $C(k) = C_0 - 0.75C_0(k\text{-}1)/N_{generations}$.)

Set counter $m = 0$

While $J(\boldsymbol{B_i}') < J(\boldsymbol{B_i})$ and $m \leq N_S$

$\boldsymbol{B_i} = \boldsymbol{B_i}'$

$\boldsymbol{B_i}' = \boldsymbol{B_i} + C(k)u$

$m = m + 1$

end

end

Reproduction

Sort the bacteria in ascending order of J. Replace the worst (highest J values) $N_B/2$ bacteria with copies of the best $N_B/2$ bacteria.

Elimination and dispersal (reduces the probability of stagnation)

For $i = 1, \ldots, N_B$

If $r \leq P_{ed}$ then replace $\boldsymbol{B_i}$ with a randomly generated bacterium within the bounds specified in Eqs. (9) and (10). $P_{ed} \in [0,1]$ is the criterion for elimination and dispersal; r is a random number uniformly distributed on [0,1].

end

end

MODELING THE CONTROL TORQUES

In this work, the control torques are modeled as linear combinations of Chebyshev polynomials, with each torque represented as

$$M_i(t) = \sum_{j=1}^{N_T} c_j T_j(t) \tag{18}$$

where the T_j are Chebyshev polynomials, and the c_j are coefficients determined from the torque values calculated using any of the three heuristic algorithms (i.e., the decision variables in this formulation consist of the final time t_f, and the N_T torque values for each control).

For each particle (or solution vector), the time variable t in the domain $[0, t_f]$ is converted to τ in the domain [-1,1], which is the domain of the Chebyshev functions, via the linear relation

$$\tau = \frac{2t-(t_f+t_0)}{t_f-t_0} \tag{19}$$

and the Chebyshev coefficients are calculated using

$$c_j = \frac{2}{N_T}\sum_{k=1}^{N_T} f\left[\cos\left(\frac{\pi\left(k+\frac{1}{2}\right)}{N_T}\right)\right]\cos\left(\frac{\pi j\left(k+\frac{1}{2}\right)}{N_T}\right) = \frac{2}{N_T}\sum_{k=1}^{N_T} f_k \cos\left(\frac{\pi j\left(k+\frac{1}{2}\right)}{N_T}\right) \tag{20}$$

where the f_k are the torque values calculated by the optimizer at times t_k. Chebyshev polynomials are used here because of the property that all the Tj have extrema at ±1 on the domain $-1 \le \tau \le 1$, making the representation in Eq. (18) readily compatible with the torque limits. Eqs. (18)-(20) are employed in the explicit integration to compute the control torques. Each T_j value is calculated using the Clenshaw recursion algorithm.[15]

Another choice for representing the control torques is B-splines. These have been used successfully in conjunction with PSOs for several types of trajectory optimization and robot maneuver planning problems,[12] where the authors included the spline-degree as one of the unknown parameters to be determined during the optimization. Of particular interest is their solution for the robotic-arm problem that has a bang-bang control structure.

CONSTRAINTS

Penalty functions serve well for handling both equality and inequality constraints in conjunction with all three of the heuristic methods. The form of penalty function employed here is essentially linear in the constraint violation. For an equality constraint of the form

$$f_i(\boldsymbol{x}) = 0 \tag{21}$$

where **x** is the vector of decision variables, the penalty function found to work best in this problem is

$$F_i = k_i|\max\{0, f_i(\boldsymbol{x}) - \Delta_i| \tag{22}$$

and similarly, for inequality constraints of the form

$$g_j(\boldsymbol{x}) < 0 \tag{23}$$

the penalty function is

$$G_j(\boldsymbol{x}) = \ell_j\left|max\{0,\ g_j(\boldsymbol{x}) - \Delta_j\}\right| \tag{24}$$

where the k_i and $\boldsymbol{\ell}_j$ are weights, whose values are selected to make the penalty terms have the same order of magnitude as the principal part of the cost function; the user-specified tolerances Δ_i, Δ_j result in zero penalties for $f_i < \Delta_I$ and $g_j < \Delta_j$. The resulting modified cost function is then

$$J' = J + \sum_{i=1}^{N_{eq}} F_i + \sum_{j=1}^{N_{ineq}} G_j \tag{25}$$

where N_{eq} and N_{ineq} are the numbers of equality and inequality constraints, respectively.

RESULTS

Using the pseudospectral code DIDO,[16] a sample case first reported in Ref. 9 was examined. This is a rest-to-rest maneuver of an axisymmetric spacecraft, with initial orientation given by $\varepsilon_{1,i} = 0$, $\varepsilon_{2,i} = 0$, $\varepsilon_{3,i} = 0$, $\varepsilon_{4,i} = 1$, and final orientation given by $\varepsilon_{1,f} = 0$, $\varepsilon_{2,f} = 0$, $\varepsilon_{3,f} = \sin(3\pi/8)$, $\varepsilon_{4,f} = \cos(3\pi/8)$. Three "keep-out" cones are included, centered on the Sun, Earth, and Moon, depicted in Fig. 1. The corresponding cone half-angles are 47 deg, 33 deg., and 23 deg., respectively (these values correspond to constraints on the Swift telescope axis). Fig. 1 also shows the optimal path of the telescope axis. Corresponding to this motion are the control and state histories, shown in Fig. 2. As seen in Fig. 3, the Hamiltonian remains essentially constant at a value

of -1, as expected if the path constraints never become active. This solution was generated with no initial guess for the controls or states; required computation time was 4189 sec. Note that the optimal sensor path passes between the Sun and Moon constraint cones, which are separated by 0.1 deg., without intersecting either cone.

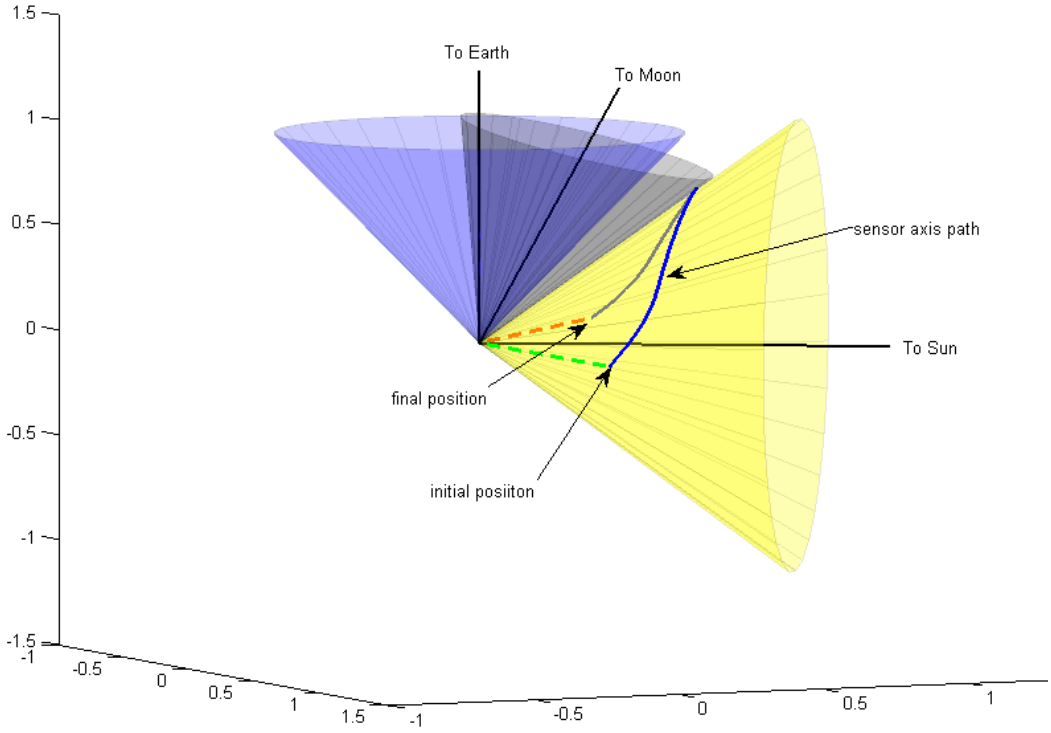

Figure 1. Time-optimal path of the sensor axis.

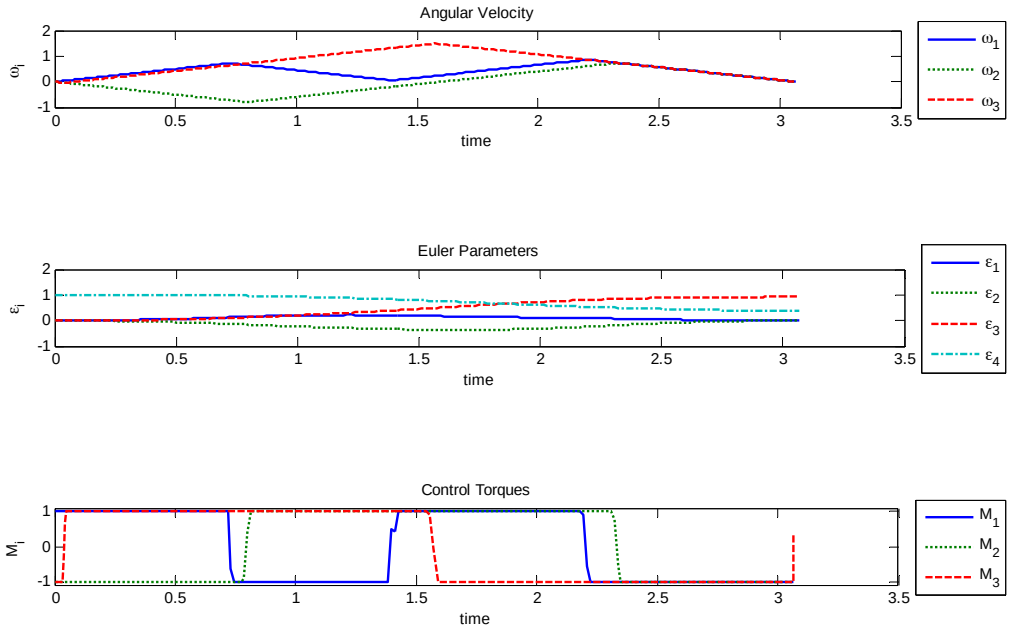

Figure 2. State and torque histories (from DIDO) for the path shown in Fig. 1.

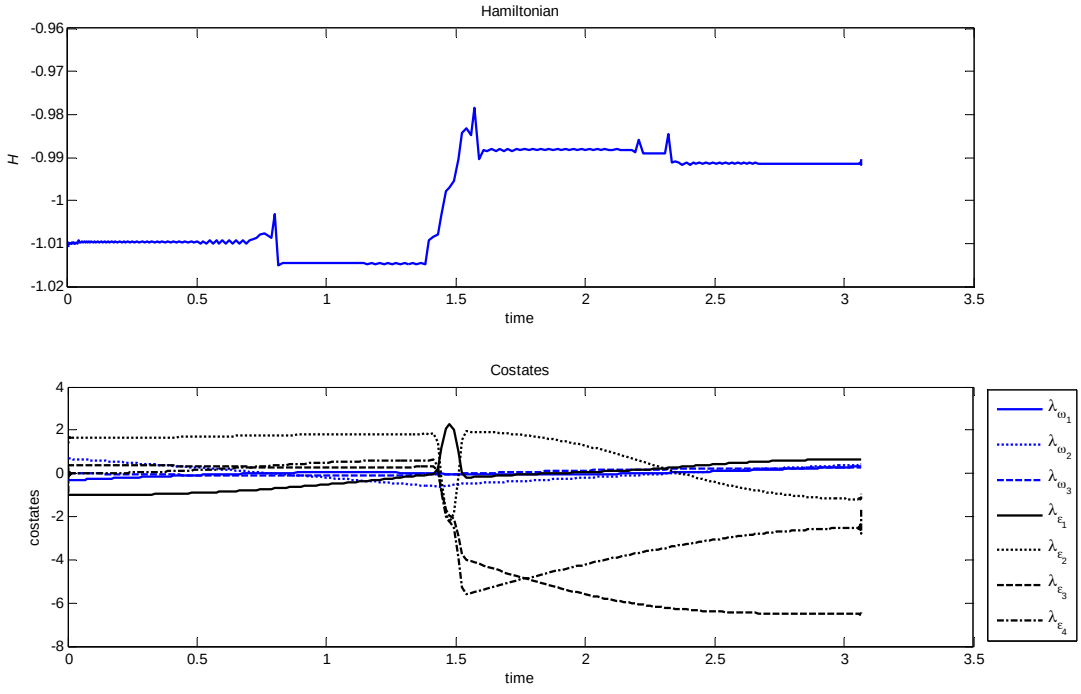

Figure 3. Hamiltonian and costate histories (from DIDO) for the path shown in Fig. 1.

Using the PSO with 20 discrete control torques (per axis), 30 particles and 200 iterations, an approximate solution was generated, with results shown in Figs. 4 and 5. (Processor times for DIDO alone and for the hybrid methods are summarized in Table 1.) While the torque history is not bang-bang (as in Fig. 2), a careful examination shows that each torque component follows approximately the same pattern as in Fig. 2. For example, M_3 is mostly positive in the first half of the time history, and mostly negative in the second half. This results in ω_3 rising to a maximum value and then decreasing back to zero. The required computation time was 196 sec. This solution (including the approximate histories for the Euler parameters, angular velocities, and torques) was then used as the initial guess for DIDO, which now terminated in 1344 sec of processor time; then overall total processor time (PSO + DIDO) was thus 1540 sec (approximately one-third of what DIDO required with no initial guess). This example is typical of the results found for other cases (i.e., with different orientations of the constraint cones): the PSO code required approximately 1300 sec. of processor time and produced solutions that permitted DIDO to converge in less than half the time needed without an initial guess. In some cases, DIDO was unable to find any solution unless an approximate guess from the PSO was provided.

Using the DE optimizer, with 20 discrete control torques (per axis), 30 solution vectors and 500 iterations produced results qualitatively similar to those in Figs. 4 and 5, with a required processor time of 272 sec.; however, the approximate solution was of such poor quality that when used as the initial guess for DIDO, resulted in a much longer computation time (DIDO processor time = 6908 sec, total processor time = 7180 sec.). Repeated attempts, with different numbers of iterations, and different values of *CR* and *F* failed to produce useful approximate solutions.

With the BFO algorithm employed in the first stage, with 20 discrete control torques (per axis), 30 bacteria and 200 iterations, the approximate solution (torque and angular velocity components) are as shown in Figs. 6 and 7, respectively. This approximate solution required 167 sec of processor time. The resulting pseudospectral solution, identical to that in Figs. (1)-(3), required only 1034 sec.

It was also found that an approximate solution for the *unconstrained* problem could also work effectively as an initial guess for the constrained problem. This is not surprising, since the unconstrained solution meets the dynamic constraints and the end-point constraints, allowing the pseudospectral method to search just within the subspace constrained by the "keep-out" cones. Nevertheless, constrained initial guesses are preferred to help ensure that the first-stage solution results from a thorough exploration of the entire constrained solution space.

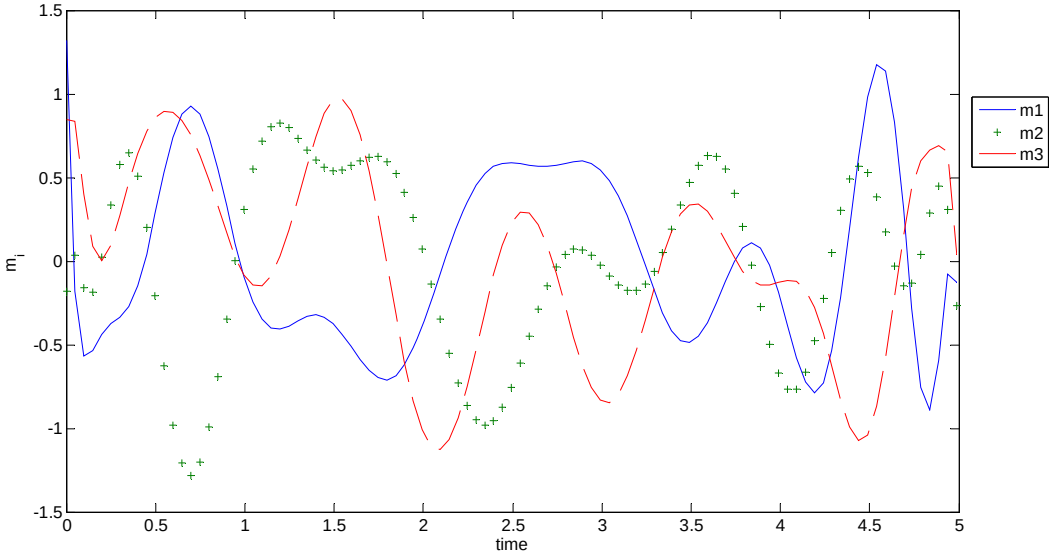

Figure 4. Approximate control solution generated by the PSO.

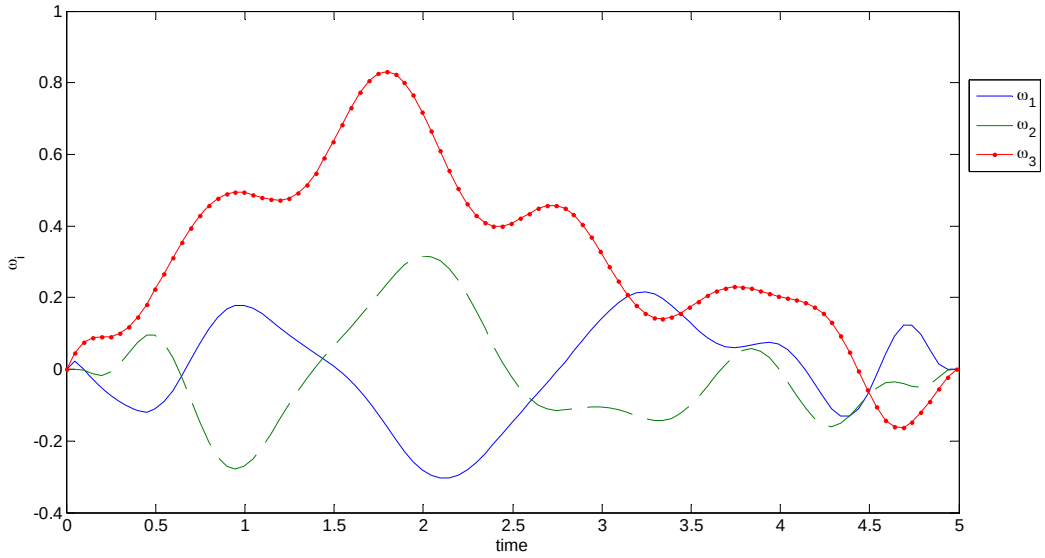

Figure 5. Approximate angular velocities generated by the PSO.

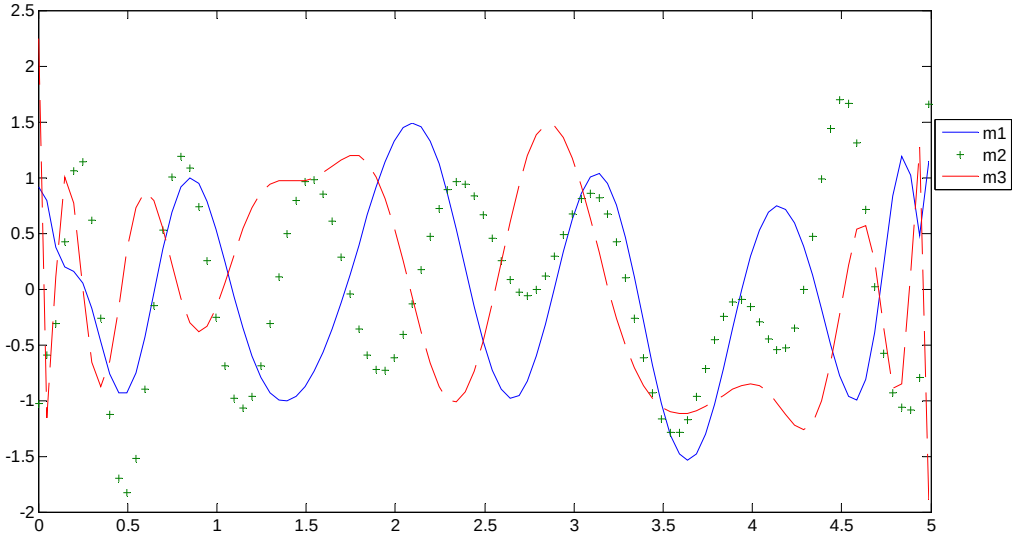

Figure 6. Approximate control solution generated by the BFO.

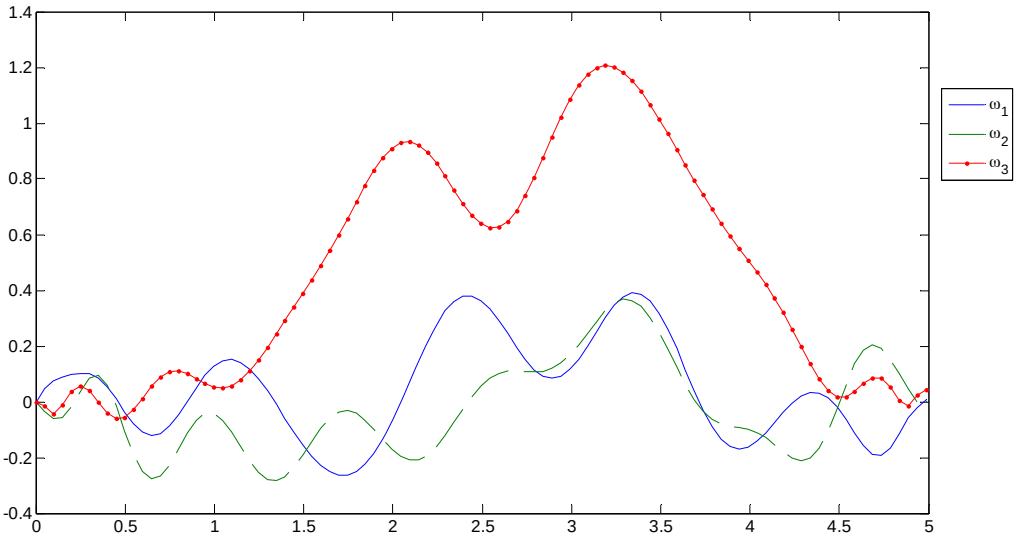

Figure 7. Approximate angular velocities generated by the BFO.

Table 1. Computation times

Method	Computation time (sec)
Pseudospectral alone	4189
PSO → Pseudospectral	196 + 1344 = 1540
DE → Pseudospectral	272 + 6908 = 7180
BFO → Pseudospectral	167 + 1034 = 1101

CONCLUSION

For constrained, time-optimal spacecraft maneuvers, use of a global stochastic optimizer in conjunction with a pseudospectral method can significantly reduce overall computation time. In some cases, the pseudospectral method was unable to find a solution without an approximate guess from the first-stage optimizer. Guesses that meet only the dynamic constraints (i.e., the equations of motion) are found to be suitable guesses.

In the cases considered in this work, using a bacteria foraging oprtimizer (BFO) or a particle swarm optimizer (PSO) for the first stage showed better performance than did a differential evolution (DE) method. All three methods require some "tuning" (i.e., selection of certain parameter values) for a specific problem, and further study of DEs, which are simpler to code and execute faster, could prove useful for this problem. An alternative representation of the control torques using B-splines is worth investigating since that approach has been found by others to work well in problems with a bang-bang control structure.

REFERENCES

[1]Bilimoria, K.D. and Wie, B., "Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 3, 1993, pp. 446-452.

[2]Shen, H. and Tsiotras, P., "Time-Optimal Control of an Axi-Symmetric Rigid Spacecraft sing Two Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 5, 1999, pp. 682-694.

[3]Proulx, R.J. and Ross, I.M., "Time-Optimal Reorientation of Asymmetric Rigid Bodies," *Advances in the Astronautical Sciences*, Vol. 109, part II, 2002, pp. 1207-1227.

[4]Scrivener, S.L. and Thompson, R.C., "Time-Optimal Reorientation of a Rigid Spacecraft Using Collocation and Nonlinear Programming," *Advances in the Astronautical Sciences*, Vol. 85, part III, 1993, pp. 1905-1924.

[5]Liu, S.-W. and Singh T., "Fuel/Time Optimal Control of Spacecraft Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 20., No. 2, 1997, pp. 394-397.

[6]Bai, X. and Junkins, J.L., ""New Results for Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, 2009, 1071-1076.

[7]Hablani, H B., "Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 6, 1999, pp. 759-767.

[8]Mengali, G. and Quarta, A.A.,"Spacecraft Control With Constrained Fast Reorientation and Accurate Pointing," *The Aeronautical Journal*, Vol. 108, No. 1080, 2004, pp. 85-91.

[9]Melton, R.G., "Constrained Time-Optimal Slewing Maneuvers for Rigid Spacecraft," Advances in the Astronautical Sciences, 2010, Vol. 135, pp. 107-126 (paper AAS 09-309).

[10]Swift. http://heasarc.nasa.gov/docs/swift/swiftsc.html. Accessed July 3, 2009.

[11]Kennedy, J, Eberhart, R., "Particle Swarm Opimization," *Proceedings of IEEE International Conference on Neural Networks*, IV, 1995, pp. 1942-1948.

[12]Ghosh, P. and Conway, B.A., "Numerical Trajectory Optimization with Swarm Intelligence and Dynamic Assignment of Solution Structure," *Journal of Guidance, Dynamics, and Control*, to appear.

[13]Storn, R., and Price, K., "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, 1997, Vol. 11, pp. 341-359.

[14]Chen, H, Zhu, Y., and Hu, K., "Adaptive Bacterial Foraging Optimization," *Abstract and Applied Analysis*, Vol, 2011, article ID 108269, doi:10.1155/2011/108269.

[15]Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes: The Art of Scientific Computing 3rd ediiton*, Cambridge University Press, 2007, sect. 5.8.

[16]Ross, I.M., *A Beginner's Guide to DIDO (ver 7.3): A MATLAB©Application Package for Solving Optimal Control Problems*, Document #TR-711, Elissar, LLC, 2007.