Acta Astronautica 94 (2014) 294–301

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Hybrid methods for determining time-optimal, constrained spacecraft reorientation maneuvers

Robert G. Melton *

The Pennsylvania State University, University Park, 229 Hammond Bldg, Pennsylvania 16802, USA

ARTICLE INFO

Article history:
Received 2 August 2012
Accepted 10 May 2013
Available online 24 May 2013

Keywords:
Time-optimal attitude control
Heuristic methods
Pseudospectral method
Particle swarm optimization
Bacteria foraging optimization

ABSTRACT

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, which then 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.

1. 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 $\lambda-$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 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 $\lambda-$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

* Tel.: +1 814 865 1185; fax: +1 814 865 7092.
E-mail address: rgmelton@psu.edu

0094-5765/$ - see front matter © 2013 IAA. Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.actaastro.2013.05.007

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 afterglow 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.

2. 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

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

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

$$\begin{bmatrix}\dot{\epsilon}_1\\ \dot{\epsilon}_2\\ \dot{\epsilon}_3\\ \dot{\epsilon}_4\end{bmatrix} = \begin{bmatrix}\epsilon_4 & -\epsilon_3 & \epsilon_2 & \epsilon_1\\ \epsilon_3 & \epsilon_4 & -\epsilon_1 & \epsilon_2\\ -\epsilon_2 & \epsilon_1 & \epsilon_4 & \epsilon_3\\ -\epsilon_1 & -\epsilon_2 & -\epsilon_3 & \epsilon_4\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ 0\end{bmatrix} \tag{3}$$

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 with boundary conditions

$$\begin{aligned}&\omega_1(0) = \omega_2(0) = \omega_3(0) = 0\\ &\epsilon_1(0) = \epsilon_2(0) = \epsilon_3(0) = 0, \quad \epsilon_4(0) = 1\\ &\omega_1(t_f) = \omega_2(t_f) = \omega_3(t_f) = 0\\ &\epsilon_1(t_f) = \epsilon_2(t_f) = 0, \quad \epsilon_3(t_f) = \sin\left(\frac{3\pi}{8}\right),\\ &\epsilon_4(t_f) = \cos\left(\frac{3\pi}{8}\right)\end{aligned} \tag{4}$$

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

$$\begin{aligned}\mathcal{H} = {}& \lambda_{\omega_1}\dot{\omega}_1 + \lambda_{\omega_1}\dot{\omega}_2 + \lambda_{\omega_3}\dot{\omega}_3 + \lambda_{\epsilon_1}\dot{\epsilon}_1\\ &+\lambda_{\epsilon_2}\dot{\epsilon}_2 + \lambda_{\epsilon_3}\dot{\epsilon}_3 + \lambda_{\epsilon_4}\dot{\epsilon}_4\end{aligned} \tag{5}$$

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 [11, pp. 4–15]

$$\begin{aligned}\sigma_1 &= 1 - 2(\epsilon_1^2 + \epsilon_3^2)\\ \sigma_2 &= 2(\epsilon_1\epsilon_2 + \epsilon_3\epsilon_4)\\ \sigma_3 &= 2(\epsilon_3\epsilon_1 - \epsilon_2\epsilon_4)\end{aligned} \tag{7}$$

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 h).

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.

3. 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 [12], PSO has gained widespread use in a variety of disciplines, most recently in trajectory optimization [13]. 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

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,D} \\ \vdots & \ddots & \vdots \\ p_{N,1} & \cdots & p_{N,D} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} \tag{8}$$

with each row $\mathbf{p}_i$, 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

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

and upper bounds

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

the initial population is set to

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

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

$$b_{Uv} = b_{Up} - b_{Lp}, \qquad b_{Lv} = -b_{Uv} \tag{12}$$

which keep the velocity (i.e., the rate of change of the "position" per iteration) within the appropriate numerical range.

One initializes the array of best particles, $\mathbf{P}_{Best} = \mathbf{P}$, the N-element array of best-cost values $\mathbf{J}_{Best}$, and the global-best value G^*, then sets G^* and the elements of $\mathbf{J}_{Best}$ to arbitrarily large values. Finally, one sets the number of iterations N_{iter}.

In the following iteration algorithm, each particle's new velocity is determined by a combination of three effects: the inertial component, which tends to keep the velocity unchanged; the cognitive component, which moves in the direction of the best position that this particle has ever obtained; and the social component, which moves in the direction of the best position obtained by any particle in the population. The relative weights of the these components are dictated by the so-called accelerator coefficients, c_I, c_C, and c_S, respectively, employing random numbers r_1, r_2 and r_3, uniformly distributed on [0,1]. Next, the particle's position is updated by adding the new velocity (times one iteration). The function *Limit* checks each element of the argument vector ($\mathbf{p}_i$ or $\mathbf{v}_i$) to determine if it exceeds the corresponding lower or upper bound, and if so, sets the element equal to that bound. At the end of all iterations, the particle $\mathbf{G}_{Best}$ is deemed the optimal solution.

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.

PSO Iteration Algorithm.

for $k = 1 : N_{iter}$ **do**
  for $i = 1 : N$ **do**
    $J_i = J(\mathbf{p}_i)$
    if $J_i < \mathbf{J}_{Best,i}$ **then**
      $\mathbf{J}_{Best,i} = J_i$
      $\mathbf{P}_{Best,i} = \mathbf{p}_i$
    end if
    if $J_i < G^*$ **then**
      $G^* = J_i$
      $\mathbf{G}_{Best} = \mathbf{p}_i$
    end if
    $c_I = (1 + r_1)/2, \qquad c_C = 1.49445 r_2, \qquad c_S = 1.49445 r_3$
    $\mathbf{v}_i = c_I \mathbf{v}_i + c_C(\mathbf{p}_{Best,i} - \mathbf{p}_i) + c_S(\mathbf{G}_{Best} - \mathbf{p}_i)$
    Limit($\mathbf{b}_{Lv} \le \mathbf{v}_i \le \mathbf{b}_{Uv}$)
    $\mathbf{p}_i = \mathbf{p}_i + \mathbf{v}_i$
    Limit($\mathbf{b}_{Lp} \le \mathbf{p}_i \le \mathbf{b}_{Up}$)
  end for
end for

4. Differential evolution

Differential evolution (DE), first proposed by Storn and Price [14], 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. In this algorithm, 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). The optimal solution is the value of $\mathbf{q}$ with minimum J.

DE Iteration Algorithm.

for $k=1:N_{iter}$ **do**
  for $i=1:N$ **do**
    Randomly select three different solution vectors from $\mathbf{P}_1$: $\mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c$.
    Form the trial vector: Randomly select from 1 to D elements $(j_1, j_2, \ldots)$ and set

$$\mathbf{q}(j)=\begin{cases}\mathbf{p}_a(j)+F(\mathbf{p}_b(j)-\mathbf{p}_c(j)), & r<CR\\ \mathbf{p}_a(j), & \text{otherwise}\end{cases}$$

    if $J(\mathbf{q})<J(\mathbf{p}_i)$ **then** $\mathbf{P}_2(i)=\mathbf{q}$
    end if
  end for
  $\mathbf{P}_1=\mathbf{P}_2$.
end for

5. 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. [15]. Here, the vector of parameters to be determined is called a bacterium **B**.

One initializes the population of N bacteria, using Eqs. (9)–(11), sets the number of swim steps N_S during chemotaxis (the process in which bacteria direct their movements in order to find nutrients), and then sets the swim-step size $C_0=K/N_S$. The value of K should be the maximum allowable change in any element of **B** during one iteration; for this work, because of the scaling used in the equations of motion, K=1. The value of $P_{ed}\in[0,1]$ is the criterion for elimination and dispersal, an important step for reducing the probability of stagnation; r is a random number uniformly distributed on [0,1].

BFO Iteration Algorithm.

for $k=1:N_{iter}$ **do**
  Tumble and Swim (**chemotaxis**)
  for $i=1:N$ **do**
    Generate $\mathbf{u}_i$, a random vector with uniformly distributed elements on [−1,1], with $|\mathbf{u}_i|=1$
    $\mathbf{B}'_i=\mathbf{B}_i+C(k)\mathbf{u}_i$, a trial bacterium
    $C(k)=C_0-0.75C_0(k-1)/N_{iter}$
    Set $m=0$ (step counter)
    while $J(\mathbf{B}'_i)<J(\mathbf{B}_i)$ and $m\le N_S$ **do**
      $\mathbf{B}_i=\mathbf{B}'_i$
      $\mathbf{B}'_i=\mathbf{B}_i+C(k)\mathbf{u}_i$
      $m=m+1$
    end while
  end for
  Reproduction
  Sort the bacteria in ascending order of J. Replace the worst $N/2$ bacteria (those with the highest J values) with copies of the best $N/2$ bacteria.
  Elimination and dispersal
  for $i=1:N$ **do**
    if $r<P_{ed}$ *bfthen* replace B_i with a randomly generated bacterium within the bounds specified in Eqs. (9) and (10).
    end if
  end for
end for

6. Modeling the control torques

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

$$M_i(t)=\sum_{j=1}^{N_T} c_j T_j(t) \tag{13}$$

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

$$\tau=\frac{2t-(t_f+t_0)}{t_f-t_0} \tag{14}$$

and the Chebyshev coefficients are calculated using

$$c_j=\frac{2}{N_T}\sum_{k=1}^{N_T} f\left[\cos\left(\frac{k+\frac{1}{2}}{N_T}\right)\right]\cos\left(\frac{\pi j\left(k+\frac{1}{2}\right)}{N_T}\right)$$
$$=\frac{2}{N_T}\sum_{k=1}^{N_T} f_k\cos\left(\frac{\pi j\left(k+\frac{1}{2}\right)}{N_T}\right) \tag{15}$$

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 T_j have extrema at ± 1 on the domain $-1\le\tau\le 1$, making the representation in Eq. (13) readily compatible with the torque limits. Eqs. (13)–(15) are employed in the explicit integration to compute the control torques. Each T_j value is calculated using the Clenshaw recursion algorithm [16]. 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 [13], 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.

7. 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

$$f_i(x)=0 \tag{16}$$

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

$$F_i=k_i|\max\{0, f_i(x)-\Delta_i\}| \tag{17}$$

and similarly, for inequality constraints of the form

$$g_j(x) < 0 \tag{18}$$

the penalty function is

$$G_j(x) = l_j|\max\{0, g_j(x) - \Delta_j\}| \tag{19}$$

where the k_i and l_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

$$J' = J + \sum_{i=1}^{N_{eq}} F_i + \sum_{j=1}^{N_{ineq}} G_j \tag{20}$$

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

8. Results

Using the pseudospectral code DIDO [17] a sample case first reported in Ref. [9] was examined. This is a rest-to-rest maneuver of an axisymmetric spacecraft, with initial and final orientations given by Eq. (4). 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

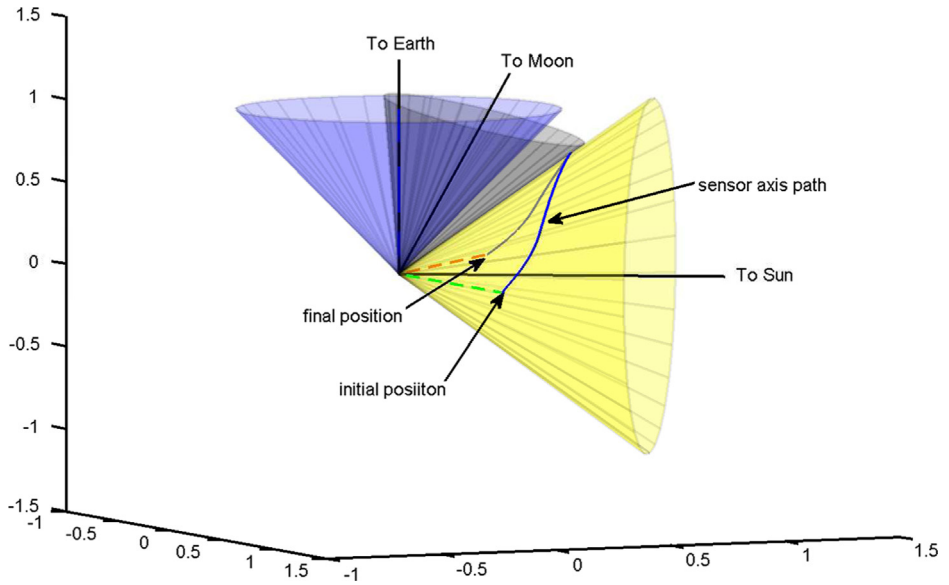

Fig. 1. Constrained time-optimal path of the sensor axis.

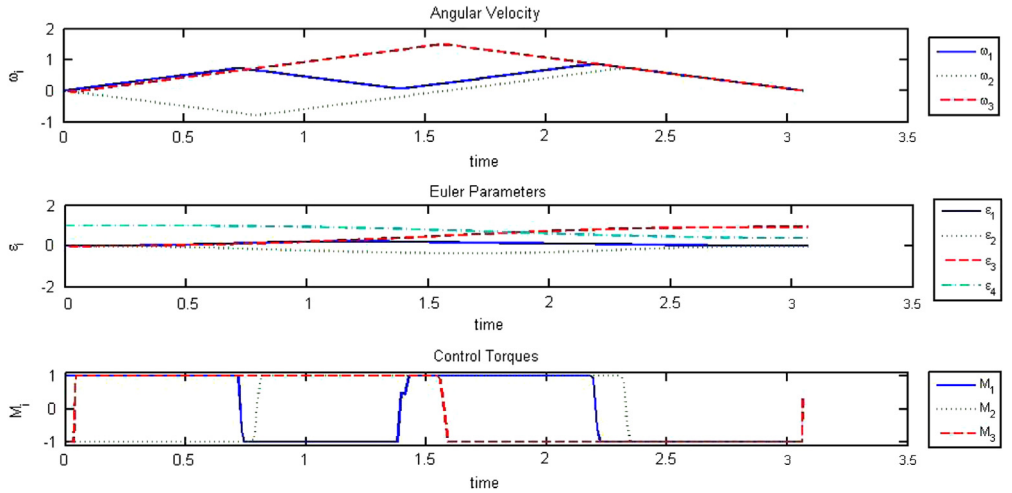

Fig. 2. State and torque histories (from DIDO) for the path shown in Fig. 1.

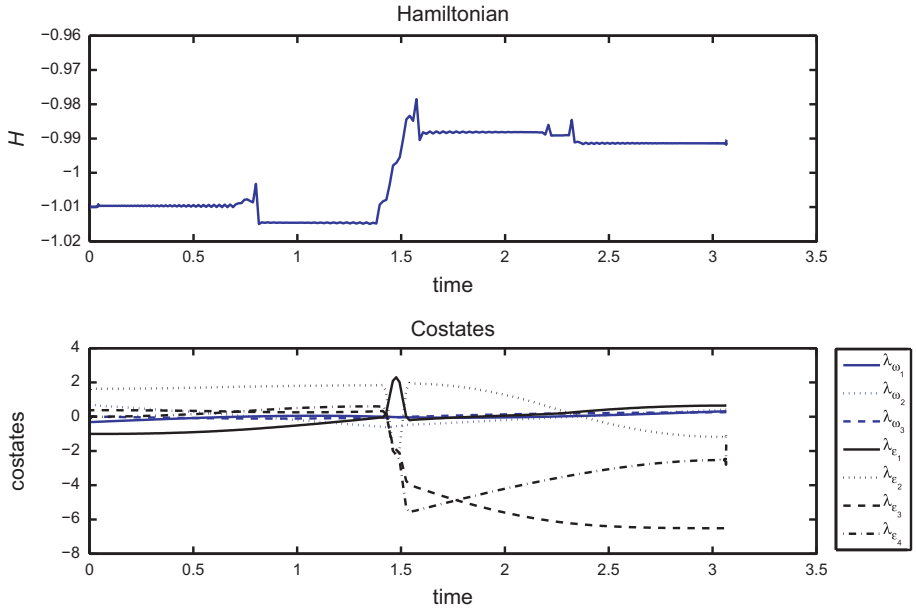

Fig. 3. Hamiltonian and costate histories (from DIDO) for the path shown in Fig. 1.

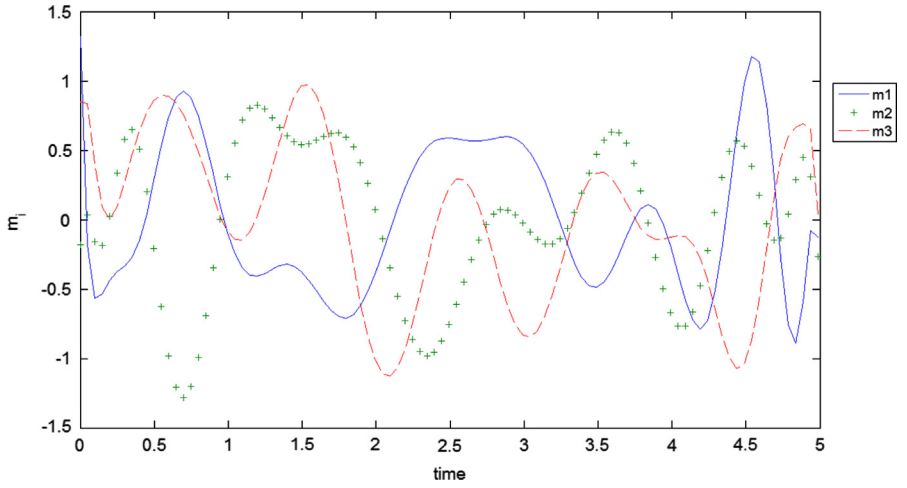

Fig. 4. Approximate control solution generated by the PSO.

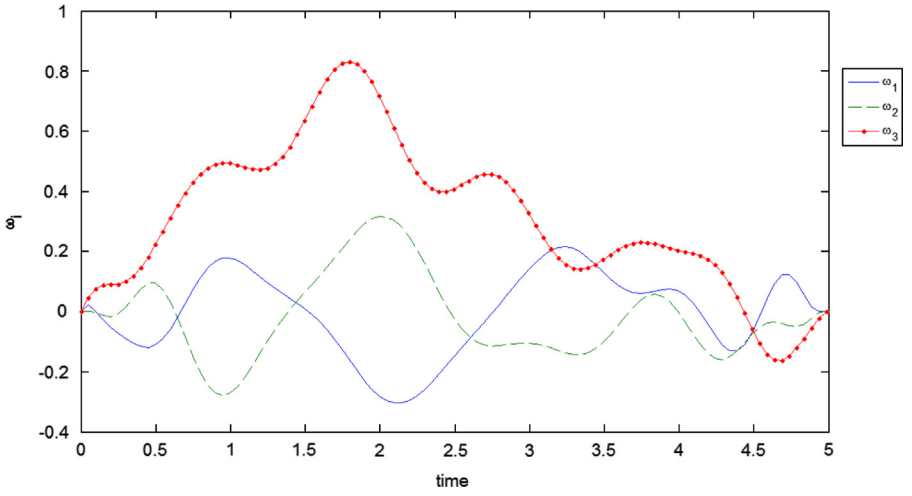

Fig. 5. Approximate angular velocities generated by the PSO.

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 s. 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.

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 s. 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 s of processor time; then overall total processor time (PSO + DIDO) was thus 1540 s (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 s. 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

Table 1
Computation times.

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

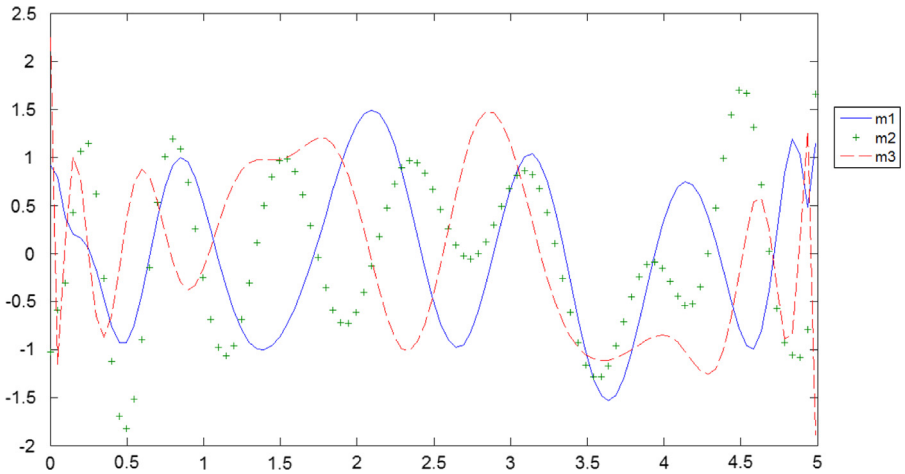

Fig. 6. Approximate control solution generated by the BFO.

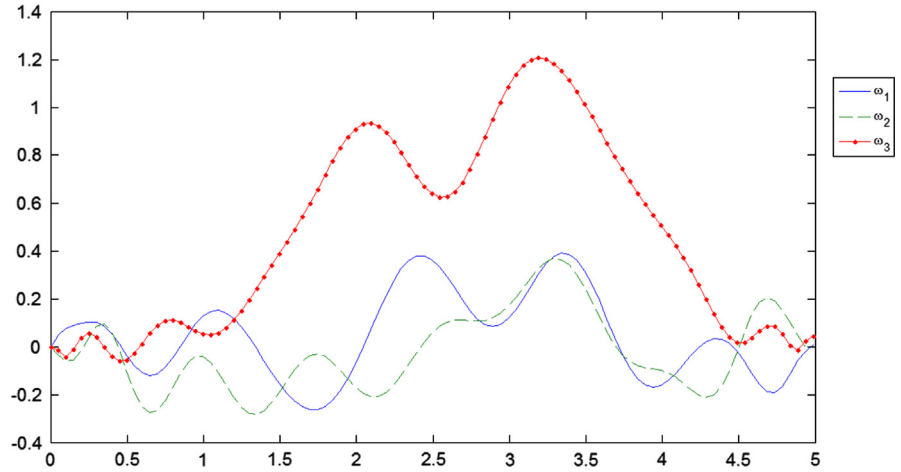

Fig. 7. Approximate angular velocities generated by the BFO.

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 s.; 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 s, total processor time = 7180 s.). 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 s of processor time. The resulting pseudospectral solution, identical to that in Figs. 2 and 3, required only 1034 s.

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.

9. 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 optimizer (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] K.D. Bilimoria, B. Wie, Time-optimal three-axis reorientation of a rigid spacecraft, J. Guidance Control Dyn. 16 (3) (1993) 446–452.
[2] H. Shen, P. Tsiotras, Time-optimal control of an axi-symmetric rigid spacecraft using two controls, J. Guidance Control Dyn. 22 (5) (1999) 682–694.
[3] R.J. Proulx, I.M. Ross, Time-optimal reorientation of asymmetric rigid bodies, Adv. Astronaut. Sci. 109 (part II) (2002) 1207–1227.
[4] S.L. Scrivener, R.C. Thompson, Time-optimal reorientation of a rigid spacecraft using collocation and nonlinear programming, Adv. Astronaut. Sci. 85 (part III) (1993) 1905–1924.
[5] S.-W. Liu, T. Singh, Fuel/time optimal control of spacecraft maneuvers, J. Guidance Control Dyn. 20 (2) (1997) 394–397.
[6] X. Bai, J. Junkins, New results for time-optimal three-axis reorientation of a rigid spacecraft, J. Guidance Control Dyn. 32 (4) (2009) 1071–1076, http://dx.doi.org/10.2514/1.43097.
[7] H. Hablani, Attitude commands avoiding bright objects and maintaining communication with ground station, J. Guidance Control Dyn. 22 (6) (1999) 759–767.
[8] G. Mengali, A. Quarta, Spacecraft control with constrained fast reorientation and accurate pointing, Aeronaut. J. 108 (1080) (2004) 85–91.
[9] R. Melton, Constrained time-optimal slewing maneuvers for rigid spacecraft, Adv. Astronaut. Sci. 135 (2010) 107–126.
[10] ⟨http://heasarc.nasa.gov/docs/swift/swiftsc.html⟩, accessed: July 3, 2009.
[11] T. Kane, P. Likins, D. Levinson, Spacecraft Dynamics, McGraw-Hill, New York, 1983.
[12] J. Kennedy, R. Eberhart, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Netw. IV (1995) 1942–1948.
[13] P. Ghosh, B. Conway, Numerical trajectory optimization with swarm intelligence and dynamic assignment of solution structure, J. Guidance Control Dyn. 35 (4) (2012) 1178–1191, http://dx.doi.org/10.2514/1.55594.
[14] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[15] H. Chen, Y. Zhu, K. Hu, Adaptive bacterial foraging optimization, Abstract Appl. Anal., 2011, article ID 108269 http://dx.doi.org/10.1155/2011/108269.
[16] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd edition, Cambridge University Press, 2007.
[17] I. Ross, A beginner's guide to DIDO (ver 7.3): a matlab application package for solving optimal control problems, Document #TR-711, Elissar, LLC, 2007.