Chapter 7

Advanced Applications

Most effective numerical methods are based on the following premise:

A hard problem can be broken into a sequence of easy subproblems.

For example, to compute the root of a *nonlinear* constraint $c(x) = 0$ using Newton's method, one must solve the sequence of *linear* approximations $x_{k+1} = x_k - c(x_k)/c'(x_k)$. Similarly, when using a nonlinear programming (NLP) algorithm to solve a *nonlinear optimization* problem, one solves a sequence of (a) *quadratic programming* subproblems (an SQP method) or (b) *unconstrained* subproblems (a barrier method). In like fashion an optimal control solution can be obtained by solving a sequence of NLP subproblems. Of course there may be no clear definition of an "easy subproblem." For Newton's method is it better to compute $c'(x_k)$ or use a secant approximation? Is a quadratic program easier than an unconstrained subproblem? Ultimately the solution technique should consist of a sequence of subproblems that can be solved efficiently and reliably.

This chapter presents a number of advanced applications that require using a combination of the techniques presented in the book.

7.1 Optimal Lunar Swingby Trajectories

7.1.1 Background and Motivation

With the growth of the space age, it has become desirable to place spacecraft into a wide variety of mission orbits. Typically a launch vehicle is used to deploy a satellite into a low altitude park orbit. It then becomes necessary to transfer from the park orbit to the mission orbit. A major goal of mission design is to compute this *orbit transfer* to minimize some performance objective. The most common objective is to minimize the fuel consumed by the orbit transfer. For spacecraft using propulsion systems whose thrust is high compared to the mass of the vehicle, i.e., so-called *high thrust* systems, the duration of the burns is very short compared to the duration of the orbit transfer itself. Consequently it is common to model high thrust systems using an instantaneous or *impulsive* velocity change. In essence a minimum fuel orbit transfer is approximated using a so-called *minimum* Δv *transfer.* In 1925, the German scientist Walter Hohmann demonstrated that a minimum Δv transfer be-

tween two circular orbits in the same plane could be achieved using two impulsive velocity increments.

Although the original *Hohmann transfer* was strictly between coplanar orbits, the term is often loosely used to describe any two-impulse transfer. For example, one of the most common applications involves transferring between a circular orbit with an altitude of 150 nm and inclination of 28.5 deg (a standard shuttle park orbit) to a geosynchronous orbit which is circular at an altitude of 19323 nm and inclination of 0 deg. The Hohmann transfer for this application is illustrated in Figure 6.21. However, the park and mission orbits are not coplanar in this case, and some of the requisite inclination change must be accomplished by each of the Δv. For this transfer nearly all of the inclination change ($\approx$ 26 deg) is achieved by the second impulse. In fact for any orbit transfer it is most efficient to change the orbital plane at a high altitude when the velocity is smallest.

Because changing the orbit plane is the most expensive portion of the transfer one is led to consider a *three-burn transfer*. For this type of transfer, the second burn does all of the plane change at a very high altitude. In fact it was demonstrated in Betts [12] that a three-impulse solution is more efficient than two impulses for some transfers that require large plane changes. Figure 7.1 illustrates the situation. A critical property of a three-burn transfer is that the second burn be located a "long" distance from the primary body (earth). Thus one is led to consider locating the second impulse at or near the moon. In effect the lunar gravitational attraction can be used in lieu of the second burn, thereby producing a very efficient trajectory. The concept of using a lunar gravity assist to design a nominal mission trajectory has been considered by many authors (cf. [172]). Lee et al. [129] discuss using a lunar swingby; however, as with most analyses they do not minimize the impulsive velocity. A more detailed presentation is given in [22].

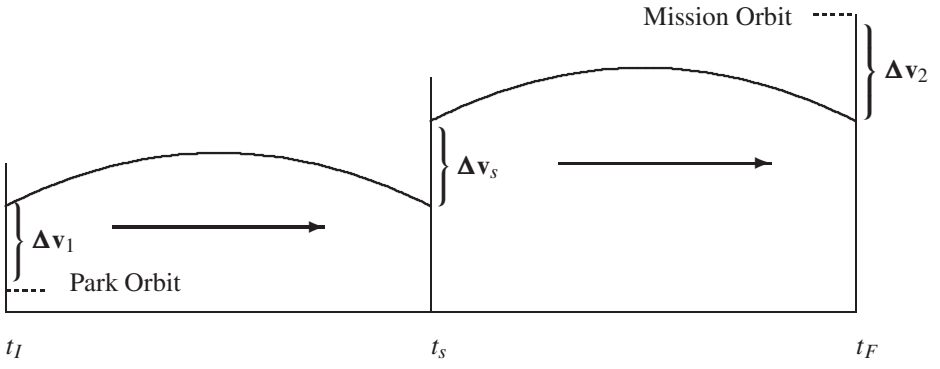


Figure 7.1. *Optimal three-impulse transfer orbit.*

This application deals with an approach for computing optimal lunar swingby trajectories between two earth orbits. First some representative optimal transfers are presented that illustrate the performance benefits. Then the problem formulation is explained and a number of challenging numerical issues are addressed. To facilitate using the examples as benchmark problems, the dynamics are modeled using three-body mechanics but do not require more sophisticated planetary ephemerides. The overall solution procedure requires solution of a dense nonlinear program, an optimal control problem, and an optimal estimation problem.

7.1.2 Optimal Lunar Transfer Examples

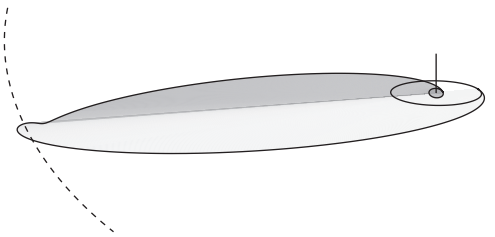
For the sake of illustration examples of lunar swingby transfers to four different mission orbits are presented. All of the transfers begin in a circular park orbit at an altitude of 150nm, with an inclination of 28 deg. The outbound transfer orbit is established by $\mathbf{\Delta v}_1$, and after passing around the moon the inbound transfer returns to the mission orbit which is established by applying $\mathbf{\Delta v}_2$. All of the transfers minimize the total Δv, that is,

$$F = \|\mathbf{\Delta v}_1\| + \|\mathbf{\Delta v}_2\|. \tag{7.1}$$

To obtain meaningful results it is also necessary to impose some limit on the total transfer time. An upper bound of 15 days is imposed on the total transfer time for the examples.

Synchronous Equatorial

Results are presented for a geosynchronous orbit which is circular at an altitude of 19323 nm and inclination of 0 deg. Figure 7.2 illustrates the optimal lunar swingby trajectory and for comparison summarizes the total Δv of the swingby as well as a standard two-impulse Hohmann transfer. It is interesting to observe that even for this case, which requires only 28.5 degrees of plane change, the swingby solution saves 188.01 fps of total Δv.



	$\|\Delta V_1\|$	$\|\Delta V_2\|$	Total (fps)
Hohmann	8056.67	5851.44	13908.12
Swingby	10201.39	3518.72	13720.11

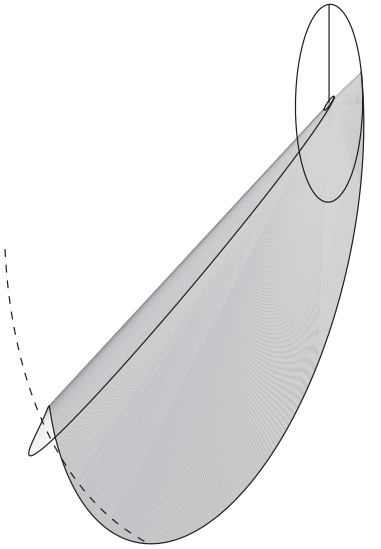
Figure 7.2. *Optimal swingby transfer to geosynchronous orbit.*

Polar, 24 hr (A)

Figure 7.3 illustrates the optimal lunar swingby trajectory to an orbit which is circular at an altitude of 19323 nm and inclination of 90 deg. This solution is characterized by an outbound transfer trajectory from the descending node of the park orbit and as such will be referred to as solution "A." The total plane change for this transfer is 61.5 deg, which is achieved using 3059.56 fps less than the corresponding Hohmann transfer.

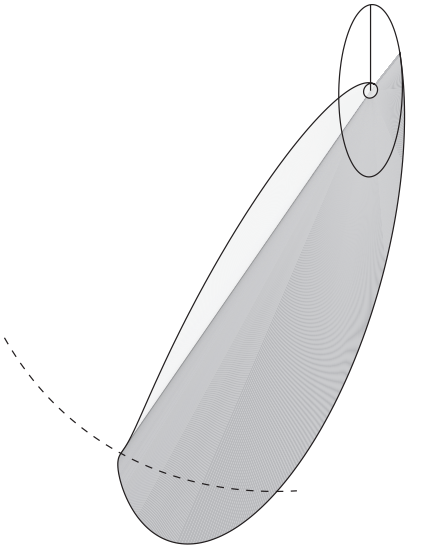
Polar, 24 hr (B)

Figure 7.4 illustrates the optimal lunar swingby trajectory to the same mission orbit as in section 7.1.2. In contrast to solution "A," this trajectory is characterized by an outbound



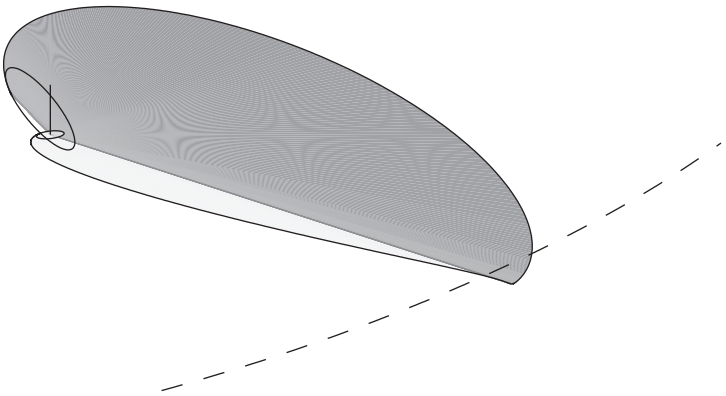
	$\|\Delta V_1\|$	$\|\Delta V_2\|$	Total (fps)
Hohmann	8113.34	8610.82	16724.17
Swingby	10225.16	3440.44	13665.61

Figure 7.3. *Optimal swingby transfer to polar,* 24 *hr orbit.*



	$\|\Delta V_1\|$	$\|\Delta V_2\|$	Total (fps)
Hohmann	8113.33	8610.77	16724.10
Swingby	10251.24	3459.66	13710.90

Figure 7.4. *Optimal swingby transfer to polar,* 24 *hr orbit.*



	$\|\Delta V_1\|$	$\|\Delta V_2\|$	Total (fps)
Hohmann	546.99	39682.76	40229.76
Swingby	10392.39	4826.32	15218.71

Figure 7.5. *Optimal swingby transfer to Molniya orbit.*

transfer from the ascending node of the park orbit and as such will be referred to as solution "B." Although the swingby solution "B" is slightly less efficient than solution "A" it still is 3013.20 fps less than the Hohmann transfer. It is also worth noting the very small difference in the Hohmann solutions between "A" and "B," which can be attributed to the small (yet different) lunar perturbations.

Retrograde Molniya

The final example as illustrated in Figure 7.5 is a retrograde Molniya mission orbit. Specifically the orbit is elliptical with an (osculating) apogee altitude of 21450 nm, a perigee altitude of 350 nm, an argument of perigee of 270 deg, and a retrograde inclination of 116.6 deg. This particular orbit has a period of approximately 12 hr and requires an orbital plane change of 88.1 deg. For this example the lunar swingby saves 25011.05 fps of total Δv. In this extreme case the Hohmann transfer is approximately 264% more expensive.

7.1.3 Equations of Motion

The dynamic behavior of a spacecraft can be modeled using a simplified version of the N-body problem as described in Battin [6]. It is convenient to use an earth centered Cartesian coordinate system, with boldface notation used to distinguish a vector from a scalar. Let $\mathbf{r}$ and $\mathbf{v}$ denote the position and velocity of the spacecraft, and let $\mathbf{r}_L$ and $\mathbf{v}_L$ the position and velocity of the moon.[8] The motion is described by

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{7.2}$$

$$\dot{\mathbf{v}} = -\frac{\mu_e}{r^3}\mathbf{r} + \mathbf{g}_L, \tag{7.3}$$

[8]Subscript convention: x_p, park; x_m, mission; x_L, lunar; $x_\imath$, inbound; x_o, outbound; x_s, swingby.

$$\dot{\mathbf{r}}_L = \mathbf{v}_L, \tag{7.4}$$

$$\dot{\mathbf{v}}_L = -\frac{\mu_\circ}{r_L^3}\mathbf{r}_L, \tag{7.5}$$

where $\|\mathbf{r}\| = r$, $\|\mathbf{r}_L\| = r_L$, and $\mu_\circ = \mu_e + \mu_L = Gm_e + Gm_L$, where G is the universal gravitation constant with m_e and m_L denoting the masses of the earth and moon, respectively. The disturbing acceleration caused by the gravitation of the moon is

$$\mathbf{g}_L = -\mu_L\left[\frac{1}{d^3}\mathbf{d} + \frac{1}{r_L^3}\mathbf{r}_L\right], \tag{7.6}$$

where

$$\mathbf{d} = \mathbf{r} - \mathbf{r}_L \tag{7.7}$$

is a vector from the moon to the vehicle with magnitude $\|\mathbf{d}\| = d$. To avoid losing precision when evaluating (7.6), Battin [6] suggests defining the function

$$F(q) = q\left[\frac{3 + 3q + q^2}{1 + (\sqrt{1+q})^3}\right], \tag{7.8}$$

where

$$q = \frac{\mathbf{r}^{\mathsf{T}}(\mathbf{r} - 2\mathbf{r}_L)}{\mathbf{r}_L^{\mathsf{T}}\mathbf{r}_L}. \tag{7.9}$$

The perturbing acceleration (7.6) is then given by

$$\mathbf{g}_L = -\frac{\mu_L}{d^3}\left[\mathbf{r} + F(q)\mathbf{r}_L\right]. \tag{7.10}$$

7.1.4 Kepler Orbit Propagation

As written the differential equations (7.4)–(7.5) constitute a *two-body* approximation to the lunar motion. It is well known that this system has an analytic solution, and for convenience we briefly summarize the approach (cf. [6], [78]). In general, the technique is applicable to any two-body system using the appropriate definitions for the gravitational constant $\mu_\circ$ and coordinates $(\mathbf{r}, \mathbf{v})$, and consequently we omit the subscript for generality. Given a Cartesian state vector $\mathbf{r}_\circ, \mathbf{v}_\circ$ at time $t_\circ$ called the *reference epoch* and a specified change in eccentric anomaly $\Delta E = E - E_\circ$, a new state vector $(\mathbf{r}, \mathbf{v})$, with corresponding time change $\Delta t = t - t_\circ$, can be computed as follows:

$$r_\circ = \|\mathbf{r}_\circ\|, \tag{7.11}$$

$$\sigma_\circ = \frac{1}{\sqrt{\mu_\circ}}\mathbf{r}_\circ^{\mathsf{T}}\mathbf{v}_\circ, \tag{7.12}$$

$$v_\circ^2 = \mathbf{v}_\circ^{\mathsf{T}}\mathbf{v}_\circ, \tag{7.13}$$

$$\frac{1}{a} = \frac{2}{r_\circ} - \left[\frac{v_\circ^2}{\mu_\circ}\right], \tag{7.14}$$

$$\rho = 1 - \frac{r_\circ}{a}, \tag{7.15}$$

$$C = a(1-\cos\Delta E), \tag{7.16}$$

$$S = \sqrt{a}\sin\Delta E, \tag{7.17}$$

$$F = 1 - \frac{C}{r_\circ}, \tag{7.18}$$

$$G = \frac{1}{\sqrt{\mu_\circ}}(r_\circ S + \sigma_\circ C), \tag{7.19}$$

$$r = r_\circ + \rho C + \sigma_\circ S, \tag{7.20}$$

$$F_t = -\frac{\sqrt{\mu_\circ}}{r r_\circ}S, \tag{7.21}$$

$$G_t = 1 - \frac{C}{r}, \tag{7.22}$$

$$\mathbf{r} = F\mathbf{r}_\circ + G\mathbf{v}_\circ, \tag{7.23}$$

$$\mathbf{v} = F_t\mathbf{r}_\circ + G_t\mathbf{v}_\circ, \tag{7.24}$$

$$\Delta t = \sqrt{\frac{a^3}{\mu_\circ}}\left[\Delta E + \frac{\sigma_\circ C}{a\sqrt{a}} - \rho\frac{S}{\sqrt{a}}\right]. \tag{7.25}$$

Notice that the sequence of calculations (7.11)–(7.25) constitutes an *explicit* definition for the states, as well as the corresponding time increment, which can be expressed as

$$\mathbf{r} = \mathbf{h}_r(\Delta E), \tag{7.26}$$

$$\mathbf{v} = \mathbf{h}_v(\Delta E), \tag{7.27}$$

$$\Delta t = h_t(\Delta E). \tag{7.28}$$

The propagation is explicit with respect to the independent variable ΔE but is *implicit* with respect to the time t. To be more precise let us rewrite (7.28) as

$$t_E = t_\circ + h_t(\Delta E) \tag{7.29}$$

since $\Delta t = t_E - t_\circ$, where t_E is the final time corresponding to the eccentric anomaly increment ΔE.

7.1.5 Differential-Algebraic Formulation of Three-Body Dynamics

The original system of 12 differential equations (7.2)–(7.5) can be recast as a differential-algebraic (DAE) system involving the six states $\mathbf{r}(t)$ and $\mathbf{v}(t)$ and the single algebraic variable $\Delta E(t)$:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{7.30}$$

$$\dot{\mathbf{v}} = -\frac{\mu_e}{r^3}\mathbf{r} + \mathbf{g}_L, \tag{7.31}$$

$$0 = t - t_\circ - h_t(\Delta E). \tag{7.32}$$

Observe that the implicit algebraic equation (7.32) is a modified form of Kepler's transcendental equation. The gravitational perturbation on the spacecraft $\mathbf{g}_L$ is defined by the position vectors $\mathbf{r}$ and $\mathbf{r}_L$ using (7.8)–(7.10). The lunar position vector is completely determined by (7.26) using the Kepler propagation procedure outlined in (7.11)–(7.25). Thus the complete functional dependency is given by

$$\mathbf{g}_L = \mathbf{g}_L(\mathbf{r}, \Delta E).$$

7.1.6 Boundary Conditions

Denote the state variables at a particular time (e.g., the beginning or end of the trajectory) by $\widetilde{\mathbf{r}}$ and $\widetilde{\mathbf{v}}$. Boundary conditions defining both the park and mission orbits are typically stated in terms of *osculating* orbit elements, i.e., nonlinear functions of $\widetilde{\mathbf{r}}$ and $\widetilde{\mathbf{v}}$. There are many equivalent ways to specify both the park and the mission orbits. A circular orbit with specified radius $\bar{r}$ can be achieved by imposing the following boundary constraints:

$$\bar{r} = \|\widetilde{\mathbf{r}}\|, \tag{7.33a}$$

$$\sqrt{\frac{\mu_e}{\bar{r}}} = \|\widetilde{\mathbf{v}}\|, \tag{7.33b}$$

$$0 = \frac{\widetilde{\mathbf{r}}^{\mathsf{T}}\widetilde{\mathbf{v}}}{\sqrt{\mu_e \bar{r}}}. \tag{7.33c}$$

Equation (7.33a) fixes the magnitude of the position vector, (7.33b) defines the circular velocity, and (7.33c) ensures the flight path angle (and eccentricity) is zero. Observe that when (7.33a) and (7.33b) are satisfied (7.33c) is just

$$\frac{\widetilde{\mathbf{r}}^{\mathsf{T}}\widetilde{\mathbf{v}}}{\sqrt{\mu_e \bar{r}}} = \frac{\widetilde{\mathbf{r}}^{\mathsf{T}}\widetilde{\mathbf{v}}}{\|\widetilde{\mathbf{r}}\|\|\widetilde{\mathbf{v}}\|},$$

which is equivalent to making the normalized position and velocity vectors orthogonal. However, as written the denominator in (7.33c) is a *constant*, which is the preferable, i.e., "more linear," way to pose the constraint.

To achieve a particular inclination $\bar{i}$ we must enforce the boundary condition

$$\cos\bar{i} = \widehat{\mathbf{h}}_3, \tag{7.34a}$$

where the angular momentum vector $\mathbf{h}$ and north vector $\mathbf{i}_z$ are

$$\mathbf{h} = \widetilde{\mathbf{r}} \times \widetilde{\mathbf{v}}, \tag{7.34b}$$

$$\widehat{\mathbf{h}} = \frac{\mathbf{h}}{\|\mathbf{h}\|}, \tag{7.34c}$$

$$\mathbf{i}_z = (0,0,1)^{\mathsf{T}}, \tag{7.34d}$$

$$\widehat{\mathbf{h}}_3 = \mathbf{i}_z^{\mathsf{T}} \widehat{\mathbf{h}}. \tag{7.34e}$$

Observe that (7.34a) is stated in terms of $\cos\bar{i}$ rather than the inclination $\bar{i}$ itself, thereby avoiding ambiguities when computing the inverse cosine.

If the desired orbit is elliptic, (7.33a)–(7.33c) are not applicable. Instead to achieve a specified semimajor axis $\bar{a}$, and eccentricity $\bar{e}$ (> 0), we must impose the boundary conditions

$$\bar{a} = \left(\frac{2}{\|\widetilde{\mathbf{r}}\|} - \frac{\|\widetilde{\mathbf{v}}\|^2}{\mu_e} \right)^{-1}, \tag{7.35a}$$

$$\bar{e} = \|\mathbf{e}\|, \tag{7.35b}$$

where the eccentricity vector is given by

$$\mathbf{e} = \frac{1}{\mu_e} (\widetilde{\mathbf{v}} \times \mathbf{h}) - \frac{\widetilde{\mathbf{r}}}{\|\widetilde{\mathbf{r}}\|}, \tag{7.35c}$$

and the angular momentum is given by (7.34b).

For elliptic orbits ($\|\mathbf{e}\| > 0$), the eccentricity vector is directed toward periapsis, thereby defining the orientation of the principal axis. In general, the argument of periapsis ω is an angle measured in the orbital plane from the ascending node to periapsis. The particular case $\bar{\omega} = 270^\circ$ can be enforced by imposing the conditions

$$0 = \widehat{\mathbf{e}}^{\mathsf{T}} \mathbf{i}_n, \tag{7.36a}$$

$$0 > \widehat{\mathbf{e}}_3, \tag{7.36b}$$

where

$$\widehat{\mathbf{e}} = \frac{\mathbf{e}}{\bar{e}}, \tag{7.36c}$$

$$\widehat{\mathbf{e}}_3 = \mathbf{i}_z^{\mathsf{T}} \widehat{\mathbf{e}}, \tag{7.36d}$$

$$\mathbf{i}_n = \mathbf{i}_z \times \widehat{\mathbf{h}}, \tag{7.36e}$$

and $\mathbf{e}$ is given by (7.35c), with $\widehat{\mathbf{h}}$ defined by (7.34c), and $\mathbf{i}_z$ from (7.34d). Again notice that the denominator of (7.36c) involves the *constant* $\bar{e}$ instead of the quantity $\|\mathbf{e}\|$ appearing on the right-hand side of (7.35b).

7.1.7 A Four-Step Solution Technique

To compute a solution to an *optimal lunar swingby* let us first pose a sequence of "easier" subproblems. It is important to remark that the proposed technique is a heuristic—other equally valid methods can be postulated. However, the heuristic has proven to be both reliable and efficient. Section 7.1.8 will describe *how* each subproblem is solved.

The *four-step solution technique* can be summarized as follows:

Step 1: Three-Impulse, Conic Solution
Solve a small NLP with analytic propagation ignoring lunar gravity.

Step 2: Three-Body Approximation to Conic Solution
Solve an "inverse problem" to fit three-body dynamics to the conic solution.

Step 3: Optimal Three-Body Solution with Fixed Swingby Time
Use the solution from Step 2 to initialize.

Step 4: Optimal Three-Body Solution
Compute solution with free swingby time, using Step 3 as an initial guess.

Step 1: Three-Impulse, Conic Solution

Since the optimal trajectory depends on the relative geometry of three bodies, namely the earth, moon, and spacecraft, it is critical that the solution process be initiated with a reasonable geometric configuration. This goal can be achieved by using a very simplified model of the dynamics. In particular for Step 1, it is assumed that the spacecraft dynamics are determined entirely by the gravitational attraction of the primary body, earth. Further, it is assumed that the lunar swingby can be approximated by a simple velocity increment. Because all of the dynamics are modeled using a two-body approximation, the analytic trajectory propagation approach outlined in Section 7.1.4 can be exploited. With these simplifying assumptions one can pose the following very simple NLP problem.

Optimization Variables. The problem can be formulated using 24 variables to define the trajectory as illustrated in Figure 7.6:

$$(\mathbf{r}_o, \mathbf{v}_o, \Delta\mathbf{v}_1, \Delta E_o) \quad \text{State at Park Orbit Departure,} \tag{7.37}$$

$$(\mathbf{r}_i, \mathbf{v}_i, \Delta\mathbf{v}_2, \Delta E_i) \quad \text{State at Mission Orbit Arrival,} \tag{7.38}$$

$$(\Delta\mathbf{v}_s, \Delta E_L) \quad \text{Swingby Velocity Increment and Lunar Angle to Intercept.} \tag{7.39}$$

The outbound (earth to moon) transfer orbit is defined by the variables in (7.37), namely the Cartesian state at the beginning of the outbound transfer $\mathbf{r}_o, \mathbf{v}_o$, the impulsive velocity $\Delta\mathbf{v}_1$, and the eccentric anomaly change for the outbound trajectory ΔE_o. Equation (7.38) defines the corresponding variables for the inbound (moon to earth) transfer. The velocity increment provided by the moon at swingby is defined as $\Delta\mathbf{v}_s$, and the location of the moon relative to a reference epoch is defined by the angle ΔE_L.

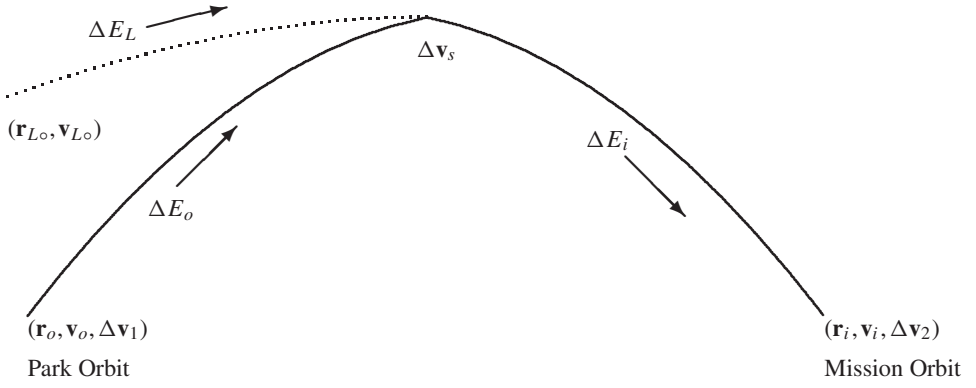


Figure 7.6. *Three-impulse, conic solution.*

Park Orbit Conditions. In order to enforce the boundary conditions at the park orbit, the following NLP constraints must be imposed:

$$\mathbf{r}_p = \mathbf{r}_o \qquad \text{Position Continuity,} \tag{7.40}$$

$$\mathbf{v}_p = \mathbf{v}_o - \Delta \mathbf{v}_1 \qquad \text{Impulsive Velocity Change,} \tag{7.41}$$

$$\boldsymbol{\phi}_p\left(\mathbf{r}_p, \mathbf{v}_p\right) = \mathbf{0} \qquad \text{Park Orbit Constraints.} \tag{7.42}$$

Equation (7.40) ensures that the park orbit and outbound transfer orbit position is continuous. The impulsive velocity change at departure is enforced by (7.41). The park orbit state vector must also satisfy the nonlinear constraints denoted $\boldsymbol{\phi}_p$ in (7.42). For the circular park orbit illustrated here, the vector $\boldsymbol{\phi}_p$ is given by (7.33a), (7.33b), (7.33c), and (7.34a).

Mission Orbit Conditions. The boundary conditions imposed by the mission orbit lead to a set of NLP constraints similar to those enforced at departure, namely,

$$\mathbf{r}_m = \mathbf{r}_i \qquad \text{Position Continuity,} \tag{7.43}$$

$$\mathbf{v}_m = \mathbf{v}_i + \Delta \mathbf{v}_2 \qquad \text{Impulsive Velocity Change,} \tag{7.44}$$

$$\boldsymbol{\phi}_m\left(\mathbf{r}_m, \mathbf{v}_m\right) = \mathbf{0} \qquad \text{Mission Orbit Constraints.} \tag{7.45}$$

Constraints (7.43) and (7.44) are analogous to (7.40) and (7.41). The mission orbit constraints denoted by the vector $\boldsymbol{\phi}_m$ in (7.45) are computed using the appropriate expressions from Section 7.1.6.

Lunar Conditions. The conditions at the moon used to approximate the lunar swingby all involve expressions for the state vector computed using the Kepler propagation described by (7.26) and (7.27):

$$\mathbf{h}_r(\mathbf{r}_o, \mathbf{v}_o, \Delta E_o) = \mathbf{h}_r(\mathbf{r}_i, \mathbf{v}_i, \Delta E_i) \qquad \text{Outbound/Inbound Position,} \tag{7.46}$$

$$\mathbf{h}_r(\mathbf{r}_o, \mathbf{v}_o, \Delta E_o) = \mathbf{h}_r(\mathbf{r}_{L\circ}, \mathbf{v}_{L\circ}, \Delta E_L) \qquad \text{Outbound/Lunar Position,} \tag{7.47}$$

$$\begin{aligned} \mathbf{h}_v(\mathbf{r}_i, \mathbf{v}_i, \Delta E_i) &= \mathbf{h}_v(\mathbf{r}_o, \mathbf{v}_o, \Delta E_o) \\ &\quad + \mathbf{h}_v(\mathbf{r}_{L\circ}, \mathbf{v}_{L\circ}, \Delta E_L) + \Delta \mathbf{v}_s \end{aligned} \qquad \text{Velocity Change.} \tag{7.48}$$

The constraints (7.46) force the outbound and inbound transfer trajectories to have the same position at the swingby. Observe that the outbound position $\mathbf{h}_r$ is completely determined by the departure state $\mathbf{r}_o, \mathbf{v}_o$, and the outbound eccentric anomaly change ΔE_o, with similar comments applicable to the inbound leg. While (7.46) ensures continuity between the outbound and inbound trajectories, it is also necessary that the swingby occur at the moon's position. This is achieved using constraint (7.47). Of course this constraint will locate the swingby at the *center* of the moon, which can be achieved only by ignoring the lunar gravity. Finally, one must model the velocity change at the moon, and this is expressed by (7.48). Observe that (7.48) involves the Kepler velocities $\mathbf{h}_v$ and an impulsive change $\Delta\mathbf{v}_s$. This constraint requires the velocity *after* the swingby to equal the vector sum of (a) the velocity *before* the swingby, plus (b) the velocity of the moon, plus (c) the impulsive change.

Objective. The objective function for this simplified model problem is to *minimize* the total Δv, i.e.,

$$F = \|\Delta\mathbf{v}_1\| + \|\Delta\mathbf{v}_2\| + \|\Delta\mathbf{v}_s\|. \tag{7.49}$$

The solution to this problem should define a geometric configuration of the earth-moon system that is optimal when ignoring the lunar gravitational effects.

Step 2: Three-Body Approximation

The solution to the simplified model problem computed in Step 1 is designed to construct an approximate solution to the overall problem. Unfortunately, by design, the simplified model ignores one of the primary dynamic aspects of the real problem. Specifically, the conic solution solves

$$\ddot{\mathbf{r}} = -\frac{\mu_e}{r^3}\mathbf{r}, \tag{7.50a}$$

$$\ddot{\mathbf{r}}_L = -\frac{\mu_\circ}{r_L^3}\mathbf{r}_L \tag{7.50b}$$

but *not* the three-body dynamics

$$\ddot{\mathbf{r}} = -\frac{\mu_e}{r^3}\mathbf{r} + \mathbf{g}_L, \tag{7.51a}$$

$$\ddot{\mathbf{r}}_L = -\frac{\mu_\circ}{r_L^3}\mathbf{r}_L. \tag{7.51b}$$

Let us denote the conic solution from (7.50a) by $\widetilde{\mathbf{r}}(t)$. If one assumes that the conic trajectory is approximately correct, then it is reasonable to find a "nearby" trajectory that does satisfy the correct dynamics given by (7.51a). This goal can be achieved by "fitting" the three-body trajectory to the conic, i.e., by minimizing

$$F = \sum_{k=1}^{N} \|\mathbf{r}_k - \widetilde{\mathbf{r}}_k\|^2 \tag{7.52}$$

subject to

$$\ddot{\mathbf{r}} = -\frac{\mu_e}{r^3}\mathbf{r} + \mathbf{g}_L, \tag{7.53}$$

$$\ddot{\mathbf{r}}_L = -\frac{\mu_\circ}{r_L^3}\mathbf{r}_L, \tag{7.54}$$

$$r_{Lmin} \le \|\mathbf{r} - \mathbf{r}_L\|, \tag{7.55}$$

where $\widetilde{\mathbf{r}}_k = \widetilde{\mathbf{r}}(t_k)$ is the spacecraft position on the conic trajectory and $\mathbf{r}_k = \mathbf{r}(t_k)$ is the state evaluated at the same time points t_k on the three-body trajectory. Observe that in this *inverse problem* an algebraic inequality constraint (7.55) is included to ensure the trajectory is sufficiently above the lunar surface where r_{Lmin} is a lower bound on the distance from the spacecraft to the moon. The conic solution serves as a good initial guess for this inverse problem, so it is reasonable to set $\mathbf{r}_k^{(0)} = \widetilde{\mathbf{r}}_k$, provided the N data points *exclude* the single time point at the moon. Clearly the lunar acceleration $\mathbf{g}_L$ cannot be evaluated when $\|\mathbf{d}\| = \|\mathbf{r} - \mathbf{r}_L\| = \mathbf{0}$ in (7.6).

Step 3: Fixed Swingby Time

The solution obtained from Step 2 provides an excellent initial guess for Step 3. In particular it supplies a time history for $\mathbf{r}(t), \mathbf{v}(t)$ that satisfies the full three-body dynamic equations (7.2)–(7.5). Furthermore, by construction the boundary conditions at the park and mission orbits will be approximately satisfied. Thus one can pose a problem with two distinct phases described by either the three-body dynamic equations (7.2)–(7.5) or the differential-algebraic system (7.30)–(7.32). Figure 7.7 illustrates the dynamic simulation.

Phase 1: *Outbound Transfer*. The outbound transfer begins at the free initial time t_I, which must satisfy the following park orbit conditions analogous to those used for Step 1:

$$\mathbf{r}_p = \mathbf{r}_o \qquad \text{Position Continuity,} \tag{7.56}$$

$$\mathbf{v}_p = \mathbf{v}_o - \Delta\mathbf{v}_1 \qquad \text{Impulsive Velocity Change,} \tag{7.57}$$

$$\boldsymbol{\phi}_p\left(\mathbf{r}_p, \mathbf{v}_p\right) = \mathbf{0} \qquad \text{Park Orbit Constraints.} \tag{7.58}$$

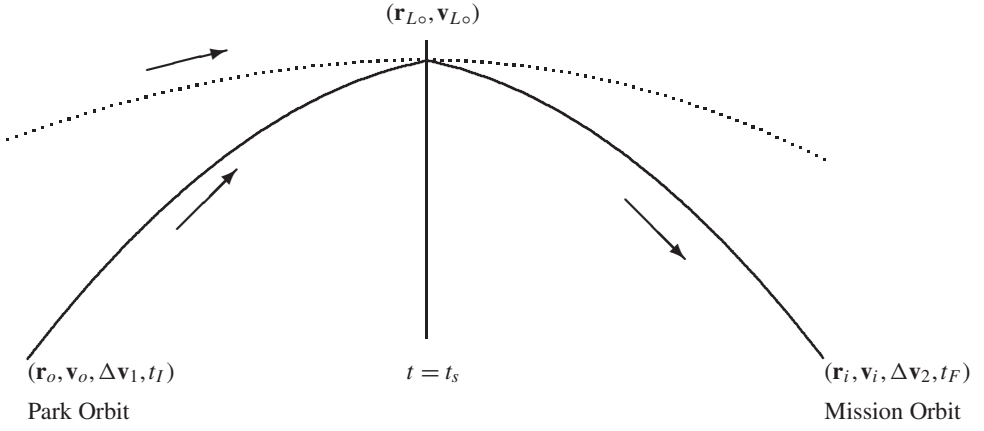


Figure 7.7. *Fixed swingby time solution.*

The phase terminates at the fixed time t_s, which must satisfy the lunar conditions

$$\|\mathbf{r}-\mathbf{r}_L\| \geq r_{Lmin} \qquad \text{Closest Approach,} \tag{7.59}$$

$$(\mathbf{v}-\mathbf{v}_L)^{\mathsf{T}}(\mathbf{r}-\mathbf{r}_L)=0 \qquad \text{Lunar Flight Path Angle.} \tag{7.60}$$

Observe that by forcing the lunar flight path angle to be zero (7.60) ensures that the closest approach to the moon will occur at t_s.

Phase 2: *Inbound Transfer.* The inbound transfer begins at the fixed time t_s and must satisfy the conditions

$$(\mathbf{r},\mathbf{v},\mathbf{r}_L,\mathbf{v}_L)^{(-)}=(\mathbf{r},\mathbf{v},\mathbf{r}_L,\mathbf{v}_L)^{(+)} \qquad \text{State Continuity,} \tag{7.61}$$

$$(\mathbf{r}_L,\mathbf{v}_L)=(\overline{\mathbf{r}}_L,\overline{\mathbf{v}}_L) \qquad \text{Lunar State.} \tag{7.62}$$

Equation (7.61) ensures continuity in all of the states across the phase boundary, and since the time t_s is fixed (7.62) enforces consistency with the lunar reference epoch $(\overline{\mathbf{r}}_L,\overline{\mathbf{v}}_L)$.

Phase 2 terminates at the free time t_F and at that point must satisfy the mission orbit conditions

$$\mathbf{r}_m=\mathbf{r}_i \qquad \text{Position Continuity,} \tag{7.63}$$

$$\mathbf{v}_m=\mathbf{v}_i+\Delta\mathbf{v}_2 \qquad \text{Impulsive Velocity Change,} \tag{7.64}$$

$$\boldsymbol{\phi}_m(\mathbf{r}_m,\mathbf{v}_m)=\mathbf{0} \qquad \text{Mission Orbit Constraints,} \tag{7.65}$$

$$t_{max}\geq t_F-t_I \qquad \text{Mission Duration.} \tag{7.66}$$

The objective for this dynamic optimization problem is to minimize

$$F=\|\Delta\mathbf{v}_1\|+\|\Delta\mathbf{v}_2\|. \tag{7.67}$$

Step 4: Optimal Three-Body Solution

The completion of Step 3 will provide an excellent initial guess for the full optimal three-body lunar swingby. Indeed, only two modifications to the formulation in Step 3 are required. First, of course, the time of closest approach to the moon t_s must be free. Second, one must ensure that the lunar state at the (free) initial time is consistent with the reference epoch for the moon. Thus at the beginning of Phase 1, the following additional boundary conditions must be satisfied:

$$\mathbf{r}_L=\mathbf{h}_r(\mathbf{r}_\circ,\mathbf{v}_\circ,\Delta E_\circ), \tag{7.68}$$

$$\mathbf{v}_L=\mathbf{h}_v(\mathbf{r}_\circ,\mathbf{v}_\circ,\Delta E_\circ). \tag{7.69}$$

Typically any convenient reference epoch for the moon can be chosen.

7.1.8 Solving the Subproblems

The preceding sections posed a set of subproblems that can be solved to obtain an optimal lunar swingby trajectory. But how does one efficiently solve the subproblems? In particular it is necessary to solve

- *an optimal control problem* in Steps 3 and 4, using the SNLP method presented in Chapter 4,
- *a parameter estimation (inverse) problem* in Step 2, using the algorithm described in Chapter 5, and
- *an NLP problem* in Step 1, using the methods in Chapter 1.

There are a number of efficiency issues that must be addressed within the context of an SNLP algorithm. The second step of the algorithm requires the solution of an NLP; however, among the many possible NLP algorithms it is desirable to choose the most efficient. The computational experience as discussed in Section 4.13 provides some insight to guide the choice. To reinforce the observations of Section 4.13, let us compare the performance of two different NLP algorithms when solving the small, dense NLP subproblems required by Step 1 (Section 7.1.7). Table 7.1 summarizes the performance of two different NLP algorithms, when used to solve three different NLP problems. In particular results are presented for the sequential quadratic programming (SQP) algorithm described in Chapter 1. Two different techniques were used to compute the Hessian matrix, namely a finite difference approximation denoted as "Newton" and a quasi-Newton BFGS (Broyden–Fletcher–Goldfarb–Shanno) recursive update. The second algorithm is the primal-dual interior-point or barrier algorithm described in Chapter 2. This algorithm was also tested using Newton and BFGS Hessian approximations.

Table 7.1. *NLP algorithm performance comparison.*

Step 1–Small, Dense NLP Subproblems

Mission	Equatorial	Polar	Molniya
SQP-Newton	(10,4)	(16,10)	(135,44)
SQP-BFGS	(24,19)	(36,31)	(186,96)
Barrier-Newton	(24,22)	(57,55)	(70,68)†
Barrier-BFGS	(242,241)†	(58,56)	(286,284)

Key: (Gradient Eval., Hessian Eval.) † No Solution

For each algorithm the table presents the number of gradient and Hessian evaluations needed to reach a solution. Although the number of problems in this test set is quite small, the basic findings are consistent with much more extensive testing as described in [33]. Generally the SQP algorithm was both more efficient and more robust than the barrier method. Although it is difficult to prove, one speculates that an SQP method is simply a better choice for solving very nonlinear optimization problems as discussed in Section 4.13. Conversely, our experience suggests a barrier algorithm may be preferable for problems with linear constraints, especially when there are many inequalities. The testing also suggests that a quasi-Newton Hessian approximation requires significantly more iterations to converge. While this is not surprising, it is extremely important when considering the very large, sparse NLP problems that arise when using discretization methods for optimal control.

A quasi-Newton Hessian approximation suffers from another deficiency first mentioned in Section 4.13 which is not demonstrated by this comparison. The results given utilize a quasi-Newton approximation to the *full* Hessian. Unfortunately this can become

prohibitively expensive for large-scale optimization problems, especially when there are many degrees of freedom.

A second, more serious performance issue occurs when comparing NLP algorithms for use within the context of an SNLP. Specifically a *barrier algorithm cannot exploit a good guess* as discussed in Section 4.13. To illustrate this point consider the solution of the optimal control subproblem for the polar mission summarized in Tables 7.2 and 7.3. Both cases were initiated using the same information. Specifically the initial trajectory was constructed as the solution from a Step 2 inverse problem. Using this three-body solution trajectory a variable stepsize numerical integration algorithm was used to construct the initial grid points. Referring to Table 7.2, the first grid had 594 points, leading to an NLP problem with 7136 optimization variables and 7715 nonlinear constraints. The solution to this coarse grid problem requires 18 gradient evaluations (NGC), 10 Hessian evaluations (NHC), and 3794 function evaluations (NFE), including those needed for finite difference derivatives. The resulting solution had a relative discretization error of $\epsilon = 1 \times 10^{-4}$ and was computed in 30.1 sec. The grid was refined two times, with the final grid containing 1113 points. Table 7.3 presents exactly the same history, when a barrier algorithm is used to solve the NLP subproblems. For the SQP algorithm as the mesh is refined, the number of Hessian evaluations and iterations decreases—only 1 Hessian is needed on the second and third grids. Each coarse grid solution provides a very good guess, and the Newton method is within its region of quadratic convergence. For the barrier method this is not true! The second mesh required an additional 49 Hessian evaluations because the initial guess was perturbed. The overall penalty in computation time is catastrophic in this example. In addition, because the initial guesses were perturbed the barrier algorithm converged to a different local solution than did the SQP algorithm. All of the computational results were obtained using a Dell M60 laptop computer, with a Linux operating system.

Table 7.2. *Mesh refinement with an SQP algorithm.*

				SQP				
k	M	n	m	NGC	NHC	NFE	ϵ	Time (sec)
1	594	7136	7715	18	10	3794	1×10^{-4}	30.1
2	881	10580	11446	4	1	454	4×10^{-7}	7.8
3	1113	13364	14462	4	1	454	1×10^{-8}	10.6
Total				26	12	4702		48.6

Table 7.3. *Mesh refinement with a barrier algorithm.*

				Barrier†				
k	M	n	m	NGC	NHC	NFE	ϵ	Time (sec)
1	594	7720	7715	328	319	112475	1×10^{-5}	749.3
2	881	12985	12980	57	49	17637	2×10^{-7}	233.7
3	1113	14090	14085	6	2	881	1×10^{-8}	18.5
Total				391	370	130993		1001.6

† Different Local Solution than SQP

Is Mesh Refinement Needed?

In light of the apparent conflict between mesh refinement and a barrier algorithm it is important to review why mesh refinement is needed. Let us consider the solution of the

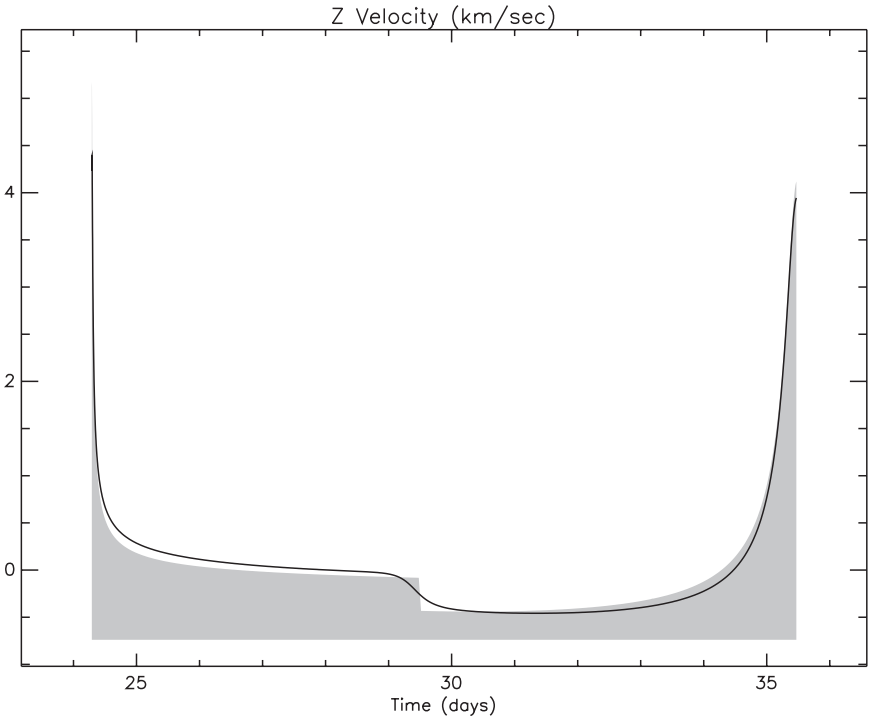


Figure 7.8. *Velocity discontinuity.*

Step 2 inverse problem solution for the polar mission. To review, the conic trajectory available from the solution of Step 1 can be used to construct an initial guess. In particular by sampling the conic at 600 equal ΔE increments, one obtains an NLS problem with 1800 residuals. To avoid a singularity, it is necessary to omit the point at the moon. The shaded region in Figure 7.8 illustrates the discontinuous behavior in one component of the velocity from the conic trajectory, and the solid line shows the smoothed approximation that results after "fitting" a three-body solution to the conic. Table 7.4 summarizes the behavior of the SOCS mesh-refinement procedure for this example, and Figure 7.9 illustrates what the procedure does to both the discretization error and the mesh distribution. Initially the discretization error is very large in the vicinity of the moon. Clearly, this error can be attributed to the approximate nature of the conic trajectory—i.e., the position goes through the center of the moon and the velocity change is impulsive. During the first few refinement iterations, grid points are added in the vicinity of the discontinuity, and this leads to a significant reduction in the discretization error as measured by the value of ϵ in Table 7.4. In fact after the first refinement iteration only six grid points were added, all in the neighborhood of the discontinuity, and this reduced the discretization error by nearly four orders of magnitude. It is also worth noting that solving the NLP subproblem after adding these grid points was significantly more expensive (taking 17 Hessian evaluations), because the entire solution had to be adjusted to account for this effect. Clearly, mesh refinement is needed in order to address the singularities in the vicinity of the moon.

Table 7.4. *Mesh refinement.*

k	M	n	m	NGC	NHC	NFE	ϵ	Time (sec)
1	599	7188	7775	12	2	144	6×10^{-1}	3.5
2	606	7272	7866	21	17	525	9×10^{-5}	1.1
3	606	7272	7866	4	2	724	1×10^{-6}	7.6
4	742	8904	9634	4	1	448	1×10^{-8}	5.6
Total				41	22	1841		27.7

k	Refinement No	M	Grid Pts	n	NLP vars
m	NLP cons.	NGC	Grad Eval	NHC	Hess Eval
NFE	Func Eval	ϵ	Disc. Error	Time	CPU

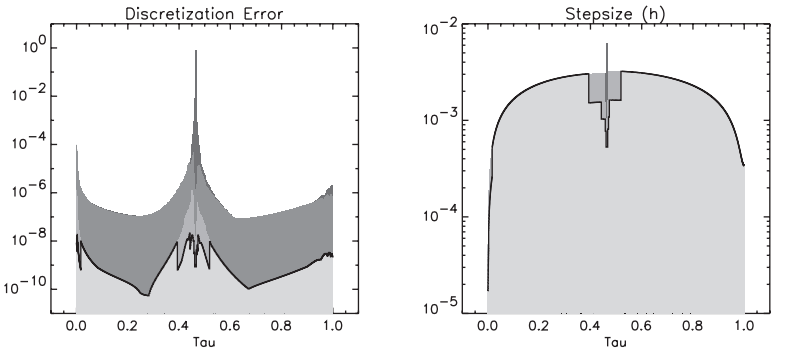


Figure 7.9. *Mesh refinement.*

DAE or ODE Formulation?

The three-body dynamics of the system can be described by the ODEs summarized in Section 7.1.3. However, the differential-algebraic system presented in Section 7.1.5 can also be used when solving the subproblems in Steps 2, 3, and 4. Is one formulation preferable to the other in terms of either computational speed and/or solution accuracy? To address this question, Tables 7.5 and 7.6 compare the formulations. Specifically, the optimal control problems in Steps 3 and 4 for the Molniya mission were solved using the ODE and DAE formulations. Both were initialized with the same Step 2 solution trajectory.

The comparison reveals a number of issues. First, the DAE formulation required a final grid with 2056 points, whereas the ODE formulation achieved the same accuracy with 1190 points. There are 12 ODEs and the DAE system has only 7 equations. Since the number of grid points is related to the nonlinearity of the equations as well as the order of interpolation, this suggests the DAE system may be more nonlinear. However, the total solution time is dictated by the total number of NLP variables in the discretized subproblem. The ODE formulation requires 14291 variables for the final iteration. In contrast, the DAE formulation needs 16456 variables. Thus the size of the NLP problems is nearly the same, even though one formulation involves nearly twice as many dynamic equations. Overall, there was no clearcut difference between the ODE and DAE formulations in either speed or accuracy for the applications considered here.

Table 7.5. *Steps* 3 *and* 4*, optimal solution for Molniya mission.*

ODE Formulation

M	n	NGC	NHC	Time (sec)
630	7568	26	7	32.02
807	9692	7	2	12.30
940	11288	4	1	8.66
1190	14288	4	1	11.22
1190	14291	10	4	45.49

Total Time = 109.69 sec

Table 7.6. *Steps* 3 *and* 4*, optimal solution for Molniya mission.*

DAE Formulation

M	n	NGC	NHC	Time (sec)
1097	8782	30	14	45.27
1530	12246	4	1	7.98
2056	16454	4	1	11.82
2056	16456	7	2	31.02

Total Time = 96.09 sec

Reference Epoch

The reference epoch for all numerical results corresponds to a Julian date of 2453561.5 (July 10, 2005). All results utilize an ICRF (International Celestial Reference Frame). The lunar ephemeris was constructed using a least squares fit of two-body dynamics to the JPL DE405 lunar ephemeris [161] over a 60 day period beginning at the reference epoch. The resulting lunar state vector and corresponding equatorial radii and gravitational constants are given in Table 7.7.

Table 7.7. *Mission parameters.*

μ_e	398600.436380820	km^3/sec^2
R_e	6378.14000000000	km
μ_L	4902.79881586123	km^3/sec^2
R_L	1737.40000000000	km
r_{ox}	$-.3398817123704749\times10^6$	km
r_{oy}	$.1956358580374241\times10^6$	km
r_{oz}	$.1139974125070158\times10^6$	km
v_{ox}	$-.5195857465292167\times10^0$	km/sec
v_{oy}	$-.7186784200912856\times10^0$	km/sec
v_{oz}	$-.3742849669631482\times10^0$	km/sec

7.2 Multiple-Pass Aero-Assisted Orbit Transfer

Example 7.1 MULTIPLE-PASS AERO-ASSISTED ORBIT TRANSFER. A favorite summer pastime while at a seaside beach or lakefront is "stone skipping." As a flat stone hits the surface of the water a rapid change in direction takes place that alters the motion for the next "skip," and a "good throw" will result in many skips before the stone loses energy. An analogous situation occurs in orbit mechanics when a spacecraft reenters the atmosphere. When a spacecraft is outside of the sensible atmosphere, i.e., *exo-atmospheric*, there are no aerodynamic forces on it. However, as the vehicle gets closer to the earth, the *endo-atmospheric* motion is dominated by aerodynamic effects. This rapid change in the environment can dramatically change the path of the vehicle, much like a stone hitting the surface of a lake. Trajectories of this type present challenging optimization problems, primarily because the environmental state of the vehicle changes during the dynamic process. In fact, very similar computational issues arise when modeling chemical or thermodynamic systems that change state during the course of a process. The concept of a phase becomes critical when constructing a mathematical model of such dynamic systems.

Many authors have studied the use of atmospheric forces to achieve orbital plane change. An extensive study of the subject, as well as many pertinent references, can be found in the paper by Rao, Tang, and Hallman [149]. For the particular scenario considered here, the motion begins with the vehicle in orbit. After using a rocket to slow the spacecraft, it reenters the earth's atmosphere and then skips out again, much like a stone on a lake. After executing three additional passes through the atmosphere, the spacecraft propulsion system is used to add velocity, thereby inserting the vehicle into a different orbit. The basic goal is to change the orbit of the vehicle and minimize the fuel required to execute the transfer. To be specific let us focus on one of the cases described in [149]. In particular, the spacecraft begins in a geosynchronous orbit which is circular with zero inclination at an altitude of 19323 nm.[9] After four passes through the atmosphere, it is required that the vehicle be in a circular orbit at an altitude of 100 nm with an inclination of 89 deg. The goal is to minimize the fuel consumed to perform the transfer.

This orbit transfer can be modeled using nine distinct phases alternating between orbital (exo-atmospheric) and atmospheric (endo-atmospheric). During phases 1, 3, 5, 7, and 9 the dynamics do not include aerodynamic forces and are described in Section 7.2.1. The dynamics used in phases 2, 4, 6, and 8 are presented in Section 7.2.2.

7.2.1 Orbital Phases

As in [149] let us use a spherical nonrotating earth model, in which case the orbital motion can be defined by the simple two-body dynamics

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{7.70}$$

$$\dot{\mathbf{v}} = -\frac{\mu}{r^3}\mathbf{r}, \tag{7.71}$$

where $\mathbf{r}$ is the earth centered inertial (ECI) position vector and $\mathbf{v}$ is the ECI velocity vector. This system can be solved analytically as described in Section 7.1.4. Given a Cartesian state vector $\mathbf{r}_\circ, \mathbf{v}_\circ$ at time $t_\circ$ and a specified change in eccentric anomaly $\Delta E = E - E_\circ$, a

[9] 1 nautical mile = 6076.1154855643 ft.

new state vector $(\mathbf{r},\mathbf{v})$ with corresponding time change $\Delta t = t - t_\circ$ can be computed; i.e., from (7.26)–(7.28) we can write

$$\mathbf{r} = \mathbf{h}_r(\mathbf{r}_\circ,\mathbf{v}_\circ,\Delta E), \tag{7.72}$$

$$\mathbf{v} = \mathbf{h}_v(\mathbf{r}_\circ,\mathbf{v}_\circ,\Delta E), \tag{7.73}$$

$$\Delta t = h_t(\mathbf{r}_\circ,\mathbf{v}_\circ,\Delta E). \tag{7.74}$$

Thus each orbital phase can be propagated analytically from the state at the beginning of the phase $(\mathbf{r}_\circ,\mathbf{v}_\circ)$ to the final state $(\mathbf{r},\mathbf{v})$. In fact, for this application it is not necessary to compute the time change Δt (except for display), so it is convenient to simply set the initial eccentric anomaly $E_\circ = 0$ and treat ΔE as the independent variable during the orbital phases.

7.2.2 Atmospheric Phases

During the atmospheric phases (2, 4, 6, and 8), it is more convenient to express the dynamics using an intrinsic or flight path coordinate system as in (6.1)–(6.6). In general, atmospheric phase k is defined in the domain $t_k^+ \le t \le t_{k+1}^-$. However, since the atmospheric dynamics are independent of time it is convenient to model the dynamics with respect to the beginning of the phase and simply set $t_k^+ = 0$ in a manner similar to the orbital phases where $E_k^+ = 0$. The state vector $(h,\phi,\theta,v,\gamma,\psi)$ is comprised of the altitude, longitude, geocentric latitude, velocity, flight path angle, and azimuth, respectively. The dynamics are described by the DAEs

$$\dot h = v\sin\gamma, \tag{7.75}$$

$$\dot\phi = \frac{v\cos\gamma\sin\psi}{r\cos\theta}, \tag{7.76}$$

$$\dot\theta = \frac{v\cos\gamma\cos\psi}{r}, \tag{7.77}$$

$$\dot v = -\frac{D}{m} - g\sin\gamma, \tag{7.78}$$

$$\dot\gamma = -\frac{1}{v}\left[\frac{qS}{m}u_2 + \left(g - \frac{v^2}{r}\right)\cos\gamma\right], \tag{7.79}$$

$$\dot\psi = \frac{1}{v}\left[\frac{-qS}{m\cos\gamma}u_1 + \frac{v^2}{r}\cos\gamma\sin\psi\tan\theta\right], \tag{7.80}$$

$$C_{LU} \ge C_L, \tag{7.81}$$

$$Q_U \ge Q, \tag{7.82}$$

where the stagnation point heating rate (BTU/(ft^2 sec)) is computed using the equation

$$Q = 17600\left(\frac{\rho}{\rho_E}\right)^{\frac{1}{2}}\left(\frac{v}{v_E}\right)^{3.15} \tag{7.83}$$

and the control variables (u_1,u_2) are defined in terms of the lift coefficient C_L and bank

angle β by

$$u_1 = -C_L \sin\beta, \tag{7.84}$$
$$u_2 = -C_L \cos\beta \tag{7.85}$$

with the inverse transformations given by

$$C_L = \sqrt{u_1^2 + u_2^2}, \tag{7.86}$$
$$\beta = \tan^{-1}(u_1/u_2). \tag{7.87}$$

The additional quantities

$$q = \frac{1}{2}\rho v^2, \tag{7.88}$$
$$D = qSC_D, \tag{7.89}$$
$$L = qSC_L, \tag{7.90}$$
$$C_D = C_{D0} + KC_L^2, \tag{7.91}$$
$$r = h + R_E, \tag{7.92}$$
$$g = \frac{\mu}{r^2}, \tag{7.93}$$
$$\alpha = \frac{C_L}{C_{L\alpha}} \tag{7.94}$$

complete the description of the dynamic model. The atmospheric density ρ is computed using a smooth atmosphere model [46], and the sensible limit of the atmosphere is assumed to be 60 nm. It is important to note that the particular choice of control variables (7.84)–(7.85) suggested in [149] is preferable to lift coefficient and bank angle (C_L, β), which is the more obvious engineering choice. The "winding" problem associated with angles as illustrated in Figure 6.15 is avoided. This is particularly important in order to achieve robust convergence for this application. The peak heating rate is limited by the path inequality constraint (7.82) to the value $Q_U = 400$ BTU/(ft^2 sec). The parametric values given in Table 7.8 complete the definition of the dynamic model.

To improve robustness it is also useful to limit the dynamic variables by imposing the following simple bounds:

$$0 \le h(t) \le 60\,\text{nm}, \tag{7.95a}$$
$$170\,\text{deg} \le \phi(t) \le 190\,\text{deg}, \tag{7.95b}$$
$$-20\,\text{deg} \le \theta(t) \le 80\,\text{deg}, \tag{7.95c}$$
$$25000\,\text{ft/sec} \le v(t) \le 35000\,\text{ft/sec}, \tag{7.95d}$$
$$-5\,\text{deg} \le \gamma(t) \le 5\,\text{deg}, \tag{7.95e}$$
$$0\,\text{deg} \le \psi(t) \le 40\,\text{deg}, \tag{7.95f}$$
$$-1.1C_{LU} \le u_1(t) \le 1.1C_{LU}, \tag{7.95g}$$
$$-1.1C_{LU} \le u_2(t) \le 1.1C_{LU}. \tag{7.95h}$$

Table 7.8. *Dynamic model parameters.*

g_0	32.174 ft/sec^2	m_0	519.5 slug
I_{sp}	310 sec	R_E	20926430 ft
μ	1.40895×10^{16} ft^3/sec^2	ρ_E	.0023769 slug/ft^3
S	125.84 ft^2	C_{D0}	.032
K	1.4	$C_{L\alpha}$	.5699
C_{LU}	0.4	v_E	$\sqrt{\mu/R_E}$ ft/sec

7.2.3 Boundary Conditions

The trajectory begins in a circular orbit at an altitude of 19323 nm, i.e., an ECI state vector

$$\mathbf{r}^{\mathsf{T}}(t_0) = (1.38335209528\times10^8, 0, 0), \qquad \mathbf{v}^{\mathsf{T}}(t_0) = (0, 1.00920971977\times10^4, 0). \tag{7.96}$$

The first de-orbit burn is approximated by an instantaneous velocity change as in examples (6.6) and (7.1). To be more precise, we use the *impulsive* Δv approximation

$$\mathbf{v}(t_1) = \mathbf{v}(t_0) + \mathbf{\Delta v}_1, \tag{7.97}$$

where $\mathbf{v}(t_0)$ is the velocity before the burn, $\mathbf{v}(t_1)$ is the velocity after the burn, $\mathbf{\Delta v}$ is the velocity added by the burn, and $t_0 = t_1$. The velocity change is related to the mass by the boundary condition

$$m(t_0) = m(t_1)\exp\left(\frac{\|\mathbf{\Delta v}_1\|}{g_0 I_{sp}}\right), \tag{7.98}$$

where $m(t_0) = m_0$ is given in Table 7.8. The final value for ΔE_1^+ must satisfy the boundary condition

$$h(t_2^-) = \|\mathbf{r}(\Delta E_1^+)\| - R_E = 60 \text{ nm}. \tag{7.99}$$

As with many orbital problems, this application has more than one local solution. To distinguish between transfers that differ only because of orbital symmetry, let us (arbitrarily) focus on solutions that begin with the initial de-orbit burn at a *descending node*. This can be achieved by imposing a bound on the z-component of the velocity change

$$\Delta v_{z1} \le 0, \tag{7.100}$$

which ensures that the z-component of the inertial velocity is negative.

Now, motion during orbital and atmospheric phases is described using two different coordinate systems; however, at the phase boundaries the coordinate systems must be consistent. Thus for each atmospheric phase k, where $k = 2,4,6,8$, the following consistency conditions must be satisfied:

$$60 \text{ nm} = h(t_k^+), \qquad h(t_{k+1}^-) = 60 \text{ nm}, \tag{7.101a}$$

$$\phi\left[\mathbf{r}(t_k^-), \mathbf{v}(t_k^-)\right] = \phi(t_k^+), \qquad \phi(t_{k+1}^-) = \phi\left[\mathbf{r}(t_{k+1}^+), \mathbf{v}(t_{k+1}^+)\right], \tag{7.101b}$$

$$\theta\left[\mathbf{r}(t_k^-), \mathbf{v}(t_k^-)\right] = \theta(t_k^+), \qquad \theta(t_{k+1}^-) = \theta\left[\mathbf{r}(t_{k+1}^+), \mathbf{v}(t_{k+1}^+)\right], \tag{7.101c}$$

$$v\left[\mathbf{r}(t_k^-), \mathbf{v}(t_k^-)\right] = v(t_k^+), \qquad v(t_{k+1}^-) = v\left[\mathbf{r}(t_{k+1}^+), \mathbf{v}(t_{k+1}^+)\right], \tag{7.101d}$$

$$\gamma\left[\mathbf{r}(t_k^-),\mathbf{v}(t_k^-)\right]=\gamma(t_k^+), \qquad \gamma(t_{k+1}^-)=\gamma\left[\mathbf{r}(t_{k+1}^+),\mathbf{v}(t_{k+1}^+)\right], \tag{7.101e}$$

$$\psi\left[\mathbf{r}(t_k^-),\mathbf{v}(t_k^-)\right]=\psi(t_k^+), \qquad \psi(t_{k+1}^-)=\psi\left[\mathbf{r}(t_{k+1}^+),\mathbf{v}(t_{k+1}^+)\right]. \tag{7.101f}$$

Observe that the conditions at the beginning of an atmospheric phase require computing flight path coordinates using the ECI state from the previous phase (e.g., $\phi\left[\mathbf{r}(t_k^-),\mathbf{v}(t_k^-)\right]$). Conversely, at the end of the atmospheric phase, these *linkage conditions* are computed using the ECI state vector from the next, orbital, phase. In general the transformation matrix $\mathbf{Q}_{LE}$ from local horizontal (LH) to earth centered inertial (ECI) coordinates at the position $\mathbf{r}$ is given by

$$\widehat{\mathbf{z}}=-r^{-1}\mathbf{r}, \tag{7.102}$$

$$\widehat{\mathbf{x}}=\|\mathbf{e}_3-\hat{z}_3\widehat{\mathbf{z}}\|^{-1}(\mathbf{e}_3-\hat{z}_3\widehat{\mathbf{z}}), \tag{7.103}$$

$$\widehat{\mathbf{y}}=\widehat{\mathbf{z}}\times\widehat{\mathbf{x}}, \tag{7.104}$$

$$\mathbf{Q}_{LE}(\mathbf{r})=\left[\widehat{\mathbf{x}}\quad\widehat{\mathbf{y}}\quad\widehat{\mathbf{z}}\right], \tag{7.105}$$

where $\mathbf{e}_3^{\mathrm{T}}=(0,0,1)$. Thus the intrinsic or flight path coordinates $(h,\phi,\theta,v,\gamma,\psi)$ can be computed from the ECI state $(\mathbf{r},\mathbf{v})$ as follows:

$$r=\|\mathbf{r}\|, \tag{7.106a}$$

$$\widetilde{\mathbf{v}}=\mathbf{Q}_{LE}^{\mathrm{T}}(\mathbf{r})\mathbf{v}, \tag{7.106b}$$

$$h=r-R_E, \tag{7.106c}$$

$$\phi=\tan^{-1}(r_2/r_1), \tag{7.106d}$$

$$\theta=\sin^{-1}(r_3/r), \tag{7.106e}$$

$$v=\|\mathbf{v}\|, \tag{7.106f}$$

$$\gamma=\sin^{-1}(-\tilde{v}_3/v), \tag{7.106g}$$

$$\psi=\tan^{-1}(\tilde{v}_2/\tilde{v}_1). \tag{7.106h}$$

The inverse of transformation (7.106a)–(7.106h) can be used to compute ECI coordinates given intrinsic quantities as follows:

$$r=h+R_E, \tag{7.107a}$$

$$r_1=r\cos\theta\cos\phi, \tag{7.107b}$$

$$r_2=r\cos\theta\sin\phi, \tag{7.107c}$$

$$r_3=r\sin\theta, \tag{7.107d}$$

$$\tilde{v}_1=v\cos\gamma\cos\psi, \tag{7.107e}$$

$$\tilde{v}_2=v\cos\gamma\sin\psi, \tag{7.107f}$$

$$\tilde{v}_3=-v\sin\gamma, \tag{7.107g}$$

$$\mathbf{v}=\mathbf{Q}_{LE}(\mathbf{r})\widetilde{\mathbf{v}}. \tag{7.107h}$$

To ensure that an atmospheric phase begins with decreasing altitude and terminates with altitude increasing, the flight path angle at the phase boundaries is restricted by the boundary conditions

$$0\geq\gamma(t_k^+), \qquad \gamma(t_{k+1}^-)\geq 0. \tag{7.108}$$

The vehicle mass during all atmospheric phases and the final phase must be consistent with the final mass after the first burn (in phase 1), so we must have

$$m(t_1) = m(t_2) = m(t_4) = m(t_6) = m(t_8) = m(t_9). \tag{7.109}$$

The intermediate orbital phases 3, 5, and 7 all begin and end at the atmospheric limit. Therefore we must impose conditions similar to (7.99) at the beginning and end of these phases:

$$\|\mathbf{r}(0)\| - R_E = \|\mathbf{r}(\Delta E_k^+)\| - R_E = 60 \text{ nm}. \tag{7.110}$$

The final outbound orbital phase is a mirror image of the first phase. The altitude at the beginning of the phase is constrained:

$$\|\mathbf{r}(0)\| - R_E = 60 \text{ nm}. \tag{7.111}$$

At the final time since there is no change in the position $\mathbf{r}(t_f) = \mathbf{r}(t_9^+)$. However, the final velocity is altered by the final burn, and so we have

$$\mathbf{v}(t_f) = \mathbf{v}(t_9^+) + \mathbf{\Delta v}_2, \tag{7.112}$$

with a corresponding mass change defined by the condition

$$m(t_9) = m(t_f) \exp\left(\frac{\|\mathbf{\Delta v}_2\|}{g_0 I_{sp}}\right). \tag{7.113}$$

Furthermore the final orbit conditions must be satisfied:

$$\|\mathbf{r}(t_f)\| - R_E = 100 \text{ nm}, \tag{7.114}$$

$$\|\mathbf{v}(t_f)\| = \sqrt{\frac{\mu}{r_f}}, \tag{7.115}$$

$$\mathbf{r}^{\mathsf{T}}(t_f)\mathbf{v}(t_9^+) = 0, \tag{7.116}$$

$$\mathbf{r}^{\mathsf{T}}(t_f)\mathbf{\Delta v}_2 = 0, \tag{7.117}$$

$$i(t_f) = i_F, \tag{7.118}$$

where the inclination in (7.118) can be computed as in (7.34a)–(7.34e).

The objective is to maximize the final mass, that is,

$$F = m(t_f). \tag{7.119}$$

7.2.4 Initial Guess

The efficient solution for a nonlinear problem such as this can be dictated by how good the initial guess is. Of course it is also important that the initial guess be relatively easy to compute, at least compared to the actual problem. For this example a reasonably simple approximate solution can be obtained by first solving a small NLP problem. In particular, let us approximate the trajectory by *ignoring the atmosphere* and model the action of the atmosphere by impulsive velocity changes. The basic approach is illustrated schematically in Figure 7.10.

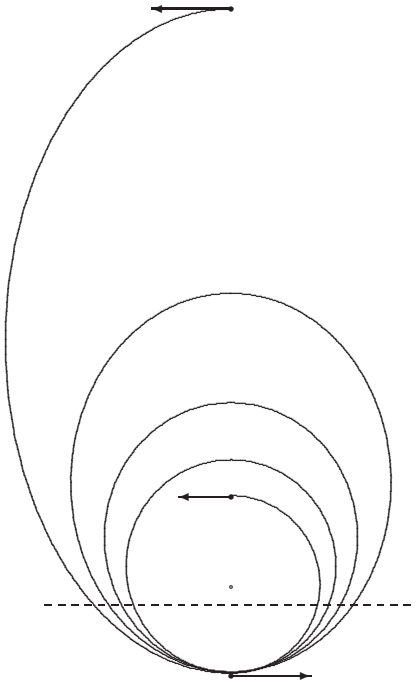


Figure 7.10. *Bi-elliptic transfer approximate solution.*

For this simple model we permit impulsive velocity changes to occur at the initial and final points and at each perigee location. Furthermore, we assume that there is no $\Delta\mathbf{v}$ in the x-direction and that all of the impulsive changes at perigee are identical. Thus we pose an NLP problem with $n = 2 \times (1 + 1 + 1) = 6$ variables. To fix the orbit geometry, analytic propagation is used with the inbound and outbound segments having $\Delta E = \pi$ and the intermediate arcs of length $\Delta E = 2\pi$. The desired initial and final conditions are imposed as constraints. Furthermore, we arbitrarily insist that each of the perigee burns occur at a fixed altitude of 40 nm, which is "inside" the atmosphere. This simple bi-elliptic orbit transfer problem can be solved easily to minimize the total $\Delta\mathbf{v}$ using the same NLP algorithm used by the complete $\mathbb{SOCS}$ algorithm. Table 7.9 summarizes the bi-elliptic

Table 7.9. *Bi-elliptic transfer solution.*

Orbit	Perigee (nm)	Apogee (nm)	Inclination (deg)
Initial	19323.0	19323.0	0.00000
Inbound	40.0000	19323.0	86.4340
First Intermediate	40.0000	8032.62	86.9445
Second Intermediate	40.0000	3733.28	87.5242
Third Intermediate	40.0000	1479.40	88.1873
Outbound	40.0000	100.000	88.9536
Final	100.000	100.000	89.0000

transfer solution. The bi-elliptic solution not only provides reasonable estimates for the orbit phases but also can be used to construct estimates for the states at the beginning and end of each atmosphere phase. Linear interpolation between the states at the beginning and end of each phase is used for all of the differential variables. Guessing constant values for $C_L = .25$ and $\beta = 0$ yields estimates for the control variables of $u_1(t) = 0$ and $u_2(t) = -.25$.

7.2.5 Numerical Results

The nine-phase problem described has been solved using $\mathbb{SOCS}$, and Table 7.10 presents a summary of the algorithm performance. The problem was initialized using the approach described in Section 7.2.4. A total of seven mesh-refinement iterations were performed as summarized in each row of Table 7.10. Column two gives the number of grid points. All of the orbital phases (1, 3, 5, 7, 9) utilized the analytic Kepler propagation scheme, and within $\mathbb{SOCS}$ this approach is implemented using two grid points—the initial and final phase boundaries. No mesh refinement is performed on analytic phases, and consequently the number of grid points remains unchanged. In contrast, the atmospheric phases (2, 4, 6, 8) are initiated using a trapezoidal discretization with 10 points equally spaced within the phase. The resulting sparse NLP has $n = 402$ variables and at the solution the number of degrees of freedom $n_d = n - \hat{m} = 77$. The total number of QP subproblems (NQP) required to solve the first coarse grid NLP was 173, and it took 3.4 CPU sec on a Dell M90 laptop. After two refinement iterations the discretization scheme is changed to an HSC method, which is used for all subsequent refinement iterations. The discretization error ϵ was reduced from 1.3×10^{-1} on the first grid to 9.2×10^{-8} on the last. The overall solution was computed in 16.95 CPU sec.

The optimal impulsive velocity increments are

$$\Delta\mathbf{v}_1^* = (-1.8392052\times10^2, -7.1911864\times10^3, -4.3121550\times10^3)^{\mathsf{T}}, \tag{7.120}$$

$$\Delta\mathbf{v}_2^* = (3.8614396\times10^0, -2.0351389\times10^1, -1.1615574\times10^2)^{\mathsf{T}} \tag{7.121}$$

and the total (minimum) $\Delta\mathbf{v}$ for this example is

$$\|\Delta\mathbf{v}_1^*\| = 8386.9940 \text{ ft/sec}, \tag{7.122}$$

$$\|\Delta\mathbf{v}_2^*\| = 117.98833 \text{ ft/sec}, \tag{7.123}$$

$$\|\Delta\mathbf{v}_1^*\| + \|\Delta\mathbf{v}_2^*\| = 8504.9823 \text{ ft/sec}, \tag{7.124}$$

Table 7.10. *Mesh-refinement summary.*

k	M	n	n_d	NQP	ϵ	Time (sec)
1	(2,10,2,10,2,10,2,10,2)	402	77	173	1.3×10^{-1}	3.4
2	(2,19,2,19,2,19,2,19,2)	690	144	21	7.5×10^{-3}	0.4
3	(2,19,2,19,2,19,2,19,2)	834	289	32	9.8×10^{-4}	1.7
4	(2,37,2,37,2,37,2,37,2)	1554	569	17	3.9×10^{-5}	1.4
5	(2,73,2,73,2,73,2,73,2)	2994	1138	17	1.5×10^{-6}	3.4
6	(2,95,2,131,2,136,2,145,2)	5144	1976	11	1.2×10^{-7}	4.6
7	(2,95,2,131,2,136,2,150,2)	5194	1996	5	9.2×10^{-8}	1.9
Total	522			4979		16.95

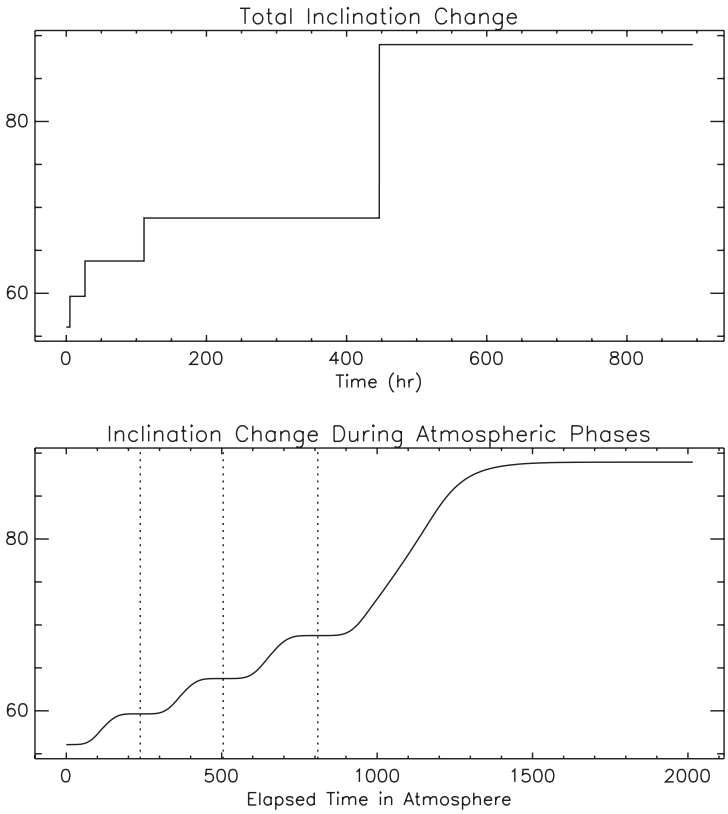


Figure 7.11. *Inclination change.*

which corresponds to masses given by

$$m^*(t_1) = 224.07393 \text{ slug}, \tag{7.125}$$

$$m^*(t_f) = 221.43883 \text{ slug}. \tag{7.126}$$

The dynamic histories of particular interest are illustrated in Figures 7.11, 7.12, and 7.13. Since the time scale for the atmospheric portions of the transfer are significantly shorter than the orbit transit times, it is convenient to display the results using elapsed time in the atmosphere as the independent variable. The phase boundaries are indicated using a vertical dotted line. In Figure 7.11 the inclination is plotted as a function of the total time, and also the elapsed time during the atmosphere passes. This figure clearly illustrates that the inclination change is accomplished primarily in the last atmospheric pass. It is also clear that the inclination changes appear to occur almost instantaneously because the duration of the atmospheric passes is much shorter than the entire transfer time. This also suggests that approximating the aerodynamic pass by an impulsive velocity as was done in the bi-elliptic initialization procedure is a reasonable approximation. Figure 7.12 plots the state variables, and it is clear from the altitude plot that each atmospheric pass begins at the atmospheric limit of 60 nm. The velocity decreases monotonically because of

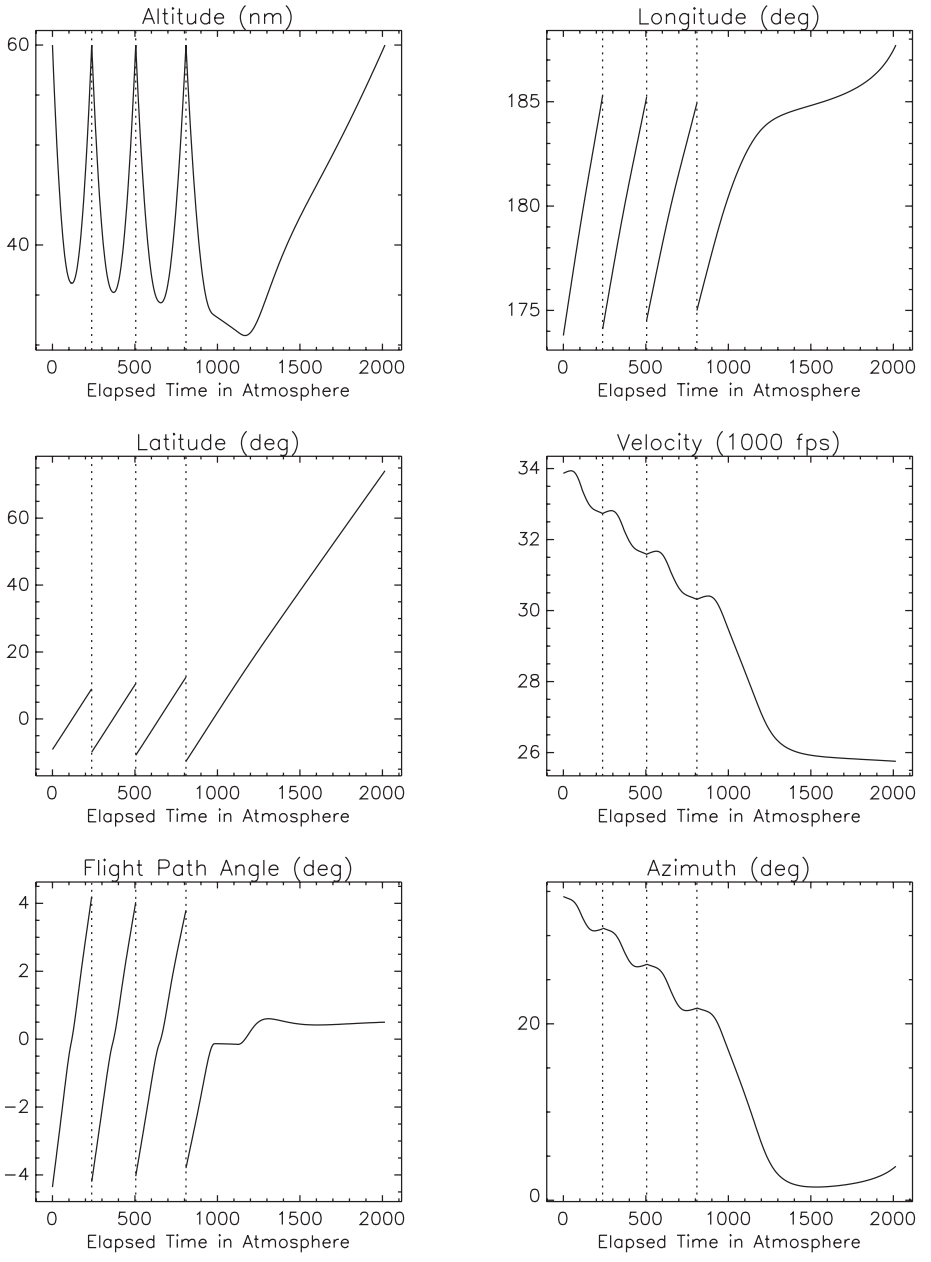


Figure 7.12. *States during four atmospheric passes.*

atmospheric drag, and for each phase the flight path angle begins negative and ends positive as required. Figure 7.13 presents the actual controls $u_1(t)$ and $u_2(t)$, as well as the derived values for angle of attack and bank angle. Also included are the time histories for the

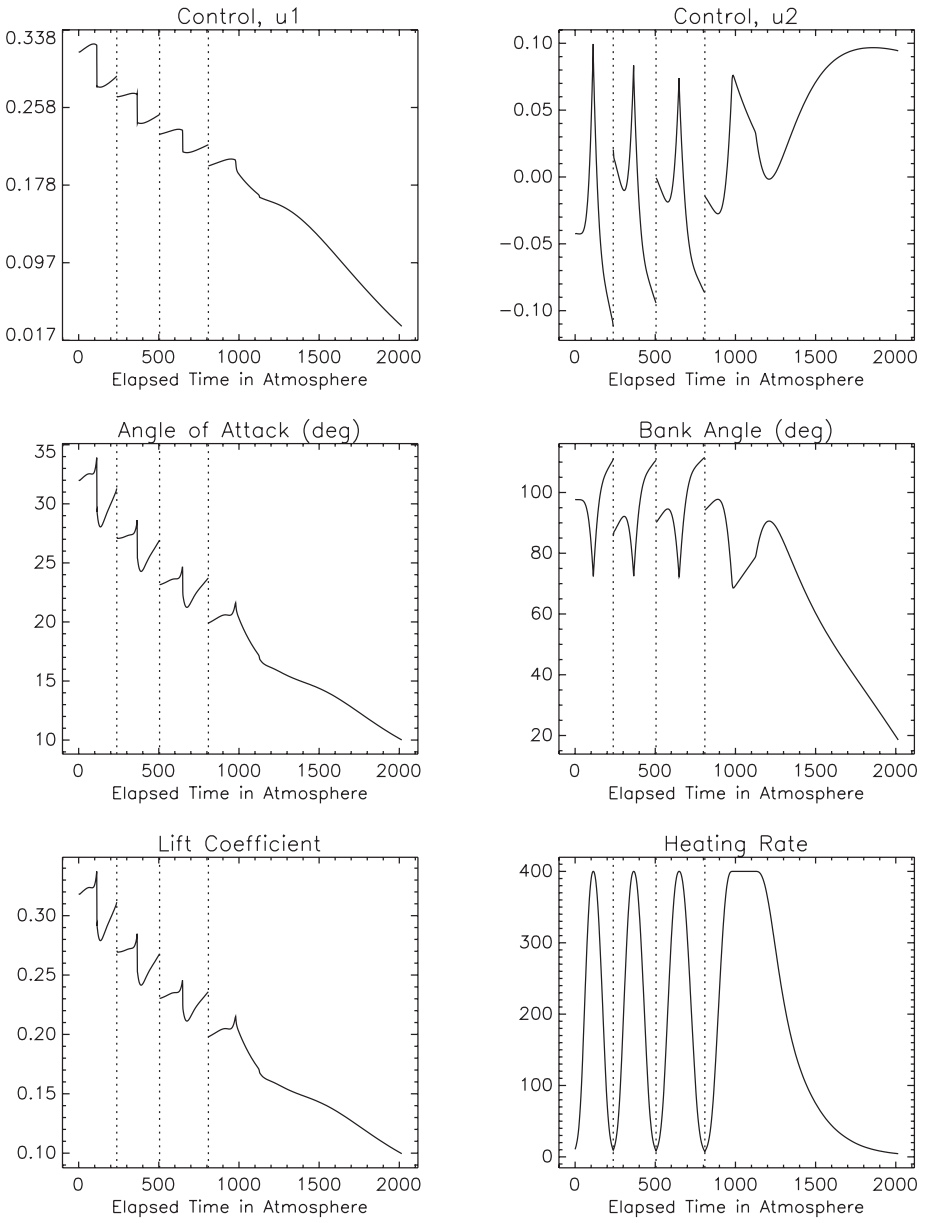


Figure 7.13. *Controls during four atmospheric passes.*

path constraint functions C_L and Q that appear in (7.81) and (7.82). For comparison, the same problem was solved using a single atmospheric pass, and the corresponding dynamic histories are illustrated in Figures 7.14 and 7.15. The final mass using a single pass was $m^*(t_f) = 212.16080$ slug. Finally, Figure 7.16 illustrates the entire multipass aero-assisted

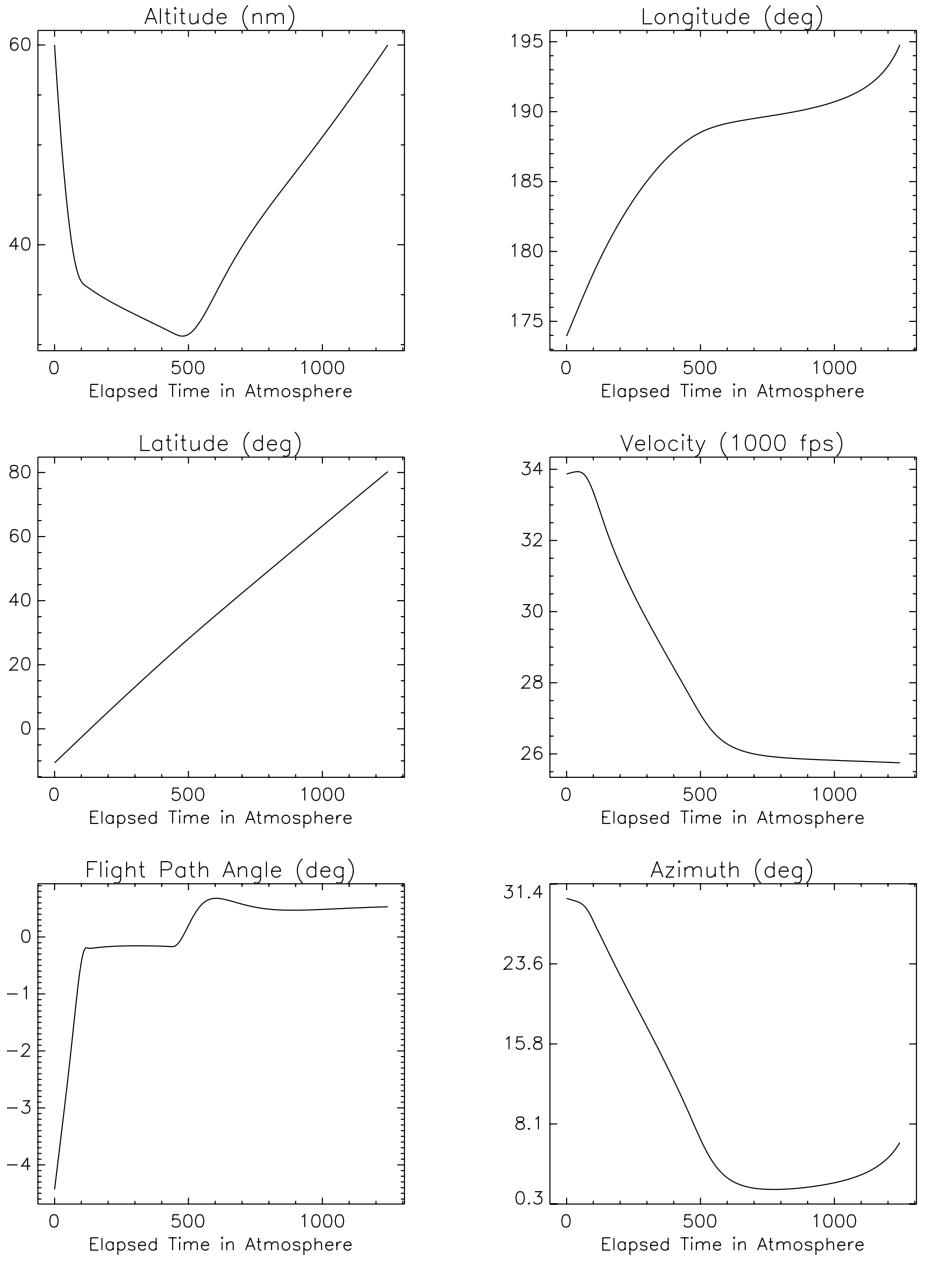


Figure 7.14. *States during a single atmospheric pass.*

transfer trajectory and Figure 7.17 illustrates the single pass solution. The trajectory plane shading is defined by the instantaneous inclination, which illustrates the transition from $i \approx 56$ (deg) shown in light gray to $i = 89$ (deg) in dark.

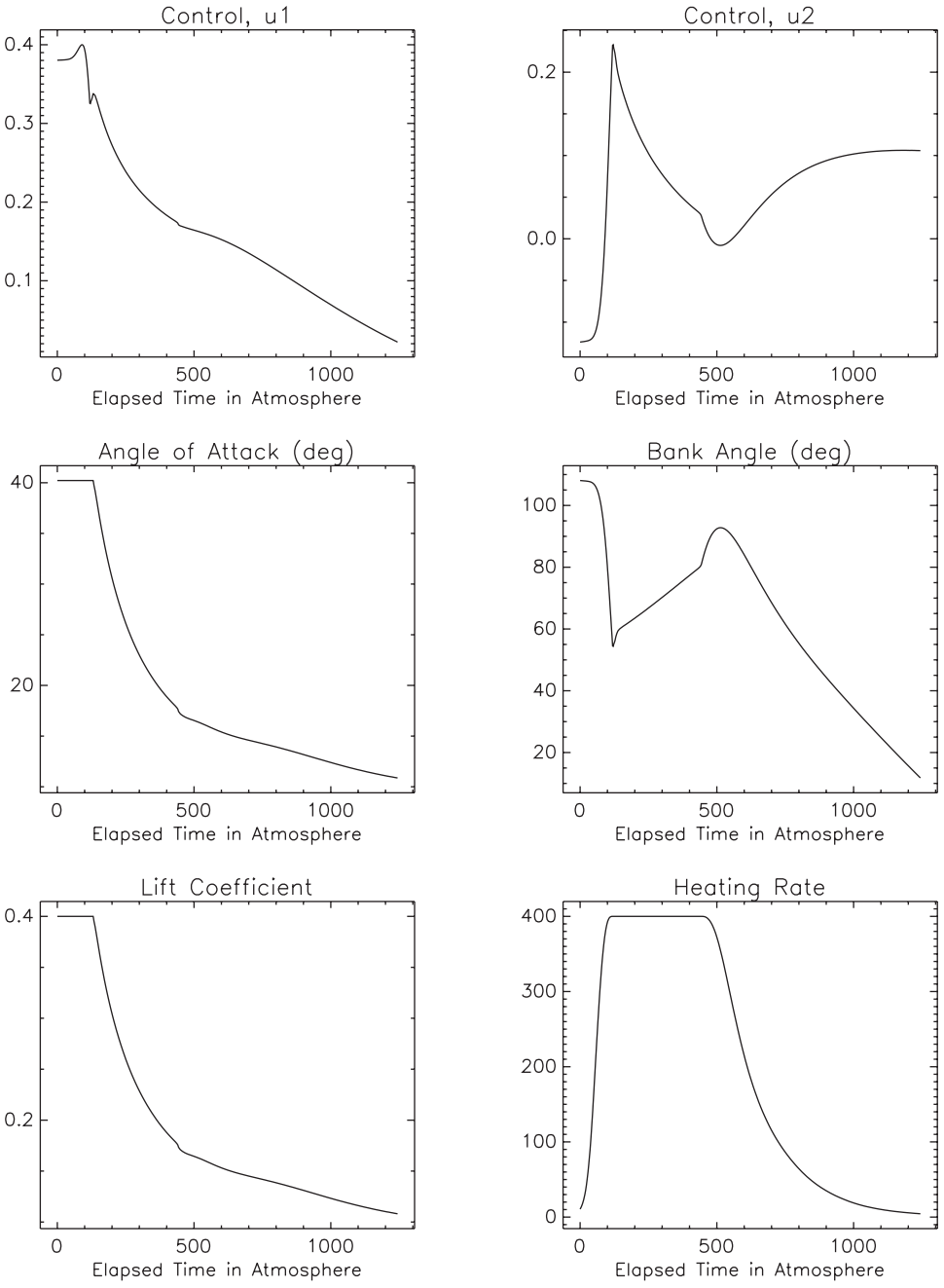


Figure 7.15. *Controls during a single atmospheric pass.*

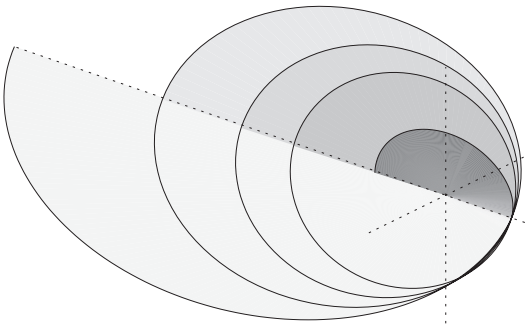


Figure 7.16. *Optimal four-pass aero-assisted transfer.*

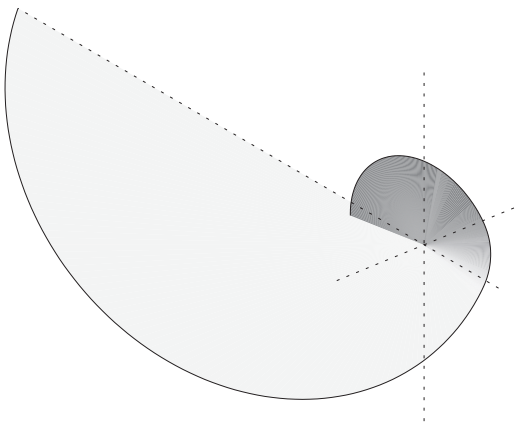


Figure 7.17. *Optimal single-pass aero-assisted transfer.*

7.3 Delay Differential Equations

For many physical processes the mathematical description involves "time lags" or "retarded arguments." Thus instead of the usual ODE model

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), t]$$

the problem description entails a delay differential equation (DDE) model such as

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{y}(t-\tau), t, t-\tau]. \tag{7.127}$$

Observe that the dynamics are expressed in terms of the state $\mathbf{y}(t-\tau)$ evaluated at the delay time $t-\tau$, where $\tau > 0$. This *fixed delay* model is one of the simplest types of DDEs. Clearly more complicated descriptions can be envisioned if, for example, one considers multiple delay times τ_k and/or nonlinear delay arguments such as $\mathbf{y}(\boldsymbol{\phi}[t-\tau])$. Furthermore, one can consider delay-differential-algebraic equation (DDAE) problem formulations [3]. To fully appreciate the computational challenges caused by stiffness and propagation of discontinuities it is instructive to review the paper by Shampine and Gahinet [157].

Although a complete discussion of the subject is beyond the scope of this book, it is instructive to illustrate some of the issues that arise. To do so we will focus on examples having fixed delay that can be converted to BVPs with ODEs using an approach called *method of steps*. It should be emphasized that this is not the only (or perhaps the best) approach for these problems.

Example 7.2 ENZYME KINETICS. Let us focus on a particular example originally published by Okamoto and Hayashi [137] and cited by Hairer, Norsett, and Wanner [106, pp. 348–349] that describes enzyme kinetics. The enzyme concentrations on the time interval $0 \le x \le 160$ are described by the DDE system

$$s_1' = I - z s_1(x), \tag{7.128}$$
$$s_2' = z s_1(x) - c_2 s_2(x), \tag{7.129}$$
$$s_3' = c_2 s_2(x) - c_3 s_3(x), \tag{7.130}$$
$$s_4' = c_3 s_3(x) - c_4 s_4(x), \tag{7.131}$$

where the process is inhibited by

$$z = \frac{c_1}{1 + \alpha[s_4(x-4)]^3} \tag{7.132}$$

with $I = 10.5$, $c_1 = c_2 = c_3 = 1$, $c_4 = 0.5$, and $\alpha = 0.0005$. As $x \to \infty$, the system approaches an equilibrium state $\mathbf{s}(x) \to \tilde{\mathbf{s}}$ with $\tilde{s}_1 = I(1 + .004I^3)$, $\tilde{s}_2 = \tilde{s}_3 = I$, and $\tilde{s}_4 = 2I$. In this example the number of delay equations $L = 4$, and the number of delay intervals $N = 40$, with a delay time $\tau = 4$.

For systems such as this with constant delay, it is possible to reformulate the problem as a BVP involving just ODEs, using a technique called the *method of steps*. Specifically let us consider a fixed time domain $0 \le t \le \tau$ and then transform the original time into multiples of the delay time, that is,

$$x = t + k\tau \tag{7.133}$$

for $k = 0, \ldots, N-1$, where N is the total number of delay steps. Observe that when $k = 0$, $0 \le x \le \tau$, and when $k = 1$, $\tau \le x \le 2\tau$, and so forth. Using this transformation, each delay interval has been mapped onto the fixed interval $0 \le t \le \tau$. Using this mapping, the original time domain can be "folded" onto a single interval, by defining new dynamic variables according to

$$\begin{aligned} y_1(t) &= s_1(t), \\ &\vdots \\ y_5(t) &= s_1(t+\tau), \\ &\vdots \\ y_{j+kL}(t) &= s_j[t + k\tau] = s_j(x) \end{aligned} \tag{7.134}$$

for $j = 1, \ldots, L$, and $k = 0, \ldots, N-1$. Now if we differentiate (7.134), we obtain

$$\dot{y}_{j+kL} = \frac{dy_{j+kL}}{dt} = \frac{ds_j}{dt} = \frac{ds_j}{dx}\frac{dx}{dt} = s_j'. \tag{7.135}$$

Using the new dynamic variables, the original system of delay equations (7.128)–(7.132) is replaced by the system of ODEs

$$\dot{y}_{1+kL} = I - z y_{1+kL}, \tag{7.136}$$
$$\dot{y}_{2+kL} = z y_{1+kL} - c_2 y_{2+kL}, \tag{7.137}$$
$$\dot{y}_{3+kL} = c_2 y_{2+kL} - c_3 y_{3+kL}, \tag{7.138}$$
$$\dot{y}_{4+kL} = c_3 y_{3+kL} - c_4 y_{4+kL}, \tag{7.139}$$

where the delay term becomes

$$z = \frac{c_1}{1+\alpha[y_{4+(k-1)L}]^3}. \tag{7.140}$$

The delay term given by (7.140) follows from (7.132) by noting that

$$s_4(x-4) = s_4[(t+k\tau)-\tau] = s_4[t+(k-1)\tau] = y_{4+(k-1)L}. \tag{7.141}$$

Of course the problem statement is not complete without the appropriate boundary conditions. By construction the dynamic variables y describe behavior on a single delay interval. Furthermore at the boundary of neighboring delay intervals since $\tau + k\tau = (0) + (k+1)\tau$ we must have

$$y_{j+kL}(\tau) = s_j[\tau + k\tau] = s_j[(0)+(k+1)\tau] = y_{j+(k+1)L}(0) \tag{7.142}$$

for $j = 1, \ldots, L$, and $k = 0, \ldots, N-1$.

DDEs pose one additional complication not shared by ODEs, namely how to get started. For an ODE, it is sufficient to specify the initial values $\mathbf{y}(0)$. In contrast for a DDE, one must specify function histories over the entire startup region $-\tau \le x \le 0$. For the particular example illustrated here, we define the startup functions by

$$y_{1-L}(t) = s_1(x) = 60, \tag{7.143}$$
$$y_{2-L}(t) = s_2(x) = 10, \tag{7.144}$$
$$y_{3-L}(t) = s_3(x) = 10, \tag{7.145}$$
$$y_{4-L}(t) = s_4(x) = 20 \tag{7.146}$$

for $-\tau \le t \le 0$.

To summarize, the original problem was posed in terms of a system of DDEs for the dynamic variables $\mathbf{s}(x)$ defined on the time domain $0 \le x \le 160 = 40 \times 4$. The problem can be recast as a BVP for a system of ODEs in the dynamic variables $\mathbf{y}(t)$ which is defined on the domain $0 \le t \le \tau = 4$. The approach is attractive since it provides a mechanism to extend methods for optimal control and estimation of ODEs to DDEs. In spite of its attractive simplicity the method of steps does present a number of potentially problematic issues. First, a DDE IVP is replaced by an ODE BVP. Second, the number of ODEs (LN) is much larger than the number of DDEs (L). Finally, the technique used to control ODE integration (discretization) error may not be appropriate or efficient for a DDE system.

To illustrate the process, let us take a two step approach. First, let us treat the constants $c_1 = c_2 = c_3 = 1$, $c_4 = 0.5$ as known and solve the DDE system. Then as a

second step, let us construct an inverse problem and attempt to estimate the parameters $\mathbf{p} = (c_1, c_2, c_3, c_4)$. Treating the solution to the first step as a truth model $\widetilde{\mathbf{y}}$, data for the second step can be constructed by adding noise to the truth model, i.e.,

$$\widehat{\mathbf{y}}(\theta_i) = \widetilde{\mathbf{y}}(\theta_i) + \boldsymbol{\nu}, \tag{7.147}$$

where $\boldsymbol{\nu} \sim \mathcal{N}(0, \sigma)$ is a vector of independent identically distributed Gaussian random variables with mean zero and variance $\sigma = 0.01$. For this example 10 evaluation times θ_i are equally spaced over the time interval. The objective is to minimize

$$F = \frac{1}{2} \sum_i [\mathbf{y}_i - \widehat{\mathbf{y}}_i]^{\mathsf{T}} [\mathbf{y}_i - \widehat{\mathbf{y}}_i] \tag{7.148}$$

by choosing the parameters $\mathbf{p}$ and dynamic variables $\mathbf{y}(t)$ while satisfying the differential equations (7.136)–(7.140) and boundary conditions (7.142). As initial conditions we impose

$$y_1(0) = 60, \tag{7.149}$$
$$y_2(0) = 10, \tag{7.150}$$
$$y_3(0) = 10, \tag{7.151}$$
$$y_4(0) = 20 \tag{7.152}$$

with the startup functions given by (7.143)–(7.146). The resulting problem has 160 state variables $\mathbf{y}$ and 156 boundary conditions (7.142). The objective function has 1600 total residuals.

A piecewise constant initial guess is used to initiate the iterative process. Thus with two grid points the initial guess is just $y_j^{(0)}(0) = y_j^{(0)}(\tau) = 0$ for $j = 5, \ldots, 160$. On the first delay interval when $j = 1, 2, 3, 4$ and the initial conditions (7.149)–(7.152) provide the guess $y_j^{(0)}(0) = y_j^{(0)}(\tau) = y_j(0)$. Eight mesh-refinement iterations are required to achieve the desired accuracy of $\epsilon \leq \delta = 1 \times 10^{-7}$. Table 7.11 presents a summary of the algorithm performance using $\mathbb{SOCS}$ to solve Step 1. Table 7.12 presents similar information for the parameter estimation step. It is worth noting that by exploiting the right-hand-side sparsity, it is possible to compute finite difference gradient information quite efficiently for this application as indicated by the values in Table 7.13. Figure 7.18 illustrates the solution,

Table 7.11. *Mesh-refinement summary (nominal).*

k	M	n	NGC	NFE	ϵ	Time (sec)
1	2	320	16	87	1.04×10^{0}	.13
2	3	480	12	74	3.07×10^{-1}	.09
3	3	480	15	196	5.51×10^{-2}	.14
4	5	800	3	29	4.88×10^{-3}	.08
5	9	1440	2	20	2.99×10^{-4}	.12
6	17	2720	1	11	1.96×10^{-5}	.20
7	33	5280	1	11	1.23×10^{-6}	.42
8	65	10400	1	11	7.68×10^{-8}	1.34
Total	65		51	439		2.52

Table 7.12. *Mesh-refinement summary (parameter estimation).*

k	M	n	NGC	NFE	ϵ	Time (sec)
1	2	324	20	260	2.91×10^{-1}	.48
2	3	484	23	314	4.39×10^{-1}	1.42
3	3	484	19	733	5.07×10^{-2}	2.31
4	5	804	6	402	3.88×10^{-3}	.63
5	9	1444	4	154	2.26×10^{-4}	.42
6	17	2724	4	154	1.53×10^{-5}	.92
7	33	5284	3	129	9.68×10^{-7}	1.84
8	65	10404	3	129	6.06×10^{-8}	5.59
Total	65		82	2275		13.61

Table 7.13. *Number of index sets γ.*

Discretization	Trapezoidal	Hermite–Simpson
Step 1	3	8
Step 2	5	12

with the delay intervals "unfolded" to correspond with the original problem statement. The estimated parameter values for this case are

$$\mathbf{p} = (1.0000097 \times 10^{0}, 9.9999697 \times 10^{-1}, 9.9999450 \times 10^{-1}, 5.0000131 \times 10^{-1})^{\mathsf{T}}, \quad (7.153)$$

which corresponds to the minimum value $F^* = 8.8747308 \times 10^{-2}$.

Example 7.3 IMMUNOLOGY EXAMPLE. A second example originally published in Russian by G. I. Marchuk is also cited by Hairer, Norsett, and Wanner [106, pp. 349–351]. The dynamics are described by the DDE system

$$\frac{dV}{dx} = (h_1 - h_2 F)V, \quad (7.154)$$

$$\frac{dC}{dx} = \xi(m)h_3 F(x-\tau)V(x-\tau) - h_5(C-1), \quad (7.155)$$

$$\frac{dF}{dx} = h_4(C-F) - h_8 FV, \quad (7.156)$$

$$\frac{dm}{dx} = h_6 V - h_7 m. \quad (7.157)$$

The dynamics model the struggle of viruses $V(t)$, antibodies $F(t)$, and plasma cells $C(t)$ in a person infected with a viral disease. The relative characteristic damage is represented by $m(t)$, where the first term in (7.157) accounts for damaging and the second term for recuperation. The fact that plasma cell creation slows down when the organism is damaged by the viral infection is modeled by the term

$$\xi(m) = \begin{cases} 1 & \text{if } m \le 0.1, \\ (1-m)\frac{10}{9} & \text{if } 0.1 \le m \le 1. \end{cases} \quad (7.158)$$

Equation (7.154) is referred to as a predator-prey equation, and (7.155) models the creation of new plasma cells with a time lag due to infection. Notice that an equilibrium occurs

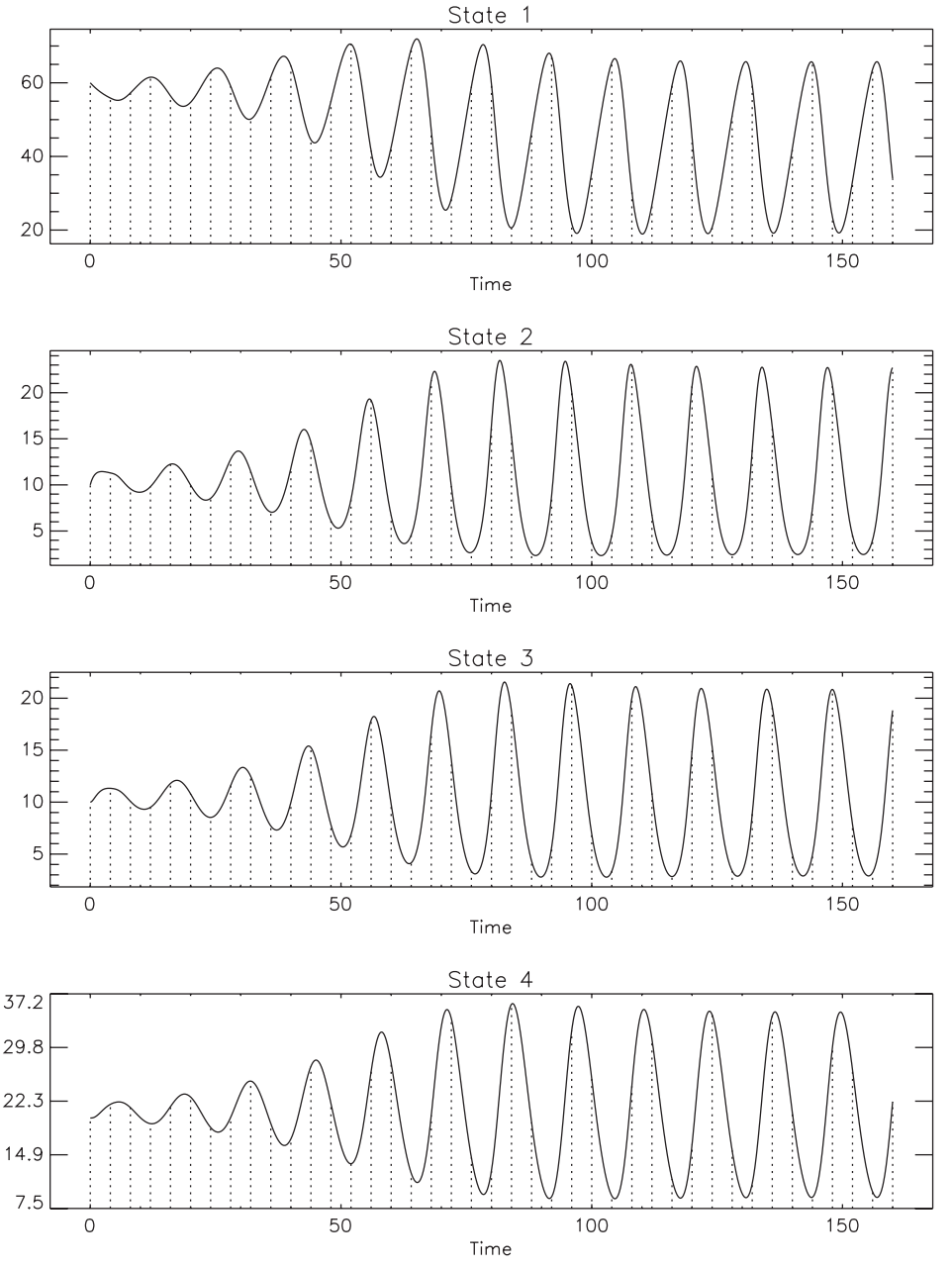


Figure 7.18. *Enzyme kinetic delay equation solution.*

with $C = 1$ if the first term in (7.155) is omitted. Equation (7.156) models three effects, namely creation of antibodies from plasma cells ($h_4 C$), decrease in plasma cells due to aging ($-h_4 F$), and binding with antigens ($-h_8 FV$).

The DDE system (7.154)–(7.157) can be converted to a system of ODEs using the same technique introduced for the previous example. In particular the number of delay equations $L = 4$, and if we identify the variables

$$(y_{1+kL}, y_{2+kL}, y_{3+kL}, y_{4+kL}) = (V, C, F, m)$$

on delay intervals given by (7.133), one obtains the following equations:

$$\dot{y}_{1+kL} = \left[h_1 - h_2 y_{3+kL}\right] y_{1+kL}, \tag{7.159}$$

$$\dot{y}_{2+kL} = \xi(y_{4+kL}) h_3 y_{3+(k-1)L} y_{1+(k-1)L} - h_5\left[y_{2+kL} - 1\right], \tag{7.160}$$

$$\dot{y}_{3+kL} = h_4\left[y_{2+kL} - y_{3+kL}\right] - h_8 y_{3+kL} y_{1+kL}, \tag{7.161}$$

$$\dot{y}_{4+kL} = h_6 y_{1+kL} - h_7 y_{4+kL} \tag{7.162}$$

for $k = 0, \ldots, N-1$, where $N = 120$ is the total number of delay steps. For the specific case of interest, $\tau = 0.5$, $h_1 = 2$, $h_2 = 0.8$, $h_3 = 10^4$, $h_4 = 0.17$, $h_5 = 0.5$, $h_6 = 300$, $h_7 = 0.12$, and $h_8 = 8$. As startup conditions we use

$$y_{1-L}(t) = \max\left(0, 10^{-6} + t\right), \tag{7.163}$$

$$y_{2-L}(t) = 1, \tag{7.164}$$

$$y_{3-L}(t) = 1, \tag{7.165}$$

$$y_{4-L}(t) = 0 \tag{7.166}$$

for $-\tau \le t \le 0$. As before, we must also impose the "wrapping" boundary conditions (7.142). In summary, the complete solution to this DDE requires solving a nonlinear BVP with 480 ODEs and 476 nonlinear boundary conditions. After the continuous problem is discretized the resulting sparse NLP has no degrees of freedom, i.e., no objective function. Nevertheless, the constraints are very nonlinear, and as such initiating the iteration with a "good guess" is very important.

The serial nature of the problem suggests a very natural way to proceed. As a first step let us solve a problem with only one delay interval. Thus first solve

$$\dot{y}_1 = \left[h_1 - h_2 y_3\right] y_1, \tag{7.167a}$$

$$\dot{y}_2 = \xi(y_4) h_3 y_{3-L} y_{1-L} - h_5\left[y_2 - 1\right], \tag{7.167b}$$

$$\dot{y}_3 = h_4\left[y_2 - y_3\right] - h_8 y_3 y_1, \tag{7.167c}$$

$$\dot{y}_4 = h_6 y_1 - h_7 y_4. \tag{7.167d}$$

This problem involving four differential equations can be solved using a coarse discretization, e.g., a trapezoidal method with two grid points. Since the goal is to construct an initial guess for the "real problem," mesh refinement is not needed at this stage. To initiate this first step it is reasonable to guess constant values for the states. Denote the approximate solution obtained by this first step

$$\left[\tilde{y}_1(t), \tilde{y}_2(t), \tilde{y}_3(t), \tilde{y}_4(t)\right] \qquad \text{for } 0 \le t \le \tau. \tag{7.168}$$

Now let us solve a problem with two delay intervals, i.e., the system

$$\dot{y}_1 = \big[h_1 - h_2 y_3\big] y_1, \tag{7.169a}$$
$$\dot{y}_2 = \xi(y_4) h_3 y_{3-L} y_{1-L} - h_5\big[y_2 - 1\big], \tag{7.169b}$$
$$\dot{y}_3 = h_4\big[y_2 - y_3\big] - h_8 y_3 y_1, \tag{7.169c}$$
$$\dot{y}_4 = h_6 y_1 - h_7 y_4, \tag{7.169d}$$
$$\dot{y}_5 = \big[h_1 - h_2 y_7\big] y_5, \tag{7.169e}$$
$$\dot{y}_6 = \xi(y_8) h_3 y_{7-L} y_{5-L} - h_5\big[y_6 - 1\big], \tag{7.169f}$$
$$\dot{y}_7 = h_4\big[y_6 - y_7\big] - h_8 y_7 y_5, \tag{7.169g}$$
$$\dot{y}_8 = h_6 y_5 - h_7 y_8 \tag{7.169h}$$

subject to the boundary conditions

$$y_j(\tau) = y_{j+L}(0) \tag{7.170}$$

for $j = 1, \ldots, L$. It is now necessary to supply an initial guess for this two-delay problem. Clearly the solution (7.168) satisfies (7.169a)–(7.169d). Also from (7.170) we can construct a guess for the remaining variables

$$\begin{aligned}\big[\tilde{y}_1(\tau), \tilde{y}_2(\tau), \tilde{y}_3(\tau), \tilde{y}_4(\tau)\big] &= \Big[y_5^{(0)}(0), y_6^{(0)}(0), y_7^{(0)}(0), y_8^{(0)}(0)\Big] \\ &= \Big[y_5^{(0)}(\tau), y_6^{(0)}(\tau), y_7^{(0)}(\tau), y_8^{(0)}(\tau)\Big].\end{aligned} \tag{7.171}$$

In effect this is a constant "prediction" for the next delay interval.

Obviously, this stepwise procedure can be continued. The procedure requires solving a sequence of BVPs. The number of differential equations grows from step to step. The solution to the BVP computed at step k provides an excellent initial guess for the BVP that must be solved at step $(k+1)$. In essence the technique can be viewed as an extension of the predictor-corrector method for solving ODEs (cf. (3.42)). Even though the number of ODEs increases from step to step, each solution can be computed quite efficiently. The desired accuracy is achieved using mesh refinement only on the final step of the process. Table 7.14 summarizes the final solution refinement behavior. Figure 7.19 illustrates the solution, with the delay intervals "unfolded" to correspond with the original problem statement.

Table 7.14. *Mesh-refinement summary.*

k	M	n	NGC	NFE	ϵ	Time (sec)
1	2	324	9	47	2.36×10^{-2}	.57
2	3	484	26	159	3.40×10^{-3}	.97
3	3	484	36	520	2.09×10^{-4}	1.57
4	5	804	4	54	2.69×10^{-5}	.53
5	9	1444	5	67	1.03×10^{-5}	1.33
6	15	2724	1	15	1.30×10^{-6}	.51
7	26	5284	1	15	1.82×10^{-7}	1.63
8	42	10404	1	15	7.17×10^{-8}	2.20
Total	42		83	892		9.31

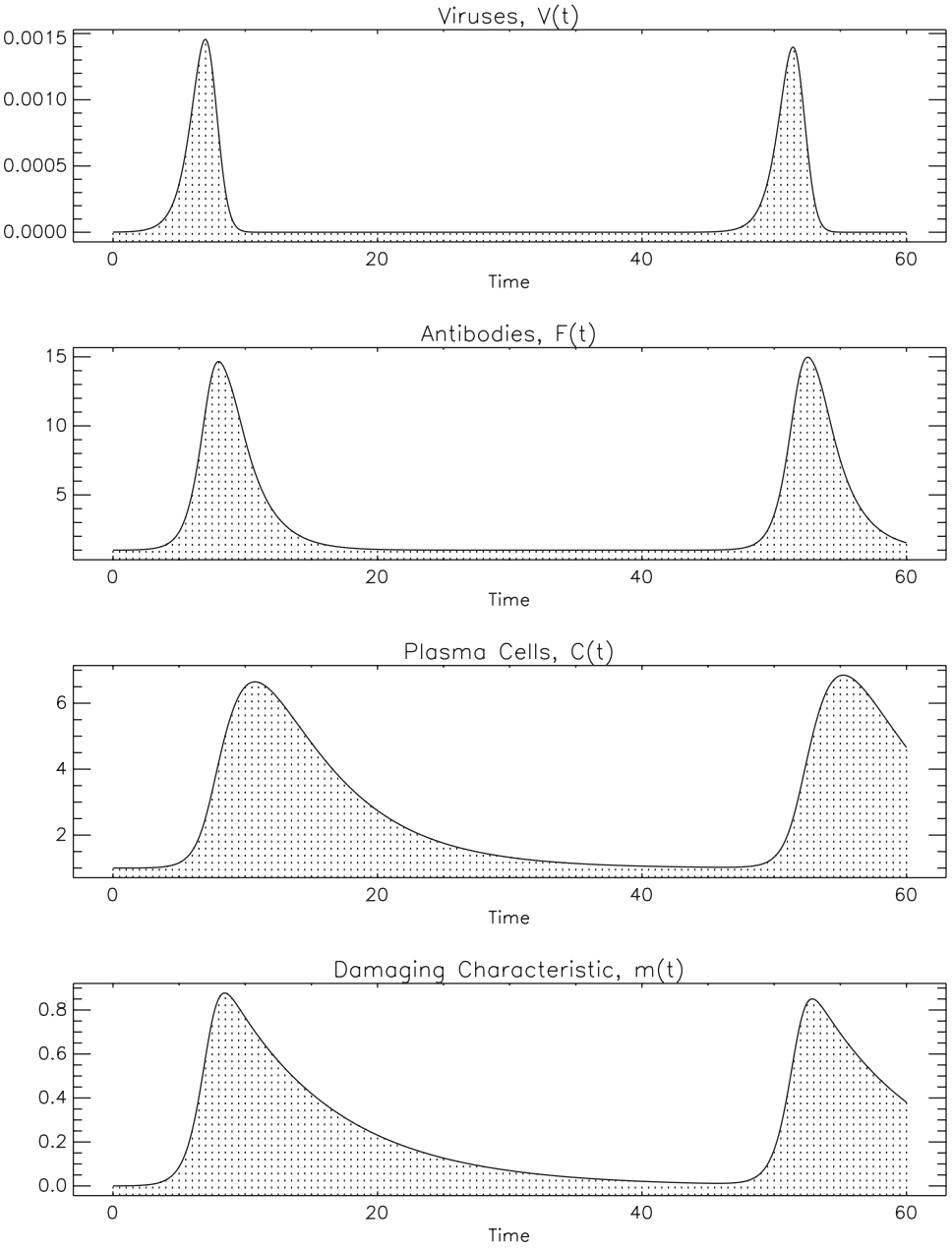


Figure 7.19. *Marchuk delay equation solution.*

Example 7.4 FINITE HORIZON OPTIMAL CONTROL. Deshmukh, Ma, and Butcher [72] present an example they describe as follows:

> The mathematical models of certain engineering processes and systems are represented by delay differential equations with time periodic coefficients. Such processes and systems include the machine tool dynamics in metal cutting operations such as milling and turning with periodically varying cutting speed or impedance and parametric control of robots, etc. Delay differential equations have been used to model nonlinear systems where finite delay in feedback control can have adverse effects on closed loop stability.

They propose a linear time-periodic delay differential system

$$\mathbf{x}'(\alpha) = \mathbf{A}_1(\alpha)\mathbf{x}(\alpha) + \mathbf{A}_2(\alpha)\mathbf{x}(\alpha-\tau) + \mathbf{B}(\alpha)\mathbf{v}(\alpha), \tag{7.172}$$

$$\mathbf{x}(\alpha) = \boldsymbol{\phi}(\alpha) \qquad \text{for } -\tau \le \alpha \le 0, \tag{7.173}$$

where $\mathbf{x}(\alpha)$ is the n-dimensional state vector, and $\mathbf{v}(\alpha)$ is the m-dimensional control vector. Differentiation with respect to time α is denoted by $\mathbf{x}' = d\mathbf{x}/d\alpha$. The matrices $\mathbf{A}_1(\alpha) = \mathbf{A}_1(\alpha - T)$ and $\mathbf{A}_2(\alpha) = \mathbf{A}_2(\alpha - T)$ are $n \times n$ periodic matrices with period T, and the startup vector function $\boldsymbol{\phi}(\alpha)$ is defined on the interval $[-\tau, 0]$. The $n \times m$ matrix $\mathbf{B}(\alpha) = \mathbf{B}(\alpha - T)$ is also periodic. In [72] the authors also assume a single fixed delay equal to the natural period of the parametric excitations $\tau = T > 0$. The objective is to minimize the quadratic

$$J = \frac{1}{2}\mathbf{x}^{\mathsf{T}}(\alpha_F)\mathbf{S}\mathbf{x}(\alpha_F) + \frac{1}{2}\int_0^{\alpha_F} \Big[\mathbf{x}^{\mathsf{T}}(\alpha)\mathbf{Q}(\alpha)\mathbf{x}(\alpha) + \mathbf{v}^{\mathsf{T}}(\alpha)\mathbf{R}(\alpha)\mathbf{v}(\alpha)\Big]\, d\alpha, \tag{7.174}$$

over the *finite horizon* $0 \le \alpha \le \alpha_F$. $\mathbf{S}$ and $\mathbf{Q}(\alpha)$ are $n \times n$ symmetric positive semidefinite matrices, $\mathbf{R}(\alpha)$ is $m \times m$ symmetric positive definite, and both $\mathbf{Q}(\alpha)$ and $\mathbf{R}(\alpha)$ are periodic with period T.

To illustrate their approach the authors in [72] address an example with two delay intervals that describes a controlled delay Mathieu equation

$$\begin{bmatrix} x_1'(\alpha) \\ x_2'(\alpha) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4\pi^2(a + c\cos 2\pi\alpha) & 0 \end{bmatrix} \begin{bmatrix} x_1(\alpha) \\ x_2(\alpha) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 4\pi^2 b\cos 2\pi\alpha & 0 \end{bmatrix} \begin{bmatrix} x_1(\alpha-1) \\ x_2(\alpha-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(\alpha) \tag{7.175}$$

with startup function

$$\begin{bmatrix} x_1(\alpha) \\ x_2(\alpha) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{for } -1 \le \alpha \le 0 \tag{7.176}$$

for $k = 1, \ldots, N$. The system parameters are given by the values $a = 0.2$, $b = 0.5$, and $c = 0.2$ and the uncontrolled system is unstable. The delay interval and period are both one, $\tau = T = 1$. The goal is to drive the final state to zero, which is reflected by the objective

$$J = \frac{10^4}{2}\mathbf{x}^{\mathsf{T}}(\alpha_F)\mathbf{x}(\alpha_F) + \int_0^{\alpha_F} \Big[\mathbf{x}^{\mathsf{T}}(\alpha)\mathbf{x}(\alpha) + v^2(\alpha)\Big]\, d\alpha, \tag{7.177}$$

where the final time is $\alpha_F = N\tau = N$. Two cases will be considered, one with two delay intervals $N = 2$ for comparison with the original authors, and a second, more realistic case with $N = 50$ intervals.

To apply the method of steps we introduce the expression $\alpha = t + (k-1)\tau$ that relates the original time α to the time on a delay interval t with $0 \le t \le \tau$. Then let us define the expanded state vector

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{y}_1(t) \\ \mathbf{y}_2(t) \\ \vdots \\ \mathbf{y}_N(t) \end{bmatrix} \tag{7.178}$$

with a corresponding definition for the expanded control $\mathbf{u}(t)$. Departing from our standard notational convention the components of $\mathbf{y}$ and $\mathbf{u}$ are defined by

$$\mathbf{y}_k(t) = \mathbf{x}(\alpha) \qquad \text{for } (k-1)\tau \le \alpha \le k\tau, \tag{7.179a}$$

$$\mathbf{u}_k(t) = \mathbf{v}(\alpha) \qquad \text{for } (k-1)\tau \le \alpha \le k\tau, \tag{7.179b}$$

where the subscript $k = 1, \ldots, N$ corresponds to a particular delay interval. Applying this transformation to (7.172) leads to the system of nN ODEs

$$\dot{\mathbf{y}}_k = \mathbf{A}_1(\alpha)\mathbf{y}_k + \mathbf{A}_2(\alpha)\mathbf{y}_{k-1} + \mathbf{B}(\alpha)\mathbf{u}_k \tag{7.180}$$

for $k = 1, \ldots, N$ defined on the domain $0 \le t \le \tau$, with $\alpha = t + (k-1)\tau$. To ensure consistency with the original DDE we impose the additional boundary conditions

$$\mathbf{y}_k(0) = \mathbf{y}_{k-1}(\tau), \tag{7.181a}$$

$$\mathbf{u}_k(0) = \mathbf{u}_{k-1}(\tau). \tag{7.181b}$$

The startup conditions are given by

$$\mathbf{y}_0(t) = \boldsymbol{\phi}(t) \qquad \text{for } -\tau \le t \le 0 \tag{7.182}$$

and the objective function (7.174) is recast as

$$J = \frac{1}{2}\mathbf{y}_N^{\mathsf{T}}(\tau)\mathbf{S}\mathbf{y}_N(\tau) + \frac{1}{2}\int_0^\tau \sum_{k=1}^{N} \left[\mathbf{y}_k^{\mathsf{T}}\mathbf{Q}(\alpha)\mathbf{y}_k + \mathbf{u}_k^{\mathsf{T}}\mathbf{R}(\alpha)\mathbf{u}_k\right] dt. \tag{7.183}$$

Thus the original DDE control problem has be transformed to a standard optimal control problem defined by the ODE system (7.180), with boundary conditions (7.181a), (7.181b), and (7.182), and objective (7.183). As such, the $\mathbb{SOCS}$ algorithm is directly applicable. Furthermore, observe that when the linear ODE system (7.180) is discretized, the resulting finite-dimensional NLP constraints are *linear* functions of the NLP variables. Also, after discretization the objective function is a *quadratic* function of the NLP variables. Since the boundary conditions are also linear functions of the NLP variables the resulting finite-dimensional NLP problem is just a QP problem (cf. Section 1.10). As such, the QP subproblem can be solved in one step, regardless of the initial guess. To exploit this computational efficiency we set the NLP algorithm option to M (cf. Section 2.6.2), thereby omitting an unnecessary step to locate a feasible point. Second, the gradient, Jacobian, and Hessian information needed to define the QP subproblem is normally computed using sparse central differences. However, since the truncation error for all linear and quadratic functions is zero, we have $|f'''(x)| = 0$ in (1.165). Thus to minimize the finite difference

error given by (1.165) a "large" perturbation size should be used. For this case we choose $\delta = 1$. Furthermore, with limited experimentation one can simply pick a fixed discretization mesh size, thereby avoiding mesh refinement and the underlying solution of many NLP subproblems. In summary we choose to discretize the problem with 100 equally spaced grid points, using the HSC method. As an initial guess for all of the NLP variables we simply use zero, with the exception of the boundary values (7.176). Figure 7.20 displays the solutions obtained for both cases. The solution with two delay intervals is shown in the top plot, and the other three plots contain the time history for the state and control over 50 delay intervals. For the latter case the optimal objective value is $F^* = 4.5677520 \times 10^1$ with discretization error $\epsilon = 6.61 \times 10^{-8}$.

7.4 In-Flight Dynamic Optimization of Wing Trailing Edge Surface Positions

Example 7.5 TRAILING EDGE VARIABLE CAMBER. When designing a commercial aircraft, one major consideration is the aerodynamic efficiency. Ideally, the shape of the aircraft is designed to optimize some measure of aerodynamic performance such as the ratio of lift to drag (L/D). However, aerodynamic forces depend on many factors, so for example an airfoil designed to maximize L/D when flying at 32,000 ft and Mach .84 is *not optimal* when the aircraft flies at any other condition (e.g., 35,000 ft and $M = .83$). This deficiency can be addressed by changing the shape of the airfoil to give the best performance at the actual operating condition. One technique that can be implemented on modern commercial aircraft is to vary the camber of the wing using minor changes in some of the control surfaces. An approach that was tested for an aircraft using a single control surface is described by Gilyard [101] and Gilyard, Georgie, and Barnicki [102]. Similar ideas are presented by Martins and Catalano [132].

In spite of its obvious appeal, practical implementation of this *trailing edge variable camber* (TEVC) approach must address a number of nontrivial technical challenges. First, the expected change in drag caused by changing the camber of the wing is very small—typically less than 1% of the base drag. Second, on Boeing commercial aircraft as many as four different control surfaces can be used to modify the camber. Third, the displacement of the control surfaces is small—typically less than 2 deg—and because of mechanical limitations the flap positions can be specified only to within 1/4 degree. Finally, the *base drag* is affected by many random factors, including wind and atmospheric affects, engine performance, vehicle weight, etc.

This example describes a rather straightforward camber estimation process developed in collaboration with Paul H. Carpenter.[10] After reaching the operating flight condition (typically cruise altitude and velocity), the camber control surfaces are "swept" through a predetermined time profile. During this period, which typically lasts from 3 to 5 min, dynamic behavior is observed and measurement data is recorded at a high frequency (e.g. 20 samples per second.) Using a high-fidelity model of the dynamics, the trajectory is reconstructed. The solution to this *inverse problem* yields an estimate for the time history of the aerodynamic coefficients (C_D, C_L, and C_y). From the time history of the aerodynamic coefficients, we then construct a parametric representation for the drag coefficient

[10] Aero Performance Tools and Methods, The Boeing Company, P.O. Box 3707, MS 67-FE, Seattle, WA 98124-2207.

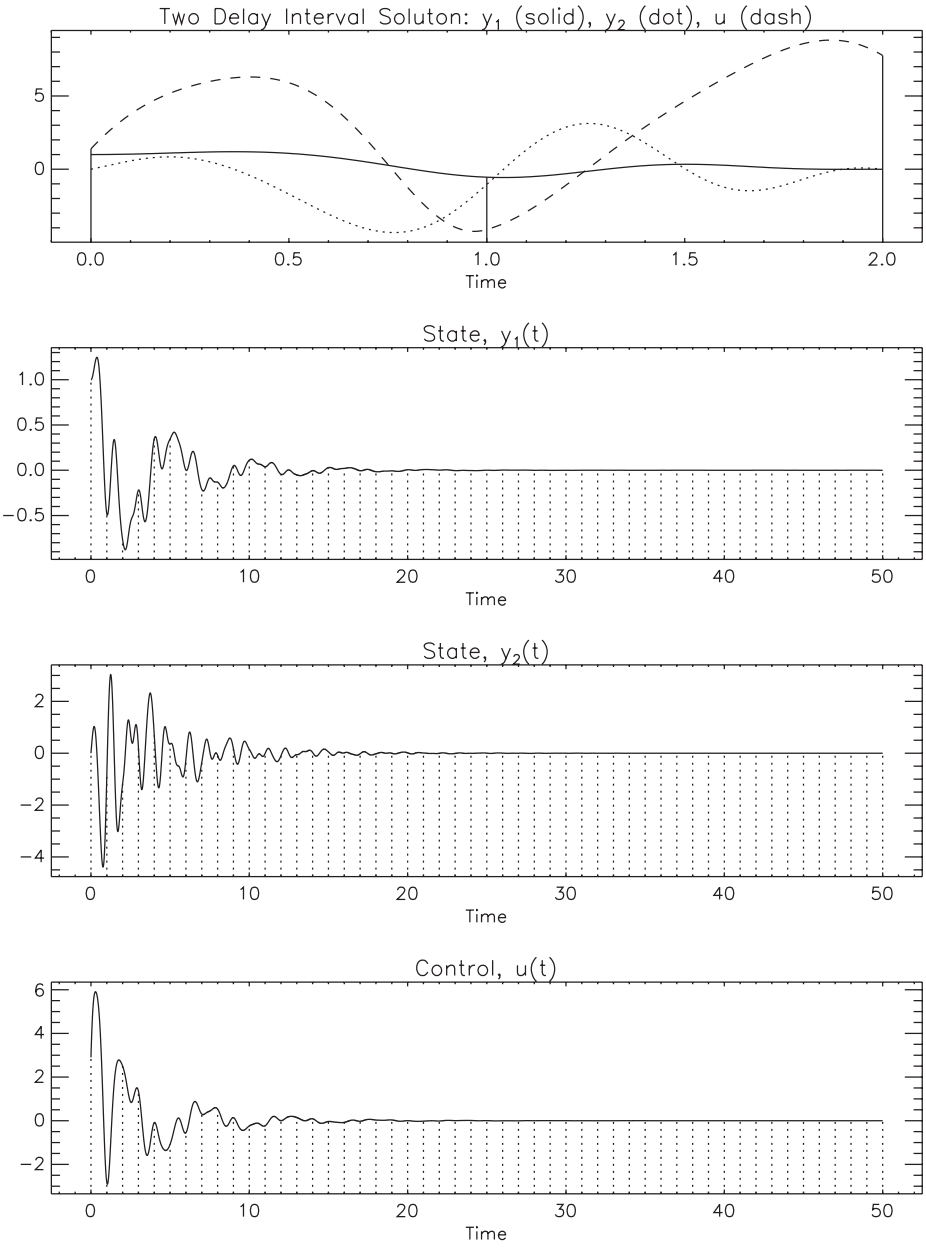


Figure 7.20. *Finite horizon optimal control delay equation solutions.*

as a function of the camber control variables. This drag model is constructed by solving a scattered data least squares approximation using a constrained multivariate B-spline representation. As the final step, we compute the camber control variables that minimize the drag model.

7.4.1 Aircraft Dynamics for Drag Estimation

In general, the dynamic behavior of an aircraft can be described by a set of differential equations. Using a six degree of freedom model as a starting point a number of changes can be introduced in order to construct a dynamic model that is appropriate for the estimation process. The dynamics can be formulated as the following *differential-algebraic system*:

$$\dot{z} = -u\sin\theta + v\sin\phi\cos\theta + w\cos\theta\cos\phi, \tag{7.184}$$

$$\dot{u} = vr - wq - g\sin\theta + \frac{F_x}{m}, \tag{7.185}$$

$$\dot{v} = wp - ur + g\sin\phi\cos\theta + \frac{F_y}{m}, \tag{7.186}$$

$$\dot{w} = uq - vp + g\cos\phi\cos\theta + \frac{F_z}{m}, \tag{7.187}$$

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta, \tag{7.188}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi, \tag{7.189}$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta, \tag{7.190}$$

$$\dot{W} = -\mu(h, M, T_g), \tag{7.191}$$

$$0 = n_x - \left[\frac{F_x}{m} + \Delta n_x\right], \tag{7.192}$$

$$0 = n_y - \left[\frac{F_y}{m} + \Delta n_y\right], \tag{7.193}$$

$$0 = n_z - \left[\frac{F_z}{m} + \Delta n_z\right], \tag{7.194}$$

$$0 = V_g - \sqrt{u^2 + v^2 + w^2 - \dot{z}^2}, \tag{7.195}$$

$$0 = \delta - \arctan\left(\frac{v\cos\phi - w\sin\phi}{u\cos\theta + v\sin\phi\sin\theta + w\cos\phi\sin\theta}\right). \tag{7.196}$$

Observe that the time varying roll, pitch, and yaw rates $p(t)$, $q(t)$, and $r(t)$ are treated as algebraic variables rather than differential variables; i.e., they are *algebraic states*. The approximation is reasonable since the rates are very small at a cruise condition, and this eliminates the need to estimate the moments $L(t)$, $M(t)$, and $N(t)$.

The total forces that appear in (7.185)–(7.187) are expressed in the body axes as the sum of aerodynamic and thrust terms, i.e.,

$$F_x = (-C_D\cos\alpha + C_L\sin\alpha)\bar{q}S + \frac{1}{\kappa}T_g - D_r\cos\beta\cos\alpha, \tag{7.197}$$

$$F_y = C_y\bar{q}S - D_r\sin\beta, \tag{7.198}$$

$$F_z = (-C_L\cos\alpha - C_D\sin\alpha)\bar{q}S - \frac{1}{\kappa}\tan\eta_{xz}T_g - D_r\cos\beta\sin\alpha. \tag{7.199}$$

Assuming the left and right engines have equal gross thrust T_g and net thrust T_n, and η_{xy} and η_{xz} are constants that define the wing incidence angles for a particular engine, the ram

drag, lift, and drag are given by

$$D_r = T_g - T_n, \qquad \kappa = \sqrt{\tan^2 \eta_{xz} + \tan^2 \eta_{xy} + 1},$$
$$L = C_L \bar{q} S, \qquad D = C_D \bar{q} S.$$

The lift and drag forces are defined in terms of the drag coefficient C_D, the lift coefficient C_L, the reference area S, and the dynamic pressure $\bar{q}$. The vehicle weight W is related to the mass m by the expression $W = mg$.

As the aircraft moves the aerodynamic forces change as a function of time, in particular the total pressure and static pressure. Using values for $P_{total}(t)$, and $P_{static}(t)$ it is possible to construct auxiliary quantities. Specifically the following equations can be evaluated in sequence:

$$M = \sqrt{5}\left[\left(\frac{P_{total}}{P_{static}}\right)^{\frac{1}{3.5}} - 1\right]^{\frac{1}{2}}, \tag{7.200}$$

$$V_e = (14.3791496702540) M \sqrt{P_{static}}, \tag{7.201}$$

$$\bar{q} = \frac{V_e^2}{295.3714}. \tag{7.202}$$

These expressions define the Mach number $M(t)$, the equivalent airspeed $V_e(t)$ (knots), and the dynamic pressure $\bar{q}(t)$ (psf). The *ground speed* is given by

$$V_g = \sqrt{u^2 + v^2 + w^2 - \dot{z}^2} \tag{7.203}$$

and the *drift angle* is given by

$$\delta = \arctan\left(\frac{v\cos\phi - w\sin\phi}{u\cos\theta + v\sin\phi\sin\theta + w\cos\phi\sin\theta}\right). \tag{7.204}$$

These two consistency conditions appear as algebraic equations (7.195) and (7.196). A similar ambiguity in the decomposition of the total forces (7.197)–(7.199) is resolved by using information about the total sensed accelerations

$$n_x = \left[\frac{F_x}{m} + \Delta n_x\right], \tag{7.205}$$

$$n_y = \left[\frac{F_y}{m} + \Delta n_y\right], \tag{7.206}$$

$$n_z = \left[\frac{F_z}{m} + \Delta n_z\right], \tag{7.207}$$

where the constants Δn_x, Δn_y, and Δn_z represent observation instrumentation biases. These expressions also appear as the algebraic equations (7.192)–(7.194). A summary of the nomenclature is presented in Table 7.15.

Table 7.15. *Nomenclature.*

α	Angle of Attack (deg)	P_{total}	Total Pressure (psf)
β	Sideslip (Yaw) Angle (deg)	P_{static}	Static Pressure (psf)
δ	Drift Angle (deg)	q	Pitch Rate Y-Stability Axis (rad/sec)
μ	Total Fuel Flow (lb/sec)	r	Yaw Rate Z-Stability Axis (rad/sec)
ϕ	Euler Roll (Bank) Angle (deg)	T_g	Total Gross Thrust (lb)
ψ	Euler Yaw Angle (deg)	T_n	Total Net Thrust (lb)
θ	Euler Pitch Angle (deg)	u	X-Velocity (fps)
C_D	Reference Drag Coefficient	u_1	Aileron Deflection (deg)
C_L	Reference Lift Coefficient	u_2	Flaperon Deflection (deg)
C_y	Reference Side Force Coefficient	u_3	Inboard Flap Deflection (deg)
n_x	Acceleration X-Direction (g)	v	Y-Velocity (fps)
n_y	Acceleration Y-Direction (g)	V_g	Ground Speed (knots)
n_z	Acceleration Z-Direction (g)	w	Z-Velocity (fps)
p	Roll Rate X-Stability Axis (rad/sec)	z	Z-Position (ft)

The motion is described by 8 differential variables $(z,u,v,w,\phi,\theta,\psi,W)$ and 18 algebraic variables

$$(T_g,T_n,\mu,C_L,C_D,C_y,p,q,r,n_x,n_y,n_z,\alpha,\beta,V_g,\delta,P_{total},P_{static}).$$

In addition, the three observation biases Δn_x, Δn_y, and Δn_z must be determined. The total set of 26 dynamic variables and 3 parameters must satisfy the 8 differential equations (7.184)–(7.191) and the 5 algebraic equations (7.192)–(7.196).

7.4.2 Step 1: Reference Trajectory Estimation

The goal of the first step is to compute time histories for the complete set of dynamic variables such that the trajectory dynamics defined by the DAEs (7.184)–(7.196) are satisfied. Denote the dynamic variables by the vector $\mathbf{z}(t)$ with corresponding observations by the vector $\widehat{\mathbf{z}}(t)$. These quantities must be chosen to minimize the error at the observation data points, i.e.,

$$F=\sum_{k=1}^{N}\sum_{\substack{j\\ j\subset\mathcal{O}}}\left[\frac{(\mathbf{z}_j(t_k)-\widehat{\mathbf{z}}_j(t_k))}{\sigma_j}\right]^2. \tag{7.208}$$

For a typical flight test of 300 sec duration, with 20 data points per second, the total number of points $N\sim 6000$. Statistical information about the observation data is included using specified values for the standard deviations σ_j. Note the summation includes all dynamic variables for which data are available (the set of observations $\mathcal{O}$) but *excludes* the quantities $u(t)$, $v(t)$, $W(t)$, $C_L(t)$, $C_D(t)$, and $C_y(t)$. Thus observation data are available for only 20 of the 26 total dynamic variables. Note that for this step information about the camber control surfaces, i.e., the camber variable data $\widehat{\mathbf{u}}_k$, is *not* used. The solution to this least squares problem is the best estimate for the time histories, which are denoted as $C_L^*(t)$, $C_D^*(t)$, and $C_y^*(t)$. This large-scale parameter estimation problem can be solved using the method described in Chapter 5.

7.4.3 Step 2: Aerodynamic Drag Model Approximation

The first step provides estimates for the aerodynamic coefficients as a function of time; however, it is desirable to construct a parametric model for the drag coefficient. The aerodynamic properties of the aircraft can be altered by control surfaces on the wing. For the Boeing 777 aircraft, two control surfaces, namely the aileron and flaperon, can be utilized. For the 787 aircraft, the inboard flap can also be used in addition to aileron and flaperon. Thus we propose constructing the tensor product B-spline model

$$\tilde{C}_D(u_1,u_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} a_{ij} B_i(u_1) B_j(u_2) \tag{7.209a}$$

for aircraft with two control surfaces and

$$\tilde{C}_D(u_1,u_2,u_3) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} a_{ijk} B_i(u_1) B_j(u_2) B_k(u_3) \tag{7.209b}$$

for aircraft with three control surfaces. To ensure that the model $\tilde{C}_D$ has a minimum it must satisfy the constraints

$$\frac{\partial^2 \tilde{C}_D}{\partial u_j^2} \geq 0 \tag{7.210}$$

with respect to the camber control surfaces u_j. The number of coefficients a in model (7.209a) or (7.209b) is dictated by two factors: the order of the B-spline representation, and the number of internal knots.

In order to compute the coefficients a_{ij} and thereby define the aerodynamic model (7.209a) or (7.209b) we utilize the results of the reference trajectory estimation, i.e., Step 1. Specifically we choose the coefficients a_{ij} to minimize

$$F = \sum_{k=1}^{N} \left\{ C_D^*(t_k) - \tilde{C}_D\left[\hat{u}_1(t_k), \hat{u}_2(t_k)\right] \right\}^2 \tag{7.211a}$$

for aircraft with two control surfaces and

$$F = \sum_{k=1}^{N} \left\{ C_D^*(t_k) - \tilde{C}_D\left[\hat{u}_1(t_k), \hat{u}_2(t_k), \hat{u}_3(t_k)\right] \right\}^2 \tag{7.211b}$$

for aircraft with three control surfaces and satisfy the curvature constraints (7.210). Note that the objective function (7.211a) or (7.211b) involves the quantity $C_D^*(t_k)$, which is available from the reference trajectory reconstruction in Step 1. Since the variation in drag is

attributed to the changes in the camber variables, this information is incorporated into the model using the data $\hat{u}_j(t_k)$. This constrained scattered data fit problem can be solved using the sparse NLP techniques introduced in Chapter 2.

7.4.4 Step 3: Optimal Camber Prediction

The final step in the process is to determine the best performance over an extended period of time at cruise using a constant setting for the camber variables. This can be achieved if we determine the camber to minimize drag. By construction the camber variables appear *only* in the quantity $\tilde{C}_D(\mathbf{u})$, so the optimal values with respect to $\mathbf{u}$ can be computed just by minimizing

$$F = \tilde{C}_D(u_1, u_2) \tag{7.212a}$$

for aircraft with two control surfaces and

$$F = \tilde{C}_D(u_1, u_2, u_3) \tag{7.212b}$$

for aircraft with three control surfaces, at fixed values of $\bar{C}_L$ and $\bar{M}$. As a practical matter we impose simple bounds on this two or three variable optimization problem. Let us denote the minimum value as

$$C_D^{\circledast} = \tilde{C}_D(u_1^*, u_2^*). \tag{7.213}$$

7.4.5 Numerical Results

777-200ER Flight Test

The approach described has been implemented using flight test data for a particular Boeing 777-200ER airplane. To illustrate the method a series of representative results are plotted in Figures 7.21–7.24. Figures 7.21–7.23 share a common format. The left column of each figure displays the time history for a dynamic variable, as well as the measured (observation) for the same quantity. The right column presents the error between the data and the reconstructed variable. So, for example, the first row in Figure 7.21 illustrates the variation in the horizontal velocity during the test. The right column plots the error between the measured velocity and the reconstructed value. For this example the error in velocity varies between $\pm.1$ (fps). The average error in the reconstructed velocity over the entire test .00020674815 is displayed in the title. Figure 7.23 displays the quantities $C_L^*(t) - C_L^{\circledast}$ and $C_D^*(t) - C_D^{\circledast}$, which are the estimated time histories for the aerodynamic coefficients relative to their respective optimal values. Using the reconstructed aerodynamic coefficient data, the tensor product spline model (7.209a) is computed. Figure 7.24 displays the percent change relative to the minimum, i.e., the quantity

$$\Delta C_D = 100 \times \frac{(\tilde{C}_D(u_1, u_2) - C_D^{\circledast})}{C_D^{\circledast}}.$$

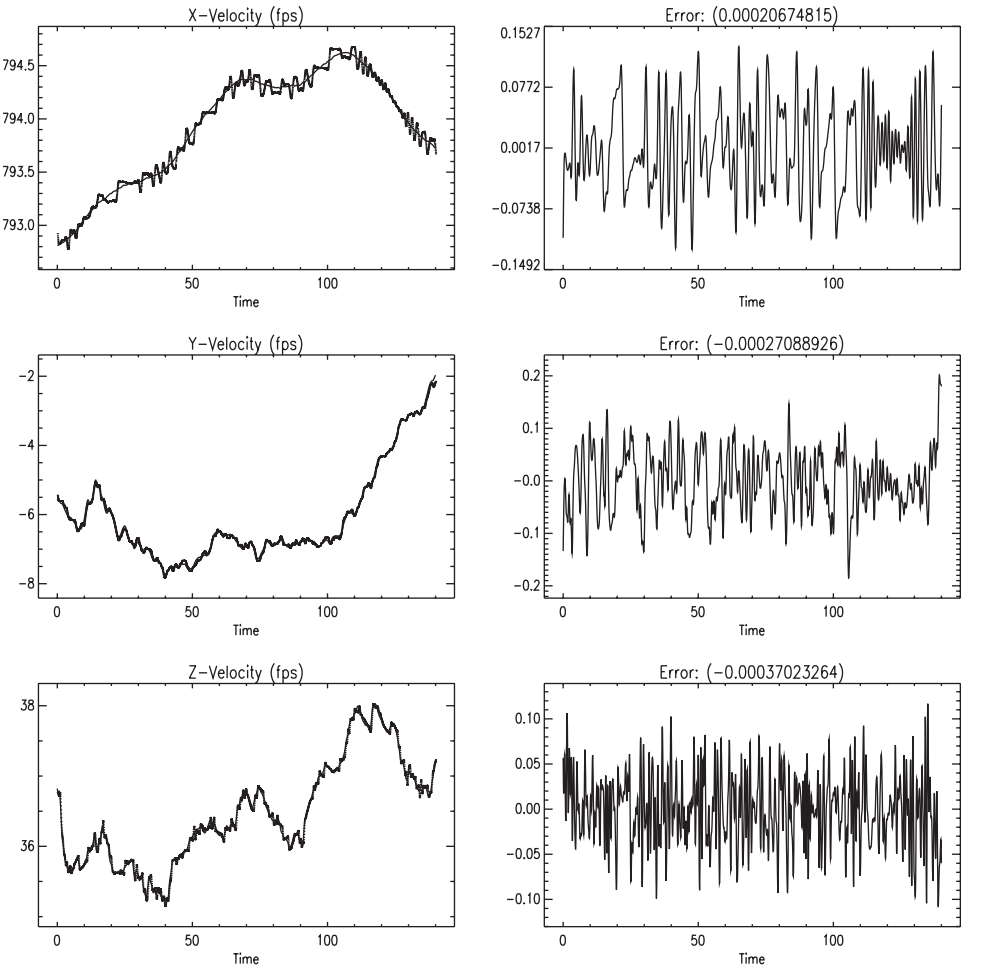


Figure 7.21.

Performance Comparison

Step 1 requires the solution of a large-scale parameter estimation problem, and Figure 7.25 illustrates how the algorithm proceeds. The overall solution required 10 iterations. The first three iterations locate a feasible point; that is, the defect constraints are solved. An initial guess for the optimization variables $\mathbf{x}$ (4.105) is constructed by linearly interpolating the observation data. In Figure 7.25 we see that the initial constraint error $\|\mathbf{c}(\mathbf{x}^{(1)})\|_2 = .137$, and after two iterations it is reduced to $\|\mathbf{c}(\mathbf{x}^{(3)})\|_2 = 8.5 \times 10^{-9}$. The next six iterations are directed toward minimizing the objective function while also satisfying the constraints. At the beginning of this portion of the algorithm, the constraints are satisfied $\|\mathbf{c}(\mathbf{x}^{(4)})\|_\infty = 2.3 \times 10^{-9}$, and they are also satisfied at the solution $\|\mathbf{c}(\mathbf{x}^{(10)})\|_\infty = 7.1 \times 10^{-15}$. During the optimization process the objective function is reduced from $F(\mathbf{x}^{(4)}) = 2.92 \times 10^{-5}$ to $F(\mathbf{x}^{(10)}) = 8.1 \times 10^{-7}$, and the error in the projected gradient is reduced from 9.9×10^{-5}

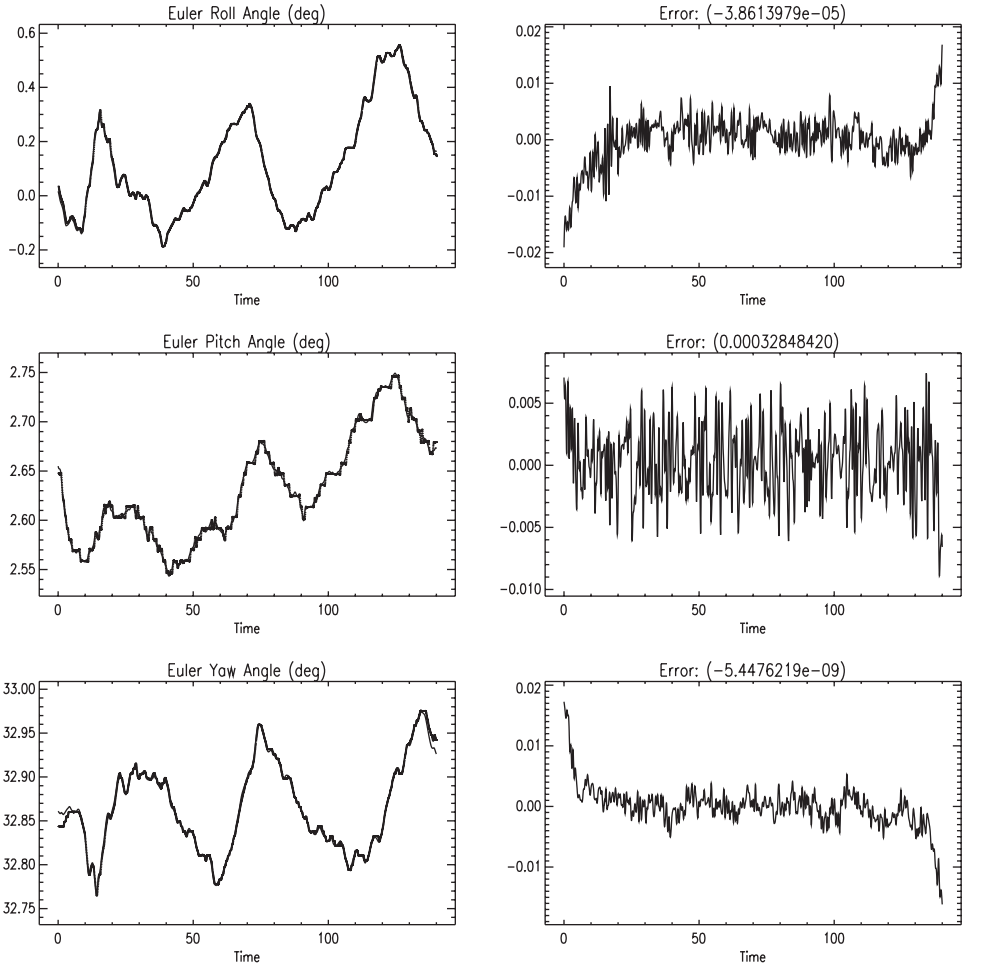


Figure 7.22.

to 5.1×10^{-16}. However, note that constraint feasibility is not maintained during the optimization steps, and at the seventh iteration $\|\mathbf{c}(\mathbf{x}^{(7)})\|_\infty = 5.9 \times 10^{-5}$. Although the values quoted have all been scaled to improve numerical conditioning, it is important to note that the optimal value of the objective function is *nonzero*. In fact, at the solution the five largest (unscaled) residuals are, respectively, $5.29, -4.32, 4.25, -4.21, 3.91$. Because the objective is nonzero and the residuals are nonlinear, a traditional Gauss–Newton method would exhibit linear convergence. In contrast, the $\mathbb{SOPE}$ algorithm demonstrates quadratic convergence, which is achieved because the full Hessian matrix (2.56) is utilized. This behavior is illustrated in Figure 7.25 by the dramatic reduction in the projected gradient and constraint error during the final iterations.

In order to fully appreciate the algorithmic behavior it is worthwhile to look a little deeper into the problem characteristics which are summarized in Table 7.16. At each iteration the optimization algorithm must solve a large system of linear equations, the KKT

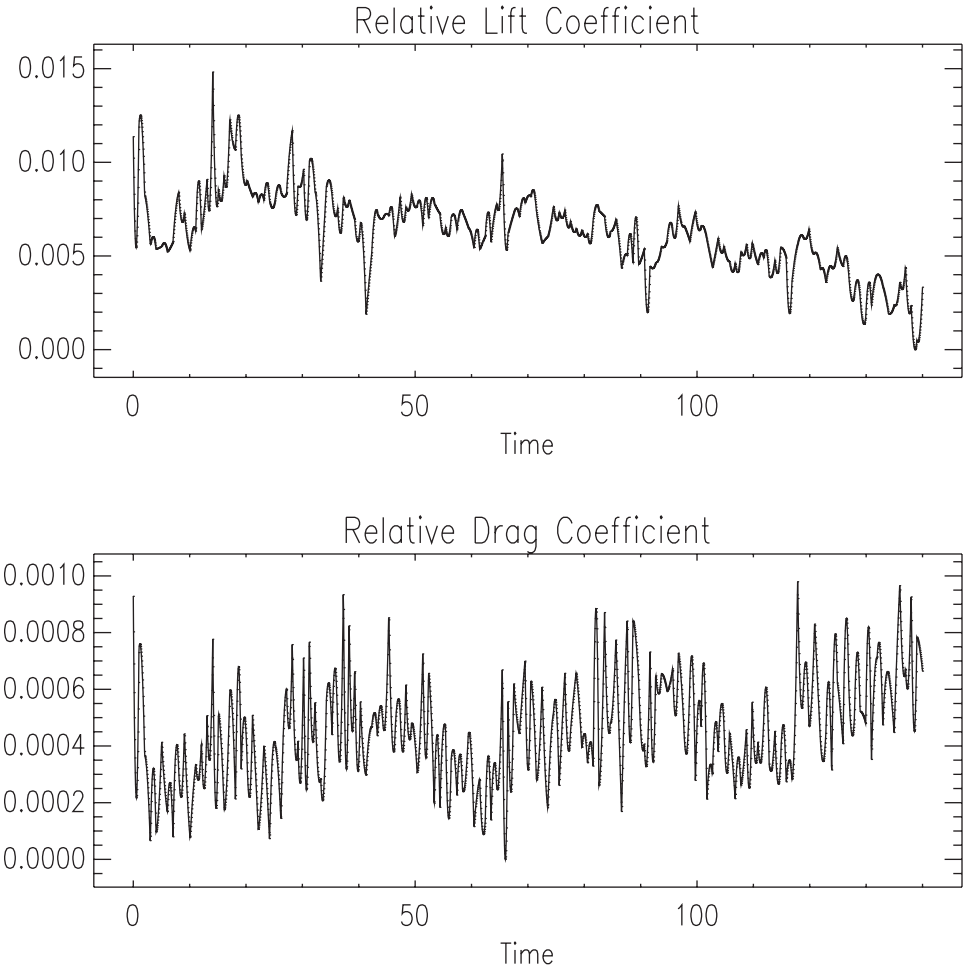


Figure 7.23.

system (7.215). For this example there are 176050 equations. Because there are 18735 variables, the matrix $\mathbf{V}$ is 18735×18735. Since there are 147290 residuals the matrix $\mathbf{R}$ is 147290×18735. The total number of defect constraints is 10025, and consequently the matrix $\mathbf{G}$ is 10025×18735. Observe that of the 18735 variables, 10025 are determined by constraints, and one is fixed by the initial condition on airplane weight $W(0) = w_0$, leaving 8709 degrees of freedom available to minimize the objective function (7.208). Even though the KKT system is large, it is also *sparse*. It is imperative that sparsity is exploited in order to efficiently solve the problem. This is achieved in two ways.

First, we must compute the matrices $\mathbf{V}$, $\mathbf{R}$, and $\mathbf{G}$, and this is done using finite difference approximations. Now a standard finite difference technique would require at least 18735 perturbations just to compute a forward approximation, which is clearly prohibitive. However, by exploiting the sparse difference technique described in Section 5.4 it is possible to compute all first and second derivative information with just 170 perturbations!

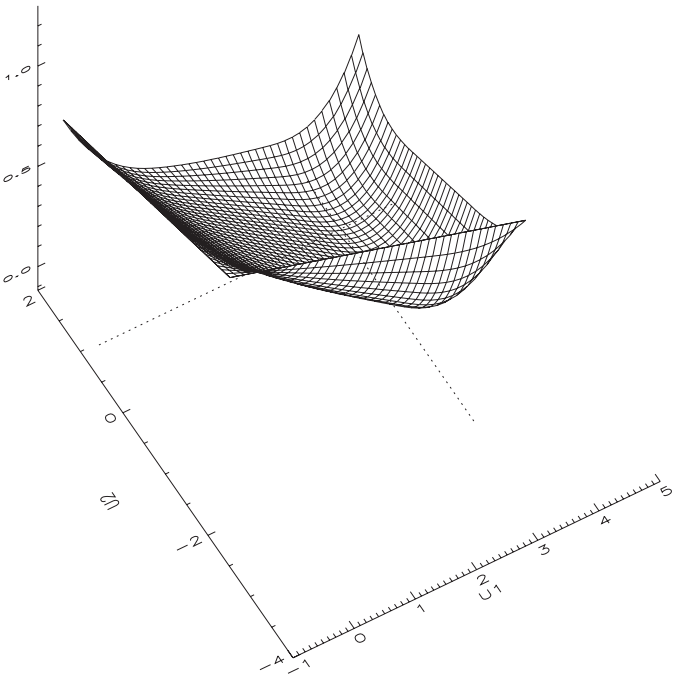


Figure 7.24. *Aerodynamic model surface change.*

This is possible because the total number of index sets for this problem is just 17 and is determined by the right-hand-side sparsity template given by

$$\mathcal{T} = \text{struct}\left[\begin{array}{c|c|c} \frac{\partial \mathbf{f}}{\partial \mathbf{y}} & \frac{\partial \mathbf{f}}{\partial \mathbf{u}} & \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{y}} & \frac{\partial \mathbf{g}}{\partial \mathbf{u}} & \frac{\partial \mathbf{g}}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{w}}{\partial \mathbf{y}} & \frac{\partial \mathbf{w}}{\partial \mathbf{u}} & \frac{\partial \mathbf{w}}{\partial \mathbf{p}} \end{array}\right] = \left[\begin{array}{ccccccccc} \times\times\times\times\times & & & & & & & & \\ \times\times\ \times & \times\times\times & \times\times & & \times\times & & \times\times & \times\times & \\ \times\ \times\times\times & \times\times\times & & \times\times & \times & & \times & \times\times & \\ \times\times\ \times\times & \times\times\times & \times\times & \times\times & & & \times\times & \times\times & \\ & \times\times & & & \times\times\times & & & & \\ & \times & & & \times\times & & & & \\ & \times\times & & & \times\times & & & & \\ & & & \times & & & & & \\ & & \times\times\times & \times\times & & \times & \times\times & \times\times\times & \\ & & \times\times\times & & \times & & \times & \times\times & \times \\ & & \times\times\times & \times\times & & & \times\times\times & \times\times & \times \\ \times\times\times\times\times & & & & & & & \times & \\ \times\times\times\times\times & & & & & & & & \times \end{array}\right]. \tag{7.214}$$

The combined benefit of sparse finite differences in conjunction with quadratic convergence means that the entire problem is solved using only 707 evaluations, or 612901 evaluations of the right-hand-side dynamic functions given in (7.184)–(7.196).

The second computational benefit accrues from the matrix sparsity itself. It is well known that the computational complexity for solving a system of n *dense* linear equations is $\mathcal{O}(n^3)$. In contrast for a sparse linear system the cost is $\mathcal{O}(\kappa n)$, where κ is a factor related to sparsity. By using a symmetric indefinite decomposition of the underlying KKT matrix, as described in Section 2.10 the time required to solve the linear equations on all 10

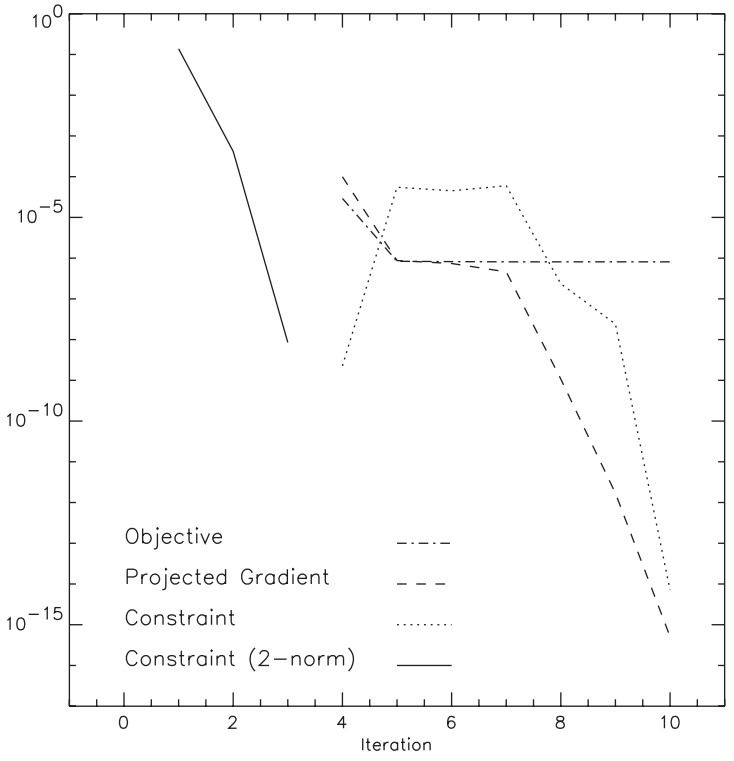


Figure 7.25. *Iteration history.*

Table 7.16. *Step* 1 *computational performance characteristics.*

Number of KKT Equations	176050
Number of Residuals	147290
Number of Variables	18735
Number of Constraints	10025
Number of Degrees of Freedom	8709
Number of Index Sets	17
Number of Function Evaluations per Jacobian	34
Number of Function Evaluations per Jacobian/Hessian	170
Total Number of Function Evaluations	707
Total Number of Right-Hand-Side Evaluations	612901
Number of Iterations	10
Number of Hessian Evaluations	3
Total Time Inside NLP (sec)	28.37
Total CPU Time (sec)	30.11

iterations was only 28.37 sec, and the entire solution was obtained in just 30.11 sec. This would not be possible without exploiting matrix sparsity.

In order to solve the large-scale least squares problems encountered in Step 1 of the process one must face the computational issues associated with the normal matrix $\mathbf{R}^T\mathbf{R}$. It is well known that simply forming the matrix is ill-conditioned, and accurate solution

of linear systems involving the normal matrix can be problematic. This ill-conditioning is avoided by using a symmetric indefinite decomposition of the underlying KKT matrix, as described in Section 2.10. Thus a new estimate for the variables (4.105) is given by $\overline{\mathbf{x}} = \mathbf{x} + \mathbf{p}$, which is the solution to the QP subproblem (2.61)–(2.62). This approach requires solving the linear system

$$\begin{bmatrix} \mathbf{V} & \mathbf{R}^{\mathsf{T}} & \mathbf{G}^{\mathsf{T}} \\ \mathbf{R} & -\mathbf{I} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\mathbf{p} \\ -\mathbf{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{\mathsf{T}}\mathbf{r} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{7.215}$$

where $\mathbf{G}$ is the Jacobian matrix of the defect constraints. In contrast when the normal matrix is formed explicitly, the search direction can be computed by solving the linear system

$$\begin{bmatrix} \mathbf{H}_L & \mathbf{G}^{\mathsf{T}} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\mathbf{p} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{\mathsf{T}}\mathbf{r} \\ \mathbf{0} \end{bmatrix}. \tag{7.216}$$

The optimization search direction $\mathbf{p}$ can be computed by solving the KKT system written in either the *sparse tableau* format (7.215) or in the *normal matrix* format (7.216). The sparse tableau format is well-conditioned, whereas the normal matrix format is ill-conditioned. On the other hand the sparse tableau format is much larger than the normal matrix format. To assess the relative merits of each approach for this example a series of cases were solved using various options. Three different discretization stepsizes were used, namely $h = 1, 2, 4$ seconds. Two different (fourth order) discretization techniques, specifically HSC and HSS, were tested. In each case both the sparse tableau and the normal matrix formats were tried. The results are summarized in Table 7.17. Three of the

Table 7.17. *Algorithm parameter options.*

Case	h^a	Disc	KKT Option	N^b	CPU^c
1	1	C^d	N	47154	334.780
2	1	C	T	351434	F
3	1	S^e	N^f	59314	594.830
4	1	S	T^g	363594	F
5	2	C	N	23532	287.780
6	2	C	T	327812	102.540
7	2	S	N	29596	277.700
8	2	S	T	314139	70.5900
9	4	C	N	11752	F
10	4	C	T	316032	85.4500
11	4	S	N	14776	436.830
12	4	S	T	319056	62.9000

[a] Discretization stepsize
[b] Dimension of KKT matrix
[c] CPU seconds
[d] HSC discretization
[e] HSS discretization
[f] Normal matrix KKT option (7.216)
[g] Sparse tableau KKT option (7.215)

cases failed to converge either because of storage or iteration limits. Cases 6, 8, 10, and 12 suggest that the sparse tableau format is more efficient provided sufficient storage is available. When storage is limited, the normal matrix approach can be utilized; however, there is a significant CPU time penalty which can be attributed to the ill-conditioning of this alternative. The default settings for production usage employ the sparse tableau format with the separated Hermite–Simpson discretization.

Constructing the parametric representation for the drag coefficient as described in Section 7.4.3 also entails solving a sparse constrained optimization problem just as in Step 1. Specifically we must compute the coefficients a_{ij} that define the spline model (7.209a). In this case it is possible to compute the necessary derivatives analytically; however, the same sparse optimization algorithm is used.