Using Direct Transcription to Compute Optimal Low Thrust Transfers Between Libration Point Orbits

John T. Betts *

February 4, 2016

Abstract

The direct transcription method has been used to solve many challenging optimal control problems. One such example involves the calculation of a low thrust orbit transfer between libration point orbits. The recent implementation of high order discretization techniques is first described and then illustrated by computing optimal low thrust trajectories between orbits about the L_1 and L_2 Earth-Moon libration points.

1 The Optimal Control Problem

The primary focus of this paper is the presentation of efficient numerical methods to solve the optimal control problem. The goal is to choose the control functions $\mathbf{u}(t)$ to minimize the objective

$$F = \int_{t_I}^{t_F} w\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t\right] dt \tag{1.1}$$

subject to the state equations

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \tag{1.2}$$

and the boundary conditions

$$\mathbf{0} = \boldsymbol{\psi}_I[\mathbf{y}(t_I), \mathbf{u}(t_I), \mathbf{p}, t_I] \tag{1.3}$$
$$\mathbf{0} = \boldsymbol{\psi}_F[\mathbf{y}(t_F), \mathbf{u}(t_F), \mathbf{p}, t_F]. \tag{1.4}$$

Denote the initial and final states by $\mathbf{y}(t_I) = \mathbf{y}_I$ and $\mathbf{y}(t_F) = \mathbf{y}_F$ respectively, with corresponding notation for other quantities. Although the problem can be stated in many equivalent formats, this *Lagrange* formulation is sufficiently general for our purpose.

2 Transcription Method

All numerical results presented here were obtained using the $\mathbb{SOS}$ (Sparse Optimizaton Suite) software. This implements a *direct transcription method* as described in [3]. The fundamental idea of a *transcription method* is to introduce a discrete approximation to the differential equations (1.2) in terms of the dynamic variables $\mathbf{y}(t)$, and $\mathbf{u}(t)$ evaluated at the discrete times

$$t_I = t_1 < t_2 < \cdots < t_M = t_F, \tag{2.1}$$

referred to as node, mesh, or grid points. In so doing the differential equation (1.2) is transcribed into a set algebraic constraints, defined in terms of a finite set of variables. Thus the optimal control problem is converted into a large nonlinear programming (NLP) problem. In principle any nonlinear programming method can be used to solve the discretized problem, but to do so the NLP must evaluate both first and second derivatives of the relevant discretization equations. These Jacobian and Hessian matrices are both large and sparse. To efficiently solve the NLP it is critical to exploit a computational benefit that 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

*Applied Mathematical Analysis, LLC; <http://www.appliedmathematicalanalysis.com/>

factor related to sparsity. The $\mathbb{SOS}$ software has two nonlinear programming algorithms, a sparse Schur-complement sequential quadratic programming (SQP) algorithm, and a primal-dual interior point (barrier) algorithm. All numerical results presented use the SQP algorithm, and although we will not discuss details of the underlying sparse nonlinear programming algorithm the reader is referred to [3, Chapter 1-2]. In fact since the computational expense of the entire algorithm is dominated by the quantity $\mathcal{O}(\kappa n)$ the remaining discussion will be focused on how to keep both κ and n as small as possible.

To summarize, the transcription method has three fundamental steps:

Direct Transcription: Transcribe the optimal control problem into a nonlinear programming (NLP) problem by discretization;

Sparse Nonlinear Program: Solve the sparse (SQP or Barrier) NLP

Mesh Refinement: Assess the accuracy of the approximation (i.e. the finite dimensional problem), and if necessary refine the discretization, and then repeat the optimization steps.

3 Runge-Kutta Methods

Let us begin by introducing methods for discretization of the differential equations (1.2). A popular family of one-step methods called *S-stage Runge–Kutta* [9] can be defined as follows:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \sum_{j=1}^{S} \beta_j \mathbf{f}_{kj}, \tag{3.1}$$

where for $1 \leq j \leq S$

$$\mathbf{y}_{kj} = \mathbf{y}_k + h_k \sum_{\ell=1}^{S} \alpha_{j\ell} \mathbf{f}_{k\ell}, \tag{3.2}$$

$$\mathbf{f}_{kj} = \mathbf{f}\left[\mathbf{y}_{kj}, \mathbf{u}_{kj}, t_{kj}\right], \tag{3.3}$$

$$\mathbf{u}_{kj} = \mathbf{u}\left(t_{kj}\right), \tag{3.4}$$

$$t_{kj} = t_k + h_k \rho_j \tag{3.5}$$

$$h_k = t_{k+1} - t_k. \tag{3.6}$$

S is referred to as the "stage," and the intermediate values $\mathbf{y}_{kj}$ called *internal stages* and can be considered approximations to the solution at t_{kj}. In these expressions, $\{\rho_j, \beta_j, \alpha_{j\ell}\}$ are known constants with $0 \leq \rho_1 \leq \rho_2 \leq \cdots \leq \rho_S \leq 1$. A convenient way to define the coefficients is to use the Butcher array

$$\begin{array}{c|ccc} \rho_1 & \alpha_{11} & \dots & \alpha_{1S} \\ \vdots & \vdots & & \vdots \\ \rho_S & \alpha_{S1} & \dots & \alpha_{SS} \\ \hline & \beta_1 & \dots & \beta_S \end{array}.$$

The schemes are called *explicit* if $\alpha_{j\ell} = 0$ for $l \geq j$ and *implicit* otherwise. The focus here is on a particular family of implicit Runge-Kutta (IRK) schemes called Lobatto IIIA methods. The Lobatto IIIA family has the following properties:

- The methods are symmetric with $\rho_1 = 0$ and $\rho_S = 1$.
- The coefficients $\alpha_{1j} = 0$ and $\alpha_{Sj} = \beta_j$ for $j = 1, \ldots, S$.
- As such the internal variables $\mathbf{y}_{j1}$ and $\mathbf{y}_{jS}$ for $j = 1, \ldots, S$ as well as the implicit constraints (3.2) can be analytically eliminated.
- The methods are collocation methods as described below. The variable and constraint definitions introduced are consistent with the collocation conditions given as equations 5.71a and 5.71b, in [1, p 218].
- The method with S stages has (nonstiff) order $\eta = 2S - 2$.

The numerical values of the Lobatto IIIA coefficients for $S = 2, 3, 4, 5$ are given in appendix A.

Variable Phase Length

For many optimal control problems it is convenient to break the problem into *phases* either for numerical purposes or to describe different physical processes. In general the length of a phase is defined by t_I and t_F either of which can be an optimization variable. Therefore let us define a second time transformation

$$t = t_I + \tau(t_F - t_I) = t_I + \tau\sigma \tag{3.7}$$

where the phase length $\sigma \doteq t_F - t_I$ and $0 \leq \tau \leq 1$. Thus for $\Delta\tau_k = (\tau_{k+1} - \tau_k)$ we have

$$h_k = (\tau_{k+1} - \tau_k)(t_F - t_I) = \Delta\tau_k\sigma \tag{3.8}$$

In light of the transformation (3.7)

$$\mathbf{y}' = \frac{d\mathbf{y}}{d\tau} = \frac{d\mathbf{y}}{dt}\frac{dt}{d\tau} = \sigma\dot{\mathbf{y}} \tag{3.9}$$

and so the original ODE (1.2) becomes

$$\mathbf{y}' = \sigma\mathbf{f}[\mathbf{y}(\tau), \mathbf{u}(\tau), \tau] \tag{3.10}$$

Collocation Methods

The Runge–Kutta scheme (3.1)–(3.5) is often motivated in another way. Suppose we consider approximating the solution of the ODE (1.2) by a function $\mathbf{z}(t)$. In what follows it will be convenient denote component-wise operations using $z(t)$. As an approximation, let us use a polynomial of degree S (order $S+1$) over each step $t_k \leq t \leq t_{k+1}$:

$$z(t) = a_0 + a_1(t - t_k) + \cdots + a_S(t - t_k)^S. \tag{3.11}$$

The coefficients $(a_0, a_1, \ldots, a_S)$ are chosen such that the approximation matches at the beginning of the step t_k, that is,

$$z(t_k) = y_k, \tag{3.12}$$

and has derivatives that match at the internal stage points (3.5)

$$\frac{dz(t_{kj})}{dt} = f\left[\mathbf{y}_{kj}, \mathbf{u}_{kj}, t_{kj}\right] = f_{kj}. \tag{3.13}$$

Observe that within a particular step $t_k \leq t \leq t_{k+1}$ the parameter $0 \leq \rho \leq 1$ defines the *local* time parameterization $t = t_k + h_k\rho$ and so from (3.11) it follows that

$$z(t) = a_0 + a_1h_k\rho_j + \cdots + a_Sh_k^S\rho_j^S \tag{3.14}$$

Similarly from (3.11)

$$\frac{dz(t)}{dt} = a_1 + \cdots + a_{S-1}(S-1)(t - t_k)^{S-2} + a_SS(t - t_k)^{S-1} \tag{3.15}$$

and so substituting (3.5), (3.13) gives

$$f_{kj} = a_1 + \cdots + a_{S-1}(S-1)h_k^{S-2}\rho_j^{S-2} + a_SSh_k^{S-1}\rho_j^{S-1} \tag{3.16}$$

Moreover, it is demonstrated in Ref [1, p. 219] that when the derivatives match (cf (3.13)) the function values $y_{kj} = z_{kj}$ also match for $1 \leq j \leq S$. Thus it follows from (3.14) and (3.5) that

$$y_{kj} = a_0 + a_1h_k\rho_j + \cdots + a_Sh_k^S\rho_j^S \tag{3.17}$$

The conditions (3.13) are called *collocation* conditions and the resulting method is referred to as a *collocation method*. The Runge–Kutta scheme (3.1)–(3.5) is a collocation method [1], and the solution produced by the method is a piecewise polynomial.

The focus of a collocation method is on a polynomial representation for the differential *state* variables. When the state is a polynomial of degree S over each step $t_k \le t \le t_{k+1}$ it is natural to use a polynomial approximation of degree $S-1$ for the algebraic variables $\mathbf{u}(t)$ similar to (3.11)

$$v(t) = b_0 + b_1(t-t_k) + \cdots + b_{S-1}(t-t_k)^{S-1} \tag{3.18}$$

for $j = 0, \ldots, S-1$ and the coefficients $(b_0, b_1, \ldots, b_{S-1})$ are determined such that the approximation matches at the intermediate points (3.5) for $j = 1, \ldots, S$

$$v(t_{kj}) = u_{kj}. \tag{3.19}$$

3.1 Lobatto IIIA, $S = 2$

The simplest Lobatto IIIA method has two stages and is of order $\eta = 2$. It is commonly referred to as the *trapezoidal* method (abbreviated TRP or LA2). The nonlinear programming constraints, called defects, and the corresponding NLP variables are as follows:

Defect Constraints

$$\mathbf{0} \equiv \boldsymbol{\zeta}_k = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{\Delta\tau_k}{2}\left[\sigma\mathbf{f}_k + \sigma\mathbf{f}_{k+1}\right] \tag{3.20a}$$

Variables

$$\mathbf{x}^\mathsf{T} = (\ldots, \mathbf{y}_k, \mathbf{u}_k, \mathbf{y}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{p}, t_I, t_F, \ldots) \tag{3.20b}$$

3.2 Lobatto IIIA, $S = 3$

There are three common forms when there are three stages all having order $\eta = 4$. We abbreviate the primary form LA3.

Primary Form

Defect Constraints

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \Delta\tau_k\left[\beta_1\sigma\mathbf{f}_k + \beta_2\sigma\mathbf{f}_{k2} + \beta_3\sigma\mathbf{f}_{k+1}\right] \tag{3.21a}$$
$$\mathbf{0} = \mathbf{y}_{k2} - \mathbf{y}_k - \Delta\tau_k\left[\alpha_{21}\sigma\mathbf{f}_k + \alpha_{22}\sigma\mathbf{f}_{k2} + \alpha_{23}\sigma\mathbf{f}_{k+1}\right] \tag{3.21b}$$

where

$$\mathbf{f}_{k2} = \mathbf{f}\left[\mathbf{y}_{k2}, \mathbf{u}_{k2}, t_{k2}\right] \tag{3.21c}$$
$$t_{k2} = t_k + h_k\rho_2 = t_k + \frac{1}{2}h_k \tag{3.21d}$$
$$\mathbf{u}_{k2} = \mathbf{u}(t_{k2}) \tag{3.21e}$$

Variables

$$\mathbf{x}^\mathsf{T} = (\ldots, \mathbf{y}_k, \mathbf{u}_k, \mathbf{y}_{k2}, \mathbf{u}_{k2}, \mathbf{y}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{p}, t_I, t_F, \ldots) \tag{3.21f}$$

Hermite-Simpson (Separated)
By solving (3.21a) for the quantity $\mathbf{f}_{k2}$ and then substituting the result into (3.21b) one obtains the second form. The method is referred to as *Hermite-Simpson (Separated)* or simply *Separated Simpson* and abbreviated HSS [3, Sec. 4.6.6].

Defect Constraints

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \Delta\tau_k\left[\beta_1\sigma\mathbf{f}_k + \beta_2\sigma\mathbf{f}_{k2} + \beta_3\sigma\mathbf{f}_{k+1}\right] \tag{3.22a}$$
$$\mathbf{0} = \mathbf{y}_{k2} - \frac{1}{2}(\mathbf{y}_k + \mathbf{y}_{k+1}) - \frac{\Delta\tau_k}{8}(\sigma\mathbf{f}_k - \sigma\mathbf{f}_{k+1}) \tag{3.22b}$$

where the internal stage values and NLP variables are given by (3.21c)-(3.21f).

Hermite-Simpson (Compressed)
The third form is obtained by solving (3.22b) for the internal state $\mathbf{y}_{k2}$ and simply using this to evaluate $\mathbf{f}_{k2}$. This eliminates the explicit internal stage constraints (3.21b) and also the internal stage variables $\mathbf{y}_{k2}$. Referred to as *Hermite-Simpson (Compressed)* or simply *Compressed Simpson* it is abbreviated HSC [3, Sec. 4.6.5]. This form benefits from a smaller number of NLP variables and constraints, however at the expense of matrix sparsity.

Defect Constraints

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \Delta\tau_k \left[\beta_1 \sigma \mathbf{f}_k + \beta_2 \sigma \mathbf{f}_{k2} + \beta_3 \sigma \mathbf{f}_{k+1}\right] \tag{3.23a}$$

where

$$\mathbf{y}_{k2} = \frac{1}{2}(\mathbf{y}_k + \mathbf{y}_{k+1}) + \frac{h_k}{8}(\mathbf{f}_k - \mathbf{f}_{k+1}) \tag{3.23b}$$
$$\mathbf{f}_{k2} = \mathbf{f}\left[\mathbf{y}_{k2}, \mathbf{u}_{k2}, t_{k2}\right] \tag{3.23c}$$
$$t_{k2} = t_k + h_k \rho_2 = t_k + \frac{1}{2} h_k \tag{3.23d}$$
$$\mathbf{u}_{k2} = \mathbf{u}(t_{k2}) \tag{3.23e}$$

Variables

$$\mathbf{x}^\mathsf{T} = (\ldots, \mathbf{y}_k, \mathbf{u}_k, \mathbf{u}_{k2}, \mathbf{y}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{p}, t_I, t_F, \ldots) \tag{3.23f}$$

3.3 Lobatto IIIA, $S = 4$

This sixth order scheme is abbreviated LA4.

Defect Constraints

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \Delta\tau_k \left[\beta_1 \sigma \mathbf{f}_k + \beta_2 \sigma \mathbf{f}_{k2} + \beta_3 \sigma \mathbf{f}_{k3} + \beta_4 \sigma \mathbf{f}_{k+1}\right] \tag{3.24a}$$
$$\mathbf{0} = \mathbf{y}_{k2} - \mathbf{y}_k - \Delta\tau_k \left[\alpha_{21} \sigma \mathbf{f}_k + \alpha_{22} \sigma \mathbf{f}_{k2} + \alpha_{23} \sigma \mathbf{f}_{k3} + \alpha_{24} \sigma \mathbf{f}_{k+1}\right] \tag{3.24b}$$
$$\mathbf{0} = \mathbf{y}_{k3} - \mathbf{y}_k - \Delta\tau_k \left[\alpha_{31} \sigma \mathbf{f}_k + \alpha_{32} \sigma \mathbf{f}_{k2} + \alpha_{33} \sigma \mathbf{f}_{k3} + \alpha_{34} \sigma \mathbf{f}_{k+1}\right] \tag{3.24c}$$

where

$$\mathbf{f}_{k2} = \mathbf{f}\left[\mathbf{y}_{k2}, \mathbf{u}_{k2}, t_{k2}\right] \tag{3.24d}$$
$$t_{k2} = t_k + h_k \rho_2 \tag{3.24e}$$
$$\mathbf{u}_{k2} = \mathbf{u}(t_{k2}) \tag{3.24f}$$
$$\mathbf{f}_{k3} = \mathbf{f}\left[\mathbf{y}_{k3}, \mathbf{u}_{k3}, t_{k3}\right] \tag{3.24g}$$
$$t_{k3} = t_k + h_k \rho_3 \tag{3.24h}$$
$$\mathbf{u}_{k3} = \mathbf{u}(t_{k3}) \tag{3.24i}$$

Variables

$$\mathbf{x}^\mathsf{T} = (\ldots, \mathbf{y}_k, \mathbf{u}_k, \mathbf{y}_{k2}, \mathbf{u}_{k2}, \mathbf{y}_{k3}, \mathbf{u}_{k3}, \mathbf{y}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{p}, t_I, t_F, \ldots) \tag{3.24j}$$

3.4 Lobatto IIIA, $S = 5$

This eighth order scheme is abbreviated LA5.

Defect Constraints

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \Delta\tau_k \left[\beta_1 \sigma \mathbf{f}_k + \beta_2 \sigma \mathbf{f}_{k2} + \beta_3 \sigma \mathbf{f}_{k3} + \beta_4 \sigma \mathbf{f}_{k4} + \beta_5 \sigma \mathbf{f}_{k+1}\right] \tag{3.25a}$$
$$\mathbf{0} = \mathbf{y}_{k2} - \mathbf{y}_k - \Delta\tau_k \left[\alpha_{21} \sigma \mathbf{f}_k + \alpha_{22} \sigma \mathbf{f}_{k2} + \alpha_{23} \sigma \mathbf{f}_{k3} + \alpha_{24} \sigma \mathbf{f}_{k4} + \alpha_{25} \sigma \mathbf{f}_{k+1}\right] \tag{3.25b}$$
$$\mathbf{0} = \mathbf{y}_{k3} - \mathbf{y}_k - \Delta\tau_k \left[\alpha_{31} \sigma \mathbf{f}_k + \alpha_{32} \sigma \mathbf{f}_{k2} + \alpha_{33} \sigma \mathbf{f}_{k3} + \alpha_{34} \sigma \mathbf{f}_{k4} + \alpha_{35} \sigma \mathbf{f}_{k+1}\right] \tag{3.25c}$$
$$\mathbf{0} = \mathbf{y}_{k4} - \mathbf{y}_k - \Delta\tau_k \left[\alpha_{41} \sigma \mathbf{f}_k + \alpha_{42} \sigma \mathbf{f}_{k2} + \alpha_{43} \sigma \mathbf{f}_{k3} + \alpha_{44} \sigma \mathbf{f}_{k4} + \alpha_{45} \sigma \mathbf{f}_{k+1}\right] \tag{3.25d}$$

where

$$\mathbf{f}_{k2} = \mathbf{f}\left[\mathbf{y}_{k2}, \mathbf{u}_{k2}, t_{k2}\right] \tag{3.25e}$$
$$t_{k2} = t_k + h_k\rho_2 \tag{3.25f}$$
$$\mathbf{u}_{k2} = \mathbf{u}(t_{k2}) \tag{3.25g}$$
$$\mathbf{f}_{k3} = \mathbf{f}\left[\mathbf{y}_{k3}, \mathbf{u}_{k3}, t_{k3}\right] \tag{3.25h}$$
$$t_{k3} = t_k + h_k\rho_3 \tag{3.25i}$$
$$\mathbf{u}_{k3} = \mathbf{u}(t_{k3}) \tag{3.25j}$$
$$\mathbf{f}_{k4} = \mathbf{f}\left[\mathbf{y}_{k4}, \mathbf{u}_{k4}, t_{k4}\right] \tag{3.25k}$$
$$t_{k4} = t_k + h_k\rho_4 \tag{3.25l}$$
$$\mathbf{u}_{k4} = \mathbf{u}(t_{k4}) \tag{3.25m}$$

Variables

$$\mathbf{x}^{\mathsf{T}} = (\ldots, \mathbf{y}_k, \mathbf{u}_k, \mathbf{y}_{k2}, \mathbf{u}_{k2}, \mathbf{y}_{k3}, \mathbf{u}_{k3}, \mathbf{y}_{k4}, \mathbf{u}_{k4}, \mathbf{y}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{p}, t_I, t_F, \ldots) \tag{3.25n}$$

Quadrature Equations

The IRK methods have been introduced as a way to solve ODE's. When treating problems involving integral expressions such as

$$\mathcal{I} = \int_{t_I}^{t_F} \mathbf{w}[\mathbf{y}(t), \mathbf{u}(t), t]dt \tag{3.26}$$

it is common to introduce new dynamic variables $\mathbf{r}(t)$ and then solve the following augmented system:

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), t] \tag{3.27}$$
$$\dot{\mathbf{r}} = \mathbf{w}[\mathbf{y}(t), \mathbf{u}(t), t] \tag{3.28}$$

in conjunction with the initial conditions

$$\mathbf{r}(t_I) = 0 \tag{3.29}$$

and it then follows that

$$\mathbf{r}(t_F) = \mathcal{I}. \tag{3.30}$$

If we apply a recursive scheme to the augmented system we can write

$$\mathbf{r}(t_F) = \mathbf{r}_M = \sum_{k=1}^{M-1} \mathbf{r}_{k+1} - \mathbf{r}_k \tag{3.31}$$

for the subset of dynamic variables in (3.28). It then follows from (3.20a),(3.21a),(3.24a), and (3.25a) that

$$\mathbf{r}_{k+1} - \mathbf{r}_k = \begin{cases} \Delta\tau_k \left[\beta_1\sigma\mathbf{w}_k + \beta_2\sigma\mathbf{w}_{k+1}\right] & S = 2 \\ \Delta\tau_k \left[\beta_1\sigma\mathbf{w}_k + \beta_2\sigma\mathbf{w}_{k2} + \beta_3\sigma\mathbf{w}_{k+1}\right] & S = 3 \\ \Delta\tau_k \left[\beta_1\sigma\mathbf{w}_k + \beta_2\sigma\mathbf{w}_{k2} + \beta_3\sigma\mathbf{w}_{k3} + \beta_4\sigma\mathbf{w}_{k+1}\right] & S = 4 \\ \Delta\tau_k \left[\beta_1\sigma\mathbf{w}_k + \beta_2\sigma\mathbf{w}_{k2} + \beta_3\sigma\mathbf{w}_{k3} + \beta_4\sigma\mathbf{w}_{k4} + \beta_5\sigma\mathbf{w}_{k+1}\right] & S = 5 \end{cases} \tag{3.32}$$

Now it is important to note that the dynamic variables $\mathbf{r}(t)$ *do not appear* in the functions $\mathbf{w}$ and so there is no need to introduce the values at the grid points $\mathbf{r}_k$, and the internal stage points $\mathbf{r}_{k2}, \mathbf{r}_{k3}, \ldots$ when evaluating the integrands, i.e $\mathbf{w}_k, \mathbf{w}_{k2}, \mathbf{w}_{k3}, , \mathbf{w}_{k4}$ etc.

4 Nonlinear Programming

The general nonlinear programming (NLP) problem can be stated as follows: Find the n-vector $\mathbf{x}^\mathsf{T} = (x_1, \ldots, x_n)$ to minimize the scalar objective function

$$F(\mathbf{x}) \tag{4.1}$$

subject to the m constraints

$$\mathbf{c}_L \le \mathbf{c}(\mathbf{x}) \le \mathbf{c}_U \tag{4.2}$$

and the simple bounds

$$\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U. \tag{4.3}$$

Equality constraints can be imposed by setting $\mathbf{c}_L = \mathbf{c}_U$.

In the preceding section we outlined the quantities that must be treated as constraints and variables when an ODE is approximated by discretization. So for example, when using the trapezoidal method with M grid points, we must impose the defect constraints (3.20a) $k = 1, \ldots, M-1$. In the trapezoidal case the goal is to choose the NLP variables (3.20b) to minimize the objective function and satisfy the defect constraints (3.20a) as well as any boundary conditions. To solve the nonlinear programming problem it is also necessary to compute the derivatives of the objective and constraint functions. When a finite difference method is used to construct the Jacobian, it is natural to identify the constraint functions as the quantities being differentiated. In other words, if we define

$$\mathbf{q} = \begin{bmatrix} \mathbf{c} \\ F \end{bmatrix} \tag{4.4}$$

then we can use finite differences to compute

$$\mathbf{D} = \frac{\partial \mathbf{q}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{G} \\ \mathbf{g}^\mathsf{T} \end{bmatrix} \tag{4.5}$$

where $\mathbf{G}$ is the constraint Jacobian and $\mathbf{g}$ is the objective gradient. The Hessian of the Lagrangian $\mathbf{H}_L$ can also be constructed using differencing techniques as described in [3, Sec. 2.2].

However to exploit separability we write

$$\begin{bmatrix} \mathbf{c}(\mathbf{x}) \\ F(\mathbf{x}) \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x}). \tag{4.6}$$

By isolating the linear terms $\mathbf{A}\mathbf{x}$ from the nonlinear terms $\mathbf{B}\mathbf{q}(\mathbf{x})$, it is then easy to demonstrate that for all of the Lobatto methods the elements of the vector $\mathbf{q}(\mathbf{x})$ are of the form

$$\mathbf{q}(\mathbf{x}) = \begin{bmatrix} \vdots \\ \sigma\mathbf{f}_{k1} \\ \sigma\mathbf{w}_{k1} \\ \sigma\mathbf{f}_{k2} \\ \sigma\mathbf{w}_{k2} \\ \vdots \\ \sigma\mathbf{f}_{kS} \\ \sigma\mathbf{w}_{kS} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \sigma\mathbf{f}_{k} \\ \sigma\mathbf{w}_{k} \\ \sigma\mathbf{f}_{k2} \\ \sigma\mathbf{w}_{k2} \\ \vdots \\ \sigma\mathbf{f}_{k+1} \\ \sigma\mathbf{w}_{k+1} \\ \vdots \end{bmatrix} \tag{4.7}$$

The constant matrices $\mathbf{A}$ and $\mathbf{B}$ are defined by the method coefficients $\alpha_{j\ell}$, and β_j. It then follows that

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{g}^\mathsf{T} \end{bmatrix} = \mathbf{A} + \mathbf{B}\mathbf{D} \tag{4.8}$$

where the finite difference matrix

$$\mathbf{D} = \frac{\partial \mathbf{q}}{\partial \mathbf{x}} \tag{4.9}$$

involves the right hand side quantities at the grid points and internal stage points. In particular these quantities are often sparse with a structure defined by the *sparsity template*

$$\mathcal{T} = \text{struct}\left[\begin{array}{c|c|c} \frac{\partial \mathbf{f}}{\partial \mathbf{y}} & \frac{\partial \mathbf{f}}{\partial \mathbf{u}} & \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{y}} & \frac{\partial \mathbf{g}}{\partial \mathbf{u}} & \frac{\partial \mathbf{g}}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{w}}{\partial \mathbf{y}} & \frac{\partial \mathbf{w}}{\partial \mathbf{u}} & \frac{\partial \mathbf{w}}{\partial \mathbf{p}} \end{array}\right] \tag{4.10}$$

and consequently $\mathbf{D}$ can be computed very efficiently using sparse finite difference techniques. In particular, when using sparse differencing the number of perturbations is dictated by the number of *index sets* γ, and for optimal control problems $\gamma \ll n$. For the low thrust problem below $\gamma = 6$ and this value does not change as the mesh size increases, even though the number of NLP variables n can become very large. In short we use the same number of perturbations whether the grid is coarse or fine!

5 Mesh Refinement

There are two primary goals for the final step in a direct transcription method, namely to

- decide whether the discrete solution is "accurate" enough, and
- if not do something about it.

To define "accuracy" in this context consider a single interval $t_k \le t \le t_k + h_k$. Suppose the NLP problem has produced a solution to the ODEs (1.2) so that,

$$\mathbf{y}(t_k + h_k) = \mathbf{y}(t_k) + \int_{t_k}^{t_k+h_k} \dot{\mathbf{y}} dt = \mathbf{y}(t_k) + \int_{t_k}^{t_k+h_k} \mathbf{f}\left[\mathbf{y}, \mathbf{u}, t\right] dt. \tag{5.1}$$

Unfortunately the expression for $\mathbf{y}(t_k + h_k)$ involves the true value for both $\mathbf{y}$ and $\mathbf{u}$ which are unknown. Consequently, we may consider the approximation

$$\widehat{\mathbf{y}}(t_k + h_k) \equiv \mathbf{y}(t_k) + \int_{t_k}^{t_k+h_k} \mathbf{f}\left[\mathbf{z}(t), \mathbf{v}(t), t\right] dt, \tag{5.2}$$

or the alternative expression

$$\widetilde{\mathbf{y}}(t_k + h_k) \equiv \mathbf{y}(t_k) + \int_{t_k}^{t_k+h_k} \mathbf{f}\left[\mathbf{y}(t), \mathbf{v}(t), t\right] dt, \tag{5.3}$$

Observe that the collocation solution $\mathbf{z}(t)$ and $\mathbf{v}(t)$ appears in the integrand of (5.2) whereas (5.3) involves the real solution state $\mathbf{y}(t)$ and the collocation control approximation $\mathbf{v}(t)$. In either case the "error" over the step is measured by the difference $\mathbf{y}(t_k + h_k) - \widehat{\mathbf{y}}(t_k + h_k)$ or $\mathbf{y}(t_k + h_k) - \widetilde{\mathbf{y}}(t_k + h_k)$. Motivated by this we define the *absolute local error* on a particular step by

$$\boldsymbol{\eta}_{i,k} = \left| \int_{t_k}^{t_{k+1}} \dot{\mathbf{z}}_i(s) - \mathbf{f}_i\left[\mathbf{z}(s), \mathbf{v}(s), s\right] ds \right| \tag{5.4}$$

Notice that the arguments of the integrand use the collocation approximations (3.11) and (3.18) for the state and control evaluated at intermediate points in the interval. From this expression for the absolute error, we can define the *relative local error* by

$$\epsilon_k \approx \max_i \frac{\boldsymbol{\eta}_{i,k}}{(w_i + 1)}, \tag{5.5}$$

where the scale weight $w_i = \max_{k=1}^{M} \left[|\tilde{y}_{i,k}|, |\dot{\tilde{y}}_{i,k}|\right]$ defines the maximum value for the ith state variable or its derivative over the M grid points in the phase. In the $\mathbb{SOS}$ software implementation ϵ_k is computed in every interval $t_k \le t \le t_k + h_k$ by evaluating $\boldsymbol{\eta}_{i,k}$ using a high precision Romberg quadrature method with Richardson extrapolation.

There are many complicated and often conflicting factors that determine the overall computational efficiency of the direct transcription algorithm. The cost of solving a sparse linear system $\mathbf{Ax} = \mathbf{b}$ is $\mathcal{O}(\kappa n)$, and typically the solution of a nonlinear program will require many iterations each requiring the solution

of a sparse system. This suggests that the number of NLP variables n should be kept small. However, the cost also depends on the sparsity κ and consequently a small dense formulation may actually be more costly than a large sparse formulation. Furthermore, the number of NLP iterations is a function of the nonlinearity of the differential equations, as well as the quality of the initial guess. It is often more expensive to solve a small system of very nonlinear ODE's, when compared to a large system of linear ODE's. Similarly a "good" initial guess for the NLP variables may converge in one or two iterations regardless of the system nonlinearity. The situation is also unclear when comparing different discretization methods. For example, is it better to use two steps of a Lobatto two stage method, or a single step of a Lobatto three stage method? The number of NLP variables n is the same for both approaches. Since the two stage method is of order two and the three stage method is of order four, a naive analysis would suggest that the fourth order method is preferable. However, this conclusion ignores the impact of high order derivatives in a nonlinear differential equation. In fact, it is often better to utilize a low order scheme in regions with rapidly changing dynamics. Furthermore, when a "bad" initial guess is used to begin the NLP a low order method can be more robust, that is, converging to a solution in fewer iterations. The $\mathbb{SOS}$ software incorporates a number of options that permit the user to tailor the algorithm to the physical application. The basic mesh refinement procedure can be summarized as follows:

Mesh Refinement Algorithm

Estimate Discretization Error
Compute error ϵ_k for all intervals
Terminate if $\max_k \epsilon_k \leq \delta$
Change Discretization Method or Stepsize
Change the discretization method (if possible), otherwise
Reduce the stepsize(s) h_k by subdividing one or more intervals

The goal of the mesh refinement algorithm is to reduce the local error (5.5) below a user specified relative tolerance δ in all intervals. There are two mechanisms for achieving this goal, namely changing the discretization method, or reducing the stepsize h_k. In $\mathbb{SOS}$ the user can specify a sequence of methods for each phase in the problem description. For example suppose it is desirable to use a trapezoidal method for the first two refinement iterations, followed by the Compressed Simpson Method for the remaining iterations. This sequence is abbreviated as follows:

(TRP),2;(HSC),20.

As a second option if the sequence is

(LA2),-2;(LA3),-4;(LA5),-20

the Lobatto IIIA (two-stage) discretization will be used until $\epsilon_k > 10^{-2}$, followed by the Lobatto IIIA (three-stage) method if $\epsilon_k > 10^{-4}$, followed by the Lobatto IIIA (five-stage) method if $\epsilon_k > 10^{-20}$. It is worth recalling that the mesh refinement procedure occurs only after solving an NLP problem, and is terminated when the solution is sufficiently accurate. Thus for the second example, suppose the first refinement iteration using the LA2 discretization terminates with a discretization error of $\epsilon_k = .5 \times 10^{-4}$. The second and all subsequent refinement iterations will utilize an LA5 method until converging, provided of course the requested accuracy $\delta > 10^{-20}$.

If the discretization method is not changed on a particular refinement iteration, the error can be reduced by subdividing one or more intervals in the current mesh. In $\mathbb{SOS}$ there are three approaches. The default scheme described in [3, Sec. 4.7.4], solves an integer programming problem in order to minimize the maximum error over all intervals. A second approach attempts to distribute the error equally in all intervals using inverse interpolation of the existing error distribution. When solving delay-differential equations it is occasionally useful to use a simple bisection scheme which is also available.

6 Optimal Low Thrust Transfers Between Libration Point Orbits

6.1 Introduction

To illustrate the techniques discussed above let us consider an example which is noteworthy in two respects. First we address a problem of ongoing practical interest. Low thrust propulsion systems have been studied for many practical missions [4], and in addition there are a number of missions that are either currently operating in libration point orbits or proposed for the future. Secondly, this example represents the real manifestation of a class of *hypersensitive* optimal control problems. A series of investigations by Rao and Mease [10]

demonstrate the solution is characterized by an initial and terminal boundary layer region, separated by a long duration equilibrium segment. The techniques they use are related to singular perturbation methods and example 4.4 [3, pp. 170–171] illustrates the solution of a "toy problem" with this structure. The use of a high order Lobatto discretization was investigated by Herman and Conway [8], however, their approach did not exploit sparsity. Finally, because of periodic behavior and dynamic sensitivity there are many local solutions. In short this example is both numerically challenging and of great practical interest.

6.2 Dynamic Model

A formulation of an optimal low thrust transfer between libration point orbits is presented by Epenoy [7]. The dynamic model is based on the Planar Circular Restricted Three Body Problem (PCR3BP) with Earth as one primary and the Moon as the second. The equations of motion are constructed in a rotating reference frame, in which the x-axis extends from the barycenter of the Earth-Moon system to the Moon, and the y-axis completes the right hand coordinate frame. A set of non-dimensional units is chosen such that the unit of distance is the distance between the two primaries, the unit of mass is the sum of the primaries' masses, and the unit of time is such that the angular velocity of the primaries around their barycenter is one. Thus the Moon has mass μ and is fixed at the coordinates $(1-\mu, 0)$ while the Earth has mass $(1-\mu)$ and is fixed at the coordinates $(-\mu, 0)$. The mass parameter is defined as

$$\mu = \frac{M_m}{M_e + M_m} = 0.0121506683 \tag{6.1}$$

where M_e and M_m are the masses of the Earth and Moon respectively.

The equations of motion in the rotating frame are

$$\dot{x} = v_x \tag{6.2}$$

$$\dot{y} = v_y \tag{6.3}$$

$$\dot{v}_x = x + 2v_y - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x+\mu-1)}{r_2^3} + u_1 \tag{6.4}$$

$$\dot{v}_y = y - 2v_x - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} + u_2 \tag{6.5}$$

where dot denotes the non-dimensional time derivative relative to an observer in the rotating frame. The position in the rotating frame is denoted by (x, y) with corresponding relative velocity (v_x, v_y). The distances from the Earth and Moon respectively are given by

$$r_1 = \sqrt{(x+\mu)^2 + y^2} \tag{6.6}$$

$$r_2 = \sqrt{(x+\mu-1)^2 + y^2} \tag{6.7}$$

The dynamics are defined by the state vector $\mathbf{z}^\mathsf{T} = (x, y, v_x, v_y)$ in the domain $t_0 \le t \le t_f$ where both the initial and final times are fixed. The control variables $\mathbf{u}^\mathsf{T} = (u_1, u_2)$ denote the spacecraft acceleration in the rotating frame. Thus the dynamics (6.2)-(6.5) are given by

$$\dot{\mathbf{z}} = \mathbf{f}[\mathbf{z}, \mathbf{u}]. \tag{6.8}$$

The initial and terminal states are fixed by the following boundary conditions:

$$\mathbf{z}(t_0) = \boldsymbol{\xi}_1(\tau_0) \tag{6.9}$$

$$\mathbf{z}(t_f) = \boldsymbol{\xi}_2(\tau_f) \tag{6.10}$$

where the states on the Lyapunov orbits are denoted by $\boldsymbol{\xi}_1(\tau_0)$ and $\boldsymbol{\xi}_2(\tau_f)$ respectively. The Lyapunov states are computed by means of Lindstedt-Poincare approximation as functions of the parameters τ_0 and τ_f. These non-dimensional times determine the departure and arrival location. The goal is to minimize the energy consumed during the transfer, i.e.

$$F = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^\mathsf{T}\mathbf{u}\, dt. \tag{6.11}$$

For the PCR3BP it is also convenient to introduce a time scaling. In particular we fix $t_0 = 0$ and define

$$t_F = 2\pi \frac{T_f}{P_M} \tag{6.12}$$

where T_f is the duration of the transfer and $P_M = 27.321577$ days is the orbital period of the moon. Numerical results will be constructed for two cases, namely a *short transfer* where $T_f = 12$ days, and a *long transfer* where $T_f = 44$ days. After time scaling $t_F = 2.759659$ for the short transfer and $t_F = 10.11874803$ for the long transfer.

6.3 Lyapunov Orbits

The libration points are defined in the rotating frame at $L_1 = (x_1, 0)$ and $L_2 = (x_2, 0)$ where x_1 and x_2 are roots of the nonlinear equations

$$0 = x_1 - \frac{1-\mu}{\mu + x_1} + \frac{\mu}{x_1 - 1 + \mu} \tag{6.13}$$

and

$$0 = x_2 - \frac{1-\mu}{\mu + x_2} - \frac{\mu}{x_2 - 1 + \mu} \tag{6.14}$$

respectively. For the particular case of interest with μ given by (6.1) we have $x_1 = 0.83691471889320190$ and $x_2 = 1.1556824834786137$.

The specific Lyapunov orbits given in Epenoy [7] correspond to orbits around the Earth-Moon libration points L_1 and L_2 with the same value for the Jacobi constant, namely 3.178. FORTRAN code implementing the calculation of the functions $\boldsymbol{\xi}_1(\tau_0)$ and $\boldsymbol{\xi}_2(\tau_f)$ was graciously supplied by R. Epenoy. Tables 6.1 and 6.2 present a summary of the extreme points in these orbits. Note also that the functions $\boldsymbol{\xi}_1(\tau_0)$ and $\boldsymbol{\xi}_2(\tau_f)$ are periodic functions of the parameters τ_0 and τ_f respectively. Thus $\boldsymbol{\xi}_1(\tau_0) = \boldsymbol{\xi}_1(\tau_0 + kT_1)$ where $T_1 = 2.776024944790721$ for $k = 0, \pm 1, \pm 2, \ldots$. Similarly for the orbit around L_2 we have $\boldsymbol{\xi}_2(\tau_f) = \boldsymbol{\xi}_2(\tau_f + kT_2)$ where $T_2 = 3.385292341000000$.

Table 6.1: Lyapunov Orbit Around L_1

Location	τ_0	x	y
$\max x$	0.000000000000000	0.8604642913942989	0.000000000000000
$\min y$	0.6799402049577115	0.8460673759976699	-0.07126150424351556
$\min x$	1.388012472385421	0.8203198878278488	0.000000000000000
$\max y$	2.096084695912298	0.8460673739125611	0.07126150424351496
$\max x$	2.776024944790721	0.8604642913942989	0.000000000000000

Table 6.2: Lyapunov Orbit Around L_2

Location	τ_0	x	y
$\max x$	0.000000000000000	1.170863515501900	0.000000000000000
$\min y$	0.8632085886843588	1.150862903956706	-0.04864318483502234
$\min x$	1.692646170500000	1.137392572452755	0.000000000000000
$\max y$	2.522083753145239	1.150862903981730	0.04864318483502232
$\max x$	3.385292341000000	1.170863515501900	0.000000000000000

6.4 Adjoint Equations

For the sake of reference in what follows it is useful to define the adjoint equations. The Hamiltonian is given by

$$H = F + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{f} = \frac{1}{2}\left[u_1^2 + u_2^2\right] + \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 \tag{6.15}$$

and the adjoint equations are:

$$\dot{\lambda}_1 = -\frac{\partial H}{\partial x} = -\lambda_3 \frac{\partial f_3}{\partial x} - \lambda_4 \frac{\partial f_4}{\partial x} = -\lambda_3 \frac{\partial f_3}{\partial x} - \lambda_4 \frac{\partial f_4}{\partial x} \tag{6.16}$$

$$\dot{\lambda}_2 = -\frac{\partial H}{\partial y} = -\lambda_3 \frac{\partial f_3}{\partial y} - \lambda_4 \frac{\partial f_4}{\partial y} = -\lambda_3 \frac{\partial f_3}{\partial y} - \lambda_4 \frac{\partial f_4}{\partial y} \tag{6.17}$$

$$\dot{\lambda}_3 = -\frac{\partial H}{\partial v_x} = -\lambda_1 \frac{\partial f_1}{\partial v_x} - \lambda_4 \frac{\partial f_4}{\partial v_x} = -\lambda_1 \{1\} - \lambda_4 \{-2\} = -\lambda_1 + 2\lambda_4 \tag{6.18}$$

$$\dot{\lambda}_4 = -\frac{\partial H}{\partial v_y} = -\lambda_2 \frac{\partial f_2}{\partial v_y} - \lambda_3 \frac{\partial f_3}{\partial v_y} = -\lambda_2 \{1\} - \lambda_3 \{2\} = -\lambda_2 - 2\lambda_3 \tag{6.19}$$

To evaluate these expressions first define:

$$\frac{\partial}{\partial x}\left\{r_1^{-3}\right\} = -3r_1^{-4}\frac{\partial r_1}{\partial x} = -\frac{3(x+\mu)}{r_1^5} \tag{6.20}$$

$$\frac{\partial}{\partial y}\left\{r_1^{-3}\right\} = -3r_1^{-4}\frac{\partial r_1}{\partial y} = -\frac{3y}{r_1^5} \tag{6.21}$$

$$\frac{\partial}{\partial x}\left\{r_2^{-3}\right\} = -3r_2^{-4}\frac{\partial r_2}{\partial x} = -\frac{3(x+\mu-1)}{r_2^5} \tag{6.22}$$

$$\frac{\partial}{\partial y}\left\{r_2^{-3}\right\} = -3r_2^{-4}\frac{\partial r_2}{\partial y} = -\frac{3y}{r_2^5} \tag{6.23}$$

where

$$\frac{\partial r_1}{\partial x} = \frac{1}{2r_1}\left[2(x+\mu)\right] = \frac{(x+\mu)}{r_1} \tag{6.24}$$

$$\frac{\partial r_2}{\partial x} = \frac{1}{2r_2}\left[2(x+\mu-1)\right] = \frac{(x+\mu-1)}{r_2} \tag{6.25}$$

$$\frac{\partial r_1}{\partial y} = \frac{1}{2r_1}\left[2y\right] = \frac{y}{r_1} \tag{6.26}$$

$$\frac{\partial r_2}{\partial y} = \frac{1}{2r_2}\left[2y\right] = \frac{y}{r_2} \tag{6.27}$$

The derivatives needed to define the right hand sides of the adjoint equations are then given by

$$\frac{\partial f_3}{\partial x} = 1 - \frac{\partial}{\partial x}\left\{\frac{(1-\mu)(x+\mu)}{r_1^3}\right\} - \frac{\partial}{\partial x}\left\{\frac{\mu(x+\mu-1)}{r_2^3}\right\} = 1 - d_1 - d_2 \tag{6.28}$$

$$\frac{\partial f_4}{\partial x} = -\frac{\partial}{\partial x}\left\{\frac{(1-\mu)y}{r_1^3}\right\} - \frac{\partial}{\partial x}\left\{\frac{\mu y}{r_2^3}\right\} = -d_3 - d_4 \tag{6.29}$$

$$\frac{\partial f_3}{\partial y} = -\frac{\partial}{\partial y}\left\{\frac{(1-\mu)(x+\mu)}{r_1^3}\right\} - \frac{\partial}{\partial y}\left\{\frac{\mu(x+\mu-1)}{r_2^3}\right\} = -d_5 - d_6 \tag{6.30}$$

$$\frac{\partial f_4}{\partial y} = 1 - \frac{\partial}{\partial y}\left\{\frac{(1-\mu)y}{r_1^3}\right\} - \frac{\partial}{\partial y}\left\{\frac{\mu y}{r_2^3}\right\} = 1 - d_7 - d_8 \tag{6.31}$$

where the intermediate terms are defined as follows:

$$d_1 \doteq \frac{\partial}{\partial x}\left\{\frac{(1-\mu)(x+\mu)}{r_1^3}\right\} = (1-\mu)(x+\mu)\frac{\partial}{\partial x}\left\{r_1^{-3}\right\} + (1-\mu)r_1^{-3} \tag{6.32}$$

$$d_2 \doteq \frac{\partial}{\partial x}\left\{\frac{\mu(x+\mu-1)}{r_2^3}\right\} = \mu(x+\mu-1)\frac{\partial}{\partial x}\left\{r_2^{-3}\right\} + \mu r_2^{-3} \tag{6.33}$$

$$d_3 \doteq \frac{\partial}{\partial x}\left\{\frac{(1-\mu)y}{r_1^3}\right\} = (1-\mu)y\frac{\partial}{\partial x}\left\{r_1^{-3}\right\} \tag{6.34}$$

$$d_4 \doteq \frac{\partial}{\partial x}\left\{\frac{\mu y}{r_2^3}\right\} = \mu y\frac{\partial}{\partial x}\left\{r_2^{-3}\right\} \tag{6.35}$$

$$d_5 \doteq \frac{\partial}{\partial y}\left\{\frac{(1-\mu)(x+\mu)}{r_1^3}\right\} = (1-\mu)(x+\mu)\frac{\partial}{\partial y}\left\{r_1^{-3}\right\} \tag{6.36}$$

$$d_6 \doteq \frac{\partial}{\partial y}\left\{\frac{\mu(x+\mu-1)}{r_2^3}\right\} = \mu(x+\mu-1)\frac{\partial}{\partial y}\left\{r_2^{-3}\right\} \tag{6.37}$$

$$d_7 \doteq \frac{\partial}{\partial y}\left\{\frac{(1-\mu)y}{r_1^3}\right\} = (1-\mu)y\frac{\partial}{\partial y}\left\{r_1^{-3}\right\} + (1-\mu)r_1^{-3} \tag{6.38}$$

$$d_8 \doteq \frac{\partial}{\partial y}\left\{\frac{\mu y}{r_2^3}\right\} = \mu y\frac{\partial}{\partial y}\left\{r_2^{-3}\right\} + \mu r_2^{-3} \tag{6.39}$$

The optimal controls are defined by the optimality conditions

$$\frac{\partial H}{\partial u_1} = 0 = u_1 + \lambda_3 \tag{6.40}$$

$$\frac{\partial H}{\partial u_2} = 0 = u_2 + \lambda_4 \tag{6.41}$$

7 Numerical Results

7.1 Short Transfer

When modeling the dynamic behavior of the short transfer it is convenient to break the problem into separate regions. The short transfer was modeled using two distinct *phases* as defined in Table 7.1.

Table 7.1: Short Transfer Phase Structure

Phase	Description	Domain	Free Parameters
1	L_1 Departure	$0 \le t \le t_1$	$t_1^{(-)}$, τ_0
2	L_2 Arrival	$t_1 \le t \le t_f$	$t_1^{(+)}$, t_f, τ_f

The phase structure permits modeling two distinct portions of the overall trajectory, namely departure from the L_1 Lyapunov orbit, and arrival at the L_2 Lyapunov orbit. The initial state given by the boundary condition (6.9) fixes the beginning of phase 1. It is convenient to terminate phase 1 when the trajectory passes below the moon in a posigrade direction, and so we require

$$x(t_1) = 1 - \mu \tag{7.1}$$
$$y(t_1) \le y_{min} \tag{7.2}$$
$$v_x(t_1) \ge 0 \tag{7.3}$$

where y_{min} is a bound on the closest approach to the moon. For our results we choose $y_{min} = -.04$. To ensure continuity from phase to phase we impose the continuity constraints

$$t_1^{(-)} = t_1^{(+)} \tag{7.4}$$
$$\mathbf{z}[t_1^{(-)}] = \mathbf{z}[t_1^{(+)}] \tag{7.5}$$

where conditions at the end of phase one are denoted by $t_1^{(-)}$ and the corresponding conditions at the beginning of phase two are denoted by $t_1^{(+)}$. At the end of phase two we must also satisfy

$$t_f \le t_F \tag{7.6}$$
$$\mathbf{z}(t_f) = \boldsymbol{\xi}_2(\tau_f) \tag{7.7}$$

where $t_F = 2.759659$. Note that in (7.6) the final time t_f is treated as a free parameter limited by the fixed upper bound t_F. Treating the final time as free, yields more stable intermediate iterates and the final optimal trajectory is achieved much more readily. In particular it is expected that the inequality constraint (7.6) will be *active* at the solution. It is also worth noting that formulating the problem using two phases is done strictly for numerical purposes. In particular the constraints (7.1)-(7.3) prevent intermediate trajectories that are unreasonable, and thus improve the robustness of the algorithm.

The direct transcription method requires an initial guess for all free parameters as well as the dynamic history for the state and control. For the short transfer we guess

$$\mathbf{p}^\mathsf{T} = (t_1, t_f, \tau_0, \tau_f) = (t_F/2, t_F, 2.096084695912298, 2.522083753145239)$$

where the choice of τ_0 and τ_f correspond to the maximum values of y as given in Table 6.1 and 6.2. For quantities that change dynamically during phase one a guess that linearly interpolates between the boundary values is given by

$$[t_k, x(t_k), y(t_k), v_x(t_k), v_y(t_k), u_1(t_k), u_2(t_k)] = \alpha_k \mathbf{a}^\mathsf{T} + \beta_k \mathbf{b}^\mathsf{T} \qquad k = 1, \ldots, M \tag{7.8}$$

for M grid points on the phase where

$$\beta_k = \frac{k-1}{M-1} \qquad \alpha_k = 1 - \beta_k \tag{7.9}$$
$$\mathbf{a}^\mathsf{T} = [0, \boldsymbol{\xi}_1^\mathsf{T}(\tau_o), 0, 0] \qquad \mathbf{b}^\mathsf{T} = [t_F/2, 1-\mu, -R, v_x(0), 0, 0, 0]. \tag{7.10}$$

We also make a guess for the radial distance from the moon as $R = .1$ in (7.10). On the second phase we again use (7.8) but replace the boundary quantities (7.10) with

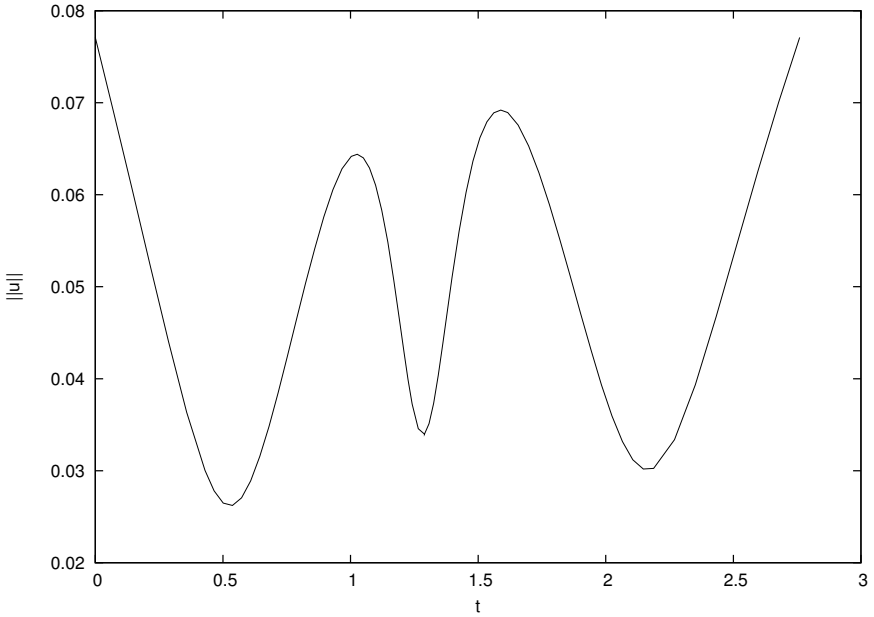
$$\mathbf{a}^\mathsf{T} = [t_F/2, 1-\mu, -R, v_x(t_f), 0, 0, 0] \qquad \mathbf{b}^\mathsf{T} = [t_F, \boldsymbol{\xi}_2^\mathsf{T}(\tau_f), 0, 0]. \tag{7.11}$$

Using this information the solution was computed using $\mathbb{SOS}$ and Table 7.2 presents a summary of the algorithm behavior when using "(LA2),-2;(LA3),-3;(LA4),-20" as a mesh refinement strategy. The first refinement iteration of the algorithm began with $M = 10$ equi-distributed grid points in each phase and the linear initial guess given by (7.8). Using an LA2 (trapezoidal) discretization produced an NLP with $m = 87$ constraints, $n = 125$ variables, and $n_d = 38$ degrees of freedom. The solution required 28 NLP iterations, as well as (NRHS = 9119) evaluations of the right hand side of the ODE. This NLP was solved in .6179 CPU seconds, and resulted in a discretization error of $\epsilon = 6.4 \times 10^{-3}$. A second mesh refinement iteration using 10 grid points in each phase with the LA3 discretization reduced the relative error to $\epsilon = 3.1 \times 10^{-3}$. The LA3 method was used again on the third refinement iteration, but with 19 grid points in each phase, distributed by the default minimax scheme. Subsequently, two additional refinement iterations were executed using the higher order LA4 (four stage Lobatto IIIA) discretization in order to reduce the error below the requested tolerance of $\epsilon = 1 \times 10^{-7}$ which corresponds to approximately eight significant figures in the solution variables. The total CPU time on a desktop computer using an Intel I7 processor (3.06 Ghz), with Linux operating system, and GNU Fortran compiler, was 3.591 seconds. A summary of the optimal solution values for this case is given in Figure 7.1, and it is worth noting that $t_1^* \neq t_F/2$ however, $t_f^* = t_F$. Figures (7.2)-(7.8) illustrate the optimal solution history.

Table 7.2: Mesh Refinement Summary–Short Transfer

k	M	Disc.	m	n	n_d	NRHS	Iter	ϵ	Time (sec)
1	(10,10)	(LA2,LA2)	87	125	38	9119	28	6.4×10^{-3}	.6179
2	(10,10)	(LA3,LA3)	159	233	74	58474	69	3.1×10^{-3}	1.289
3	(19,19)	(LA3,LA3)	303	449	146	30332	16	8.6×10^{-5}	.5749
4	(19,19)	(LA4,LA4)	447	665	218	36954	10	2.7×10^{-6}	.7068
5	(36,36)	(LA4,LA4)	855	1277	422	32430	3	4.9×10^{-8}	.4019
Total	72					167309			3.591

Figure 7.1: Optimal Solution–Short Transfer $\|\mathbf{u}(t)\|$



$$F^* = \frac{1}{2}\int_{t_0}^{t_f} \|\mathbf{u}\|^2 dt = 3.6513908 \times 10^{-3}$$

$t_1^* = 1.2892611$ $\quad t_f^* = 2.7596586$

$\tau_0^* = 1.6548720$ $\quad \tau_f^* = 3.0315362$

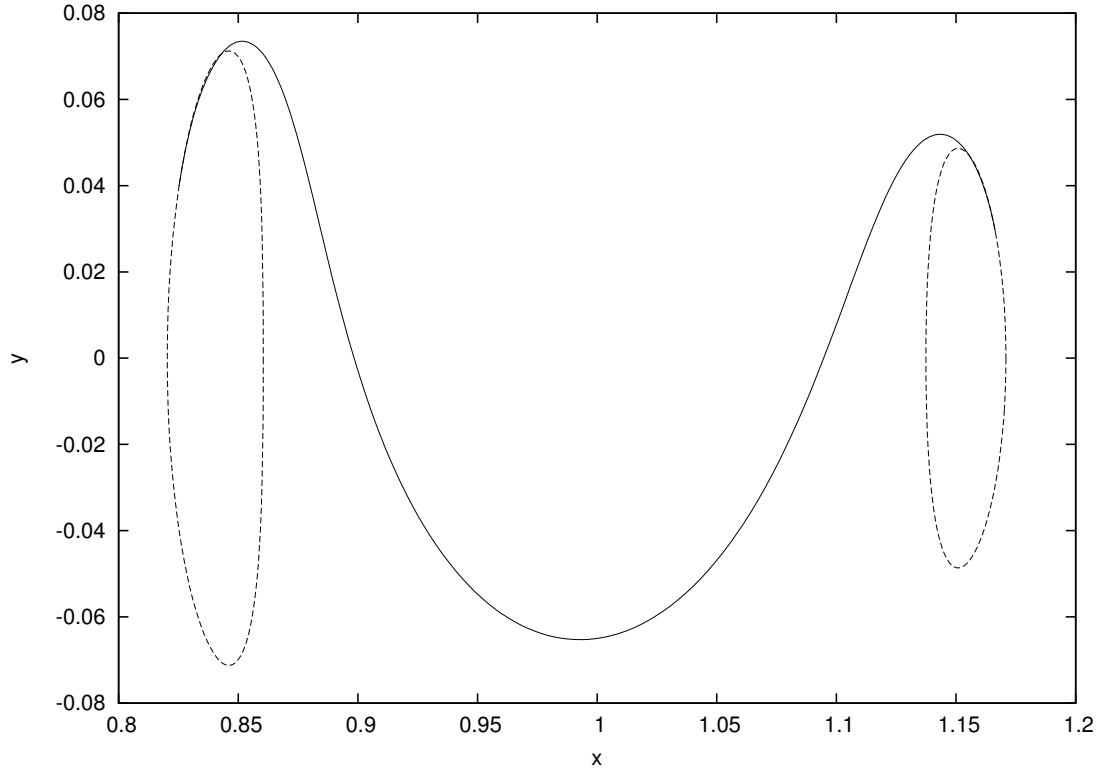


Figure 7.2: Short Transfer Trajectory

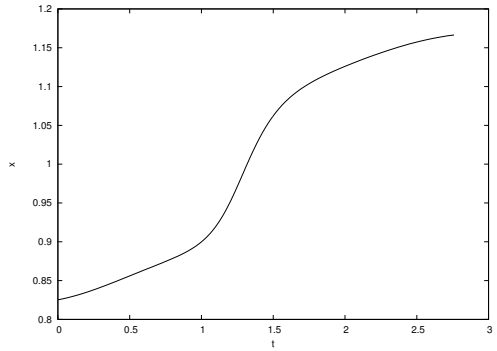


Figure 7.3: Short Transfer $x(t)$

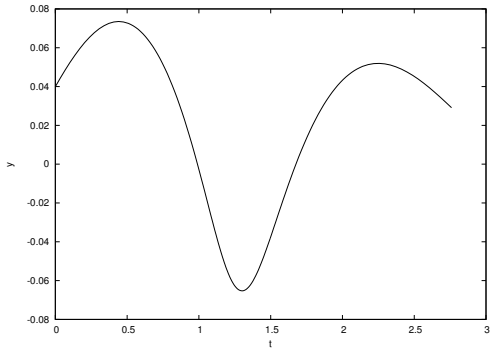


Figure 7.4: Short Transfer $y(t)$

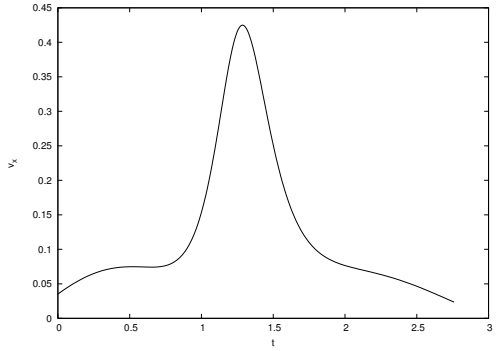


Figure 7.5: Short Transfer $v_x(t)$

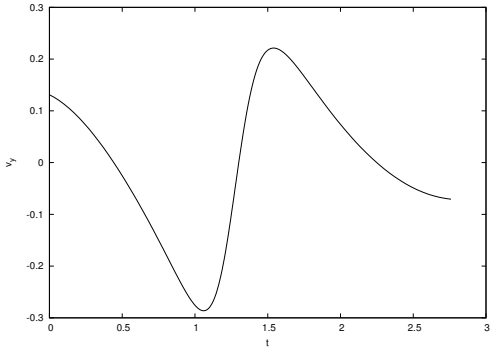


Figure 7.6: Short Transfer $v_y(t)$

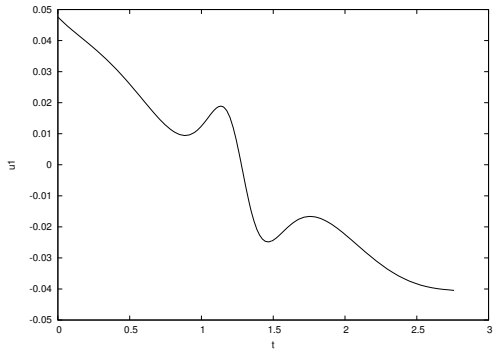


Figure 7.7: Short Transfer $u_1(t)$

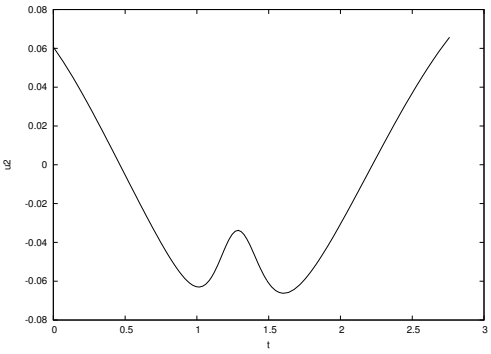


Figure 7.8: Short Transfer $u_2(t)$

7.2 Long Transfer

In contrast to the short transfer, the long transfer is modeled using four distinct *phases* as defined in Table 7.3. As before, the first phase models the departure from the L_1 orbit, and the last phase models the arrival at the L_2 orbit. However, two additional intermediate phases are introduced between the first and last phase to accommodate two revolutions about the moon.

Table 7.3: Long Transfer Phase Structure

Phase	Description	Domain	Free Parameters
1	L_1 Departure	$0 \le t \le t_1$	$t_1^{(-)}$, τ_0
2	Lunar Revolution 1	$t_1 \le t \le t_2$	$t_1^{(+)}$, $t_2^{(-)}$
3	Lunar Revolution 2	$t_2 \le t \le t_3$	$t_2^{(+)}$, $t_3^{(-)}$
4	L_2 Arrival	$t_3 \le t \le t_f$	$t_3^{(+)}$, t_f, τ_f

As before, the initial state given by the boundary condition (6.9) fixes the beginning of phase 1. However, since we are modeling two revolutions about the moon, we terminate phases one, two, and three when the trajectory passes below the moon in a posigrade direction, and so for $j = 1, 2, 3$ we require

$$x(t_j) = 1 - \mu \tag{7.12}$$
$$y(t_j) \le y_{min} \tag{7.13}$$
$$v_x(t_j) \ge 0. \tag{7.14}$$

Continuity from phase to phase is insured by imposing the continuity constraints

$$t_j^{(-)} = t_j^{(+)} \tag{7.15}$$
$$\mathbf{z}[t_j^{(-)}] = \mathbf{z}[t_j^{(+)}] \tag{7.16}$$

for all three phases. At the end of phase four we must also satisfy

$$t_f \le t_F \tag{7.17}$$
$$\mathbf{z}(t_f) = \boldsymbol{\xi}_2(\tau_f) \tag{7.18}$$

where $t_F = 10.11874803$. As before, formulating the problem using four phases is done strictly for numerical purposes. In particular the constraints (7.12)-(7.14) prevent intermediate trajectories that are unreasonable, and thus improve the robustness of the algorithm.

Since the long transfer formulation has more phases, we must supply an initial guess for all of the free parameters. Thus we guess

$$\mathbf{p}^\mathsf{T} = (t_1, t_2, t_3, t_f, \tau_0, \tau_f) = (0.1t_F, 0.5t_F, 0.9t_F, t_F, 2.096084695912298, 2.522083753145239)$$

where the choice of τ_0 and τ_f correspond to the maximum values of y as given in Table 6.1 and 6.2. For quantities that change dynamically during phase one a guess that linearly interpolates between the boundary

values is given by (7.8) as defined by the boundary values

$$\mathbf{a}^{\mathsf{T}} = [0, \boldsymbol{\xi}_1^{\mathsf{T}}(\tau_o), 0, 0] \qquad \mathbf{b}^{\mathsf{T}} = [0.1t_F, 1-\mu, -R, v_x(0), 0, 0, 0]. \tag{7.19}$$

On the last phase we again use (7.8) but replace the boundary quantities (7.19) with

$$\mathbf{a}^{\mathsf{T}} = [0.9t_F, 1-\mu, -R, v_x(t_f), 0, 0, 0] \qquad \mathbf{b}^{\mathsf{T}} = [t_F, \boldsymbol{\xi}_2^{\mathsf{T}}(\tau_f), 0, 0]. \tag{7.20}$$

While a simple linear guess of the dynamic history is acceptable on the first and last phase, for the intermediate phases it is more reasonable to supply a trajectory that circles the moon. Thus we supply a simple circle of radius R as a guess for the second and third phase dynamic history. To be more specific on phase j the value of a dynamic variable at grid point k is given by

$$\Delta t = p_j - p_{j-1} \tag{7.21}$$

$$V = \frac{2\pi R}{\Delta t} \tag{7.22}$$

$$t_k = p_{j-1} + \Delta t \left(\frac{k-1}{M-1} \right) \tag{7.23}$$

$$\theta_k = -\frac{\pi}{2} + 2\pi \left(\frac{k-1}{M-1} \right) \tag{7.24}$$

$$x(t_k) = (1-\mu) + R\cos\theta_k \tag{7.25}$$

$$y(t_k) = R\sin\theta_k \tag{7.26}$$

$$v_x(t_k) = -V\sin\theta_k \tag{7.27}$$

$$v_y(t_k) = +V\cos\theta_k \tag{7.28}$$

$$u_1(t_k) = 0 \tag{7.29}$$

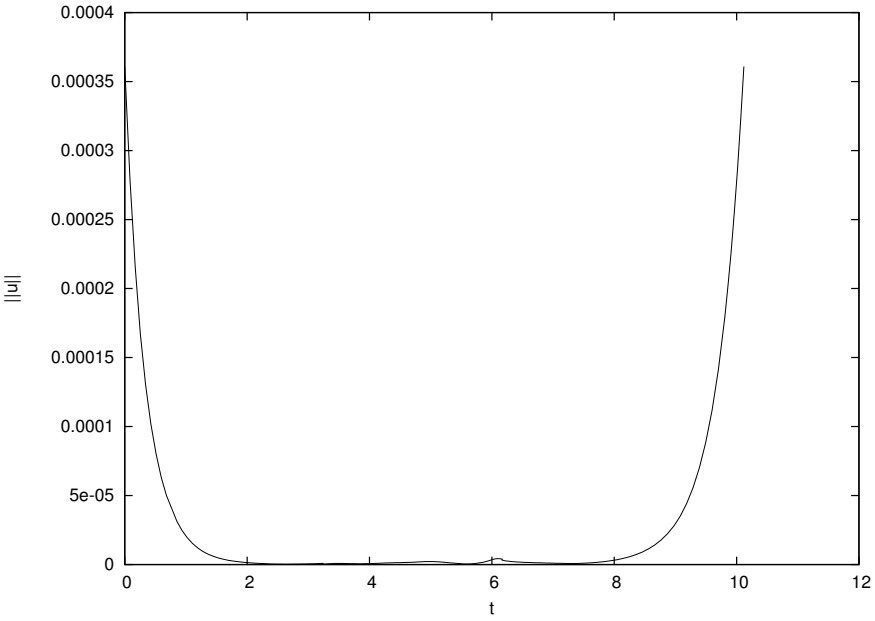
$$u_2(t_k) = 0 \tag{7.30}$$

where $j = 2, 3$ and the number of grid points $k = 1, \ldots, M$. Note that for both phase 2 and phase 3, the value of Δt is the same, since $\Delta t = p_2 - p_1 = 0.5t_F - 0.1t_F = 0.4t_F = p_3 - p_2$. As before the radius guess is $R = .1$, and the number of grid points on all phases for the first refinement iteration is $M = 20$. Using this information the solution was computed using $\mathbb{SOS}$ and Table 7.4 presents a summary of the algorithm behavior. For the first and last phase we use "(LA2),-1;(LA3),-3;(LA4),-20", and for phases two and three we use "(LA3),-3;(LA4),-4;(LA5),-20". Using an LA2 discretization in the first and last phase, and an LA3 discretization in the second and third phase, produced an NLP with $m = 482$ constraints, $n = 717$ variables, and $n_d = 235$ degrees of freedom. This NLP was solved in 2.609 CPU seconds, and resulted in a discretization error of $\epsilon = 7.9 \times 10^{-3}$. A second mesh refinement iteration with an LA4 discretization in all phases and a grid point distribution of $(20, 20, 20, 20)$ for phases 1 through 4 respectively reduced the relative error to $\epsilon = 5.7 \times 10^{-4}$. Subsequently, three additional refinement iterations were executed using the sixth order LA4 method in the first and last phase, and the eighth order LA5 method for phases two and three to reduce the error below the requested tolerance of $\epsilon = 1 \times 10^{-7}$ which corresponds to approximately eight significant figures in the solution variables. The total CPU time , was 18.112 seconds. A summary of the optimal solution values for this case is given in Figure 7.9. Figures (7.10)-(7.16) illustrate the optimal solution history.

Table 7.4: Mesh Refinement Summary–Long Transfer

k	M	Disc.	m	n	n_d	NRHS	Iter	ϵ	Time (sec)
1	(20,20,20,20)	(LA2,LA3,LA3,LA2)	482	717	235	246033	144	7.9×10^{-3}	2.609
2	(20,20,20,20)	(LA4,LA4,LA4,LA4)	938	1401	463	491828	76	5.7×10^{-4}	5.941
3	(29,20,20,25)	(LA4,LA5,LA5,LA4)	1258	1881	623	127756	12	2.0×10^{-5}	2.107
4	(49,37,32,35)	(LA4,LA5,LA5,LA4)	2082	3117	1034	193452	13	1.4×10^{-6}	5.972
5	(54,52,42,62)	(LA4,LA5,LA5,LA4)	2867	4293	1426	88870	2	9.1×10^{-8}	1.481
Total	210					1147939			18.112

Figure 7.9: Optimal Solution–Long Transfer $\|\mathbf{u}(t)\|$



$$F^* = \frac{1}{2}\int_{t_0}^{t_f} \|\mathbf{u}\|^2 dt = 2.54378004 \times 10^{-8}$$

$t_1^* = 3.2370798$	$t_2^* = 4.6847498$
$t_3^* = 6.1674429$	$t_f^* = 10.118748$
$\tau_0^* = -4.4748330 \times 10^{-3}$	$\tau_f^* = 5.2013294$

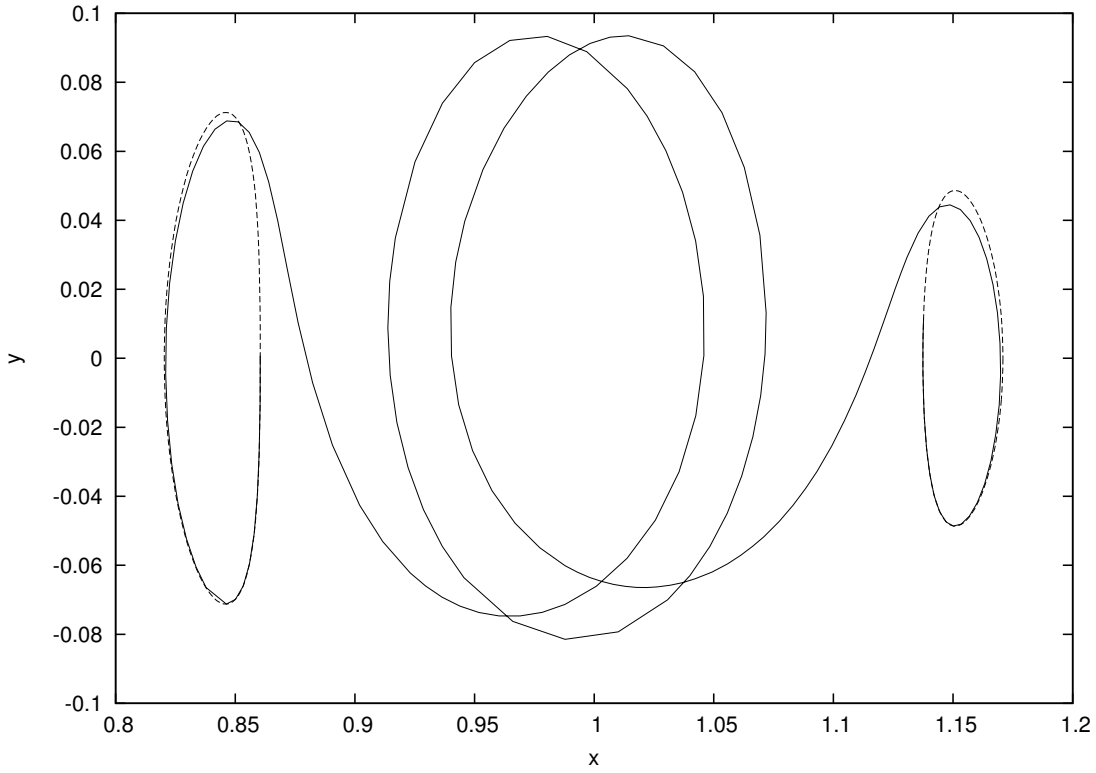


Figure 7.10: Long Transfer Trajectory

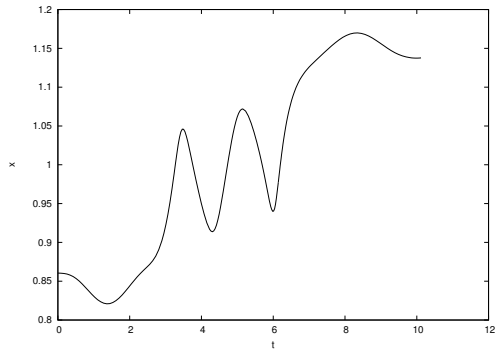


Figure 7.11: Long Transfer $x(t)$

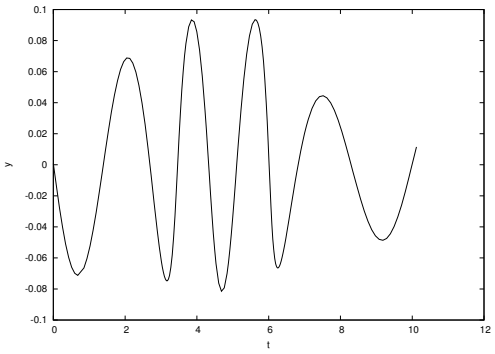


Figure 7.12: Long Transfer $y(t)$

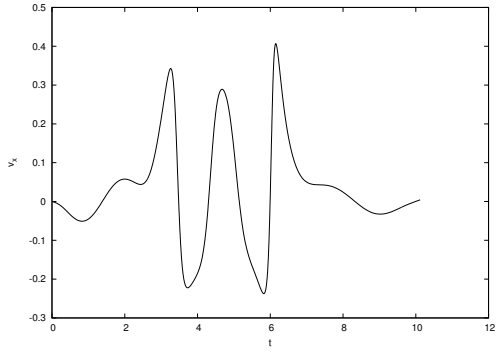


Figure 7.13: Long Transfer $v_x(t)$

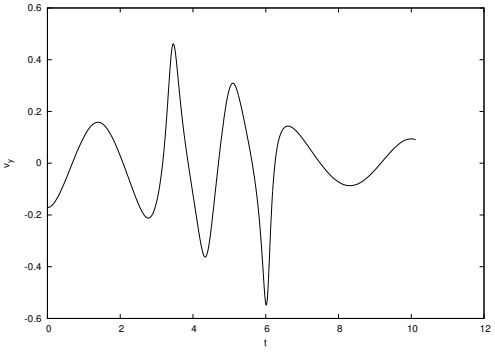


Figure 7.14: Long Transfer $v_y(t)$

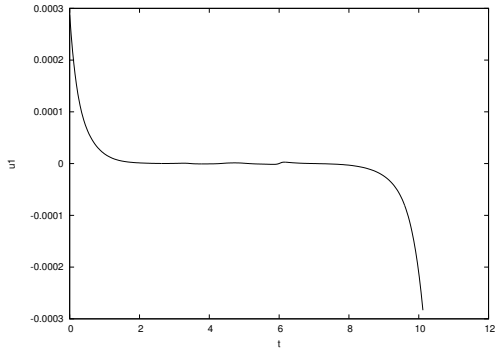


Figure 7.15: Long Transfer $u_1(t)$

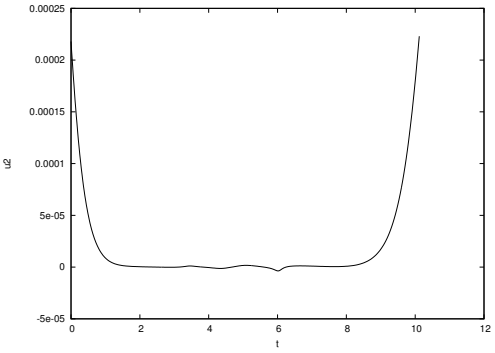


Figure 7.16: Long Transfer $u_2(t)$

8 Computational Comparisons

When solving a complicated problem such as the low-thrust transfer between libration points, there are often alternatives that can be utilized in an attempt to improve robustness and/or efficiency. This section presents a few possible alternatives for this example.

8.1 Spline Approximation

In order to evaluate the boundary condition (6.9) and (6.10) the results in the previous section utilized a FORTRAN software implementation of a Lindstedt-Poincare approximation. Since these quantities are each functions of the a single parameter it is reasonable to consider the following B-spline approximations:

$$\boldsymbol{\xi}_1(\tau_0) \approx \sum_k \boldsymbol{\alpha}_k B_k(\tau_0) \tag{8.1}$$

$$\boldsymbol{\xi}_2(\tau_f) \approx \sum_k \boldsymbol{\alpha}_k B_k(\tau_f) \tag{8.2}$$

The coefficients that define the natural cubic B-spline can be computed by evaluating the Lindstedt-Poincare approximation over the parameter range and then interpolating these values. Evaluation of this approximation can then be utilized during the optimization process. Table 8.1 summarizes the reduction in overall CPU time when B-spline approximations are used to evaluate the boundary conditions. The time spent by the $\mathbb{SOS}$ algorithm is given in the first column, whereas the time spent computing the problem functions, i.e. the ODE right hand sides $\mathbf{f}[\mathbf{z}, \mathbf{u}]$ and the boundary conditions $\boldsymbol{\xi}_1(\tau_0)$ and $\boldsymbol{\xi}_2(\tau_f)$, is summarized in the second column. Clearly there is a significant reduction in the overall solution time for both the short and long transfers when using a spline approximation.

Table 8.1: Impact of Spline Boundary Condition on CPU Time

Problem	$\mathbb{SOS}$ Algorithm (sec)	Problem Functions (sec)	Total Time (sec)
Short Transfer	1.54077	2.05368	3.59445
Short, Spline BC	2.58261	0.326948	2.90956
Long Transfer	14.5898	3.52747	18.1172
Long, Spline BC	10.7704	0.409929	11.1803

8.2 Mesh Refinement Strategy

The results presented for the short transfer in section 7.1 and the long transfer in section 7.2 were obtained using a mesh refinement strategy that exploits the benefits of the higher order Lobatto discretization. The standard default strategy used by $\mathbb{SOS}$ is "(TRP),2;(HSC),20", that is, the trapezoidal method on the first two refinement iterations, followed by the compressed Simpson method on succeeding iterations. Table 8.2 summarizes the difference between the "optimal" and "standard" strategy for both cases. Observe that the optimal strategy requires fewer grid points, and less solution time when compared to the standard strategy, for both the short and long transfers. Furthermore an additional refinement iteration was required for the short transfer. It is also apparent that for the long transfer using a high order method during phases 2 and 3 is particularly effective.

Table 8.2: Impact of Refinement Strategy on Efficiency

Problem, Strategy	Ref. Iter.	M	NRHS	Total Time (sec)
Short, Optimal	5	72	167309	3.591
Short, Standard	6	161	441931	3.619
Long, Optimal	5	210	1147939	18.112
Long, Standard	5	536	4388050	30.063

8.3 Indirect Collocation

It is well known that the optimal control problem (1.1)-(1.4) can be formulated as a two-point boundary value problem. This approach is referred to as *indirect* because it entails both the problem dynamics as well as the optimality conditions. The adjoint equations for this example are given in section 6.4, and when combined with the *transversality conditions*, a complete boundary value problem can be stated. As an alternative, let us consider an approach that does not require computation of the transversality conditions as suggested in reference [5], and illustrated in [2] and [11]. In this formulation we consider a problem with the augmented set of *differential* variables $[\mathbf{z}(t), \boldsymbol{\lambda}(t)]$ and minimize

$$F = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^{\mathsf{T}}(\boldsymbol{\lambda})\mathbf{u}(\boldsymbol{\lambda})\, dt. \tag{8.3}$$

subject to the differential equations

$$\dot{\mathbf{z}} = \mathbf{f}[\mathbf{z}, \mathbf{u}(\boldsymbol{\lambda})] \tag{8.4}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \mathbf{z}} \tag{8.5}$$

where we define

$$\mathbf{u}(t) = \mathbf{u}\left[\boldsymbol{\lambda}(t)\right] = \begin{bmatrix} -\lambda_3(t) \\ -\lambda_4(t) \end{bmatrix} \tag{8.6}$$

which follows from (6.40) and (6.41). As before the initial and terminal states are fixed by the boundary conditions (6.9) and (6.10). The collocation method can be used to solve this problem having only differential variables and no algebraic (control) variables. In so doing the adjoint equations (8.5) can be viewed as simply a different way to parameterize the control variables. Since the objective function (8.3) appears directly there is no need to compute the transversality boundary conditions.

Unfortunately an indirect formulation suffers from a number of well known drawbacks. First, one must derive the adjoint equations i.e. (6.15) - (6.41), and in general this can be a rather daunting task! Secondly, an initial guess for the adjoint variables must be provided in order to begin an iterative solution, and this can be very challenging because the adjoint variables are not physical quantities. Even with a reasonable guess for the adjoint variables, the numerical solution of the adjoint equations can be very ill-conditioned! The sensitivity of the indirect method has been recognized for some time. Computational experience with the technique in the late 1960s is summarized by Bryson and Ho [6, p. 214]:

> The main difficulty with these methods is *getting started*; i.e., finding a first estimate of the unspecified conditions at one end that produces a solution reasonably close to the specified conditions at the other end. The reason for this peculiar difficulty is the extremal solutions are often *very sensitive* to small changes in the unspecified boundary conditions.... Since the system equations and the Euler–Lagrange equations are coupled together, it is not unusual for the numerical integration, with poorly guessed initial conditions, to produce "wild" trajectories in the state space. These trajectories may be so wild that values of $x(t)$ and/or $\lambda(t)$ exceed the numerical range of the computer!

These observations by Bryson and Ho reflect experience with the most common way to treat an indirect formulation such as (8.3)-(8.5) referred to as *indirect shooting*. For a shooting method, the differential equations are propagated using a standard numerical integration algorithm. In so doing the number of free variables is small, since the dynamic history is not discretized. Indeed, Epenoy [7] describes the steps required to solve this example using a shooting method, and deal with the solution sensitivity.

The reason for this particular difficulty serves to illustrate a fundamental difference between the collocation and shooting methods. With a shooting method the ordinary differential equations are numerically integrated, step by step from the initial to final time. The steps are of the form

$$\mathbf{y}(t_k + h_k) = \mathbf{y}(t_k) + \boldsymbol{\Delta}\mathbf{y}_k + \mathbf{e}_k$$

where $\boldsymbol{\Delta}\mathbf{y}_k$ is an estimate of the change over a step of length h_k, and $\mathbf{e}_k$ is the error in this approximation. Although the numerical integration algorithm will attempt to keep $\|\mathbf{e}_k\|$ "small" rarely is the error exactly zero. Thus, after a series of steps, the accumulated error in the solution can become large, i.e. $\sum_k \|\mathbf{e}_k\| \to \infty$. When this happens the system of ODE's is considered *unstable*. In contrast, a collocation method is not a serial process, because the entire dynamic history is altered by the NLP iterations. Since the defect constraints over the entire domain are addressed simultaneously, the collocation method can be viewed as global corrector iteration. Finally, it is worth noting that since the Lobatto IIIA schemes are symmetric, there is no particular advantage to integrating "forward" or "backwards." In short, a collocation method can deal with stability much more effectively than a shooting algorithm. Furthermore, if a direct collocation solution is available, then estimates for the adjoint variables can be computed from the NLP Lagrange multipliers [3, Sec. 4.11].

Using the direct solution as an initial guess, the indirect collocation method was used for both the short and long transfer examples. Tables 8.3 and 8.4 summarize the mesh refinement history for these cases. The results for these two cases demonstrate a number of points. Clearly the single phase indirect collocation formulation converges quickly for both cases. Unfortunately, when a good initial guess is not available, many of the issues of an indirect approach are not resolved by using collocation. In particular, from a "bad" guess, the method may either converge to a local solution or fail to converge at all.

Table 8.3: Mesh Refinement Summary–Indirect Collocation, Short Transfer

k	M	Disc.	m	n	n_d	NRHS	Iter	ϵ	Time (sec)
1	71	LA3	1128	1130	2	11533	5	3.4×10^{-6}	.4859
2	141	LA3	2248	2250	2	46923	9	1.0×10^{-7}	.4459
3	146	LA3	2328	2330	2	22985	2	7.9×10^{-8}	.1309
Total	146					81441			1.0628

Table 8.4: Mesh Refinement Summary–Indirect Collocation, Long Transfer

k	M	Disc.	m	n	n_d	NRHS	Iter	ϵ	Time (sec)
1	168	HSC	1344	1346	2	90033	5	3.0×10^{-6}	.6319
2	335	HSC	2680	2682	2	922547	2	5.7×10^{-8}	1.7567
Total	335					1012580			2.3886

9 Summary

This paper describes the implementation of a direct transcription method that incorporates high order Lobatto discretization of the problem dynamics. The technique is illustrated on a challenging low thrust orbit transfer example originally studied by Epenoy [7]. In addition a number of computational alternatives are discussed.

A Lobatto IIIA Method Coefficients

$S = 2$

$$\begin{array}{c|cc} \rho_1 & \alpha_{11} & \alpha_{12} \\ \rho_2 & \alpha_{21} & \alpha_{22} \\ \hline & \beta_1 & \beta_2 \end{array} = \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$S = 3$

$$\begin{array}{c|ccc} \rho_1 & \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \rho_2 & \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \rho_3 & \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \hline & \beta_1 & \beta_2 & \beta_3 \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

$S = 4$

$$\begin{array}{c|cccc} \rho_1 & \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \rho_2 & \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \rho_3 & \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \rho_4 & \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \\ \hline & \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{array} = \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}-\frac{\sqrt{5}}{10} & \frac{11+\sqrt{5}}{120} & \frac{25-\sqrt{5}}{120} & \frac{25-13\sqrt{5}}{120} & \frac{-1+\sqrt{5}}{120} \\ \frac{1}{2}+\frac{\sqrt{5}}{10} & \frac{11-\sqrt{5}}{120} & \frac{25+13\sqrt{5}}{120} & \frac{25+\sqrt{5}}{120} & \frac{-1-\sqrt{5}}{120} \\ 1 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \\ \hline & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \end{array}$$

$S = 5$

							0	0	0	0	0	0
ρ_1	α_{11}	α_{12}	α_{13}	α_{14}	α_{15}		$\frac{1}{2} - \frac{\sqrt{21}}{14}$	$\frac{119+3\sqrt{21}}{1960}$	$\frac{343-9\sqrt{21}}{2520}$	$\frac{392-96\sqrt{21}}{2205}$	$\frac{343-69\sqrt{21}}{2520}$	$\frac{-21+3\sqrt{21}}{1960}$
ρ_2	α_{21}	α_{22}	α_{23}	α_{24}	α_{25}		$\frac{1}{2}$	$\frac{13}{320}$	$\frac{392+105\sqrt{21}}{2880}$	$\frac{8}{45}$	$\frac{392-105\sqrt{21}}{2880}$	$\frac{3}{320}$
ρ_3	α_{31}	α_{32}	α_{33}	α_{34}	α_{35}	=	$\frac{1}{2} + \frac{\sqrt{21}}{14}$	$\frac{119-3\sqrt{21}}{1960}$	$\frac{343+69\sqrt{21}}{2520}$	$\frac{392+96\sqrt{21}}{2205}$	$\frac{343+9\sqrt{21}}{2520}$	$\frac{-21-3\sqrt{21}}{1960}$
ρ_4	α_{41}	α_{42}	α_{43}	α_{44}	α_{45}		1	$\frac{1}{20}$	$\frac{49}{180}$	$\frac{16}{45}$	$\frac{49}{180}$	$\frac{1}{20}$
ρ_5	α_{51}	α_{52}	α_{53}	α_{54}	α_{55}							
	β_1	β_2	β_3	β_4	β_5			$\frac{1}{20}$	$\frac{49}{180}$	$\frac{16}{45}$	$\frac{49}{180}$	$\frac{1}{20}$

References

[1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[2] T. P. Bauer, J. T. Betts, W. P. Hallman, W. P. Huffman, and K. P. Zondervan, *Solving the Optimal Control Problem Using a Nonlinear Programming Technique Part 2: Optimal Shuttle Ascent Trajectories*, in Proceedings of the AIAA/AAS Astrodynamics Conference, AIAA-84-2038, Seattle, WA, Aug. 1984.

[3] J. T. Betts, *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*, Second Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA., 2010.

[4] ——, *Optimal Low-Thrust Orbit Transfers with Eclipsing*, Optimal Control Applications and Methods, 36 (2015), pp. 218–240.

[5] J. T. Betts, T. P. Bauer, W. P. Huffman, and K. P. Zondervan, *Solving the Optimal Control Problem Using a Nonlinear Programming Technique Part 1: General Formulation*, in Proceedings of the AIAA/AAS Astrodynamics Conference, AIAA-84-2037, Seattle, WA, Aug. 1984.

[6] A. E. Bryson, Jr. and Y.-C. Ho, *Applied Optimal Control*, John Wiley & Sons, New York, NY, 1975.

[7] R. Epenoy, *Optimal Long-Duration Low-Thrust Transfers Between Libration Point Orbits*, in 63rd International Astronautical Congress, no. 4 in IAC-12-C1.5.9, Naples, Italy, 2012.

[8] A. L. Herman and B. A. Conway, *Direct Optimization using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules*, AIAA Journal of Guidance, Control, and Dynamics, 19 (1996), pp. 592–599.

[9] L. O. Jay, *Lobatto methods*, in Encyclopedia of Applied and Computational Mathematics, Numerical Analysis of Ordinary Differential Equations, B. Engquist, ed., Springer - The Language of Science, 2013.

[10] A. V. Rao and K. D. Mease, *Eigenvector Approximate Dichotomic Basis Method for Solving Hyper-Sensitive Optimal Control Problems*, Optimal Control Applications and Methods, 20 (1999), pp. 59–77.

[11] K. P. Zondervan, T. P. Bauer, J. T. Betts, and W. P. Huffman, *Solving the Optimal Control Problem Using a Nonlinear Programming Technique Part 3: Optimal Shuttle Reentry Trajectories*, in Proceedings of the AIAA/AAS Astrodynamics Conference, AIAA-84-2039, Seattle, WA, Aug. 1984.