JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 49, No. 2, March–April 2012

Exploiting Sparsity in Direct Collocation Pseudospectral Methods for Solving Optimal Control Problems

Michael A. Patterson* and Anil V. Rao†
University of Florida, Gainesville, Florida 32611

DOI: 10.2514/1.A32071

In a direct collocation pseudospectral method, a continuous-time optimal control problem is transcribed to a finite-dimensional nonlinear programming problem. Solving this nonlinear programming problem as efficiently as possible requires that sparsity at both the first- and second-derivative levels be exploited. In this paper, a computationally efficient method is developed for computing the first and second derivatives of the nonlinear programming problem functions arising from a pseudospectral discretization of a continuous-time optimal control problem. Specifically, in this paper, expressions are derived for the objective function gradient, constraint Jacobian, and Lagrangian Hessian arising from the previously developed Radau pseudospectral method. It is shown that the computation of these derivative functions can be reduced to computing the first and second derivatives of the functions in the continuous-time optimal control problem. As a result, the method derived in this paper reduces significantly the amount of computation required to obtain the first and second derivatives required by a nonlinear programming problem solver. The approach derived in this paper is demonstrated on an example where it is found that significant computational benefits are obtained when compared against direct differentiation of the nonlinear programming problem functions. The approach developed in this paper improves the computational efficiency of solving nonlinear programming problems arising from pseudospectral discretizations of continuous-time optimal control problems.

Nomenclature

$\mathbf{a}$	=	differential equation right-hand side function
a	=	thrust acceleration, $4\pi^2 \times \text{AU/year}^2$
$\mathbf{b}$	=	boundary condition function
$\mathbf{c}$	=	path constraint function
$\mathbf{D}$	=	Radau pseudospectral differentiation matrix
g	=	integrand of cost functional
$\mathbf{h}$	=	general nonlinear programming constraint function
$\mathcal{J}$	=	continuous-time optimal control problem cost functional
K	=	number of mesh intervals
m	=	mass, $10,000 \times \text{lbm}$
$\dot{m}$	=	mass flow rate, $20,000\pi \times \text{lbm/year}$
N	=	total number of collocation points
N_k	=	polynomial degree in mesh interval k
N_z	=	number of nonzero constraint Jacobian entries
n_c	=	dimension of continuous-time path constraint
n_u	=	dimension of continuous-time control
n_y	=	dimension of continuous-time state
$\mathbf{P}$	=	general matrix
$\mathbf{Q}$	=	general matrix
$\mathbf{p}$	=	general vector
$\mathbf{p}(t)$	=	general vector function of time
$\mathbf{q}$	=	general vector
$\mathbf{q}(t)$	=	general vector function of time
r	=	radius, astronomical units $\equiv$ AU
$\dot{r}$	=	rate of change of radius, $\text{AU/year} \times 2\pi$
s	=	time on time interval $s \in [-1, +1]$
T	=	thrust, $20,000\pi \times \text{lbm} \cdot \text{AU/year}$
t_0	=	initial time
t_f	=	terminal time
t	=	time on time interval $t \in [t_0, t_f]$, dimensionless or $\text{year}/2\pi$
$\mathbf{U}_i$	=	approximation to control at collocation point i
$\mathbf{u}(t)$	=	control on time domain $t \in [t_0, t_f]$
v_r	=	radial component of velocity, $2\pi \times \text{AU/year}$
v_θ	=	tangential component of velocity, $2\pi \times \text{AU/year}$
$u_1(t)$	=	first component of control
$u_2(t)$	=	second component of control
w_j	=	j-th Legendre-Gauss-Radau quadrature weight
$\mathbf{Y}(s)$	=	state approximation on time domain $s \in [-1, +1]$
$\mathbf{y}(t)$	=	state on time domain $t \in [t_0, t_f]$
$\mathbf{y}(s)$	=	state on time domain $s \in [-1, +1]$
$\mathbf{z}$	=	nonlinear programming problem decision vector
Γ	=	matrix of defect constraint Lagrange multipliers
$\mathbf{\Lambda}$	=	matrix of nonlinear programming problem Lagrange multipliers
μ	=	sun gravitational parameter, $4\pi^2 \times \text{AU}^3/\text{year}^2$
ν	=	boundary condition Lagrange multiplier
ρ	=	nonlinear programming problem cost function
ϕ	=	optimal control problem Mayer cost function
$\mathbf{\Psi}$	=	matrix of path constraint Lagrange multipliers
$\ell^{(k)}(s)$	=	Lagrange polynomial on mesh interval $s \in [s_{k-1}, s_k]$
θ	=	angular displacement, radians
$\dot{\theta}$	=	rate of change of angular displacement, $2\pi \times$ radians/year

Presented as AAS 11-640 at the 2011 AAS/AIAA Astrodynamics Specialist Conference, Girdwood, AK, 31 July–4 August 2011; received 18 March 2011; revision received 8 June 2011; accepted for publication 9 June 2011.

*Ph.D. Candidate, Department of Mechanical and Aerospace Engineering; mpatterson@ufl.edu.

†Assistant Professor, Department of Mechanical and Aerospace Engineering; anilvrao@ufl.edu. Associate Fellow AIAA (Corresponding Author).

I. Introduction

OVER the past two decades, direct collocation methods have become popular in the numerical solution of nonlinear optimal control problems. In a direct collocation method, the state is approximated using a set of trial (basis) functions, and the dynamics are collocated at specified set of points in the time interval. Direct collocation methods are employed either as h methods [1–5], p methods [1–5], or hp methods [1–5]. In an h method, the state is approximated using many fixed low-degree polynomial (e.g., second-degree or third-degree) mesh intervals. Convergence in an h

method is then achieved by increasing the number of mesh intervals [6–8]. In a p method, the state is approximated using few mesh intervals (often a single mesh interval is used), and convergence is achieved by increasing the degree of the polynomial [9–16]. In an hp method, both the number of mesh intervals and the degree of the polynomial within each mesh interval is varied, and convergence is achieved through the appropriate combination of the number of mesh intervals and the polynomial degrees within each interval [17,18].

In recent years, interest has increased in using direct collocation pseudospectral methods [9–15,19–21]. In a pseudospectral method, the collocation points are based on accurate quadrature rules, and the basis functions are typically Chebyshev or Lagrange polynomials. Originally, pseudospectral methods were employed as p methods. For problems whose solutions are smooth and well-behaved, a pseudospectral method has a simple structure and converges at an exponential rate [22–24]. The most well-developed p-type pseudospectral methods are the Gauss pseudospectral method (GPM), [11,19], the Radau pseudospectral method (RPM) [14,15,21], and the Lobatto pseudospectral method (LPM)[9]. More recently, it has been found that computational efficiency and accuracy can be increased by using either an h [21] or an hp pseudospectral method [17,25].

Although pseudospectral methods are highly accurate, proper implementation is important in order to obtain solutions in a computationally efficient manner. Specifically, state-of-the-art, gradient-based nonlinear programming (NLP) solvers require that first and/or second derivatives of the NLP functions, or estimates of these derivatives, be supplied. In a first-derivative (quasi-Newton) NLP solver, the objective function gradient and constraint Jacobian are used together with a dense quasi-Newton approximation of the Lagrangian Hessian [typically a Broyden–Fletcher–Goldfarb–Shanno (BFGS) or Davidon–Fletcher–Powell (DFP) quasi-Newton approximation is used]. In a second-derivative (Newton) NLP solver, the first derivatives of a quasi-Newton method are used together with an accurate approximation of the Lagrangian Hessian. Examples of commonly-used, first-derivative NLP solvers include NPSOL [26] and SNOPT [27,28], whereas well-known, second-derivative NLP solvers include IPOPT [29] and KNITRO [30].

Generally speaking, first-derivative methods for solving NLPs are more commonly used than second-derivative methods because of the great challenge that arises from computing an accurate approximation to a Lagrangian Hessian. It is known, however, that providing an accurate Lagrangian Hessian can significantly improve the computational performance of an NLP solver over using a quasi-Newton method. The potential for a large increase in efficiency and reliability is particularly evident when the NLP is sparse. Although having an accurate Lagrangian Hessian is desirable, even for sparse NLPs, computing a Hessian is inefficient if not done properly. Although current uses of pseudospectral methods have exploited sparsity at the first-derivative level, sparsity at the second-derivative level has not yet been fully understood or exploited.

In this paper, an efficient approach is derived for computing the first and second derivatives of NLP functions arising from a direct collocation pseudospectral method. Specifically, we develop expressions for the objective function gradient, constraint Jacobian, and Lagrangian Hessian corresponding to the previously-developed RPM [14,15,17,21]. A key contribution of this paper is the elegant structure of the pseudospectrally-discretized NLP derivative functions. Moreover, it is shown that the NLP derivative functions can be obtained by differentiating only the functions of the continuous-time optimal control problem. Because the optimal control functions depend upon many fewer variables than the functions of the NLP, the approach developed in this paper reduces significantly the computational effort required to obtain the NLP derivative functions. In addition, the approach developed in this paper provides the complete first- and second-derivative sparse structure of the NLP. The computational advantages of our approach over direct differentiation of the NLP functions are demonstrated in an example using the NLP solver IPOPT [29].

It is noted that Betts and Huffman [31] develop an approach for exploiting sparsity in local direct collocation methods (e.g., Euler, Hermite-Simpson, and Runge-Kutta methods). According to Betts and Huffman, the NLP derivative functions and associated sparsity patterns are obtained using sparse finite differences where the functions of the optimal control problem are differentiated at the collocation points. The work of this paper builds upon the work of Betts and Huffman for pseudospectral methods. In particular, in this research, we take direct advantage of the special mathematical form of a pseudospectral method and develop expressions for the first and second derivatives of the NLP functions. Specifically, we show that the NLP derivatives functions can be reduced to evaluating the derivatives of the continuous-time optimal control functions at the discretization points (i.e., collocation points or noncollocated endpoints). As a result, our approach reduces significantly the amount of computational effort required to determine the NLP derivative functions when compared with direct differentiation of the NLP functions. Moreover, our approach is shown by example to be much more efficient, using even finite-difference approximations than directly differentiating the NLP functions using an efficient automatic differentiator. In addition, finite differencing is found to be only slightly less efficient than analytic differentiation. As a result, our approach increases significantly the utility of direct pseudospectral methods for solving optimal control problems.

This paper is organized as follows. In section II, we provide our notation and conventions used throughout this paper. In section III, we state the continuous-time Bolza optimal control problem. In section IV, we state the RPM[14–16] that is used to derive the NLP derivative functions. In section V, we derive expressions for the objective function gradient, constraint Jacobian, and Lagrangian Hessian of the NLP that arises from the discretization of the continuous-time Bolza optimal control problem of section III using the RPM of section IV. In section VI, we provide a discussion of the underlying structure of the derivative functions. In section VII, we provide an example that demonstrates the great improvement in computational efficiency obtained using the method of this paper. Finally, in section VIII, we provide conclusions on our work.

II. Notation and Conventions

Throughout this paper, the following notation and conventions will be employed. All scalars will be represented by lowercase symbols (e.g., y, u). All vector functions of time will be treated as row vectors and will be denoted by lowercase bold symbols. Thus, if $\mathbf{p}(t) \in \mathbb{R}^n$ is a vector function of time, then $\mathbf{p}(t) = [p_1(t) \cdots p_n(t)]$. Any vector that is not a function of time will be denoted as a column vector, i.e., a static vector $\mathbf{z} \in \mathbb{R}^n$ will be treated as a column vector. Next, matrices will be denoted by uppercase bold symbols. Thus, $\mathbf{P} \in \mathbb{R}^{N\times n}$ is a matrix of size $N \times n$. Furthermore, if $\mathbf{f}(\mathbf{p})$, $\mathbf{f}$: $\mathbb{R}^n \longrightarrow \mathbb{R}^m$, is a function that maps row vectors $\mathbf{p} \in \mathbb{R}^n$ to row vectors $\mathbf{f}(\mathbf{p}) \in \mathbb{R}^m$, then the result of evaluating $\mathbf{f}(\mathbf{p})$ at the points $(\mathbf{p}_1, \ldots, \mathbf{p}_N)$ is the matrix $\mathbf{F} \in \mathbb{R}^{N\times n} \equiv [\mathbf{f}(\mathbf{p}_k)]_N^1$,

$$\mathbf{F}_N^1 \equiv [\mathbf{f}(\mathbf{p}_k)]_N^1 = \begin{bmatrix} \mathbf{f}(\mathbf{p}_1) \\ \vdots \\ \mathbf{f}(\mathbf{p}_N) \end{bmatrix}$$

A single subscript i attached to a matrix denotes a particular row of the matrix, i.e., $\mathbf{P}_i$ is the i-th row of the matrix $\mathbf{P}$. A double subscript i, j attached to a matrix denotes element located in row i and column j of the matrix, i.e., $\mathbf{P}_{i,j}$ is the (i, j)-th element of the matrix $\mathbf{P}$. Furthermore, the notation $\mathbf{P}_{:,j}$ will be used to denote all of the rows and column j of a matrix $\mathbf{P}$. Finally, $\mathbf{P}^{\mathsf{T}}$ will be used to denote the transpose of a matrix $\mathbf{P}$.

Next, let $\mathbf{P}$ and $\mathbf{Q}$ be $n \times m$ matrices. Then, the element-by-element multiplication of $\mathbf{P}$ and $\mathbf{Q}$ is defined as

$$\mathbf{P} \circ \mathbf{Q} = \begin{bmatrix} p_{11}q_{11} & \cdots & p_{1m}q_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1}q_{n1} & \cdots & p_{nm}q_{nm} \end{bmatrix}$$

It is noted further that $\mathbf{P} \circ \mathbf{Q}$ is not standard matrix multiplication. Furthermore, if $\mathbf{p} \in \mathbb{R}^n$, then the operation diag$(\mathbf{p})$ denotes $n \times n$ diagonal matrix formed by the elements of $\mathbf{p}$,

$$\text{diag}\,(\mathbf{p}) = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{bmatrix}$$

Finally, the notation $\mathbf{0}_{n\times m}$ represents an $n \times m$ matrix of zeros, whereas $\mathbf{1}_{n\times m}$ represents an $n \times m$ matrix of all ones.

Next, we define the notation for derivatives of functions of vectors. First, let $f(\mathbf{p})$, $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$. Then, $\nabla_{\mathbf{p}} f(\mathbf{p}) \in \mathbb{R}^n$ is a row vector of length n and is defined as

$$\nabla_{\mathbf{p}} f(\mathbf{p}) = \begin{bmatrix} \frac{\partial f}{\partial p_1} & \cdots & \frac{\partial f}{\partial p_n} \end{bmatrix}$$

Next, let $\mathbf{f}(\mathbf{p})$, $\mathbf{f}\colon \mathbb{R}^n \to \mathbb{R}^m$, where $\mathbf{p}$ may be either a row vector or a column vector and $\mathbf{f}(\mathbf{p})$ has the same orientation (i.e., either row vector or column vector) as $\mathbf{p}$. Then, $\nabla_{\mathbf{p}}\mathbf{f}$ is the m by n matrix whose i-th row is $\nabla_{\mathbf{p}} f_i$; that is,

$$\nabla_{\mathbf{p}}\mathbf{f} = \begin{bmatrix} \nabla_{\mathbf{p}} f_1 \\ \vdots \\ \nabla_{\mathbf{p}} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \cdots & \frac{\partial f_1}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial p_1} & \cdots & \frac{\partial f_m}{\partial p_n} \end{bmatrix}$$

The following conventions will be used for second derivatives of scalar functions. Given a function, $f(\mathbf{p}, \mathbf{q})$, where $f\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ maps a pair of row vectors, $\mathbf{p} \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^m$, to a scalar, $f(\mathbf{p}, \mathbf{q}) \in \mathbb{R}$, then the mixed second derivative, $\nabla^2_{\mathbf{pq}}$, is an n by m matrix,

$$\nabla^2_{\mathbf{pq}} f = \begin{bmatrix} \frac{\partial^2 f}{\partial p_1 \partial q_1} & \cdots & \frac{\partial^2 f}{\partial p_1 \partial q_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial p_n \partial q_1} & \cdots & \frac{\partial^2 f}{\partial p_n \partial q_m} \end{bmatrix} = [\nabla^2_{\mathbf{qp}} f]^{\mathsf{T}}$$

Thus, for a function of the form $f(\mathbf{p})$, where $f\colon \mathbb{R}^n \to \mathbb{R}$ we have

$$\nabla^2_{\mathbf{pp}} f = \begin{bmatrix} \frac{\partial^2 f}{\partial p_1^2} & \cdots & \frac{\partial^2 f}{\partial p_1 \partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial p_n \partial p_1} & \cdots & \frac{\partial^2 f}{\partial p_n^2} \end{bmatrix} = [\nabla^2_{\mathbf{pp}} f]^{\mathsf{T}}$$

III. Bolza Optimal Control Problem

Consider the following general optimal control problem in Bolza form. Determine the state, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$, the control, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$, the initial time, t_0, and the terminal time, t_f, on the time interval, $t \in [t_0, t_f]$, that minimize the cost functional

$$\mathcal{J} = \phi(\mathbf{y}(t_0), t_0, y(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{y}(t), \mathbf{u}(t), t)\,\mathrm{d}t \tag{1}$$

subject to the dynamic constraints

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{a}(\mathbf{y}(t), \mathbf{u}(t), t) \tag{2}$$

the inequality path constraints

$$\mathbf{c}_{\min} \le \mathbf{c}(\mathbf{y}(t), \mathbf{u}(t), t) \le \mathbf{c}_{\max} \tag{3}$$

and the boundary conditions

$$\mathbf{b}_{\min} \le \mathbf{b}(\mathbf{y}(t_0), t_0, \mathbf{y}(t_f), t_f) \le \mathbf{b}_{\min} \tag{4}$$

The functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ are defined by the following mappings:

$$\phi\colon \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \to \mathbb{R}, \qquad g\colon \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}$$
$$\mathbf{a}\colon \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_y}, \qquad \mathbf{c}\colon \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_c}$$
$$\mathbf{b}\colon \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \to \mathbb{R}^{n_b}$$

where we remind the reader that all vector functions of time are treated as row vectors.

In this paper, it will be useful to modify the Bolza problem given in Eqs. (1–4) as follows. Let $s \in [-1, +1]$ be a new independent variable. The variable t is then defined in terms of s as

$$t = \frac{t_f - t_0}{2} s + \frac{t_f + t_0}{2} \tag{5}$$

The Bolza problem of Eqs. (1–4) is then defined in terms of the variable s as follows. Determine the state, $\mathbf{y}(s) \in \mathbb{R}^{n_y}$, the control, $\mathbf{u}(s) \in \mathbb{R}^{n_u}$, the initial time, t_0, and the terminal time, t_f, on the time interval, $s \in [-1, +1]$, that minimize the cost functional

$$\mathcal{J} = \phi(\mathbf{y}(-1), t_0, \mathbf{y}(+1), t_f) + \frac{t_f - t_0}{2} \int_{-1}^{+1} g(\mathbf{y}(s), \mathbf{u}(s), s; t_0, t_f)\,\mathrm{d}s \tag{6}$$

subject to the dynamic constraints

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}s} = \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{y}(s), \mathbf{u}(s), s; t_0, t_f) \tag{7}$$

the inequality path constraints

$$\mathbf{c}_{\min} \le \mathbf{c}(\mathbf{y}(s), \mathbf{u}(s), s; t_0, t_f) \le \mathbf{c}_{\max} \tag{8}$$

and the boundary conditions

$$\mathbf{b}_{\min} \le \mathbf{b}(\mathbf{y}(-1), t_0, \mathbf{y}(+1), t_f) \le \mathbf{b}_{\min} \tag{9}$$

Suppose now that the time interval, $s \in [-1, +1]$, is divided into a mesh consisting of K mesh intervals $[s_{k-1}, s_k]$, $k = 1, \ldots, K$, where $(s_0, \ldots, s_K)$ are the mesh points. The mesh points have the property that $-1 = s_0 < s_1 < s_2 < \cdots < s_K = s_f = +1$. Next, let $\mathbf{y}^{(k)}(s)$ and $\mathbf{u}^{(k)}(s)$ be the state and control in mesh interval k. The Bolza optimal control problem of Eqs. (6–9) can then written as follows. First, the cost functional of Eq. (6) can be written as

$$\mathcal{J} = \phi(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{K} \int_{s_{k-1}}^{s_k} g(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), \tau; t_0, t_f)\,\mathrm{d}\tau \qquad (k = 1, \ldots, K) \tag{10}$$

Next, the dynamic constraints of Eq. (7) in mesh interval k can be written as

$$\frac{\mathrm{d}\mathbf{y}^{(k)}(s)}{\mathrm{d}s} = \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{y}^{(k)}(s), \mathbf{u}^{(k)}(s), s; t_0, t_f), \qquad (k = 1, \ldots, K) \tag{11}$$

Furthermore, the path constraints of (8) in mesh interval k are given as

$$\mathbf{c}_{\min} \le \mathbf{c}(\mathbf{y}^{(k)}(s), \mathbf{u}^{(k)}(s), s; t_0, t_f) \le \mathbf{c}_{\max}, \qquad (k = 1, \ldots, K) \tag{12}$$

Finally, the boundary conditions of Eq. (9) are given as

$$\mathbf{b}_{\min} \le \mathbf{b}(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f) \le \mathbf{b}_{\max} \tag{13}$$

Because the state must be continuous at each interior mesh point, it is required that the condition, $\mathbf{y}(s_k^-) = \mathbf{y}(s_k^+)$, be satisfied at the interior mesh points, $(s_1, \ldots, s_{K-1})$.

IV. Radau Pseudospectral Method

The multiple-interval form of the continuous-time Bolza optimal control problem in section III is discretized using the previously-developed RPM as described in Garg et al. [14]. Although the RPM is chosen, with only slight modifications, the approach developed in this paper can be used with other pseudospectral methods (e.g., the GPM [11,13,19] or the LPM [9]). An advantage of using the Radau scheme is that the continuity conditions $\mathbf{y}(s_k^-) = \mathbf{y}(s_k^+)$ across mesh points are particularly easy to implement.

In the RPM, the state of the continuous-time Bolza optimal control problem is approximated in each mesh interval $k \in [1, \ldots, K]$ as

$$\mathbf{y}^{(k)}(s) \approx \mathbf{Y}^{(k)}(s) = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \ell_j^{(k)}(s), \qquad \ell_j^{(k)}(s) = \prod_{l=1 l\neq j}^{N_k+1} \frac{s - s_l^{(k)}}{s_j^{(k)} - s_l^{(k)}} \tag{14}$$

where $s \in [-1, +1]$, $\ell_j^{(k)}(s)$, and $j = 1, \ldots, N_k + 1$ is a basis of Lagrange polynomials, $(s_1^{(k)}, \ldots, s_{N_k}^{(k)})$ are the Legendre-Gauss-Radau [32] (LGR) collocation points in mesh interval k defined on the subinterval $s \in [s_{k-1}, s_k)$, and $s_{N_k+1}^{(k)} = s_k$ is a noncollocated point. Differentiating $\mathbf{Y}^{(k)}(s)$ in Eq. (14) with respect to s, we obtain

$$\frac{\mathrm{d}\mathbf{Y}^{(k)}(s)}{\mathrm{d}s} = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \frac{\mathrm{d}\ell_j^{(k)}(s)}{\mathrm{d}s} \tag{15}$$

The cost functional of Eq. (10) is then approximated using a multiple-interval LGR quadrature as

$$\mathcal{J} \approx \phi(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_K) + \sum_{k=1}^{K} \sum_{j=1}^{N_k} \frac{t_f - t_0}{2} w_j^{(k)} g(\mathbf{Y}_j^{(k)}, \mathbf{U}_j^{(k)}, s_j^{(k)}; t_0, t_f) \tag{16}$$

where $w_j^{(k)}, j = 1, \ldots, N_k$ are the LGR quadrature weights [32] in mesh interval $k \in [1, \ldots, K]$ defined on the interval $s \in [s_{k-1}, s_k)$, $\mathbf{U}_i^{(k)}$, $i = 1, \ldots, N_k$, are the approximations of the control at the N_k LGR points in mesh interval $k \in [1, \ldots, K]$, $\mathbf{Y}_1^{(1)}$ is the approximation of $\mathbf{y}(s_0 = -1)$, and $\mathbf{Y}_{N_K+1}^{(K)}$ is the approximation of $\mathbf{y}(s_K = +1)$. Collocating the dynamics of Eq. (11) at the N_k LGR points using Eq. (15), we have

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} - \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, s_i^{(k)}; t_0, t_f) = \mathbf{0} \qquad (i = 1, \ldots, N_k) \tag{17}$$

where $t_i^{(k)}$ are obtained from $s_k^{(k)}$ using Eq. (5) and

$$D_{ij}^{(k)} = \left[\frac{\mathrm{d}\ell_j^{(k)}(s)}{\mathrm{d}s}\right]_{s_i^{(k)}}, \qquad (i = 1, \ldots, N_k, \quad j = 1, \ldots, N_k + 1 \quad k = 1, \ldots, K) \tag{18}$$

is the $N_k \times (N_k + 1)$ Radau pseudospectral differentiation matrix [14] in mesh interval $k \in [1, \ldots, K]$. Next, the path constraints of Eq. (12) in mesh interval $k \in [1, \ldots, K]$ are enforced at the N_k LGR points as

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, s_i^{(k)}; t_0, t_f) \leq \mathbf{c}_{\max}, \qquad (i = 1, \ldots, N_k) \tag{19}$$

Furthermore, the boundary conditions of Eq. (13) are approximated as

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f) \leq \mathbf{b}_{\max} \tag{20}$$

It is noted that continuity in the state at the interior mesh points $k \in [1, \ldots, K-1]$ is enforced via the condition

$$\mathbf{Y}_{N_k+1}^{(k)} = \mathbf{Y}_1^{(k+1)}, \qquad (k = 1, \ldots, K-1) \tag{21}$$

where we note that the same variable is used for both $\mathbf{Y}_{N_k+1}^{(k)}$ and $\mathbf{Y}_1^{(k+1)}$. Hence, the constraint of Eq. (21) is eliminated from the problem because it is taken into account explicitly. The NLP that arises from the Radau pseudospectral approximation is then to minimize the cost function of Eq. (16) subject to the algebraic constraints of Eqs. (17–20).

Suppose now that we define the following quantities in mesh intervals, $k \in [1, \ldots, K-1]$, and the final mesh interval, K:

$$\begin{aligned}
&\mathbf{s}^{(k)} = [s_i^{(k)}]_{N_k}^1, \quad k = 1, \ldots, K-1, \qquad \mathbf{s}^{(K)} = [s_i^{(K)}]_{N_K+1}^1 \\
&\mathbf{t}^{(k)} = [t_i^{(k)}]_{N_k}^1, \quad k = 1, \ldots, K-1, \qquad \mathbf{t}^{(K)} = [t_i^{(K)}]_{N_K+1}^1 \\
&\mathbf{Y}^{(k)} = [\mathbf{Y}_i^{(k)}]_{N_k}^1, \quad k = 1, \ldots, K-1, \qquad \mathbf{Y}^{(K)} = [\mathbf{Y}_i^{(K)}]_{N_K+1}^1 \\
&\mathbf{U}^{(k)} = [\mathbf{U}_i^{(k)}]_{N_k}^1, \quad k = 1, \ldots, K \\
&\mathbf{g}^{(k)} = [g(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, s_i^{(k)}; t_0, t_f)]_{N_k}^1, \quad k = 1, \ldots, K \\
&\mathbf{A}^{(k)} = [\mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, s_i^{(k)}; t_0, t_f)]_{N_k}^1, \quad k = 1, \ldots, K \\
&\mathbf{C}^{(k)} = [\mathbf{c}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, s_i^{(k)}; t_0, t_f)]_{N_k}^1, \quad k = 1, \ldots, K \\
&\mathbf{w}^{(k)} = [w_i]_{N_k}^1, \quad k = 1, \ldots, K, \qquad N = \sum_{k=1}^{K} N_k
\end{aligned}$$

We then define the following quantities:

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}^{(1)} \\ \vdots \\ \mathbf{s}^{(K)} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} \mathbf{t}^{(1)} \\ \vdots \\ \mathbf{t}^{(K)} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}^{(1)} \\ \vdots \\ \mathbf{w}^{(K)} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}^{(1)} \\ \vdots \\ \mathbf{Y}^{(K)} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}^{(1)} \\ \vdots \\ \mathbf{U}^{(K)} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \mathbf{g}^{(1)} \\ \vdots \\ \mathbf{g}^{(K)} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{(1)} \\ \vdots \\ \mathbf{A}^{(K)} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}^{(1)} \\ \vdots \\ \mathbf{C}^{(K)} \end{bmatrix} \tag{22}$$

It is noted for completeness that $\mathbf{t} \in \mathbb{R}^{N+1}$, $\mathbf{s} \in \mathbb{R}^{N+1}$, $\mathbf{Y} \in \mathbb{R}^{(N+1)\times n_y}$, $\mathbf{U} \in \mathbb{R}^{N\times n_u}$, $\mathbf{g} \in \mathbb{R}^{N}$, $\mathbf{A} \in \mathbb{R}^{N\times n_y}$, and $\mathbf{C} \in \mathbb{R}^{N\times n_c}$. The cost function and discretized dynamic constraints given in Eqs. (16) and (17) can then be written compactly as

$$\mathcal{J} \approx \phi(\mathbf{Y}_1, t_0, \mathbf{Y}_{N+1}, t_f) + \frac{t_f - t_0}{2} \mathbf{w}^{\mathsf{T}} \mathbf{g} \tag{23}$$

$$\mathbf{\Delta} = \mathbf{D}\mathbf{Y} - \frac{t_f - t_0}{2} \mathbf{A} = \mathbf{0} \tag{24}$$

where $\mathbf{\Delta} \in \mathbb{R}^{N\times n_y}$, and $\mathbf{D}$ is the composite Radau pseudospectral differentiation matrix. A schematic of the composite Radau differentiation matrix, $\mathbf{D}$, is shown in Fig. 1, where it is seen that $\mathbf{D}$ has a block structure with nonzero elements in the row-column indices $(\sum_{l=1}^{k-1} N_l + 1, \ldots, \sum_{l=1}^{k} N_l, \ \sum_{l=1}^{k-1} N_l + 1, \ldots, \sum_{l=1}^{k} N_l + 1)$, where for every mesh interval $k \in [1, \ldots, K]$ the nonzero elements are defined by the matrix given in Eq. (18). Next, the discretized path constraints of Eq. (19) are expressed as

$$\mathbf{C}_{\min} \leq \mathbf{C} \leq \mathbf{C}_{\max} \tag{25}$$

where $\mathbf{C}_{\min}$ and $\mathbf{C}_{\max}$ are matrices of the same size as $\mathbf{C}$ and whose rows contain the vectors $\mathbf{c}_{\min}$ and $\mathbf{c}_{\max}$, respectively. Furthermore, the discretized boundary conditions of Eq. (20) can be written as

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{Y}_1, t_0, \mathbf{Y}_{N+1}, t_f) \leq \mathbf{b}_{\max} \tag{26}$$

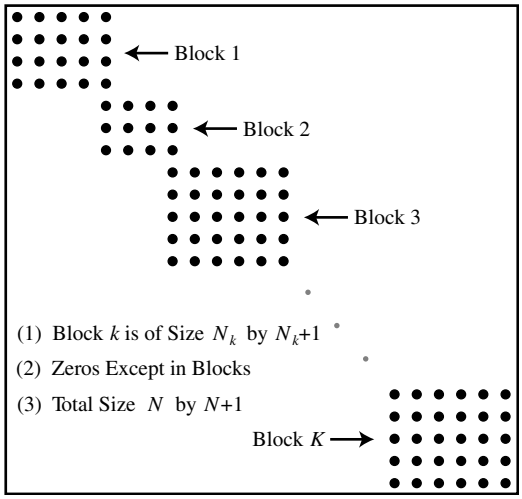


Fig. 1 Structure of composite Radau pseudospectral differentiation matrix where the mesh consists of K mesh intervals.

The NLP problem associated with the RPM is then to minimize the cost function of Eq. (23) subject to the algebraic constraints of Eqs. (24–26). Finally, let $(\alpha, \beta) \in \mathbb{R}^{N+1}$ be defined as

$$\alpha = \frac{\partial \mathbf{t}}{\partial t_0} = \frac{\mathbf{1} - \mathbf{s}}{2}, \qquad \beta = \frac{\partial \mathbf{t}}{\partial t_f} = \frac{\mathbf{1} + \mathbf{s}}{2} \tag{27}$$

where the derivatives in Eq. (27) are obtained from Eq. (5).

V. Computation of Radau Pseudospectral NLP Derivatives

The NLP problem arising from the RPM presented in section IV has the following general form. Determine the vector of decision variables $\mathbf{z} \in \mathbb{R}^{N(n_y+n_c)+2}$ that minimizes the cost function

$$f(\mathbf{z}) \tag{28}$$

subject to the constraints

$$\mathbf{h}_{\min} \le \mathbf{h}(\mathbf{z}) \le \mathbf{h}_{\max} \tag{29}$$

In the case of the RPM, the decision vector, $\mathbf{z}$, constraint function, $\mathbf{h}(\mathbf{z})$, and cost function, $f(\mathbf{z})$, are given, respectively, as

$$\mathbf{z} = \begin{bmatrix} \mathbf{Y}_{:,1} \\ \vdots \\ \mathbf{Y}_{:,n_y} \\ \mathbf{U}_{:,1} \\ \vdots \\ \mathbf{U}_{:,n_u} \\ t_0 \\ t_f \end{bmatrix}, \qquad \mathbf{h} = \begin{bmatrix} \mathbf{\Delta}_{:,1} \\ \vdots \\ \mathbf{\Delta}_{:,n_y} \\ \mathbf{C}_{:,1} \\ \vdots \\ \mathbf{C}_{:,n_c} \\ \mathbf{b}_{1:n_b} \end{bmatrix}, \qquad f(\mathbf{z}) = \phi(\mathbf{z}) + \gamma(\mathbf{z}) \tag{30}$$

where ϕ is obtained directly from Eq. (23), and γ is given as

$$\gamma = \frac{t_f - t_0}{2} \mathbf{w}^{\mathsf{T}} \mathbf{g} \tag{31}$$

We now systematically determine expressions for the gradient of the NLP objective function, the Jacobian of the NLP constraints, and the Hessian of the NLP Lagrangian. The key result of this section is that these NLP derivatives are obtained by differentiating the functions of the continuous-time Bolza optimal control problem as defined in Eqs. (1–4) as opposed to differentiating the functions of the NLP.

A. Gradient of Objective Function

The gradient of the objective function in Eq. (30) with respect to the Radau pseudospectral NLP decision vector, $\mathbf{z}$, is given as

$$\nabla_{\mathbf{z}} f = \nabla_{\mathbf{z}} \phi + \nabla_{\mathbf{z}} \gamma \tag{32}$$

The derivative, $\nabla_{\mathbf{z}} \phi$, is obtained as

$$\nabla_{\mathbf{z}} \phi = \begin{bmatrix} \nabla_{\mathbf{Y}} \phi & \nabla_{\mathbf{U}} \phi & \nabla_{t_0} \phi & \nabla_{t_f} \phi \end{bmatrix} \tag{33}$$

where

$$\nabla_{\mathbf{Y}} \phi = \begin{bmatrix} \nabla_{\mathbf{Y}_{:,1}} \phi & \cdots & \nabla_{\mathbf{Y}_{:,n_y}} \phi \end{bmatrix} \qquad \nabla_{\mathbf{U}} \phi = [\mathbf{0}_{1 \times N n_u}] \tag{34}$$

The derivatives, $\nabla_{\mathbf{Y}_{:,i}} \phi$, $\nabla_{t_0} \phi$ and $\nabla_{t_f} \phi$, are obtained as

$$\nabla_{\mathbf{Y}_{:,i}} \phi = \begin{bmatrix} \frac{\partial \phi}{\partial y_i(t_0)} & \mathbf{0}_{1 \times (N-1)} & \frac{\partial \phi}{\partial y_i(t_f)} \end{bmatrix}, \qquad i = 1, \ldots, n_y$$
$$\nabla_{t_0} \phi = \frac{\partial \phi}{\partial t_0}, \qquad \nabla_{t_f} \phi = \frac{\partial \phi}{\partial t_f} \tag{35}$$

Next, $\nabla_{\mathbf{z}} \gamma$ is given as

$$\nabla_{\mathbf{z}} \gamma = \begin{bmatrix} \nabla_{\mathbf{Y}} \gamma & \nabla_{\mathbf{U}} \gamma & \nabla_{t_0} \gamma & \nabla_{t_f} \gamma \end{bmatrix} \tag{36}$$

where

$$\nabla_{\mathbf{Y}} \gamma = \begin{bmatrix} \nabla_{\mathbf{Y}_{:,1}} \gamma & \cdots & \nabla_{\mathbf{Y}_{:,n_y}} \gamma \end{bmatrix}$$
$$\nabla_{\mathbf{U}} \gamma = \begin{bmatrix} \nabla_{\mathbf{U}_{:,1}} \gamma & \cdots & \nabla_{\mathbf{U}_{:,n_u}} \gamma \end{bmatrix} \tag{37}$$

The derivatives, $\nabla_{\mathbf{Y}_{:,i}} \gamma$, $\nabla_{\mathbf{U}_{:,j}} \gamma$, $\nabla_{t_0} \gamma$, and $\nabla_{t_f} \gamma$, are obtained as

$$\nabla_{\mathbf{Y}_{:,i}} \gamma = \begin{bmatrix} \frac{t_f - t_0}{2} \left\{ \mathbf{w} \circ \left[\frac{\partial g}{\partial y_i} \right]_N^1 \right\}^{\mathsf{T}} & 0 \end{bmatrix}, \qquad (i = 1, \ldots, n_y)$$
$$\nabla_{\mathbf{U}_{:,j}} \gamma = \frac{t_f - t_0}{2} \left\{ \mathbf{w} \circ \left[\frac{\partial g}{\partial u_j} \right]_N^1 \right\}^{\mathsf{T}}, \qquad (j = 1, \ldots, n_u)$$
$$\nabla_{t_0} \gamma = -\frac{1}{2} \mathbf{w}^{\mathsf{T}} \mathbf{g} + \frac{t_f - t_0}{2} \mathbf{w}^{\mathsf{T}} \left\{ \alpha \circ \left[\frac{\partial g}{\partial t} \right]_N^1 \right\}$$
$$\nabla_{t_f} \gamma = \frac{1}{2} \mathbf{w}^{\mathsf{T}} \mathbf{g} + \frac{t_f - t_0}{2} \mathbf{w}^{\mathsf{T}} \left\{ \beta \circ \left[\frac{\partial g}{\partial t} \right]_N^1 \right\} \tag{38}$$

It is seen from Eqs. (32–38) that computing the objective function gradient, $\nabla_{\mathbf{z}} f$, requires that the first derivatives of g be determined with respect to the continuous-time state, $\mathbf{y}$, control, $\mathbf{u}$, and time, t, whereas the first derivatives of ϕ be computed with respect to the initial state, $\mathbf{y}(t_0)$, initial time, t_0, final state, $\mathbf{y}(t_f)$, and final time, t_f. Furthermore, these derivatives are computed at either the N collocation points (in the case of g and the derivatives of g) or are computed at the endpoints (in the case of ϕ and the derivatives of ϕ). The NLP objective function and gradient is then assembled using the equations derived in this section.

B. Constraint Jacobian

The Jacobian of the constraints is defined as

$$\nabla_{\mathbf{z}} \mathbf{h} = \begin{bmatrix} \nabla_{\mathbf{z}} \mathbf{\Delta}_{:,1} \\ \vdots \\ \nabla_{\mathbf{z}} \mathbf{\Delta}_{:,n_y} \\ \nabla_{\mathbf{z}} \mathbf{C}_{:,1} \\ \vdots \\ \nabla_{\mathbf{z}} \mathbf{C}_{:,n_c} \\ \nabla_{\mathbf{z}} b_1 \\ \vdots \\ \nabla_{\mathbf{z}} b_{n_b} \end{bmatrix} \tag{39}$$

The first derivatives of the defect constraints are obtained as

$$\nabla_{\mathbf{z}}\mathbf{\Delta}_{:,l} = \begin{bmatrix} \nabla_{\mathbf{Y}}\mathbf{\Delta}_{:,l} & \nabla_{\mathbf{U}}\mathbf{\Delta}_{:,l} & \nabla_{t_0}\mathbf{\Delta}_{:,l} & \nabla_{t_f}\mathbf{\Delta}_{:,l} \end{bmatrix} \quad l = 1, \ldots, n_y \tag{40}$$

where

$$\nabla_{\mathbf{Y}}\mathbf{\Delta}_{:,l} = \begin{bmatrix} \nabla_{\mathbf{Y}_{:,1}}\mathbf{\Delta}_{:,l} & \cdots & \nabla_{\mathbf{Y}_{:,n_y}}\mathbf{\Delta}_{:,l} \end{bmatrix}$$
$$\nabla_{\mathbf{U}}\mathbf{\Delta}_{:,l} = \begin{bmatrix} \nabla_{\mathbf{U}_{:,1}}\mathbf{\Delta}_{:,l} & \cdots & \nabla_{\mathbf{U}_{:,n_u}}\mathbf{\Delta}_{:,l} \end{bmatrix} \quad l = 1, \ldots, n_y \tag{41}$$

The first derivatives, $\nabla_{\mathbf{Y}_{:,i}}\Delta_{:,l}$, $(i,\ l = 1, \ldots, n_y)$, $\nabla_{\mathbf{U}_{:,j}}\Delta_{:,l}$, $(j = 1, \ldots, n_u,\quad l = 1, \ldots, n_y)$, $\nabla_{t_0}\mathbf{\Delta}_{:,l}$, $(l = 1, \ldots, n_y)$, and $\nabla_{t_f}\mathbf{\Delta}_{:,l}$, $(l = 1, \ldots, n_y)$, can be obtained as

$$\nabla_{\mathbf{Y}_{:,i}}\mathbf{\Delta}_{:,l} = \left[\delta_{il}\mathbf{D}_{:,1:N} - \tfrac{t_f - t_0}{2}\operatorname{diag}\left(\left[\tfrac{\partial a_l}{\partial y_i}\right]_N^1\right) \quad \delta\mathbf{D}_{:,N+1} \right]$$
$$\nabla_{\mathbf{U}_{:,j}}\mathbf{\Delta}_{:,l} = -\frac{t_f - t_0}{2}\operatorname{diag}\left(\left[\frac{\partial a_l}{\partial u_j}\right]_N^1\right)$$
$$\nabla_{t_0}\mathbf{\Delta}_{:,l} = \frac{1}{2}[a_l]_N^1 - \frac{t_f - t_0}{2}\alpha \circ \left[\frac{\partial a_l}{\partial t}\right]_N^1$$
$$\nabla_{t_f}\mathbf{\Delta}_{:,l} = -\frac{1}{2}[a_l]_N^1 - \frac{t_f - t_0}{2}\beta \circ \left[\frac{\partial a_l}{\partial t}\right]_N^1 \tag{42}$$

where $(i,\ l = 1, \ldots, n_y)$, and $(j = 1, \ldots, n_u)$. Furthermore, δ_{il} is the Kronecker delta function

$$\delta_{il} = \begin{cases} 1, & i = l \\ 0, & \text{otherwise} \end{cases}$$

The first derivatives of the path constraints are given as

$$\nabla_{\mathbf{z}}\mathbf{C}_{:,p} = \begin{bmatrix} \nabla_{\mathbf{Y}}\mathbf{C}_{:,p} & \nabla_{\mathbf{U}}\mathbf{C}_{:,p} & \nabla_{t_0}\mathbf{C}_{:,p} & \nabla_{t_f}\mathbf{C}_{:,p} \end{bmatrix} \tag{43}$$

where

$$\nabla_{\mathbf{Y}}\mathbf{C}_{:,p} = \begin{bmatrix} \nabla_{\mathbf{Y}_{:,1}}\mathbf{C}_{:,p} & \cdots & \nabla_{\mathbf{Y}_{:,n_y}}\mathbf{C}_{:,p} \end{bmatrix}$$
$$\nabla_{\mathbf{U}}\mathbf{C}_{:,p} = \begin{bmatrix} \nabla_{\mathbf{U}_{:,1}}\mathbf{C}_{:,p} & \cdots & \nabla_{\mathbf{U}_{:,n_u}}\mathbf{C}_{:,p} \end{bmatrix} \quad p = 1, \ldots, n_p \tag{44}$$

The first derivatives, $\nabla_{\mathbf{Y}_{:,i}}\mathbf{C}_{:,p}$, $\nabla_{\mathbf{U}_{:,j}}\mathbf{C}_{:,p}$, $\nabla_{t_0}\mathbf{C}_{:,p}$, and $\nabla_{t_f}\mathbf{C}_{:,p}$, can be found in a sparse manner as

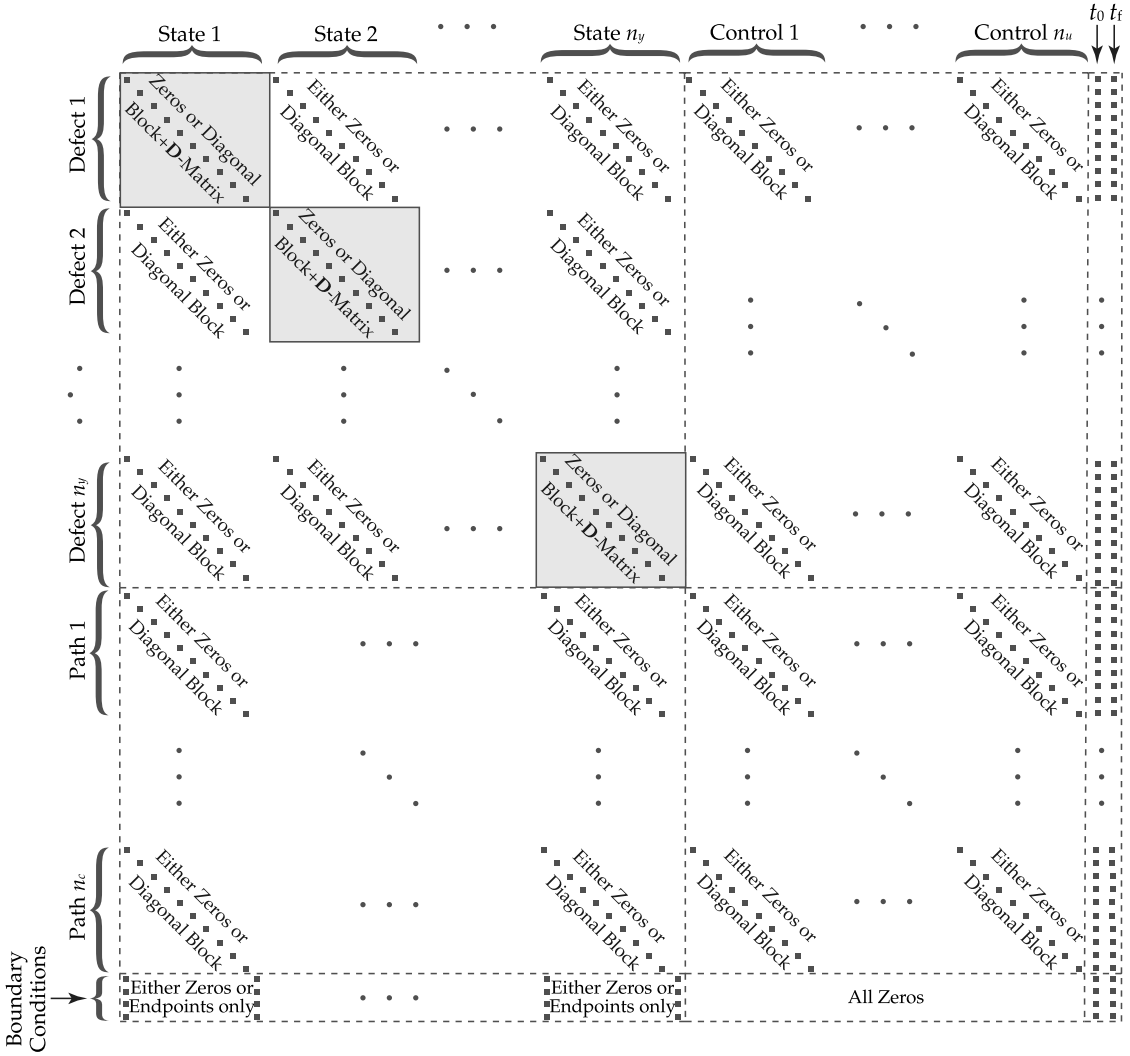


Fig. 2 General Jacobian sparsity pattern for RPM.

$$\nabla_{\mathbf{Y}_{:,i}}\mathbf{C}_{:,p} = \left[\ \mathrm{diag}\left(\left[\frac{\partial c_p}{\partial y_i}\right]_N^1\right) \quad \mathbf{0}_{N\times 1}\ \right]$$
$$\nabla_{\mathbf{U}_{:,j}}\mathbf{C}_{:,p} = \mathrm{diag}\left(\left[\frac{\partial c_p}{\partial u_j}\right]_N^1\right)$$
$$\nabla_{t_0}\mathbf{C}_{:,p} = \alpha \circ \left[\frac{\partial c_p}{\partial t}\right]_N^1$$
$$\nabla_{t_f}\mathbf{C}_{:,p} = \beta \circ \left[\frac{\partial c_p}{\partial t}\right]_N^1 \tag{45}$$

where $(i = 1, \ldots, n_y)$, $(j = 1, \ldots, n_u)$, and $(p = 1, \ldots, n_c)$. The first derivatives of the boundary conditions are given as

$$\nabla_{\mathbf{z}} b_q = [\,\nabla_{\mathbf{Y}} b_q \quad \nabla_{\mathbf{U}} b_q \quad \nabla_{t_0} b_q \quad \nabla_{t_f} b_q\,], \qquad q = 1, \ldots, n_q \tag{46}$$

where

$$\nabla_{\mathbf{Y}} b_q = \left[\ \nabla_{\mathbf{Y}_{:,1}} b_q \quad \cdots \quad \nabla_{\mathbf{Y}_{:,n_y}} b_q\ \right] \qquad \nabla_{\mathbf{U}} b_q = \left[\mathbf{0}_{1\times N n_u}\right]$$
$$q = 1, \ldots, n_q \tag{47}$$

The first derivatives, $\nabla_{\mathbf{Y}_{:,i}} b_q$, $\nabla_{t_0} b_q$, and $\nabla_{t_f} b_q$, can be found in a sparse manner as

$$\nabla_{\mathbf{Y}_{:,i}} b_q = \left[\ \frac{\partial b_q}{\partial y_i(t_0)} \quad \mathbf{0}_{1\times N-1} \quad \frac{\partial b_q}{\partial y_i(t_f)}\ \right]$$
$$\nabla_{t_0} b_q = \frac{\partial b_q}{\partial t_0}, \qquad \nabla_{t_f} b_q = \frac{\partial b_q}{\partial t_f} \tag{48}$$

where $(i = 1, \ldots, n_y)$, and $(q = 1, \ldots, n_b)$. It is seen from Eqs. (39–48) that the NLP constraint Jacobian requires that the first derivatives of $\mathbf{f}$ and $\mathbf{c}$ be determined with respect to the continuous-time state, $\mathbf{y}$, continuous-time control, $\mathbf{u}$, and continuous-time, t, and that the derivatives of $\mathbf{b}$ be computed with respect to the initial state, $\mathbf{y}(t_0)$, the initial time, t_0, the final state, $\mathbf{y}(t_f)$, and the final time, t_f. Furthermore, these derivatives are computed at either the N collocation points (in the case of the derivatives of $\mathbf{f}$ and $\mathbf{c}$) or are computed at the endpoints (in the case of $\mathbf{b}$). The NLP constraint Jacobian is then assembled using the equations derived in this section. The sparsity pattern for a general Radau pseudospectral NLP constraint Jacobian is shown in Fig. 2.

C. Lagrangian Hessian

The Lagrangian of the NLP given in Eqs. (28) and (29) is defined as

$$\mathcal{L} = \sigma f(\mathbf{z}) + \Lambda^{\mathsf{T}}\mathbf{h}(\mathbf{z}) \tag{49}$$

where $\sigma \in \mathbb{R}$, and $\mathbf{\Lambda} \in \mathbb{R}^{N(n_y+n_c)+n_b}$ is a vector of Lagrange multipliers. The vector $\mathbf{\Lambda}$ is given as

$$\mathbf{\Lambda} = \begin{bmatrix} \Gamma_{:,1} \\ \vdots \\ \Gamma_{:,n_y} \\ \mathbf{\Psi}_{:,1} \\ \vdots \\ \mathbf{\Psi}_{:,n_c} \\ \nu \end{bmatrix} \tag{50}$$

where $\Gamma_{i,j}, (i = 1, \ldots, N, j = 1, \ldots, n_y)$ are the Lagrange multipliers associated with the defect constraints of Eq. (24), $\mathbf{\Psi}_{i,j}, (i = 1, \ldots, N, i = 1, \ldots, n_c)$ are the Lagrange multipliers associated with the path constraints of Eq. (25), and $\nu_i, (i = 1, \ldots, n_b)$ are the Lagrange multipliers associated with the boundary conditions of Eq. (26). The Lagrangian can then be represented as

$$\mathcal{L} = \sigma\phi + \sigma\gamma + \sum_{i=1}^{n_y} \Gamma_{:,i}^{\mathsf{T}}\Delta_{:,i} + \sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p}^{\mathsf{T}}\mathbf{C}_{:,p} + \sum_{q=1}^{n_b} \nu_q b_q \tag{51}$$

For convenience in the discussion that follows, the Hessian of the Lagrangian will be decomposed into two parts as

$$\nabla^2_{\mathbf{zz}}\mathcal{L} = \nabla^2_{\mathbf{zz}}\mathcal{L}_E + \nabla^2_{\mathbf{zz}}\mathcal{L}_I \tag{52}$$

where $\mathcal{L}_E$ represents those parts of the Lagrangian that are functions of the endpoints functions ϕ and $\mathbf{b}$,

$$\mathcal{L}_E = \sigma\phi + \sum_{q=1}^{n_b} \nu_q b_q \tag{53}$$

whereas $\mathcal{L}_I$ represents those parts of the Lagrangian that are functions of collocation point functions, γ, $\mathbf{\Delta}$, and $\mathbf{C}$,

$$\mathcal{L}_I = \sigma\gamma + \sum_{i=1}^{n_y} \Gamma_{:,i}^{\mathsf{T}}\mathbf{\Delta}_{:,i} + \sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p}^{\mathsf{T}}\mathbf{C}_{:,p} \tag{54}$$

In the next subsections, we describe the second derivatives of the functions $\mathcal{L}_E$ and $\mathcal{L}_I$. It is noted that the Hessian is symmetric; thus, only the lower-triangular portion of $\nabla^2_{\mathbf{zz}}\mathcal{L}_E$ and $\nabla^2_{\mathbf{zz}}\mathcal{L}_I$ are computed.

1. *Hessian of Endpoint Function $\mathcal{L}_E$*

The Hessian of $\mathcal{L}_E$ with respect to the decision variable vector, $\mathbf{z}$, denoted $\nabla^2_{\mathbf{zz}}\mathcal{L}_E$, is defined as

$$\nabla^2_{\mathbf{zz}}\mathcal{L}_E = \begin{bmatrix} \nabla^2_{\mathbf{YY}}\mathcal{L}_E & (\nabla^2_{\mathbf{UY}}\mathcal{L}_E)^{\mathsf{T}} & (\nabla^2_{t_0\mathbf{Y}}\mathcal{L}_E)^{\mathsf{T}} & (\nabla^2_{t_f\mathbf{Y}}\mathcal{L}_E)^{\mathsf{T}} \\ \nabla^2_{\mathbf{UY}}\mathcal{L}_E & \nabla^2_{\mathbf{UU}}\mathcal{L}_E & (\nabla^2_{t_0\mathbf{U}}\mathcal{L}_E)^{\mathsf{T}} & (\nabla^2_{t_f\mathbf{U}}\mathcal{L}_E)^{\mathsf{T}} \\ \nabla^2_{t_0\mathbf{Y}}\mathcal{L}_E & \nabla^2_{t_0\mathbf{U}}\mathcal{L}_E & \nabla^2_{t_0t_0}\mathcal{L}_E & (\nabla^2_{t_ft_0}\mathcal{L}_E)^{\mathsf{T}} \\ \nabla^2_{t_f\mathbf{Y}}\mathcal{L}_E & \nabla^2_{t_f\mathbf{U}}\mathcal{L}_E & \nabla^2_{t_ft_0}\mathcal{L}_E & \nabla^2_{t_ft_f}\mathcal{L}_E \end{bmatrix} \tag{55}$$

where the blocks of $\nabla^2_{\mathbf{zz}}\mathcal{L}_E$ are defined as

$$\nabla^2_{\mathbf{YY}}\mathcal{L}_E = \begin{bmatrix} \nabla^2_{\mathbf{Y}_{:,1}\mathbf{Y}_{:,1}}\mathcal{L}_E & (\nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,1}}\mathcal{L}_E)^{\mathsf{T}} & \cdots & (\nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,1}}\mathcal{L}_E)^{\mathsf{T}} \\ \nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,1}}\mathcal{L}_E & \nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,2}}\mathcal{L}_E & \cdots & (\nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,2}}\mathcal{L}_E)^{\mathsf{T}} \\ \vdots & \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,1}}\mathcal{L}_E & \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,2}}\mathcal{L}_E & \cdots & \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,n_y}}\mathcal{L}_E \end{bmatrix}$$

$$\nabla^2_{\mathbf{UY}}\mathcal{L}_E = \begin{bmatrix} \nabla^2_{\mathbf{U}_{:,1}\mathbf{Y}_{:,1}}\mathcal{L}_E & \cdots & \nabla^2_{\mathbf{U}_{:,1}\mathbf{Y}_{:,n_y}}\mathcal{L}_E \\ \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{Y}_{:,1}}\mathcal{L}_E & \cdots & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{Y}_{:,n_y}}\mathcal{L}_E \end{bmatrix} = \mathbf{0}_{Nn_u\times(N+1)n_y}$$

$$\nabla^2_{\mathbf{UU}}\mathcal{L}_E = \begin{bmatrix} \nabla^2_{\mathbf{U}_{:,1}\mathbf{U}_{:,1}}\mathcal{L}_E & (\nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,1}}\mathcal{L}_E)^{\mathsf{T}} & \cdots & (\nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,1}}\mathcal{L}_E)^{\mathsf{T}} \\ \nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,1}}\mathcal{L}_E & \nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,2}}\mathcal{L}_E & \cdots & (\nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,2}}\mathcal{L}_E)^{\mathsf{T}} \\ \vdots & \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,1}}\mathcal{L}_E & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,2}}\mathcal{L}_E & \cdots & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,n_u}}\mathcal{L}_E \end{bmatrix}$$
$$= \mathbf{0}_{Nn_u\times Nn_u}$$

$$\nabla^2_{t_0\mathbf{Y}}\mathcal{L}_E = \left[\ \nabla^2_{t_0\mathbf{Y}_{:,1}}\mathcal{L}_E \quad \cdots \quad \nabla^2_{t_0\mathbf{Y}_{:,n_y}}\mathcal{L}_E\ \right]$$

$$\nabla^2_{t_0\mathbf{U}}\mathcal{L}_E = \left[\ \nabla^2_{t_0\mathbf{U}_{:,1}}\mathcal{L}_E \quad \cdots \quad \nabla^2_{t_0\mathbf{U}_{:,n_u}}\mathcal{L}_E\ \right] = \mathbf{0}_{1\times Nn_u}$$

$$\nabla^2_{t_f\mathbf{Y}}\mathcal{L}_E = \left[\ \nabla^2_{t_f\mathbf{Y}_{:,1}}\mathcal{L}_E \quad \cdots \quad \nabla^2_{t_f\mathbf{Y}_{:,n_y}}\mathcal{L}_E\ \right]$$

$$\nabla^2_{t_f\mathbf{U}}\mathcal{L}_E = \left[\ \nabla^2_{t_f\mathbf{U}_{:,1}}\mathcal{L}_E \quad \cdots \quad \nabla^2_{t_f\mathbf{U}_{:,n_u}}\mathcal{L}_E\ \right] = \mathbf{0}_{1\times Nn_u}$$

The matrices, $\nabla^2_{\mathbf{Y}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_E$, $\nabla^2_{t_0\mathbf{Y}_{:,j}}\mathcal{L}_E$, $\nabla^2_{t_0t_0}\mathcal{L}_E$, $\nabla^2_{t_f\mathbf{Y}_{:,j}}\mathcal{L}_E$, $\nabla^2_{t_ft_0}\mathcal{L}_E$, and $\nabla^2_{t_ft_f}\mathcal{L}_E$, are obtained in a sparse manner as

$$\nabla^2_{\mathbf{Y}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_E = \begin{bmatrix} \frac{\partial^2\mathcal{L}_E}{\partial y_i(t_0)\partial y_j(t_0)} & \mathbf{0}_{1\times N-1} & \frac{\partial^2\mathcal{L}_E}{\partial y_i(t_0)\partial y_j(t_f)} \\ \mathbf{0}_{N-1\times 1} & \mathbf{0}_{N-1\times N-1} & \mathbf{0}_{N-1\times 1} \\ \frac{\partial^2\mathcal{L}_E}{\partial y_i(t_f)\partial y_j(t_0)} & \mathbf{0}_{1\times N-1} & \frac{\partial^2\mathcal{L}_E}{\partial y_i(t_f)\partial y_j(t_f)} \end{bmatrix}$$

$$(i = 1, \ldots, n_y), \qquad (j = 1, \ldots, i)$$

$$\nabla^2_{t_0\mathbf{Y}_{:,j}}\mathcal{L}_E = \begin{bmatrix} \frac{\partial^2\mathcal{L}_E}{\partial t_0\partial y_j(t_0)} & \mathbf{0}_{1\times N-1} & \frac{\partial^2\mathcal{L}_E}{\partial t_0\partial y_j(t_f)} \end{bmatrix}, \qquad (j = 1, \ldots, n_y)$$

$$\nabla^2_{t_0t_0}\mathcal{L}_E = \frac{\partial^2\mathcal{L}_E}{\partial t_0^2}, \qquad \nabla^2_{t_f\mathbf{Y}_{:,j}}\mathcal{L}_E = \begin{bmatrix} \frac{\partial^2\mathcal{L}_E}{\partial t_f\partial y_j(t_0)} & \mathbf{0}_{1\times N-1} & \frac{\partial^2\mathcal{L}_E}{\partial t_f\partial y_j(t_f)} \end{bmatrix}$$

$$(j = 1, \ldots, n_y), \qquad \nabla^2_{t_ft_0}\mathcal{L}_E = \frac{\partial^2\mathcal{L}_E}{\partial t_f\partial t_0}, \qquad \nabla^2_{t_ft_f}\mathcal{L}_E = \frac{\partial^2\mathcal{L}_E}{\partial t_f^2} \tag{56}$$

where we recall that $\mathcal{L}_E$ is itself a function of the Mayer cost, ϕ, and the boundary condition function, $\mathbf{b}$. Because ϕ and $\mathbf{b}$ are functions of the continuous-time Bolza optimal control problem, the Hessian, $\nabla^2_{\mathbf{zz}}\mathcal{L}_E$, with respect to the NLP decision vector, $\mathbf{z}$, can itself be obtained by differentiating the functions of the continuous-time optimal control problem and assembling these derivatives into the correct locations of the NLP Lagrangian Hessian.

2. Hessian of Collocation Point Function $\mathcal{L}_I$

The Hessian $\nabla^2_{\mathbf{zz}}\mathcal{L}_I$ is defined as

$$\nabla^2_{\mathbf{zz}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{\mathbf{YY}}\mathcal{L}_I & (\nabla^2_{\mathbf{UY}}\mathcal{L}_I)^\mathsf{T} & (\nabla^2_{t_0\mathbf{Y}}\mathcal{L}_I)^\mathsf{T} & (\nabla^2_{t_f\mathbf{Y}}\mathcal{L}_I)^\mathsf{T} \\ \nabla^2_{\mathbf{UY}}\mathcal{L}_I & \nabla^2_{\mathbf{UU}}\mathcal{L}_I & (\nabla^2_{t_0\mathbf{U}}\mathcal{L}_I)^\mathsf{T} & (\nabla^2_{t_f\mathbf{U}}\mathcal{L}_I)^\mathsf{T} \\ \nabla^2_{t_0\mathbf{Y}}\mathcal{L}_I & \nabla^2_{t_0\mathbf{U}}\mathcal{L}_I & \nabla^2_{t_0t_0}\mathcal{L}_I & (\nabla^2_{t_ft_0}\mathcal{L}_I)^\mathsf{T} \\ \nabla^2_{t_f\mathbf{Y}}\mathcal{L}_I & \nabla^2_{t_f\mathbf{U}}\mathcal{L}_I & \nabla^2_{t_ft_0}\mathcal{L}_I & \nabla^2_{t_ft_f}\mathcal{L}_I \end{bmatrix} \tag{57}$$

where the blocks of $\nabla^2_{\mathbf{zz}}\mathcal{L}_I$ are given as

$$\nabla^2_{\mathbf{YY}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{\mathbf{Y}_{:,1}\mathbf{Y}_{:,1}}\mathcal{L}_I & (\nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,1}}\mathcal{L}_I)^\mathsf{T} & \cdots & (\nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,1}}\mathcal{L}_I)^\mathsf{T} \\ \nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,1}}\mathcal{L}_I & \nabla^2_{\mathbf{Y}_{:,2}\mathbf{Y}_{:,2}}\mathcal{L}_I & \cdots & (\nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,2}}\mathcal{L}_I)^\mathsf{T} \\ \vdots & \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,1}}\mathcal{L}_I & \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,2}}\mathcal{L}_I & \cdots & \nabla^2_{\mathbf{Y}_{:,n_y}\mathbf{Y}_{:,n_y}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{\mathbf{UY}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{\mathbf{U}_{:,1}\mathbf{Y}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{\mathbf{U}_{:,1}\mathbf{Y}_{:,n_y}}\mathcal{L}_I \\ \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{Y}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{Y}_{:,n_y}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{\mathbf{UU}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{\mathbf{U}_{:,1}\mathbf{U}_{:,1}}\mathcal{L}_I & (\nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,1}}\mathcal{L}_I)^\mathsf{T} & \cdots & (\nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,1}}\mathcal{L}_I)^\mathsf{T} \\ \nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,1}}\mathcal{L}_I & \nabla^2_{\mathbf{U}_{:,2}\mathbf{U}_{:,2}}\mathcal{L}_I & \cdots & (\nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,2}}\mathcal{L}_I)^\mathsf{T} \\ \vdots & \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,1}}\mathcal{L}_I & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,2}}\mathcal{L}_I & \cdots & \nabla^2_{\mathbf{U}_{:,n_u}\mathbf{U}_{:,n_u}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{t_0\mathbf{Y}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{t_0\mathbf{Y}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{t_0\mathbf{Y}_{:,n_y}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{t_0\mathbf{U}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{t_0\mathbf{U}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{t_0\mathbf{U}_{:,n_u}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{t_f\mathbf{Y}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{t_f\mathbf{Y}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{t_f\mathbf{Y}_{:,n_y}}\mathcal{L}_I \end{bmatrix}$$

$$\nabla^2_{t_f\mathbf{U}}\mathcal{L}_I = \begin{bmatrix} \nabla^2_{t_f\mathbf{U}_{:,1}}\mathcal{L}_I & \cdots & \nabla^2_{t_f\mathbf{U}_{:,n_u}}\mathcal{L}_I \end{bmatrix}$$

The matrices, $\nabla^2_{\mathbf{Y}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_I$, $\nabla^2_{\mathbf{U}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_I$, $\nabla^2_{\mathbf{U}_{:,i}\mathbf{U}_{:,j}}\mathcal{L}_I$, $\nabla^2_{t_0\mathbf{Y}_{:,j}}\mathcal{L}_I$, $\nabla^2_{t_0\mathbf{U}_{:,j}}\mathcal{L}_I$, $\nabla^2_{t_0t_0}\mathcal{L}_I$, $\nabla^2_{t_f\mathbf{Y}_{:,j}}\mathcal{L}_I$, $\nabla^2_{t_f\mathbf{U}_{:,j}}\mathcal{L}_I$, $\nabla^2_{t_ft_0}\mathcal{L}_I$, and $\nabla^2_{t_ft_f}\mathcal{L}_I$, are obtained in a sparse manner as

$$\nabla^2_{\mathbf{Y}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_I = \begin{bmatrix} \operatorname{diag}\left(\left[\frac{\partial^2\mathcal{L}_I}{\partial y_i\partial y_j}\right]^1_N\right) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 0 \end{bmatrix}$$

$$(i = 1, \ldots, n_y, j = 1, \ldots, i)$$

$$\nabla^2_{\mathbf{U}_{:,i}\mathbf{Y}_{:,j}}\mathcal{L}_I = \begin{bmatrix} \operatorname{diag}\left(\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial y_j}\right]^1_N\right) & \mathbf{0}_{N\times 1} \end{bmatrix}$$

$$(i = 1, \ldots, n_u, j = 1, \ldots, n_y)$$

$$\nabla^2_{\mathbf{U}_{:,i}\mathbf{U}_{:,j}}\mathcal{L}_I = \operatorname{diag}\left(\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial u_j}\right]^1_N\right)$$

$$(i = 1, \ldots, n_u, j = 1, \ldots, i)$$

$$\nabla^2_{t_0\mathbf{Y}_{:,j}}\mathcal{L}_I = \begin{bmatrix} \left\{\left[\frac{\partial^2\mathcal{L}_I}{\partial t_0\partial y_j}\right]^1_N\right\}^\mathsf{T} & 0 \end{bmatrix}$$

$$(j = 1, \ldots, n_y), \nabla^2_{t_0\mathbf{U}_{:,j}}\mathcal{L}_I = \left\{\left[\frac{\partial^2\mathcal{L}_I}{\partial t_0\partial u_j}\right]^1_N\right\}^\mathsf{T}$$

$$(j = 1, \ldots, n_u), \qquad \nabla^2_{t_0t_0}\mathcal{L}_I = \frac{\partial^2\mathcal{L}_I}{\partial t_0^2}$$

$$\nabla^2_{t_f\mathbf{Y}_{:,j}}\mathcal{L}_I = \begin{bmatrix} \left\{\left[\frac{\partial^2\mathcal{L}_I}{\partial t_f\partial y_j}\right]^1_N\right\}^\mathsf{T}\mathbf{0} \end{bmatrix} \qquad (j = 1, \ldots, n_y)$$

$$\nabla^2_{t_f\mathbf{U}_{:,j}}\mathcal{L}_I = \left\{\left[\frac{\partial^2\mathcal{L}_I}{\partial t_f\partial u_j}\right]^1_N\right\}^\mathsf{T}, \qquad (j = 1, \ldots, n_u)$$

$$\nabla^2_{t_ft_0}\mathcal{L}_I = \frac{\partial^2\mathcal{L}_I}{\partial t_f\partial t_0}, \qquad \nabla^2_{t_ft_f}\mathcal{L}_I = \frac{\partial^2\mathcal{L}_I}{\partial t_f^2} \tag{58}$$

It is seen that the derivatives given in Eq. (58) are functions of the derivatives of $\mathcal{L}_I$ with respect to the components of the continuous-time state, $\mathbf{y}(t)$, the components of the continuous-time control, $\mathbf{u}(t)$, the initial time, t_0, and the final time, t_f. The derivatives, $\left[\frac{\partial^2\mathcal{L}_I}{\partial y_i\partial y_j}\right]^1_N$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial y_j}\right]^1_N$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial u_j}\right]^1_N$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial t_0\partial y_j}\right]^1_N$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial t_0\partial u_j}\right]^1_N$, $\frac{\partial^2\mathcal{L}_I}{\partial t_0^2}$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial t_f\partial y_j}\right]^1_N$, $\left[\frac{\partial^2\mathcal{L}_I}{\partial t_f\partial u_j}\right]^1_N$, $\frac{\partial^2\mathcal{L}_I}{\partial t_f\partial t_0}$, and $\frac{\partial^2\mathcal{L}_I}{\partial t_f^2}$, are given, respectively, as

$$\left[\frac{\partial^2\mathcal{L}_I}{\partial y_i\partial y_j}\right]^1_N = \frac{t_f - t_0}{2}\left\{\sigma\mathbf{w}\circ\left[\frac{\partial^2 g}{\partial y_i\partial y_j}\right]^1_N - \sum_{l=1}^{n_y}\Gamma_{:,l}\circ\left[\frac{\partial^2 a_l}{\partial y_i\partial y_j}\right]^1_N\right\} + \sum_{p=1}^{n_c}\mathbf{\Psi}_{:,p}\circ\left[\frac{\partial^2 c_p}{\partial y_i\partial y_j}\right]^1_N, \qquad (i, j = 1, \ldots, n_y) \tag{59}$$

$$\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial y_j}\right]^1_N = \frac{t_f - t_0}{2}\left\{\sigma\mathbf{w}\circ\left[\frac{\partial^2 g}{\partial u_i\partial y_j}\right]^1_N - \sum_{l=1}^{n_y}\Gamma_{:,l}\circ\left[\frac{\partial^2 a_l}{\partial u_i\partial y_j}\right]^1_N\right\} + \sum_{p=1}^{n_c}\mathbf{\Psi}_{:,p}\circ\left[\frac{\partial^2 c_p}{\partial u_i\partial y_j}\right]^1_N, \qquad (i = 1, \ldots, n_u, j = 1, \ldots, n_y) \tag{60}$$

$$\left[\frac{\partial^2\mathcal{L}_I}{\partial u_i\partial u_j}\right]^1_N = \frac{t_f - t_0}{2}\left\{\sigma\mathbf{w}\circ\left[\frac{\partial^2 g}{\partial u_i\partial u_j}\right]^1_N - \sum_{l=1}^{n_y}\Gamma_{:,l}\circ\left[\frac{\partial^2 a_l}{\partial u_i\partial u_j}\right]^1_N\right\} + \sum_{p=1}^{n_c}\mathbf{\Psi}_{:,p}\circ\left[\frac{\partial^2 c_p}{\partial u_i\partial u_j}\right]^1_N, \qquad (i, j = 1, \ldots, n_u) \tag{61}$$

$$\left[\frac{\partial^2\mathcal{L}_I}{\partial t_0\partial y_j}\right]^1_N = \frac{1}{2}\left\{\sum_{l=1}^{n_y}\Gamma_{:,l}\circ\left[\frac{\partial a_l}{\partial y_j}\right]^1_N - \sigma\mathbf{w}\circ\left[\frac{\partial g}{\partial y_j}\right]^1_N\right\} + \frac{t_f - t_0}{2}\alpha\circ\left\{\sigma\mathbf{w}\circ\left[\frac{\partial^2 g}{\partial t\partial y_j}\right]^1_N - \sum_{l=1}^{n_y}\Gamma_{:,l}\circ\left[\frac{\partial^2 a_l}{\partial t\partial y_j}\right]^1_N\right\} + \alpha\circ\left\{\sum_{p=1}^{n_c}\mathbf{\Psi}_{:,p}\circ\left[\frac{\partial^2 c_p}{\partial t\partial y_j}\right]^1_N\right\}, \qquad (j = 1, \ldots, n_y) \tag{62}$$

$$\left[\frac{\partial^2 \mathcal{L}_I}{\partial t_0 \partial u_j}\right]^1_N = \frac{1}{2}\left\{\sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial u_j}\right]^1_N - \sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial u_j}\right]^1_N\right\} + \frac{t_f - t_0}{2}\alpha \circ \left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t \partial u_j}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t \partial u_j}\right]^1_N\right\} + \alpha \circ \left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial u_j}\right]^1_N\right\}, \qquad (j = 1, \ldots, n_u) \tag{63}$$

$$\frac{\partial^2 \mathcal{L}_I}{\partial t_0^2} = \alpha^{\mathsf{T}}\left\{\sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial t}\right]^1_N - \sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial t}\right]^1_N\right\} + \frac{t_f - t_0}{2}\alpha^{\mathsf{T}}\left[\left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t^2}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t^2}\right]^1_N\right\} \circ \alpha\right] + \alpha^{\mathsf{T}}\left[\left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial y_j}\right]^1_N\right\} \circ \alpha\right] \tag{64}$$

$$\left[\frac{\partial^2 \mathcal{L}_I}{\partial t_f \partial y_j}\right]^1_N = \frac{1}{2}\left\{\sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial y_j}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial y_j}\right]^1_N\right\} + \frac{t_f - t_0}{2}\beta \circ \left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t \partial y_j}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t \partial y_j}\right]^1_N\right\}\beta \circ \left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial y_j}\right]^1_N\right\}, \qquad (j = 1, \ldots, n_y) \tag{65}$$

$$\left[\frac{\partial^2 \mathcal{L}_I}{\partial t_f \partial u_j}\right]^1_N = \frac{1}{2}\left\{\sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial u_j}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial u_j}\right]^1_N\right\} + \frac{t_f - t_0}{2}\beta \circ \left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t \partial u_j}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t \partial u_j}\right]^1_N\right\} + \beta \circ \left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial u_j}\right]^1_N\right\}, \qquad (j = 1, \ldots, n_u) \tag{66}$$

$$\frac{\partial^2 \mathcal{L}_I}{\partial t_f \partial t_0} = \frac{1}{2}\alpha^{\mathsf{T}}\left\{\sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial t}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial t}\right]^1_N\right\} + \frac{1}{2}\beta^{\mathsf{T}}\left\{\sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial t}\right]^1_N - \sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial t}\right]^1_N\right\} + \frac{t_f - t_0}{2}\alpha^{\mathsf{T}}\left[\left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t^2}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t^2}\right]^1_N\right\} \circ \beta\right] + \alpha^{\mathsf{T}}\left[\left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial y_j}\right]^1_N\right\} \circ \beta\right] \tag{67}$$

and

$$\frac{\partial^2 \mathcal{L}_I}{\partial t_f^2} = \beta^{\mathsf{T}}\left\{\sigma \mathbf{w} \circ \left[\frac{\partial g}{\partial t}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial a_l}{\partial t}\right]^1_N\right\} + \frac{t_f - t_0}{2}\beta^{\mathsf{T}}\left[\left\{\sigma \mathbf{w} \circ \left[\frac{\partial^2 g}{\partial t^2}\right]^1_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[\frac{\partial^2 a_l}{\partial t^2}\right]^1_N\right\} \circ \beta\right] + \beta^{\mathsf{T}}\left[\left\{\sum_{p=1}^{n_c} \mathbf{\Psi}_{:,p} \circ \left[\frac{\partial^2 c_p}{\partial t \partial y_j}\right]^1_N\right\} \circ \beta\right] \tag{68}$$

It is seen from the preceding derivation that the Hessian of $\mathcal{L}_I$, with respect to the NLP decision vector $\mathbf{z}$, is a function of the first and second derivatives of the functions g and $\mathbf{a}$, and the second derivatives of the function $\mathbf{c}$, where g, $\mathbf{a}$, and $\mathbf{c}$ are defined in section III. Thus, the Hessian of $\mathcal{L}_I$ can be obtained as a function of derivatives associated with the functions of the Bolza optimal control problem as stated in the section III. Figure 3 shows the sparsity pattern of a general NLP Lagrangian Hessian obtained from the discretization of the continuous-time Bolza problem using the RPM.

VI. Discussion

Although perhaps not evident at first glance, the approach of section V only requires differentiation of the much smaller and simpler functions of the continuous-time Bolza optimal control problem of section III, as opposed to differentiation of the much larger and more complicated objective and constraint functions of the NLP. For example, using our approach, the NLP constraint Jacobian of section V.B is obtained using Eqs. (42), (45), and (48), where the first derivatives of the defect constraints and path constraints are evaluated at the N collocation points, whereas the derivatives of the boundary condition function are evaluated at the endpoints of the interval. Thus, the Jacobian is obtained by evaluating only the functions of the continuous-time Bolza optimal control problem, as opposed to differentiating the much larger and more complicated objective and constraint functions of the NLP. The simplicity of the approach developed in this paper over differentiating the NLP is particularly evident when computing the Lagrangian Hessian of section V.C. Specifically, from Eqs. (56) and (58), it is seen that the Hessian is obtained by differentiating the functions, $\mathcal{L}_I$ and $\mathcal{L}_E$, with respect to the continuous-time state, control, and time at either the endpoints (in the case $\mathcal{L}_E$) or the N collocation points (in the case of $\mathcal{L}_I$). Furthermore, because $\mathcal{L}_E$ and $\mathcal{L}_I$ are scalar functions, a variety of differentiation techniques can be used in an efficient and easy-to-understand manner. Effectively, the NLP objective function gradient, constraint Jacobian, and Lagrangian Hessian are obtained by differentiating a subset of simpler and smaller functions. Because the derivatives of these simpler and smaller functions are evaluated at only the collocation points or the endpoints of the time interval, the expressions derived in section V provide the most efficient way to compute the NLP derivative functions.

VII. Example

Consider the following variation of the orbit-raising optimal control problem taken from Bryson and Ho [33]. Minimize the cost functional

$$J = -r(t_f) \tag{69}$$

subject to the dynamic constraints

$$\dot{r} = v_r, \qquad \dot{\theta} = v_\theta / r \qquad \dot{v}_r = v_\theta^2 / r - \mu / r^2 + a u_1$$
$$\dot{v}_\theta = -v_r v_\theta / r + a u_2 \tag{70}$$

the equality path constraint

$$c = u_1^2 + u_2^2 = 1 \tag{71}$$

and the boundary conditions

$$b_1 = r(0) - 1 = 0, \qquad b_2 = \theta(0) = 0$$
$$b_3 = v_r(0) = 0, \qquad b_4 = v_\theta(0) - 1 = 0$$
$$b_5 = v_r(t_f) = 0, \qquad b_6 = \sqrt{\mu / r(t_f)} - v_\theta(t_f) = 0 \tag{72}$$

where $\mu = 1$, $T = 0.1405$, $m_0 = 1$, $\dot{m} = 0.0749$, $t_f = 3.32$, and

$$\equiv a(t) = \frac{T}{m_0 - |\dot{m}| t} \tag{73}$$

In this example, the continuous-time state and control are given, respectively, as

$$\mathbf{y}(t) = [\, r(t) \quad \theta(t) \quad v_t(t) \quad v_\theta(t) \,], \qquad \mathbf{u}(t) = [\, u_1(t) \quad u_2(t) \,]$$

whereas the right-hand side function of the dynamics, the path constraint function, and the boundary condition function are given, respectively, as

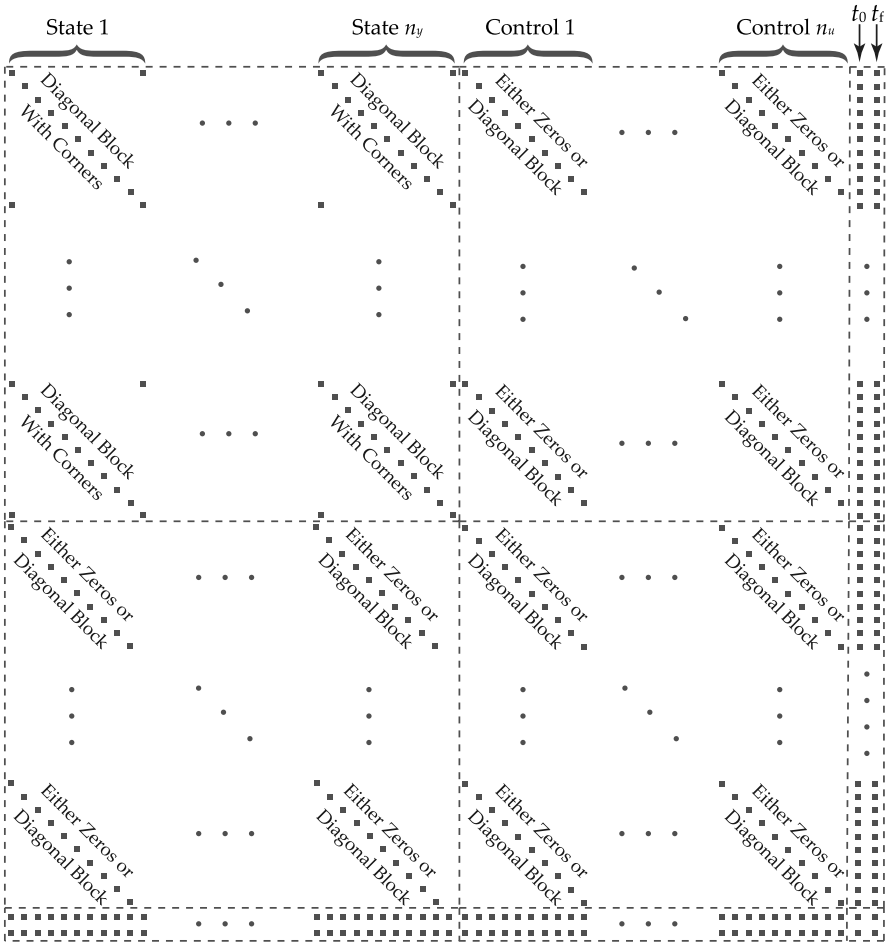


Fig. 3 General Hessian sparsity pattern for RPM.

$$\mathbf{a}(\mathbf{y}(t),\mathbf{u}(t),t)$$
$$= [\,v_r \quad v_\theta/r \quad v_\theta^2/r - \mu/r^2 + au_1 \quad -v_r v_\theta/r + au_2\,]$$
$$\mathbf{c}(\mathbf{y}(t),\mathbf{u}(t),t) = u_1^2 + u_2^2 - 1$$
$$\mathbf{b}(\mathbf{y}(t_0),t_0,\mathbf{y}(t_f),t_f)$$
$$= [\,r(0) - 1 \quad \theta(0) \quad v_r(0) \quad v_\theta(0) - 1 \quad v_r(t_f) \quad \sqrt{\mu/r(t_f)} - v_\theta(t_f)\,]$$

Finally, the lower and upper bounds on the path constraints and boundary conditions are all zero. Because the first five boundary conditions, $(b_1,\ldots,b_5)$, are simple bounds on the initial and final continuous-time state, they will be enforced in the NLP as simple bounds on the NLP variables corresponding to the initial and terminal state. The sixth boundary condition, b_6, on the other hand, is a nonlinear function of the terminal state and, thus, will be enforced in the NLP as a nonlinear constraint.

The NLP arising from the Radau pseudospectral discretization of the optimal control problem given in Eqs. (69–72) was solved using NLP solver IPOPT [29]. It is noted that IPOPT can be used as either a first-derivative NLP solver (where the objective function gradient and constraint Jacobian are supplied) or can be used as a second-derivative NLP solver (where the objective function gradient, constraint Jacobian, and Lagrangian Hessian are supplied). When used as a first-derivative, quasi-Newton NLP solver, IPOPT approximates the Lagrangian Hessian using a limited-memory BFGS update. When used as a second-derivative NLP solver, the lower-triangular portion of the sparse Lagrangian Hessian is used. It is noted that the computational efficiency and reliability of IPOPT are enhanced by providing an accurate, sparse, and efficiently-computed Lagrangian Hessian.

To see the effectiveness of the derivation presented in section V, in this example the Radau pseudospectral NLP was solved using IPOPT by either directly differentiating the NLP objective function, f, the constraints, $\mathbf{h}$, and the Lagrangian, $\mathcal{L}$, or by differentiating the functions, ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$, of the continuous-time Bolza optimal control problem as given in Eqs. (1–4), respectively, and using the method derived in section V. When the NLP functions are directly differentiated and IPOPT is applied as a first-derivative NLP, the first derivatives of f and $\mathbf{h}$ are computed using either1) first forward-differencing or 2) the forward-mode object-oriented MATLAB automatic differentiator INTLAB [34]. When the NLP functions are directly differentiated and IPOPT is applied as a second-derivative NLP solver, the first derivatives of f and $\mathbf{h}$ and the second derivatives of $\mathcal{L}$ are computed using either 3) method 1 plus a second forward-difference to approximate the Hessian of $\mathcal{L}$ or 4) method 2 plus the forward-mode, object-oriented MATLAB automatic differentiator INTLAB [34] to compute the Hessian of $\mathcal{L}$. When the Bolza optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ are differentiated and IPOPT is used as a first-derivative NLP solver, the first derivatives of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ are computed using either 5) first forward-differencing of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ or 6) analytic differentiation of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$. When the Bolza optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ are differentiated and IPOPT is used as a second-derivative NLP solver, the first and second derivatives of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$ are computed using either 7) the method of 5 plus second forward-differencing to approximate the second derivatives of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$; or 8) analytic differentiation to obtain the second derivatives of ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$.

Table 1 summarizes the different derivative methods 1–8 and the usages of IPOPT for this example, whereas the Jacobian and Hessian sparsity patterns for this example are shown, respectively, in Figs. 4 and 5. When using finite differencing or INTLAB to differentiate the NLP constraint function, only the nonlinear parts were differentiated; all known linear parts of the NLP constraint function were obtained a priori and stored for later use. When implementing the mapping derived in section V, only the functions of the continuous-time Bolza

Table 1 Summary of different derivative methods used to solve example with the NLP solver IPOPT

Method used	IPOPT mode	Functions being differentiated	Derivative approximation method
1	First derivative	NLP functions f and $\mathbf{h}$	Finite differencing
2	First derivative	NLP functions f and $\mathbf{h}$	Automatic differentiation
3	Second derivative	NLP functions f and $\mathbf{h}$	Finite differencing
4	Second derivative	NLP functions f and $\mathbf{h}$	Automatic differentiation
5	First derivative	Optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$	Finite differencing
6	First derivative	Optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$	Analytic derivatives
7	Second derivative	Optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$	Finite differencing
8	Second derivative	Optimal control functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{b}$	Analytic derivatives

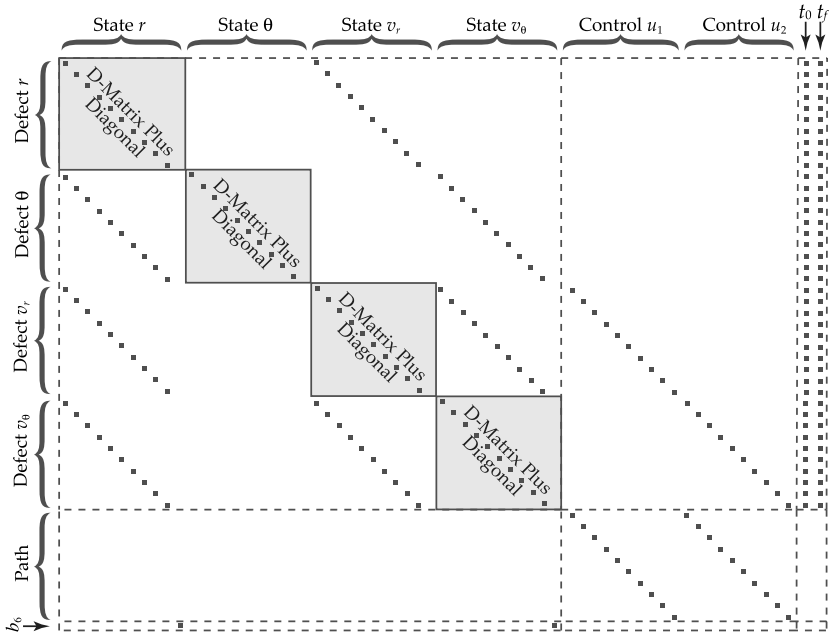


Fig. 4 NLP constraint Jacobian sparsity pattern for example problem.

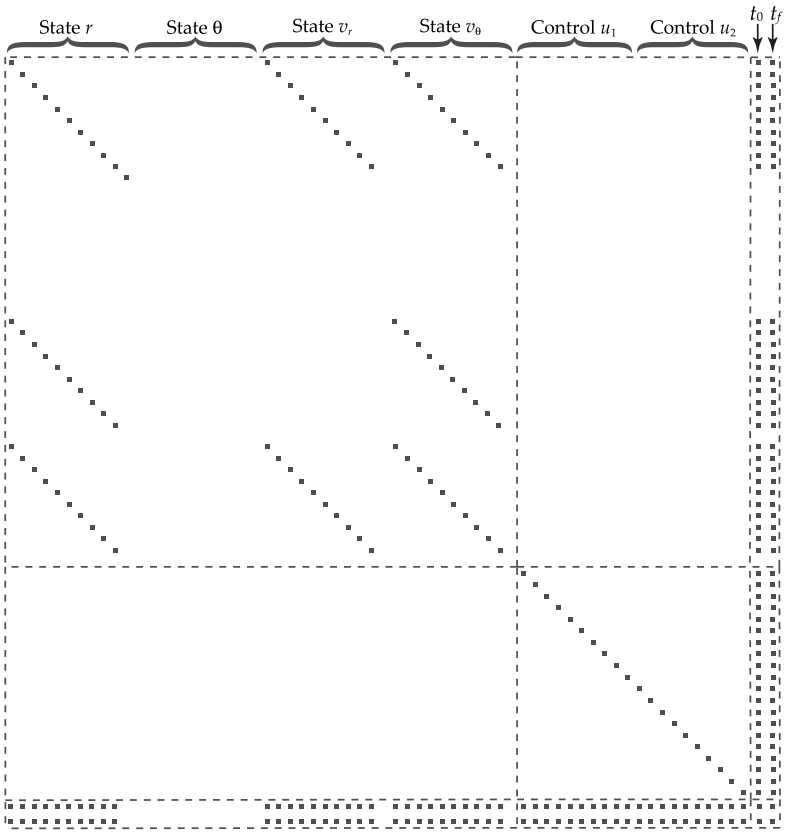
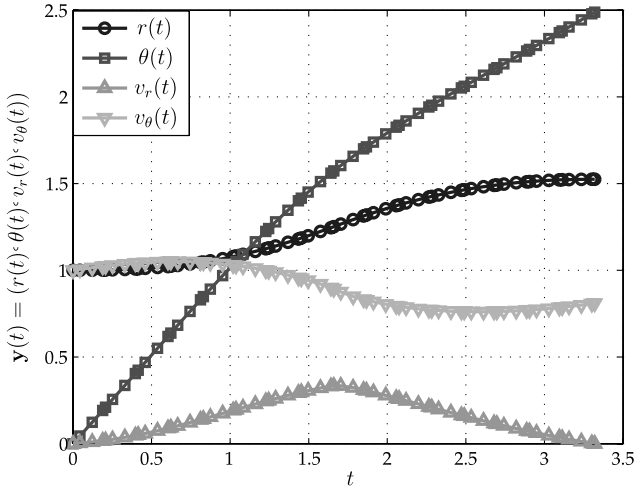
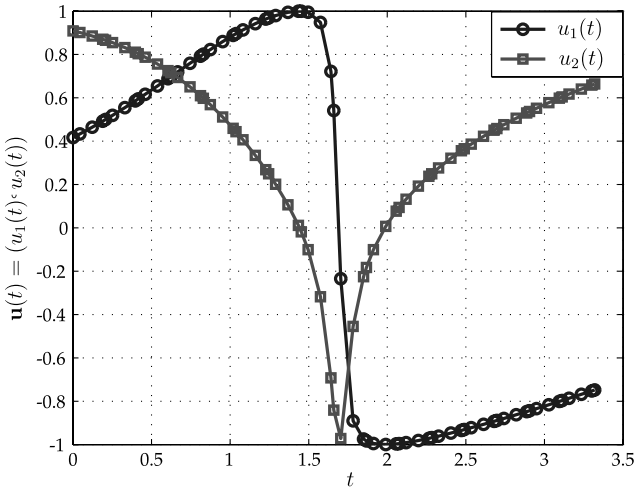


Fig. 5 NLP Lagrangian Hessian sparsity pattern for example problem.



a) State vs. time



b) Control vs. time

Fig. 6 Solution to orbit-raising optimal control problem for 16 equally-spaced sections of four LGR points each ($N = 64$).

problem shown in section III are computed; the appropriate NLP derivative matrices are then obtained by inserting these derivatives into the correct locations in the appropriate matrix using the mapping of section V. It is noted that the NLP constraint Jacobian and Lagrangian Hessian sparsity patterns, shown in Figures 4 and 5, are found using the derivation given in section V by differentiating the continuous-time Bolza optimal control problem and are implemented for all derivative methods. All computations were performed on an Intel Core 2 Duo 660 2.4 GHz computer with 2 GB of RAM running 32-bit OpenSuse Linux with MATLAB 2010a and IPOPT version 3.6, where IPOPT was compiled with the sparse symmetric linear solver MA57 [35]. Finally, for each of the methods 1–8, the values in the NLP derivatives matrices were verified using both 1) the derivative checker built into IPOPT and 2) a comparison between the derivatives obtained using the method of section V and the derivatives obtained using the automatic differentiator INTLAB [34].

The example was solved using $K = (16, 32, 64, 128, 256, 512)$ equally-spaced mesh intervals with four LGR points in each mesh interval. A typical solution to obtained for this example is shown in Fig. 6. Tables 2 and 3 summarize the computational performance using methods 1–4 and methods 5–8, respectively. In particular, Tables 2 and 3 show that differentiating the functions of the Bolza optimal control problem and using the approach of section V is significantly more computationally efficient than direct differentiation of the NLP functions. More specifically, it is seen in Tables 2 and 3 that, regardless of whether the NLP solver is used as a quasi-Newton or Newton method, the difference in computational efficiency between direct NLP function differentiation and the approach of this paper grows to several orders of magnitude. As an example, for $N = 2048$ method 1 takes 2246 s, although method 5 takes 27.1 s, whereas method 3 takes 5871 s, although method 7 takes 23.1 s. As a result, differentiating only the functions of the optimal control problem has a substantial computational benefit for large problems over direct differentiation of the NLP functions.

Next, it is useful to compare finite differencing against either automatic or analytic differentiation. First, when comparing methods 1 and 2 to methods 3 and 4 in Table 2 (that is, comparing finite differencing against automatic differentiation of the NLP functions), it is seen that using IPOPT with as a quasi-Newton method with INTLAB is significantly more efficient than using any other method where the NLP functions are differentiated directly. Correspondingly, direct differentiation of the NLP functions using IPOPT in

Table 2 Direct differentiation of the NLP functions using finite differencing and INTLAB for the example problem using the Radau pseudospectral method with $K = (16, 32, 64, 128, 256, 512)$ equally-spaced mesh intervals, $N_k = 4$ LGR points per mesh interval, and the NLP solver IPOPT

Derivative method for IPOPT	K	N	NLP major iterations	CPU time (s)
Method 1: First NLP	16	64	141	36.2
derivatives using finite differencing	32	128	121	57.9
	64	256	125	133
	128	512	176	435
	256	1024	212	1051
	512	2048	196	2246
Method 2: First NLP	16	64	118	2.9
derivatives using INTLAB	32	128	115	3.6
	64	256	136	5.7
	128	512	156	10.2
	256	1024	158	19.4
	512	2048	143	31.2
Method 3: First and second NLP	16	64	32	44.5
derivatives using finite differencing	32	128	35	100
	64	256	46	263
	128	512	49	708
	256	1024	56	2058
	512	2048	67	5871
Method 4: First and second NLP	16	64	33	2.3
derivatives using INTLAB	32	128	39	5.4
	64	256	43	17.1
	128	512	77	126
	256	1024	Out of memory	Out of memory
	512	2048	Out of memory	Out of memory

Table 3 Differentiation of the Bolza optimal control problem functions together with the approach of section V using finite differencing and analytic differentiation for the example problem using the Radau pseudospectral method with $K = (16, 32, 64, 128, 256, 512)$ equally-spaced mesh intervals, $N_k = 4$ LGR points in each mesh interval, and the NLP solver IPOPT

Derivative method for IPOPT	K	N	NLP major iterations	CPU time (s)
Method 5: First optimal control problem derivatives using finite differencing	16	64	144	1.7
	32	128	114	1.9
	64	256	132	3.5
	128	512	157	7.6
	256	1024	152	13.7
	512	2048	164	27.1
Method 6: First optimal control problem analytic derivatives	16	64	113	1.3
	32	128	106	1.7
	64	256	132	3.2
	128	512	154	6.8
	256	1024	150	12.1
	512	2048	136	21.1
Method 7: First and second optimal control problem derivatives using finite differencing	16	64	31	0.83
	32	128	35	1.3
	64	256	45	2.5
	128	512	48	4.9
	256	1024	56	11.0
	512	2048	60	23.1
Method 8: First and second optimal control problem analytic derivatives	16	64	30	0.54
	32	128	35	0.93
	64	256	41	1.6
	128	512	42	2.5
	256	1024	49	5.3
	512	2048	60	10.9

Table 4 Summary of problem sizes and densities of NLP constraint Jacobian and Lagrangian Hessian for the example problem using the Radau pseudospectral method with $K = (16, 32, 64, 128, 256, 512)$ equally-spaced mesh intervals, $N_k = 4$ LGR points in each mesh interval, and the NLP solver IPOPT

K	N	NLP variables	NLP constraints	Jacobian nonzeros	Jacobian density (%)	Hessian nonzeros	Hessian density (%)
16	64	390	321	2498	2.00	1925	1.27
32	128	774	641	4994	1.01	3845	0.642
64	256	1542	1281	9986	0.506	7685	0.323
128	512	3078	2561	19970	0.253	15365	0.162
256	1024	6150	5121	39938	0.127	30725	0.0812
512	2048	12294	10241	79874	0.0634	61445	0.0407

second-derivative mode is by far the least efficient because it is computationally costly to compute the Hessian Lagrangian in this manner. In addition to computational cost, INTLAB suffers from the problem that MATLAB runs out of memory for $N = 1024$ or $N = 2048$. Thus, even though IPOPT converges in many fewer iterations in second-derivative mode, the cost per iteration required to compute the Lagrangian Hessian is significantly higher than the cost to use the quasi-Newton Hessian approximation.

Next, Table 4 summarizes the problem size and density of the NLP constraint Jacobian and Lagrangian Hessian for the different values of K. It is interesting to observe that the densities of both the NLP constraint Jacobian and Lagrangian Hessian decrease quickly as a function of the overall problem size (number of variables and constraints). Because the number of nonzeros in the Jacobian and Hessian matrices grows slowly as a function of K, one would expect that the execution time would also grow slowly. As seen from the results in Table 3, the approach developed in section V of this paper exploits the slow growth in the number of nonzeros, thus maintaining computational tractability as the NLP increases in size. Table 2, on the other hand, shows that, when the NLP functions are directly differentiated, many unnecessary calculations are performed that degrade performance to the point where direct differentiation becomes intractable for large values of K.

The results obtained by differentiating the optimal control functions using the derivation of section V are significantly different from those obtained using direct differentiation of the NLP functions. In particular, it is seen that using either finite differencing or analytic differentiation, the computation times using the method of section V are much lower than those obtained by direct differentiation of the NLP functions. In addition, the benefit of using second analytic derivatives (a reduction in computation time by a factor of 2 over second-finite differencing) demonstrates that, with an accurate Hessian, only a small fraction of the total execution time is spent inside the NLP solver. Instead, the majority of the execution time is spent evaluating the Hessian. As a result, the speed with which IPOPT can generate a solution in second-derivative mode depends heavily upon the efficiency with which the Lagrangian Hessian can be computed. Referring again to Table 4, it is seen that the method of this paper takes advantage of the sparsity in the NLP constraint Jacobian and Lagrangian Hessian as K increases. Because the method presented in this paper has the benefit that an accurate Hessian can be computed quickly, the time required to solve the NLP is greatly reduced over direct differentiation of the NLP functions.

VIII. Conclusions

Explicit expressions have been derived for the objective function gradient, constraint Jacobian, and Lagrangian Hessian of a nonlinear programming problem that arises in direct collocation pseudospectral methods for solving continuous-time optimal control problems. A key feature of the procedure developed in this paper is that only the functions of the continuous-time optimal control problem need to be differentiated in order to determine the nonlinear programming problem derivative functions. As a result, it is possible to obtain these derivative functions much more efficiently than would be the case if the nonlinear programming problem functions were directly differentiated. In addition, the approach derived in this paper explicitly identifies the sparse structure of the nonlinear programming

problem. The approach developed in this paper can significantly improve the computational efficiency and reliability of solving the nonlinear programming problem arising from the pseudospectral approximation, particularly when using a second-derivative nonlinear programming problem solver where the Lagrangian Hessian can be exploited. An example has been studied to show the efficiency of various derivative options, and the approach developed in this paper is found to improve significantly the efficiency with which the nonlinear programming problem is solved.

Acknowledgment

The authors gratefully acknowledge support for this research from the U.S. Office of Naval Research under Grant N00014-11-1-0068.

References

[1] Babuska, I., and Suri, M., "The p- and $h-p$- Versions of the Finite Element Method, an Overview," *Computer Methods in Applied Mechanics and Engineering*, Vol. 80, Nos. 1–3, 1990, pp. 5–26. doi:10.1016/0045-7825(90)90011-A

[2] Babuska, I., and Suri, M., "The p and hp Version of the Finite Element Method, Basic Principles and Properties," *SIAM Review*, Vol. 36, No. 4, Dec. 1994, pp. 578–632. doi:10.1137/1036141

[3] Gui, W., and Babuska, I., "The h, p, and hp Versions of the Finite Element Method in 1 Dimension. Part I. The Error Analysis of the p Version," *Numerische Mathematik*, Vol. 49, No. 6, 1986, pp. 577–612. doi:10.1007/BF01389733

[4] Gui, W., and Babuska, I., "The h, p, and hp Versions of the Finite Element Method in 1 Dimension. Part II. The Error Analysis of the h and h-p Versions," *Numerische Mathematik*, Vol. 49, No. 6, 1986, pp. 613–657. doi:10.1007/BF01389734

[5] Gui, W., and Babuska, I., "*The h-Version, p-Version, and h − p-Version of the Finite-Element Method in 1-Dimension. 3. The Adaptive h − p Version*," *Numerische Mathematik*, Vol. 49, No. 6, 1986, pp. 613–657.

[6] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed., SIAM Press, Philadelphia, 2009.

[7] Jain, D., and Tsiotras, P., "Trajectory Optimization Using Multiresolution Techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, September–October 2008, pp. 1424–1436. doi:10.2514/1.32220

[8] Zhao, Y., and Tsiotras, P., "A Density-Function Based Mesh Refinement Algorithm for Solving Optimal Control Problems," *Infotech and Aerospace Conference*, AIAA Paper 2009-2019, Seattle, Washington, April 2009.

[9] Elnagar, G., Kazemi, M., and Razzaghi, M., "The Pseudospectral Legendre Method for Discretizing Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. 40, No. 10, 1995, pp. 1793–1796. doi:10.1109/9.467672

[10] Elnagar, G., and Razzaghi, M., "A Collocation-Type Method for Linear Quadratic Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 18, No. 3, May–June 1997, pp. 227–235.

[11] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, November–December 2006, pp. 1435–1440. doi:10.2514/1.20478

[12] Huntington, G. T., "Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

[13] Rao, A. V., Benson, D. A., Darby, C. L., Francolin, C., Patterson, M. A., Sanders, I., and Huntington, G. T., "Algorithm 902: GPOPS, A Matlab Software for Solving Multiple-Phase Optimal Control Problems," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, Article 22, April–June 2010, 39 p..

[14] Garg, D., Patterson, M. A., Darby, C. L., Francolin, C., Huntington, G. T., Hager, W. W., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems via a Radau Pseudospectral Method," *Computational Optimization and Applications*, October 6, 2009 (Published Online) DOI:10.1007/s10589-009-9291-0 http://www.springerlink.com/content/n851q6n343p9k60k/.

[15] Garg, D., Patterson, M. A., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," *Automatica*, Vol. 46, No. 11, November 2010, pp. 1843–1851. doi:10.1016/j.automatica.2010.06.048

[16] Garg, D., Hager, W. W., and Rao, A. V., "Pseudospectral Methods for Solving Infinite-Horizon Optimal Control Problems," *Automatica*, March 2010 (Published Online) DOI:10.1016/j.automatica.2011.01.085

[17] Darby, C. L., Hager, W. W., and Rao, A. V., "An hp–Adaptive Pseudospectral Method for Solving Optimal Control Problems," *Optimal Control Applications and Methods*, Accepted for Publication June 2010 (In Press).

[18] Darby, C. L., Hager, W. W., and Rao, A. V., "Direct Trajectory Optimization Using a Variable Low-Order Adaptive Pseudospectral Method," *Journal of Spacecraft and Rockets* Accepted for Publication November 2010 (In Press).

[19] Benson, D. A., "A Gauss Pseudospectral Transcription for Optimal Control," Ph.D. thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2004.

[20] Rao, A. V., Benson, D. A., Darby, C. L., Patterson, M. A., Sanders, I., and Huntington, G. T., User's Manual for GPOPS Version 2.2, http://gpops.sourceforge.net/, June 2009.

[21] Kameswaran, S., and Biegler, L. T., "Convergence Rates for Direct Transcription of Optimal Control Problems Using Collocation at Radau Points," *Computational Optimization and Applications*, Vol. 41, No. 1, 2008, pp. 81–126. doi:10.1007/s10589-007-9098-9

[22] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., *Spectral Methods in Fluid Dynamics*, Spinger-Verlag, Heidelberg, Germany, 1988.

[23] Fornberg, B., *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, England, 1998.

[24] Trefethen, L. N., *Spectral Methods Using MATLAB*, SIAM Press, Philadelphia, 2000.

[25] Darby, C. L., and Rao, A. V., "An Initial Examination of Using Pseudospectral Methods for Time-Scale and Differential Geometric Analysis of Nonlinear Optimal Control Problems," *2008 Guidance, Navigation, and Control Conference*, AIAA, Honolulu, Hawaii, August 2008.

[26] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., User's Guide for NPSOL (Version 4.0): A FORTRAN Package for Nonlinear Programming, Department of Operations Research, Stanford University, January 1986.

[27] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, April 2002, pp. 979–1006. doi:10.1137/S1052623499350013

[28] Gill, P. E., Murray, W., and Saunders, M. A., User's Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming, February 2006.

[29] Biegler, L. T., and Zavala, V. M., "Large-Scale Nonlinear Programming Using IPOPT: An Integrating Framework for Enterprise-Wide Optimization," *Computers and Chemical Engineering*, Vol. 33, No. 3, March 2009, pp. 575–582. doi:10.1016/j.compchemeng.2008.08.006

[30] Byrd, R. H., Nocedal, J., and Waltz, R. A., "KNITRO: An Integrated Package for Nonlinear Optimization," *Large Scale Nonlinear Optimization*, Springer-Verlag, Berlin, 2006, pp. 35–59.

[31] Betts, J. T., and Huffman, W. P., "Exploiting Sparsity in the Direct Transcription Method for Optimal Control," *Computational Optimization and Applications*, Vol. 14, No. 2, 1999, pp. 179–201. doi:10.1023/A:1008739131724

[32] Abramowitz, M., and Stegun, I., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover Publications, New York, 1965.

[33] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control*, Hemisphere Publishing, New York, 1975.

[34] Rump, S. M., "INTLAB—INTerval LABoratory," *Developments in Reliable Computing*, edited by T. Csendes, Kluwer Academic Publishers, Dordrecht, Germany, 1999, pp. 77–104.

[35] Duff, I. S., "MA57—a Code for the Solution of Sparse Symmetric Definite and Indefinite Systems," *ACM Transactions on Mathematical Software*, Vol. 30, No. 2, April–June 2004, pp. 118–144. doi:10.1145/992200.992202

D. Spencer
Associate Editor