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

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

Abstract

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

\[ a = \text{Differential Equation Right-Hand Side Function} \]
\[ a = \text{Thrust Acceleration, } 4\pi^2 \times \text{AU/Year}^2 \]

*Ph.D. Candidate, Department of Mechanical and Aerospace Engineering. E-mail: mpatterson@ufl.edu.
†Assistant Professor, Department of Mechanical and Aerospace Engineering. E-mail: anilvrao@ufl.edu. Corresponding Author. Associate Fellow AIAA.
b = Boundary Condition Function

c = Path Constraint Function

D = Radau Pseudospectral Differentiation Matrix

g = Integrand of Cost Functional

h = General Nonlinear Programming Constraint Function

J = Continuous-Time Optimal Control Problem Cost Functional

K = Number of Mesh Intervals

m = Mass, 10000 × lbm

\dot{m} = Mass Flow Rate, 20000\pi × lbm/Year

n_c = Dimension of Continuous-Time Path Constraint

n_u = Dimension of Continuous-Time Control

n_y = Dimension of Continuous-Time State

N = Total Number of Collocation Points

N_k = Polynomial Degree in Mesh Interval k

N_z = Number of Nonzero Constraint Jacobian Entries

P = General Matrix

Q = General Matrix

p = General Vector

p(t) = General Vector Function of Time

q = General Vector

q(t) = General Vector Function of Time

r = Radius, Astronomical Units \equiv \text{AU}

\dot{r} = Rate of Change of Radius, \text{AU/Year} \times 2\pi

s = Time on Time Interval \ s ∈ [−1, +1]

T = Thrust, 20000\pi \times \text{lbm} \cdot \text{AU}/\text{Year}

t_0 = Initial Time

t_f = Terminal Time

t = Time on Time Interval \ t ∈ [t_0, t_f], \text{dimensionless or Year}/2\pi

U_i = Approximation to Control at Collocation Point i

u(t) = Control on Time Domain \ t ∈ [t_0, t_f]

v_r = Radial Component of Velocity, 2\pi × \text{AU}/\text{Year}

v_\theta = Tangential Component of Velocity, 2\pi × \text{AU}/\text{year}

u_1(t) = First Component of Control

u_2(t) = Second Component of Control

w_j = j^{th} \text{Legendre-Gauss-Radau Quadrature Weight}

Y(s) = State Approximation on Time Domain \ s ∈ [−1, +1]

y(t) = State on Time Domain \ t ∈ [t_0, t_f]

y(s) = State on Time Domain \ s ∈ [−1, +1]

z = Nonlinear Programming Problem Decision Vector

\Gamma = \text{Matrix of Defect Constraint Lagrange Multipliers}

\Lambda = \text{Matrix of Nonlinear Programming Problem Lagrange Multipliers}

\mu = \text{Sun Gravitational Parameter, } 4\pi^2 \times \text{AU}^3/\text{Year}^2

\nu = \text{Boundary Condition Lagrange Multiplier}

\rho = \text{Nonlinear Programming Problem Cost Function}

\phi = \text{Optimal Control Problem Mayer Cost Function}
\[ \Psi = \text{Matrix of Path Constraint Lagrange Multipliers} \]
\[ \ell^{(k)}(s) = \text{Lagrange Polynomial on Mesh Interval } s \in [s_{k-1}, s_k] \]
\[ \theta = \text{Angular Displacement, radians} \]
\[ \dot{\theta} = \text{Rate of Change of Angular Displacement, } 2\pi \times \text{radians/year} \]

1 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 \textit{h–methods,} \textit{p–methods}, \textit{or \textit{hp–methods}.} \textit{In an \textit{h–method,}} the state is approximated using many fixed low-degree polynomial (e.g., second-degree or third-degree) mesh intervals. Convergence in an \textit{h–method} is then achieved by increasing the number of mesh intervals. \textit{In a \textit{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. \textit{In an \textit{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.

In recent years, interest has increased in using direct collocation \textit{pseudospectral methods.} \textit{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 \textit{p–methods.} \textit{For problems whose solutions are smooth and well-behaved, a pseudospectral method has a simple structure and converges at an exponential rate.} \textit{The most well developed \textit{p–type pseudospectral methods are the Gauss pseudospectral method (GPM), the Radau pseudospectral method} (RPM), and the Lobatto pseudospectral method} (LPM). More recently, it has been found that computational efficiency and accuracy can be increased by using either an \textit{h–} or an \textit{hp–} pseudospectral method.

While 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 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 BFGS or 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\textsuperscript{26} and SNOPT\textsuperscript{27,28} while well known second-derivative NLP solvers include IPOPT\textsuperscript{29} and KNITRO\textsuperscript{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. While having an accurate Lagrangian Hessian is desirable, even for sparse NLPs computing a Hessian is inefficient if not done properly. While 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 Radau pseudospectral method.\textsuperscript{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 on an example using the NLP solver IPOPT.\textsuperscript{29}

It is noted that Ref. 31 develops an approach for exploiting sparsity in local direct collo-
cation methods (e.g., Euler, Hermite-Simpson, and Runge-Kutta methods). In Ref. 31, 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 Ref. 31 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 2 we provide our notation and conventions used throughout this paper. In Section 3 we state the continuous-time Bolza optimal control problem. In Section 4, we state the Radau pseudospectral method\textsuperscript{14–16} that is used to derive the NLP derivative functions. In Section 5 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 3 using the Radau pseudospectral method of Section 4. In Section 6 we provide a discussion of the underlying structure of the derivative functions. In Section 7 we provide an example that demonstrates the great improvement in computational efficiency obtained using the method of this paper. Finally, in Section 8 we provide conclusions on our work.
2 Notation and Conventions

Throughout this paper the following notation and conventions will be employed. All scalars will be represented by lower-case symbols (e.g., \( y, u \)). All vector functions of time will be treated as row vectors and will be denoted by lower-case bold symbols. Thus, if \( p(t) \in \mathbb{R}^n \) is a vector function of time, then \( p(t) = [p_1(t) \cdots p_n(t)] \). Any vector that is not a function of time will be denoted as a column vector, i.e., a static vector \( z \in \mathbb{R}^n \) will be treated as a column vector.

Next, matrices will be denoted by upper case bold symbols. Thus, \( P \in \mathbb{R}^{N \times n} \) is a matrix of size \( N \times n \). Furthermore, if \( f(p) \), \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \), is a function that maps row vectors \( p \in \mathbb{R}^n \) to row vectors \( f(p) \in \mathbb{R}^m \), then the result of evaluating \( f(p) \) at the points \( (p_1, \ldots, p_N) \) is the matrix \( F \in \mathbb{R}^{N \times n} \equiv [f(p_k)]_N^1 \),

\[
F_N^1 \equiv [f(p_k)]_N^1 = \begin{bmatrix}
  f(p_1) \\
  \vdots \\
  f(p_N)
\end{bmatrix}.
\]

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

Next, let \( P \) and \( Q \) be \( n \times m \) matrices. Then the element-by-element multiplication of \( P \) and \( Q \) is defined as

\[
P \odot Q = \begin{bmatrix}
p_{11}q_{11} & \cdots & p_{1m}q_{1m} \\
\vdots & \ddots & \vdots \\
p_{n1}q_{n1} & \cdots & p_{nm}q_{nm}
\end{bmatrix}.
\]

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

\[
\text{diag}(p) = \begin{bmatrix}
p_1 & 0 & \cdots & 0 \\
0 & p_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & p_n
\end{bmatrix}.
\]
Finally, the notation $0_{n \times m}$ represents an $n \times m$ matrix of zeros, while $1_{n \times m}$ represent an $n \times m$ matrix of all ones.

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

$$\nabla_p f(p) = \left[ \frac{\partial f}{\partial p_1} \ldots \frac{\partial f}{\partial p_n} \right]$$

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

$$\nabla_p f = \begin{bmatrix} \nabla_p f_1 \\ \vdots \\ \nabla_p f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \ldots & \frac{\partial f_1}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial p_1} & \ldots & \frac{\partial f_m}{\partial p_n} \end{bmatrix}$$

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

$$\nabla_{pq}^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial p_1 \partial q_1} & \ldots & \frac{\partial^2 f}{\partial p_1 \partial q_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial p_n \partial q_1} & \ldots & \frac{\partial^2 f}{\partial p_n \partial q_m} \end{bmatrix} = [\nabla_{qp}^2 f]^T.$$

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

$$\nabla_{pp}^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial p_1^2} & \ldots & \frac{\partial^2 f}{\partial p_1 \partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial p_n \partial p_1} & \ldots & \frac{\partial^2 f}{\partial p_n^2} \end{bmatrix} = [\nabla_{pp}^2 f]^T.$$

### 3 Bolza Optimal Control Problem

Consider the following general optimal control problem in Bolza form. Determine the state, $y(t) \in \mathbb{R}^n$, the control $u(t) \in \mathbb{R}^m$, 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

$$J = \phi(y(t_0), t_0, y(t), t_f) + \int_{t_0}^{t_f} g(y(t), u(t), t) \, dt$$

subject to the dynamic constraints

$$\frac{dy}{dt} = a(y(t), u(t), t),$$

the inequality path constraints

$$c_{\text{min}} \leq c(y(t), u(t), t) \leq c_{\text{max}},$$

and the boundary conditions

$$b_{\text{min}} \leq b(y(t_0), t_0, y(t_f), t_f) \leq b_{\text{min}}.$$  

The functions $\phi$, $g$, $a$, $c$ and $b$ are defined by the following mappings:

$$\phi : \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \rightarrow \mathbb{R},$$

$$g : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R},$$

$$a : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_y},$$

$$c : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_c},$$

$$b : \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \rightarrow \mathbb{R}^{n_b},$$

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

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

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

$$J = \phi(y(-1), t_0, y(+1), t_f) + \frac{t_f - t_0}{2} \int_{-1}^{+1} g(y(s), u(s), s; t_0, t_f) \, ds$$

subject to the dynamic constraints

$$\frac{dy}{ds} = \frac{t_f - t_0}{2} a(y(s), u(s), s; t_0, t_f),$$
the inequality path constraints

\[ c_{\text{min}} \leq c(y(s), u(s), s; t_0, t_f) \leq c_{\text{max}}, \tag{8} \]

and the boundary conditions

\[ b_{\text{min}} \leq b(y(-1), t_0, y(+1), t_f) \leq b_{\text{min}}. \tag{9} \]

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 \( y^{(k)}(s) \) and \( u^{(k)}(s) \) 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

\[ J = \phi(y^{(1)}(-1), t_0, y^{(K)}(+1), t_f) \]

\[ + \frac{t_f - t_0}{2} \sum_{k=1}^{K} \int_{s_{k-1}}^{s_k} g(y^{(k)}(\tau), u^{(k)}(\tau), s; t_0, t_f) d\tau, \quad (k = 1, \ldots, K). \tag{10} \]

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

\[ \frac{dy^{(k)}(s)}{ds} = \frac{t_f - t_0}{2} a(y^{(k)}(s), u^{(k)}(s), s; t_0, t_f), \quad (k = 1, \ldots, K). \tag{11} \]

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

\[ c_{\text{min}} \leq c(y^{(k)}(s), u^{(k)}(s), s; t_0, t_f) \leq c_{\text{max}}, \quad (k = 1, \ldots, K). \tag{12} \]

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

\[ b_{\text{min}} \leq b(y^{(1)}(-1), t_0, y^{(K)}(+1), t_f) \leq b_{\text{max}}. \tag{13} \]

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

### 4 Radau Pseudospectral Method

The multiple-interval form of the continuous-time Bolza optimal control problem in Section 3 is discretized using the previously developed Radau pseudospectral method as described in Ref. 14.
While the Radau pseudospectral method is chosen, with only slight modifications the approach developed in this paper can be used with other pseudospectral methods (e.g., the Gauss or the Lobatto pseudospectral method). An advantage of using the Radau scheme is that the continuity conditions $y(s^-_k) = y(s^+_k)$ across mesh points are particularly easy to implement.

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

$$y^{(k)}(s) \approx Y^{(k)}(s) = \sum_{j=1}^{N_k+1} Y_j^{(k)} \ell_j^{(k)}(s), \quad \ell_j^{(k)}(s) = \prod_{l \neq j}^{N_k+1} \frac{s - s_l^{(k)}}{s_j^{(k)} - s_l^{(k)}},$$

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

$$\frac{dY^{(k)}(s)}{ds} = \sum_{j=1}^{N_k+1} Y_j^{(k)} \frac{d\ell_j^{(k)}(s)}{ds}.$$  

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

$$J \approx \phi(Y_1^{(1)}, t_0, Y_{N_k+1}^{(K)}, t_K) + \sum_{k=1}^K \sum_{j=1}^{N_k} \frac{t_f - t_0}{2} w_j^{(k)} g(Y_j^{(k)}, U_j^{(k)}, s_j^{(k)}; t_0, t_f),$$

where $w_j^{(k)}$, $j = 1, \ldots, N_k$ are the LGR quadrature weights in mesh interval $k \in [1, \ldots, K]$ defined on the interval $s \in [s_{k-1}, s_k]$, $U_i^{(k)}$, $i = 1, \ldots, N_k$, are the approximations of the control at the $N_k$ LGR points in mesh interval $k \in [1, \ldots, K]$, $Y_1^{(1)}$ is the approximation of $y(s_0 = -1)$, and $Y_{N_K+1}^{(K)}$ is the approximation of $y(s_K = +1)$. Collocating the dynamics of Eq. (11) at the $N_k$ LGR points using Eq. (15), we have

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} Y_j^{(k)} - \frac{t_f - t_0}{2} a(Y_i^{(k)}, U_i^{(k)}, s_i^{(k)}; t_0, t_f) = 0, \quad (i = 1, \ldots, N_k).$$

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

$$D_{ij}^{(k)} = \left[ \frac{d\ell_j^{(k)}(s)}{ds} \right]_{s_i^{(k)}}, \quad (i = 1, \ldots, N_k, \quad j = 1, \ldots, N_k + 1, \quad k = 1, \ldots, K).$$
is the $N_k \times (N_k + 1)$ Radau pseudospectral differentiation matrix\textsuperscript{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

$$c_{min} \leq c(Y^{(k)}_i, U^{(k)}_i, s^{(k)}_i; t_0, t_f) \leq c_{max}, \quad (i = 1, \ldots, N_k).$$ \hspace{1cm} (19)

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

$$b_{min} \leq b(Y^{(1)}_1, t_0, Y^{(K)}_{N_{K+1}}, t_f) \leq b_{max}.$$ \hspace{1cm} (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

$$Y^{(k)}_{N_k+1} = Y^{(k+1)}_1, \quad (k = 1, \ldots, K - 1),$$ \hspace{1cm} (21)

where we note that the \textit{same} variable is used for both $Y^{(k)}_{N_k+1}$ and $Y^{(k+1)}_1$. 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 $K$:

$$s^{(k)} = \left[ s^{(k)}_i \right]_{N_k}, \quad k = 1, \ldots, K - 1,$$

$$t^{(k)} = \left[ t^{(k)}_i \right]_{N_k}, \quad k = 1, \ldots, K - 1,$$

$$Y^{(k)} = \left[ Y^{(k)}_i \right]_{N_k}, \quad k = 1, \ldots, K - 1,$$

$$U^{(k)} = \left[ U^{(k)}_i \right]_{N_k}, \quad k = 1, \ldots, K,$$

$$A^{(k)} = \left[ a(Y^{(k)}_i, U^{(k)}_i, s^{(k)}_i; t_0, t_f) \right]_{N_k}, \quad k = 1, \ldots, K,$$

$$w^{(k)} = \left[ w_i \right]_{N_k}, \quad k = 1, \ldots, K,$$

$$N = \sum_{k=1}^{K} N_k.$$
We then define the following quantities:

\[ s = \begin{bmatrix} s^{(1)} \\ \vdots \\ s^{(K)} \end{bmatrix}, \quad t = \begin{bmatrix} t^{(1)} \\ \vdots \\ t^{(K)} \end{bmatrix}, \]

\[ w = \begin{bmatrix} w^{(1)} \\ \vdots \\ w^{(K)} \end{bmatrix}, \quad Y = \begin{bmatrix} Y^{(1)} \\ \vdots \\ Y^{(K)} \end{bmatrix}, \]

\[ U = \begin{bmatrix} U^{(1)} \\ \vdots \\ U^{(K)} \end{bmatrix}, \quad g = \begin{bmatrix} g^{(1)} \\ \vdots \\ g^{(K)} \end{bmatrix}, \]

\[ A = \begin{bmatrix} A^{(1)} \\ \vdots \\ A^{(K)} \end{bmatrix}, \quad C = \begin{bmatrix} C^{(1)} \\ \vdots \\ C^{(K)} \end{bmatrix}. \]

(22)

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

\[ J \approx \phi(Y_1, t_0, Y_{N+1}, t_f) + \frac{t_f - t_0}{2} w^T g \]

(23)

\[ \Delta = DY - \frac{t_f - t_0}{2} A = 0, \]

(24)

where \( \Delta \in \mathbb{R}^{N \times n_y} \) and \( D \) is the composite Radau pseudospectral differentiation matrix. A schematic of the composite Radau differentiation matrix \( D \) is shown in Fig. 1 where it is seen that \( D \) has a block structure with nonzero elements in the row-column indices \((\sum_{l=1}^{k-1} N_l + 1, \ldots, \sum_{l=1}^{k} N_l), (\sum_{l=1}^{k-1} N_l + 1, \ldots, \sum_{l=1}^{k} N_l)\), 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

\[ C_{\text{min}} \leq C \leq C_{\text{max}}, \]

(25)
where \( C_{\text{min}} \) and \( C_{\text{max}} \) are matrices of the same size as \( C \) and whose rows contain the vectors \( c_{\text{min}} \) and \( c_{\text{max}} \), respectively. Furthermore, the discretized boundary conditions of Eq. (20) can be written as

\[
b_{\text{min}} \leq b(Y_1, t_0, Y_{N+1}, t_f) \leq b_{\text{max}}.
\] (26)

The nonlinear programming problem (NLP) associated with the Radau pseudospectral method 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

\[
\alpha = \frac{\partial t}{\partial t_0} = \frac{1 - s}{2}, \quad \beta = \frac{\partial t}{\partial t_f} = \frac{1 + s}{2},
\] (27)

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

### 5 Computation of Radau Pseudospectral NLP Derivatives

The nonlinear programming problem (NLP) arising from the Radau pseudospectral method presented in Section 4 has the following general form. Determine the vector of decision variables \( z \in \mathbb{R}^{N(n_y+n_c)+2} \) that minimizes the cost function

\[
f(z)
\] (28)

subject to the constraints

\[
h_{\text{min}} \leq h(z) \leq h_{\text{max}}.
\] (29)

In the case of the Radau pseudospectral method, the decision vector, \( z \), constraint function \( h(z) \), and cost function \( f(z) \) are given, respectively, as

\[
z = \begin{bmatrix}
Y_{:,1} \\
\vdots \\
Y_{:,n_y} \\
U_{:,1} \\
\vdots \\
U_{:,n_u} \\
t_0 \\
t_f
\end{bmatrix}, \quad
h = \begin{bmatrix}
\Delta_{:,1} \\
\vdots \\
\Delta_{:,n_y} \\
C_{:,1} \\
\vdots \\
C_{:,n_c} \\
b_{1:n_b}
\end{bmatrix}, \quad
f(z) = \phi(z) + \gamma(z),
\] (30)
where $\phi$ is obtained directly from Eq. (23) and $\gamma$ is given as

$$\gamma = \frac{t_f - t_0}{2} \mathbf{w}^T \mathbf{g}. \quad (31)$$

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) of Section 3 as opposed to differentiating the functions of the NLP.

### 5.1 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

$$\nabla \mathbf{z} f = \nabla \mathbf{z} \phi + \nabla \mathbf{z} \gamma. \quad (32)$$

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

$$\nabla \mathbf{z} \phi = \begin{bmatrix} \nabla \mathbf{Y} \phi & \nabla \mathbf{U} \phi & \nabla t_0 \phi & \nabla t_f \phi \end{bmatrix}, \quad (33)$$

where

$$\nabla \mathbf{Y} \phi = \begin{bmatrix} \nabla \mathbf{Y}_{1,\phi} & \cdots & \nabla \mathbf{Y}_{1,y\phi} \end{bmatrix}$$

$$\nabla \mathbf{U} \phi = \begin{bmatrix} 0_{1 \times N_u} \end{bmatrix}. \quad (34)$$

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

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

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

$$\nabla \mathbf{z} \gamma = \begin{bmatrix} \nabla \mathbf{T} \gamma & \nabla \mathbf{U} \gamma & \nabla t_0 \gamma & \nabla t_f \gamma \end{bmatrix}, \quad (36)$$

where

$$\nabla \mathbf{Y} \gamma = \begin{bmatrix} \nabla \mathbf{Y}_{1,\gamma} & \cdots & \nabla \mathbf{Y}_{1,y} \gamma \end{bmatrix}, \quad (37)$$

$$\nabla \mathbf{U} \gamma = \begin{bmatrix} \nabla \mathbf{U}_{1,\gamma} & \cdots & \nabla \mathbf{U}_{1,u} \gamma \end{bmatrix}.$$
The derivatives $\nabla Y \gamma$, $\nabla U \gamma$, $\nabla t_0 \gamma$ and $\nabla t_f \gamma$ are obtained as

$$\nabla Y \gamma = \left[ \frac{t_f - t_0}{2} \left\{ w \circ \left[ \frac{\partial g}{\partial y_i} \right]^1_N \right\}^T 0 \right], \quad (i = 1, \ldots, n_y),$$

$$\nabla U \gamma = \frac{t_f - t_0}{2} \left\{ w \circ \left[ \frac{\partial g}{\partial u_j} \right]^1_N \right\}^T, \quad (j = 1, \ldots, n_u)$$

$$\nabla t_0 \gamma = -\frac{1}{2} w^T g + \frac{t_f - t_0}{2} w^T \left\{ \alpha \circ \left[ \frac{\partial g}{\partial t} \right]^1_N \right\},$$

$$\nabla t_f \gamma = \frac{1}{2} w^T g + \frac{t_f - t_0}{2} w^T \left\{ \beta \circ \left[ \frac{\partial g}{\partial t} \right]^1_N \right\}.$$  \hspace{1cm} (38)

It is seen from Eqs. (32)–(38) that computing the objective function gradient, $\nabla z f$, requires that the first derivatives of $g$ be determined with respect to the continuous-time state, $y$, control, $u$, and time, $t$, while the first derivatives of $\phi$ be computed with respect to the initial state, $y(t_0)$, initial time, $t_0$, final state, $y(t_f)$, 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 $\phi$ and the derivatives of $\phi$). The NLP objective function and gradient is then assembled using the equations derived in this section.

5.2 Constraint Jacobian

The Jacobian of the constraints is defined as

$$\nabla z h = \begin{bmatrix}
\nabla z \Delta_{:,1} \\
\vdots \\
\nabla z \Delta_{:,n_y} \\
\nabla z C_{:,1} \\
\vdots \\
\nabla z C_{:,n_c} \\
\nabla z b_1 \\
\vdots \\
\nabla z b_{nb}
\end{bmatrix}$$

\hspace{1cm} (39)
The first derivatives of the defect constraints are obtained as

$$\nabla z \Delta_i^l = \begin{bmatrix} \nabla Y \Delta_i^l & \nabla U \Delta_i^l & \nabla t_0 \Delta_i^l & \nabla t_f \Delta_i^l \end{bmatrix}, \quad l = 1, \ldots, n_y, \quad (40)$$

where

$$\nabla Y \Delta_i^l = \begin{bmatrix} \nabla Y_{1:i} \Delta_i^l & \cdots & \nabla Y_{n_y:i} \Delta_i^l \end{bmatrix}, \quad l = 1, \ldots, n_y, \quad (41)$$

The first derivatives \( \nabla Y_{1:i} \Delta_i^l \) \((i, l = 1, \ldots, n_y))\), \( \nabla U_{1:j} \Delta_i^l \) \((j = 1, \ldots, n_u, l = 1, \ldots, n_y))\), \( \nabla t_0 \Delta_i^l \) \((l = 1, \ldots, n_y))\), and \( \nabla t_f \Delta_i^l \) \((l = 1, \ldots, n_y))\), can be obtained as

$$\nabla Y_{1:i} \Delta_i^l = \left[ \delta_{il} \frac{D_{1:N} - t_f - t_0}{2} \text{diag} \left( \left[ \frac{\partial a_l}{\partial y_i} \right]_N \right) \right] \delta_{1:N+1},$$

$$\nabla U_{1:j} \Delta_i^l = -\frac{t_f - t_0}{2} \text{diag} \left( \left[ \frac{\partial a_l}{\partial u_j} \right]_N \right),$$

$$\nabla t_0 \Delta_i^l = \frac{1}{2} [a_l]_N - \frac{t_f - t_0}{2} \alpha \circ \left[ \frac{\partial a_l}{\partial t} \right]_N,$$

$$\nabla t_f \Delta_i^l = -\frac{1}{2} [a_l]_N - \frac{t_f - t_0}{2} \beta \circ \left[ \frac{\partial a_l}{\partial t} \right]_N,$$

\( (i, l = 1, \ldots, n_y) \) and \( (j = 1, \ldots, n_u) \). Furthermore, \( \delta_{il} \) 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

$$\nabla z C_{1:p} = \begin{bmatrix} \nabla Y C_{1:p} & \nabla U C_{1:p} & \nabla t_0 C_{1:p} & \nabla t_f C_{1:p} \end{bmatrix}, \quad (43)$$

where

$$\nabla Y C_{1:p} = \begin{bmatrix} \nabla Y_{1:1} C_{1:p} & \cdots & \nabla Y_{n_y:1} C_{1:p} \end{bmatrix}, \quad p = 1, \ldots, n_p, \quad (44)$$

$$\nabla U C_{1:p} = \begin{bmatrix} \nabla U_{1:1} C_{1:p} & \cdots & \nabla U_{n_u:1} C_{1:p} \end{bmatrix}.$$
The first derivatives $\nabla Y_i C_{:,p}$, $\nabla U_j C_{:,p}$, $\nabla t_0 C_{:,p}$ and $\nabla t_f C_{:,p}$ can be found in a sparse manner as

$$
\nabla Y_i C_{:,p} = \begin{bmatrix} \text{diag} \left( \left[ \frac{\partial c_p}{\partial y_i} \right]^T \right) & 0_{N \times 1} \end{bmatrix},
$$

$$
\nabla U_j C_{:,p} = \text{diag} \left( \left[ \frac{\partial c_p}{\partial u_j} \right]^T \right),
$$

$$
\nabla t_0 C_{:,p} = \alpha \circ \left[ \frac{\partial c_p}{\partial t_0} \right]^T,
$$

$$
\nabla t_f C_{:,p} = \beta \circ \left[ \frac{\partial c_p}{\partial t_f} \right]^T,
$$

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

$$
\nabla z_b q = \begin{bmatrix} \nabla Y b_q & \nabla U b_q & \nabla t_0 b_q & \nabla t_f b_q \end{bmatrix}, \quad q = 1, \ldots, n_q,
$$

where

$$
\nabla Y b_q = \begin{bmatrix} \nabla Y_{:,1} b_q & \cdots & \nabla Y_{:,n_y} b_q \end{bmatrix}, \quad q = 1, \ldots, n_q.
$$

The first derivatives $\nabla Y_i b_q$, $\nabla t_0 b_q$ and $\nabla t_f b_q$ can be found in a sparse manner as

$$
\nabla Y_i b_q = \begin{bmatrix} \frac{\partial b_q}{\partial y_i(t_0)} & 0_{1 \times N-1} & \frac{\partial b_q}{\partial y_i(t_f)} \end{bmatrix},
$$

$$
\nabla t_0 b_q = \frac{\partial b_q}{\partial t_0},
$$

$$
\nabla t_f b_q = \frac{\partial b_q}{\partial t_f},
$$

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 $f$ and $c$ be determined with respect to the continuous-time state, $y$, continuous-time control, $u$, and continuous-time, $t$, and that the derivatives of $b$ be computed with respect to the initial state, $y(t_0)$, the initial time, $t_0$, the final state, $y(t_f)$, and the final time, $t_f$. Furthermore, these derivatives are computed at either the $N$ collocation points (in the case of the derivatives of $f$ and $c$) or are computed at the endpoints (in the case of $b$). 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.
5.3 Lagrangian Hessian

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

\[ \mathcal{L} = \sigma f(z) + \Lambda^T h(z), \]  

(49)

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

\[
\Lambda = \begin{bmatrix}
\Gamma_{:,1} \\
\vdots \\
\Gamma_{:,n_y} \\
\Psi_{:,1} \\
\vdots \\
\Psi_{:,n_c} \\
\nu
\end{bmatrix},
\]

(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), \( \Psi_{i,j}, (i = 1, \ldots, N, i = 1, \ldots, n_c) \) 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

\[
\mathcal{L} = \sigma \phi + \sigma \gamma + \sum_{i=1}^{n_y} \Gamma_{:,i} \Delta_{:,i} + \sum_{p=1}^{n_c} \Psi_{:,p} C_{:,p} + \sum_{q=1}^{n_b} \nu_q b_q
\]

(51)

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

\[
\nabla_{zz}^2 \mathcal{L} = \nabla_{zz}^2 \mathcal{L}_E + \nabla_{zz}^2 \mathcal{L}_I,
\]

(52)

where \( \mathcal{L}_E \) represents those parts of the Lagrangian that are functions of the endpoints functions \( \phi \) and \( b \),

\[
\mathcal{L}_E = \sigma \phi + \sum_{q=1}^{n_b} \nu_q b_q,
\]

(53)

while \( \mathcal{L}_I \) represents those parts of the Lagrangian that are functions of collocation point functions, \( \gamma, \Delta \) and \( C \),

\[
\mathcal{L}_I = \sigma \gamma + \sum_{i=1}^{n_y} \Gamma_{:,i} \Delta_{:,i} + \sum_{p=1}^{n_c} \Psi_{:,p} C_{:,p}.
\]

(54)
In the next subsections we describe the second derivatives of the functions $L_E$ and $L_I$. It is noted that the Hessian is symmetric, thus, only the lower triangular portion of $\nabla^2_{zz} L_E$ and $\nabla^2_{zz} L_I$ are computed.

### 5.3.1 Hessian of Endpoint Function $L_E$

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

$$\nabla^2_{zz} L_E = \begin{bmatrix} \nabla^2_{YY} L_E & (\nabla^2_{UY} L_E)^T & (\nabla^2_{U_0 Y} L_E)^T & (\nabla^2_{U_f Y} L_E)^T \\ \nabla^2_{UY} L_E & \nabla^2_{UU} L_E & (\nabla^2_{U_0 U} L_E)^T & (\nabla^2_{U_f U} L_E)^T \\ \nabla^2_{U_0 Y} L_E & \nabla^2_{U_0 U} L_E & \nabla^2_{U_0 L} L_E & (\nabla^2_{U_f U_0} L_E)^T \\ \nabla^2_{U_f Y} L_E & \nabla^2_{U_f U} L_E & \nabla^2_{U_f L} L_E & \nabla^2_{U_f U_f} L_E \end{bmatrix},$$

(55)

where the blocks of $\nabla^2_{zz} L_E$ are defined as

$$\nabla^2_{YY} L_E = \begin{bmatrix} \nabla^2_{Y_1 Y_1} L_E & \cdots & (\nabla^2_{Y_1 Y_n} L_E)^T \\ \nabla^2_{Y_2 Y_1} L_E & \cdots & \cdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ \nabla^2_{Y_n Y_1} L_E & \cdots & \nabla^2_{Y_n Y_n} L_E \end{bmatrix},$$

$$\nabla^2_{UY} L_E = \begin{bmatrix} \nabla^2_{U_1 Y_1} L_E & \cdots & \nabla^2_{U_1 Y_n} L_E \\ \vdots & \ddots & \vdots \\ \nabla^2_{U_n Y_1} L_E & \cdots & \nabla^2_{U_n Y_n} L_E \end{bmatrix},$$

$$\nabla^2_{UU} L_E = \begin{bmatrix} \nabla^2_{U_1 U_1} L_E & \cdots & (\nabla^2_{U_1 U_n} L_E)^T \\ \nabla^2_{U_2 U_1} L_E & \cdots & \cdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ \nabla^2_{U_n U_1} L_E & \cdots & \nabla^2_{U_n U_n} L_E \end{bmatrix},$$

$$\nabla^2_{U_0 Y} L_E = \begin{bmatrix} \nabla^2_{U_0 Y_1} L_E & \cdots & \nabla^2_{U_0 Y_n} L_E \end{bmatrix},$$

$$\nabla^2_{U_0 U} L_E = \begin{bmatrix} \nabla^2_{U_0 U_1} L_E & \cdots & \nabla^2_{U_0 U_n} L_E \end{bmatrix} = 0_{N_u \times N_u},$$

$$\nabla^2_{U_f Y} L_E = \begin{bmatrix} \nabla^2_{U_f Y_1} L_E & \cdots & \nabla^2_{U_f Y_n} L_E \end{bmatrix},$$

$$\nabla^2_{U_f U} L_E = \begin{bmatrix} \nabla^2_{U_f U_1} L_E & \cdots & \nabla^2_{U_f U_n} L_E \end{bmatrix} = 0_{N_u \times N_u}.$$
The matrices $\nabla_{X,j}^2 L_E$, $\nabla_{t_0}^2 L_E$, $\nabla_{t_f}^2 L_E$, $\nabla_{t_f}^2 L_E$, $\nabla_{t_f}^2 L_E$ and $\nabla_{t_f}^2 L_E$ are obtained in a sparse manner as

$$\nabla_{X,j}^2 L_E = \begin{bmatrix}
\frac{\partial^2 L_E}{\partial y_i(t_0) \partial y_j(t_0)} & 0_{1 \times N-1} & \frac{\partial^2 L_E}{\partial y_i(t_0) \partial y_j(t_f)} \\
0_{N-1 \times 1} & 0_{N-1 \times N-1} & 0_{N-1 \times 1} \\
\frac{\partial^2 L_E}{\partial y_i(t_f) \partial y_j(t_0)} & 0_{1 \times N-1} & \frac{\partial^2 L_E}{\partial y_i(t_f) \partial y_j(t_f)}
\end{bmatrix}, \quad (i = 1, \ldots, n_y),$$

$$\nabla_{t_0}^2 L_E = \begin{bmatrix}
\frac{\partial^2 L_E}{\partial t_0 \partial y_j(t_0)} & 0_{1 \times N-1} & \frac{\partial^2 L_E}{\partial t_0 \partial y_j(t_f)}
\end{bmatrix}, \quad (j = 1, \ldots, n_y),$$

$$\nabla_{t_f}^2 L_E = \frac{\partial^2 L_E}{\partial t_f \partial t_0},$$

$$\nabla_{t_f}^2 L_E = \frac{\partial^2 L_E}{\partial t_f \partial t_0},$$

where we recall that $L_E$ is itself a function of the Mayer cost, $\phi$, and the boundary condition function, $b$. Because $\phi$ and $b$ are functions of the continuous-time Bolza optimal control problem, the Hessian $\nabla_{zz}^2 L_E$ with respect to the NLP decision vector $z$ 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.

### 5.3.2 Hessian of Collocation Point Function $L_I$

The Hessian $\nabla_{zz}^2 L_I$ is defined as

$$\nabla_{zz}^2 L_I = \begin{bmatrix}
\nabla_{YY}^2 L_I & (\nabla_{UY}^2 L_I)^T & (\nabla_{t_0}^2 L_I)^T & (\nabla_{t_f}^2 L_I)^T \\
\nabla_{UY}^2 L_I & \nabla_{UU}^2 L_I & (\nabla_{t_0}^2 L_I)^T & (\nabla_{t_f}^2 L_I)^T \\
\nabla_{t_0}^2 L_I & \nabla_{t_0}^2 L_I & \nabla_{t_0}^2 L_I & (\nabla_{t_f}^2 L_I)^T \\
\nabla_{t_f}^2 L_I & \nabla_{t_f}^2 L_I & \nabla_{t_f}^2 L_I & \nabla_{t_f}^2 L_I
\end{bmatrix},$$

(57)
where the blocks of $\nabla^2_{zz} \mathcal{L}_I$ are given as

$$
\nabla^2_{YY} \mathcal{L}_I = \begin{bmatrix}
\nabla^2_{Y_{i1}Y_{j1}} \mathcal{L}_I & \left( \nabla^2_{Y_{i1}Y_{j1}} \mathcal{L}_I \right)^T & \cdots & \left( \nabla^2_{Y_{i1}Y_{jn}} \mathcal{L}_I \right)^T \\
\nabla^2_{Y_{i2}Y_{j1}} \mathcal{L}_I & \nabla^2_{Y_{i2}Y_{j1}} \mathcal{L}_I & \cdots & \left( \nabla^2_{Y_{i2}Y_{jn}} \mathcal{L}_I \right)^T \\
\vdots & \vdots & \ddots & \vdots \\
\nabla^2_{Y_{in}Y_{j1}} \mathcal{L}_I & \nabla^2_{Y_{in}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{Y_{in}Y_{jn}} \mathcal{L}_I
\end{bmatrix},
$$

$$
\nabla^2_{UY} \mathcal{L}_I = \begin{bmatrix}
\nabla^2_{U_{i1}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{U_{i1}Y_{jn}} \mathcal{L}_I \\
\vdots & \ddots & \vdots \\
\nabla^2_{U_{in}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{U_{in}Y_{jn}} \mathcal{L}_I
\end{bmatrix},
$$

$$
\nabla^2_{UU} \mathcal{L}_I = \begin{bmatrix}
\nabla^2_{U_{i1}U_{j1}} \mathcal{L}_I & \left( \nabla^2_{U_{i1}U_{j1}} \mathcal{L}_I \right)^T & \cdots & \left( \nabla^2_{U_{i1}U_{jn}} \mathcal{L}_I \right)^T \\
\nabla^2_{U_{i2}U_{j1}} \mathcal{L}_I & \nabla^2_{U_{i2}U_{j1}} \mathcal{L}_I & \cdots & \left( \nabla^2_{U_{i2}U_{jn}} \mathcal{L}_I \right)^T \\
\vdots & \vdots & \ddots & \vdots \\
\nabla^2_{U_{in}U_{j1}} \mathcal{L}_I & \nabla^2_{U_{in}U_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{U_{in}U_{jn}} \mathcal{L}_I
\end{bmatrix},
$$

$$
\nabla^2_{YI} \mathcal{L}_I = \begin{bmatrix}
\nabla^2_{Y_{i1}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{Y_{i1}Y_{jn}} \mathcal{L}_I \\
\vdots & \ddots & \vdots \\
\nabla^2_{Y_{in}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{Y_{in}Y_{jn}} \mathcal{L}_I
\end{bmatrix},
$$

$$
\nabla^2_{UI} \mathcal{L}_I = \begin{bmatrix}
\nabla^2_{U_{i1}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{U_{i1}Y_{jn}} \mathcal{L}_I \\
\vdots & \ddots & \vdots \\
\nabla^2_{U_{in}Y_{j1}} \mathcal{L}_I & \cdots & \nabla^2_{U_{in}Y_{jn}} \mathcal{L}_I
\end{bmatrix}.
$$
The matrices $\nabla^2_{Y.i,j} L_I$, $\nabla^2_{U.i,j} L_I$, $\nabla^2_{t_0 Y.i,j} L_I$, $\nabla^2_{t_f Y.i,j} L_I$, $\nabla^2_{t_0 U.i,j} L_I$, $\nabla^2_{t_f U.i,j} L_I$, $\nabla^2_{t_0 t_f} L_I$ and $\nabla^2_{t_f t_f} L_I$ are obtained in a sparse manner as

$$
\nabla^2_{Y.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial y_i \partial y_j} \right]_N \right) & 0_{N \times 1} \\
0_{1 \times N} & 0
\end{bmatrix}, \quad (i = 1, \ldots, n_y, j = 1, \ldots, i),
$$

$$
\nabla^2_{U.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial u_i \partial y_j} \right]_N \right) & 0_{N \times 1} \\
0_{1 \times N} & 0
\end{bmatrix}, \quad (i = 1, \ldots, n_u, j = 1, \ldots, n_y),
$$

$$
\nabla^2_{t_0 Y.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial t_0 \partial y_j} \right]_N \right) & 0 \end{bmatrix}, \quad (j = 1, \ldots, n_y),
$$

$$
\nabla^2_{t_0 U.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial t_0 \partial u_j} \right]_N \right) & 0 \end{bmatrix}^T, \quad (j = 1, \ldots, n_u),
$$

$$
\nabla^2_{t_f Y.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial t_f \partial y_j} \right]_N \right) & 0 \end{bmatrix}^T, \quad (j = 1, \ldots, n_y),
$$

$$
\nabla^2_{t_f U.i,j} L_I = \begin{bmatrix}
\text{diag} \left( \left[ \frac{\partial^2 L_I}{\partial t_f \partial u_j} \right]_N \right) & 0 \end{bmatrix}^T, \quad (j = 1, \ldots, n_u),
$$

$$
\nabla^2_{t_0 t_f} L_I = \frac{\partial^2 L_I}{\partial t_0 \partial t_f},
$$

$$
\nabla^2_{t_f t_f} L_I = \frac{\partial^2 L_I}{\partial t_f^2}.
$$

It is seen that the derivatives given in Eq. (58) are functions of the derivatives of $L_I$ with respect to the components of the continuous-time state, $y(t)$, the components of the continuous-time control, $u(t)$, the initial time, $t_0$, and the final time, $t_f$. The derivatives $\left[ \frac{\partial^2 L_I}{\partial y_i \partial y_j} \right]_N'$, $\left[ \frac{\partial^2 L_I}{\partial u_i \partial y_j} \right]_N'$, $\left[ \frac{\partial^2 L_I}{\partial u_i \partial u_j} \right]_N'$,
\[
\frac{\partial^2 L_L}{\partial y_i \partial y_j} = \frac{1}{N'} \left[ \frac{\partial^2 L_L}{\partial y_i \partial y_j} \right]^{N'}_N \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \text{and} \quad \frac{\partial^2 L_L}{\partial y_i \partial y_j} \quad \text{are given, respectively, as}
\]

\[
\frac{\partial^2 L_L}{\partial y_i \partial y_j} = \frac{1}{N} \left( t_f - t_0 \right) \sigma \left( \frac{\partial^2 g}{\partial u_i \partial u_j} \right) - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial y_i \partial y_j} \right]_N + \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial y_i \partial y_j} \right]_N \quad (i, j = 1, \ldots, n_y),
\]

(59)

\[
\frac{\partial^2 L_L}{\partial u_i \partial y_j} = \frac{1}{N} \left( t_f - t_0 \right) \sigma \left( \frac{\partial^2 g}{\partial u_i \partial y_j} \right) - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial u_i \partial y_j} \right]_N + \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial u_i \partial y_j} \right]_N \quad (i = 1, \ldots, n_u, j = 1, \ldots, n_y),
\]

(60)

\[
\frac{\partial^2 L_L}{\partial u_i \partial u_j} = \frac{1}{N} \left( t_f - t_0 \right) \sigma \left( \frac{\partial^2 g}{\partial u_i \partial u_j} \right) - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial u_i \partial u_j} \right]_N + \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial u_i \partial u_j} \right]_N \quad (i = 1, \ldots, n_u, j = 1, \ldots, n_u),
\]

(61)

\[
\frac{\partial^2 L_L}{\partial t_0 \partial y_j} = \frac{1}{2} \left\{ \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial u_i}{\partial y_j} \right]_N - \sigma \left[ \frac{\partial g}{\partial y_j} \right]_N \right\} + \frac{t_f - t_0}{2} \alpha \left\{ \sigma \left[ \frac{\partial^2 g}{\partial t \partial y_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial t \partial y_j} \right]_N \right\} + \alpha \left\{ \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial t \partial y_j} \right]_N \right\}, \quad (j = 1, \ldots, n_y),
\]

(62)

\[
\frac{\partial^2 L_L}{\partial t_0 \partial u_j} = \frac{1}{2} \left\{ \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial u_i}{\partial u_j} \right]_N - \sigma \left[ \frac{\partial g}{\partial u_j} \right]_N \right\} + \frac{t_f - t_0}{2} \alpha \left\{ \sigma \left[ \frac{\partial^2 g}{\partial t \partial u_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial t \partial u_j} \right]_N \right\} + \alpha \left\{ \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial t \partial u_j} \right]_N \right\}, \quad (j = 1, \ldots, n_u),
\]

(63)

\[
\frac{\partial^2 L_L}{\partial t^2_0} = \alpha^T \left\{ \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial a_l}{\partial t} \right]_N - \sigma \left[ \frac{\partial g}{\partial t} \right]_N \right\} + \frac{t_f - t_0}{2} \alpha^T \left\{ \sigma \left[ \frac{\partial^2 g}{\partial t^2} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \left[ \frac{\partial^2 a_l}{\partial t^2} \right]_N \right\} \circ \alpha + \alpha^T \left\{ \sum_{p=1}^{n_c} \Psi_{:,p} \left[ \frac{\partial^2 c_p}{\partial t \partial y_j} \right]_N \right\} \circ \alpha
\]

(64)
\[
\left[ \frac{\partial^2 L_I}{\partial f \partial y_j} \right]_N^1 = \frac{1}{2} \left\{ \sigma w \circ \left[ \frac{\partial g}{\partial y_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial a_l}{\partial y_j} \right]_N \right\} \\
+ \frac{t_f - t_0}{2} \beta \circ \left\{ \sigma w \circ \left[ \frac{\partial^2 g}{\partial t \partial y_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial^2 a_l}{\partial t \partial y_j} \right]_N \right\} 
\]

(65)

\[
\left[ \frac{\partial^2 L_I}{\partial f \partial u_j} \right]_N^1 = \frac{1}{2} \left\{ \sigma w \circ \left[ \frac{\partial g}{\partial u_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial a_l}{\partial u_j} \right]_N \right\} \\
+ \frac{t_f - t_0}{2} \beta \circ \left\{ \sigma w \circ \left[ \frac{\partial^2 g}{\partial t \partial u_j} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial^2 a_l}{\partial t \partial u_j} \right]_N \right\} 
\]

(66)

\[
\frac{\partial^2 L_I}{\partial t f \partial t_0} = \frac{1}{2} \alpha^T \left\{ \sigma w \circ \left[ \frac{\partial g}{\partial t} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial a_l}{\partial t} \right]_N \right\} \\
+ \frac{1}{2} \beta^T \left\{ \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial a_l}{\partial t} \right]_N - \sigma w \circ \left[ \frac{\partial g}{\partial t} \right]_N \right\} \\
+ \frac{t_f - t_0}{2} \alpha^T \left\{ \left\{ \sum_{p=1}^{n_c} \Psi_{:,p} \circ \left[ \frac{\partial^2 c_p}{\partial t^2} \right]_N \right\} \circ \beta \right\} \\
+ \alpha^T \left\{ \left\{ \sum_{p=1}^{n_c} \Psi_{:,p} \circ \left[ \frac{\partial^2 c_p}{\partial t \partial y_j} \right]_N \right\} \circ \beta \right\},
\]

(67)

\[
\frac{\partial^2 L_I}{\partial t^2 f} = \beta^T \left\{ \sigma w \circ \left[ \frac{\partial g}{\partial t} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial a_l}{\partial t} \right]_N \right\} \\
+ \frac{t_f - t_0}{2} \beta^T \left\{ \sigma w \circ \left[ \frac{\partial^2 g}{\partial t^2} \right]_N - \sum_{l=1}^{n_y} \Gamma_{:,l} \circ \left[ \frac{\partial^2 a_l}{\partial t^2} \right]_N \right\} \circ \beta 
\]

(68)

It is seen from the above derivation that the Hessian of \( L_I \) with respect to the NLP decision vector \( z \) is a function of the first and second derivatives of the functions \( g \) and \( a \), and the second derivatives of the function \( c \), where \( g \), \( a \), and \( c \) are defined in the Bolza optimal control problem of Section 3. Thus, the Hessian of \( L_I \) can be obtained as a function of derivatives associated with the functions of the Bolza optimal control problem stated in Section 3. Fig. 3 shows the
sparsity pattern of a general NLP Lagrangian Hessian obtained from the discretization of the continuous-time Bolza problem using the Radau pseudospectral method.

6 Discussion

While perhaps not evident at first glance, the approach of Section 5 only requires differentiation of the much smaller and simpler functions of the continuous-time Bolza optimal control problem of Section 3 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 5.2 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, while 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 5.3. Specifically, from Eqs. (56) and (58) it is seen that the Hessian is obtained by differentiating the functions $L_I$ and $L_E$ with respect to the continuous-time state, control, and time at either the endpoints (in the case $L_E$) or the $N$ collocation points (in the case of $L_I$). Furthermore, because $L_E$ and $L_I$ are scalar functions, a variety of differentiation techniques can be utilized 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 5 provide the most efficient way to compute the NLP derivative functions.
7 Example

Consider the following variation of the orbit-raising optimal control problem taken from Ref. 33.

Minimize the cost functional

\[ J = -r(t_f) \]  \hfill (69)

subject to the dynamic constraints

\[
\begin{align*}
\dot{r} &= v_r, \\
\dot{\theta} &= v_\theta/r, \\
\dot{v}_r &= v_\theta^2/r - \mu/r^2 + au_1, \\
\dot{v}_\theta &= -v_r v_\theta/r + au_2,
\end{align*}
\]  \hfill (70)

the equality path constraint

\[ c = u_1^2 + u_2^2 = 1, \]  \hfill (71)

and the boundary conditions

\[
\begin{align*}
b_1 &= r(0) - 1 = 0, \\
b_2 &= \theta(0) = 0, \\
b_3 &= v_r(0) = 0, \\
b_4 &= v_\theta(0) - 1 = 0, \\
b_5 &= v_r(t_f) = 0, \\
b_6 &= \sqrt{\mu/r(t_f)} - v_\theta(t_f) = 0,
\end{align*}
\]  \hfill (72)

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

\[ \equiv a(t) = \frac{T}{m_0 - \vert \dot{m} \vert t}. \]  \hfill (73)

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

\[
\begin{align*}
y(t) &= \begin{bmatrix} r(t) & \theta(t) & v_r(t) & v_\theta(t) \end{bmatrix}, \\
u(t) &= \begin{bmatrix} u_1(t) & u_2(t) \end{bmatrix}.
\end{align*}
\]
while the right-hand side function of the dynamics, the path constraint function, and the boundary condition function are given, respectively, as

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

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 6th 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. 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.

In order to see the effectiveness of the derivation of Section 5, in this example the Radau pseudospectral NLP was solved using IPOPT by either directly differentiating the NLP objective function, \(f\), the constraints, \(h\), and the Lagrangian, \(L\), or by differentiating the functions \(\phi\), \(g\), \(a\), \(c\), and \(b\) of the continuous-time Bolza optimal control problem as given in Eqs. (1)–(4), respectively and using the method derived in Section 5. When the NLP functions are directly differentiated and IPOPT is applied as a first-derivative NLP, the first derivatives of \(f\) and \(h\) are computed using either

(i) first forward-differencing;
(ii) the forward-mode object-oriented MATLAB automatic differentiator INTLAB.\textsuperscript{34}

When the NLP functions are directly differentiated and IPOPT is applied as a second-derivative NLP solver, the first derivatives of $f$ and $h$ and the second derivatives of $L$ are computed using

(iii) method (i) plus a second forward-difference to approximate the Hessian of $L$;

(iv) method (ii) plus the forward-mode object-oriented MATLAB automatic differentiator INTLAB.\textsuperscript{34} to compute the Hessian of $L$.

When the Bolza optimal control functions $\phi$, $g$, $a$, $c$, and $b$ are differentiated and IPOPT is used as a first-derivative NLP solver, the first derivatives of $\phi$, $g$, $a$, $c$, and $b$ are computed using either

(v) first forward-differencing of $\phi$, $g$, $a$, $c$, and $b$;

(vi) analytic differentiation of $\phi$, $g$, $a$, $c$, and $b$.

When the Bolza optimal control functions $\phi$, $g$, $a$, $c$, and $b$ are differentiated and IPOPT is used as a second-derivative NLP solver, the first and second derivatives of $\phi$, $g$, $a$, $c$, and $b$ are computed using either

(vii) the method of (v) plus second forward-differencing to approximate the second derivatives of $\phi$, $g$, $a$, $c$, and $b$;

(viii) analytic differentiation to obtain the second derivatives of $\phi$, $g$, $a$, $c$, and $b$.

Table 1 summarizes the different derivative methods (i)–(viii) and the usages of IPOPT for this example, while the Jacobian and Hessian sparsity patterns for this example are shown, respectively, in Figs. 5 and 6. 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 5, only the functions of the continuous-time Bolza problem shown in Section 3 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 5. It is noted that the NLP constraint Jacobian and Lagrangian Hessian sparsity patterns, shown in Figs. 5 and 6, are found using the derivation given in Section 5 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. Finally, for each of the methods (i)–(viii) the values in the NLP derivatives matrices were verified using both (a) the derivative checker built into IPOPT and (b) a comparison between the derivatives obtained using the method of Section 5 and the derivatives obtained using the automatic differentiator INTLAB.

The example was solved using \( K = (16, 32, 64, 128, 256, 512) \) equally-spaced mesh intervals with 4 LGR points in each mesh interval. A typical solution to obtained for this example is shown in Fig. 4. Tables 2 and 3 summarize the computational performance using methods (i)–(iv) and methods (v)–(viii), respectively. In particular, Tables 2 and 3 show that differentiating the functions of the Bolza optimal control problem and using the approach of Section 5 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 (i) takes 2246 s while method (v) takes 27.1 s whereas method (iii) takes 5871 s while method (vii) 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 (i) and (ii) to methods (iii) and (iv) 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 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 5 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 which degrades 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 5 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 5 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 two 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.
8 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.

Acknowledgments

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

References


Figure 1: Structure of Composite Radau Pseudospectral Differentiation Matrix Where the Mesh Consists of $K$ Mesh Intervals.

(1) Block $k$ is of Size $N_k$ by $N_k+1$
(2) Zeros Except in Blocks
(3) Total Size $N$ by $N+1$
Figure 2: General Jacobian sparsity pattern for RPM.
Figure 3: General Hessian sparsity pattern for RPM.
Figure 4: Solution to Orbit-Raising Optimal Control Problem for 16 Equally-Spaced Sections of 4 LGR Points Each ($N = 64$).
Figure 5: NLP Constraint Jacobian Sparsity Pattern for Example Problem.
Figure 6: NLP Lagrangian Hessian Sparsity Pattern for Example Problem.
Table 1: Summary of Different Derivative Methods Used to Solve Example with the NLP Solver *IPOPT*.

<table>
<thead>
<tr>
<th>Method Used</th>
<th>IPOPT Mode</th>
<th>Functions Being Differentiated</th>
<th>Derivative Approximation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>First Derivative</td>
<td>NLP Functions $f$ and $h$</td>
<td>Finite-Differencing</td>
</tr>
<tr>
<td>(ii)</td>
<td>First Derivative</td>
<td>NLP Functions $f$ and $h$</td>
<td>Automatic Differentiation</td>
</tr>
<tr>
<td>(iii)</td>
<td>Second Derivative</td>
<td>NLP Functions $f$ and $h$</td>
<td>Finite-Differencing</td>
</tr>
<tr>
<td>(iv)</td>
<td>Second Derivative</td>
<td>NLP Functions $f$ and $h$</td>
<td>Automatic Differentiation</td>
</tr>
<tr>
<td>(v)</td>
<td>First Derivative</td>
<td>Optimal Control Functions $\phi$, $g$, $a$, $c$, and $b$</td>
<td>Finite-Differencing</td>
</tr>
<tr>
<td>(vi)</td>
<td>First Derivative</td>
<td>Optimal Control Functions $\phi$, $g$, $a$, $c$, and $b$</td>
<td>Analytic Derivatives</td>
</tr>
<tr>
<td>(vii)</td>
<td>Second Derivative</td>
<td>Optimal Control Functions $\phi$, $g$, $a$, $c$, and $b$</td>
<td>Finite-Differencing</td>
</tr>
<tr>
<td>(viii)</td>
<td>Second Derivative</td>
<td>Optimal Control Functions $\phi$, $g$, $a$, $c$, and $b$</td>
<td>Analytic Derivatives</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Derivative Method for IPOPT</th>
<th>$K$</th>
<th>$N$</th>
<th>NLP Major Iterations</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method (i): First NLP Derivatives Using Finite Differencing</td>
<td>16</td>
<td>64</td>
<td>141</td>
<td>36.2</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>121</td>
<td>57.9</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>125</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>176</td>
<td>435</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>212</td>
<td>1051</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>196</td>
<td>2246</td>
</tr>
<tr>
<td>Method (ii): First NLP Derivatives Using INTLAB</td>
<td>16</td>
<td>64</td>
<td>118</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>115</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>136</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>156</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>158</td>
<td>19.4</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>143</td>
<td>31.2</td>
</tr>
<tr>
<td>Method (iii): First and Second NLP Derivatives Using Finite-Differencing</td>
<td>16</td>
<td>64</td>
<td>32</td>
<td>44.5</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>35</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>46</td>
<td>263</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>49</td>
<td>708</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>56</td>
<td>2058</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>67</td>
<td>5871</td>
</tr>
<tr>
<td>Method (iv): First and Second NLP Derivatives Using INTLAB</td>
<td>16</td>
<td>64</td>
<td>33</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>39</td>
<td>5.4</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>43</td>
<td>17.1</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>77</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>Out of Memory</td>
<td>Out of Memory</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>Out of Memory</td>
<td>Out of Memory</td>
</tr>
</tbody>
</table>
Table 3: Differentiation of the Bolza Optimal Control Problem Functions Together with the Approach of Section 5 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*.

<table>
<thead>
<tr>
<th>Derivative Method for IPOPT</th>
<th>$K$</th>
<th>$N$</th>
<th>NLP Major Iterations</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method (v): First Optimal Control Problem Derivatives Using Finite-Differencing</td>
<td>16</td>
<td>64</td>
<td>144</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>114</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>132</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>157</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>152</td>
<td>13.7</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>164</td>
<td>27.1</td>
</tr>
<tr>
<td>Method (vi): First Optimal Control Problem Analytic Derivatives</td>
<td>16</td>
<td>64</td>
<td>113</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>106</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>132</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>154</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>150</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>136</td>
<td>21.1</td>
</tr>
<tr>
<td>Method (vii): First and Second Optimal Control Problem Derivatives Using Finite-Differencing</td>
<td>16</td>
<td>64</td>
<td>31</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>35</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>45</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>48</td>
<td>4.9</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>56</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>60</td>
<td>23.1</td>
</tr>
<tr>
<td>Method (viii): First and Second Optimal Control Problem Analytic Derivatives</td>
<td>16</td>
<td>64</td>
<td>30</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>128</td>
<td>35</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>256</td>
<td>41</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>512</td>
<td>42</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1024</td>
<td>49</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2048</td>
<td>60</td>
<td>10.9</td>
</tr>
</tbody>
</table>
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*.

<table>
<thead>
<tr>
<th>$K$</th>
<th>$N$</th>
<th>NLP Variables</th>
<th>NLP Constraints</th>
<th>Jacobian Non-Zeros</th>
<th>Jacobian Density (%)</th>
<th>Hessian Non-Zeros</th>
<th>Hessian Density (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>64</td>
<td>390</td>
<td>321</td>
<td>2498</td>
<td>2.00</td>
<td>1925</td>
<td>1.27</td>
</tr>
<tr>
<td>32</td>
<td>128</td>
<td>774</td>
<td>641</td>
<td>4994</td>
<td>1.01</td>
<td>3845</td>
<td>0.642</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>1542</td>
<td>1281</td>
<td>9986</td>
<td>0.506</td>
<td>7685</td>
<td>0.323</td>
</tr>
<tr>
<td>128</td>
<td>512</td>
<td>3078</td>
<td>2561</td>
<td>19970</td>
<td>0.253</td>
<td>15365</td>
<td>0.162</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>6150</td>
<td>5121</td>
<td>39938</td>
<td>0.127</td>
<td>30725</td>
<td>0.0812</td>
</tr>
<tr>
<td>512</td>
<td>2048</td>
<td>12294</td>
<td>10241</td>
<td>79874</td>
<td>0.0634</td>
<td>61445</td>
<td>0.0407</td>
</tr>
</tbody>
</table>