Optimal Control within the Context of Multidisciplinary Design, Analysis, and Optimization

Robert D. Falck* and Justin S. Gray†
NASA John H. Glenn Research Center, Cleveland, OH, 44135

Multidisciplinary design, analysis and optimization involves modeling the interactions of complex systems across a variety of disciplines. The optimization of such systems can be a computationally expensive exercise with multiple levels of nested nonlinear solvers running under an optimizer. The application of optimal control in project development often involves performing trajectory optimization for fixed vehicle designs or parametric sweeps across some key vehicle properties. This information is then relayed to the subsystem design teams who update their designs and relay some bulk characteristics back to the trajectory optimization procedure. This iteration is then repeated until the design closes. However, with increasing interest in more tightly coupled systems, such as electric and hybrid-electric aircraft propulsion and boundary layer ingestion, this process is prone to ignore subtle coupling between vehicle subsystem designs and vehicle operation on a given mission. Integrating trajectory optimization into a tightly coupled multidisciplinary design procedure can be computationally prohibitive, depending on the complexity of the subsystem analyses and the optimal control technique applied. To address these issues, a new optimal control software tool, Dymos, has been developed. Dymos is built upon NASA's OpenMDAO software and can leverage its capabilities to efficiently compute gradients for the optimization and optimize complex models in parallel on distributed memory systems. This report explains the numerical methods employed in Dymos and provides several use cases that demonstrate its performance on traditional optimal control problems and improvements in situations that often arise in multidisciplinary optimization.

Nomenclature

$[A]_i, [B]_i$	=	Hermite interpolation matrices
$[A]_d, [B]_d$	=	Hermite differentiation matrices
$\bar{d}$	=	design parameter vector
D	=	drag
$[D]$	=	Lagrange differentiation matrix
E	=	specific energy
f_{ode}	=	ordinary differential equation function
g_0	=	initial boundary constraints
g_f	=	final boundary constraints
h	=	altitude
I_{sp}	=	specific impulse
m	=	mass
M	=	Mach
L	=	lift
$[L]$	=	Lagrange interpolation matrix
p	=	path constraints
r	=	range
S	=	arc-length
S_{ref}	=	aerodynamic reference area
sos	=	speed of sound
t	=	time

*Aerospace Engineer, Mission Architecture and Design Branch, rfalck@nasa.gov, AIAA Member.
†Aerospace Engineer, Propulsion Systems Analysis Branch, justin.s.gray@nasa.gov, AIAA Member.

t_0	=	phase initial time
t_f	=	phase final time
t_p	=	phase time duration
t_{seg}	=	segment duration
T	=	thrust
v	=	speed
W	=	weight
$\bar{u}$	=	dynamic control variable vector
$\bar{x}$	=	state variable vector
α	=	angle of attack
γ	=	flight path angle
ρ	=	atmospheric density

Subscripts

$()_{lb}$	=	lower bound
$()_{ub}$	=	upper bound
$()_a$	=	values at all nodes
$()_d$	=	values at discretization nodes
$()_c$	=	values at collocation nodes
$\dot{()}$	=	time derivative
$()'$	=	polynomial approximation of time derivative

I. Introduction

The application of optimal control to a multidisciplinary design and optimization (MDO) process has typically involved multiple layers of optimization, with each discipline optimizing the design of its subsystem components given current inputs from other disciplines. The overall design is then optimized by iterating through various designs at the system level with a sequential optimization approach. This procedure can be effective when designs are not tightly coupled, but it encounters problems in the face of tightly coupled systems. Tedford and Martins demonstrated that MDO approaches which utilize optimization problems at multiple levels, such as collaborative optimization, were typically slower to converge to a solution than monolithic approaches which utilize a single overarching optimization such as Simultaneous Analysis and Design (SAND) or Multidisciplinary Design Feasible (MDF) [1]. These findings were later corroborated by Gray, Moore, Hearn, and Naylor [2].

A monolithic approach yields optimization problems which can involve very large numbers of design variables and constraints. This large design space lends itself to gradient-based optimization techniques, which can analyze such large design spaces more efficiently than gradient-free optimization approaches. Unfortunately, tools and processes used for collaborative optimization are generally not designed to pass design sensitivities between disciplines. As a result, the only way to obtain derivative information in such procedures is to finite-difference across the entire model. This is a potentially expensive undertaking that will often yield inaccurate gradient information for the optimizer. Furthermore, it has been demonstrated that gradient-based optimization of multidisciplinary systems demands a level of accuracy in the gradients that is not necessarily achievable from finite differencing, especially when nonlinear solvers are nested in the analysis[1, 3]. OpenMDAO is designed to overcome these difficulties by providing a framework for assembling complex optimization problems from many components, and by providing an efficient architecture for computing gradients across such large systems [4].

There have been several successful applications of MDO to coupled vehicle design and trajectory optimization problems. Hwang demonstrated the multidisciplinary optimization of a small satellite which maximized data transfer by optimizing the roll angle history in the face of thermal and power constraints [5]. Kao, Hwang, Martins, Moore, and Gray demonstrated the optimization of a commercial aircraft trajectory by using the steady flight assumption to re-parameterize the aircraft speed and altitude as control variables and model them via B-splines while integrating the aircraft fuel weight with an Euler-stepping approach [6]. These approaches demonstrated the potential of coupling vehicle design and trajectory optimization using a monolithic approach. Unfortunately, Hwang and Munster demonstrated that explicit time integration in OpenMDAO tends to suffer from a performance perspective compared to implicit approaches due to a large number of non-vectorized calculations [7]. Stanford University's SUAVE software [8, 9] has successfully

applied a Chebyshev-based pseudo-spectral approach to the conceptual aircraft design problem, but is domain-specific and lacks the passing of derivatives between models.

In order to provide a general optimal control capability within OpenMDAO, the authors developed a new software package, Dymos. OpenMDAO provides the interface to nonlinear programming packages like SNOPT [10], and Dymos provides methods for transcribing continuous optimal control problems into nonlinear programming problems. Its user-facing decomposition of optimal control problems is influenced by NASA's OTIS optimal control software [11, 12]. In order to minimize the number of times a system model needs to be executed during optimization, the framework implements two implicit collocation optimal control techniques: High-order Gauss-Lobatto collocation [13] and a Radau Pseudospectral Method [14]. These two techniques have been used extensively in recent decades to solve a variety of optimal control problems, typically in the form of aerospace vehicle trajectory optimization. Each technique has strengths and weaknesses that may dictate their use in a variety of situations. In addition, Dymos is designed with a modular approach that allows for new transcriptions to be developed within the same framework.

II. Numerical Methods

Implicit collocation techniques are uniquely well-suited to gradient based optimization with analytic derivatives. Explicitly propagating the equations of motion via time-marching in a manner that preserves analytic total derivatives across the integration requires a fixed number of analysis points in time with enough density to accurately capture the system dynamics [7]. This limits analysis of the dynamics to a fixed number of time-steps for the integration, subjecting the analysis to potentially large errors in some time-steps with high rates or burdening the analysis with an excessive number of time-steps. Conversely, implicit collocation schemes can achieve the same accuracy with fewer analysis points in a grid which is fixed throughout an optimization. The following sections provide an overview of the implicit techniques currently employed in Dymos.

A. General Approach

Dymos, like other optimal control software, breaks a trajectory (state time-history) into one or more *phases*. The implicit collocation methods used in Dymos further decompose each phase into one or more *segments*. On each segment, each state or control variable is assumed to be C^1 continuous such that its value in time can be modeled as a polynomial. Values of the states and controls are assigned at certain points or *nodes* within each segment, referred to as the state discretization nodes and control discretization nodes, respectively. Within each phase the design variables are the initial time (t_0) and duration (t_p), the state variables at the state discretization nodes ($\bar{x}_d$), the dynamic control variables at the control discretization nodes ($\bar{u}_d$), and the design parameters ($\bar{d}$). In addition to simple bounds on the independent design variables, the user may also constrain the value any variable, including auxiliary outputs of the ODE system via nonlinear boundary or path constraints within a phase. The objective can be set to minimize or maximize any design variable or ODE system output. The overall optimal control problem can thus be stated as shown in Table 1.

Table 1 General form of the optimal control problem solved by Dymos

Minimize	$\mathrm{J} = f_{obj}(\bar{x}, t, \bar{u}, \bar{d})$
Subject to :	
State Dynamics :	$\dot{\bar{x}} = f_{ode}(\bar{x}, t, \bar{u}, \bar{d})$
Time :	$t_{lb} \leq t \leq t_{ub}$
State Variables :	$\bar{x}_{lb} \leq \bar{x} \leq \bar{x}_{ub}$
Dynamic Controls :	$\bar{u}_{lb} \leq \bar{u} \leq \bar{u}_{ub}$
Design Parameters :	$\bar{d}_{lb} \leq \bar{d} \leq \bar{d}_{ub}$
Initial Boundary Constraints :	$\bar{g}_{0,lb} \leq g_0(\bar{x}_0, t_0, \bar{u}_0, \bar{d}) \leq \bar{g}_{0,ub}$
Final Boundary Constraints :	$\bar{g}_{f,lb} \leq g_f(\bar{x}_f, t_f, \bar{u}_f, \bar{d}) \leq \bar{g}_{f,ub}$
Path Constraints :	$\bar{p}_{lb} \leq p(\bar{x}, t, \bar{u}, \bar{d}) \leq \bar{p}_{ub}$

Each phase within Dymos is an OpenMDAO group. Phases may be collected in larger groups called Trajectories, wherein time, state, and control defect constraints can be enforced between phases to provide a continuous trajectory. Trajectories, in turn, are not necessarily the top-level system in an OpenMDAO analysis/optimization. This gives the user the ability to use multiple trajectory analyses within the optimization of a system design, as depicted in the extended design structure matrix (XDSM) [15] in Figure 1.

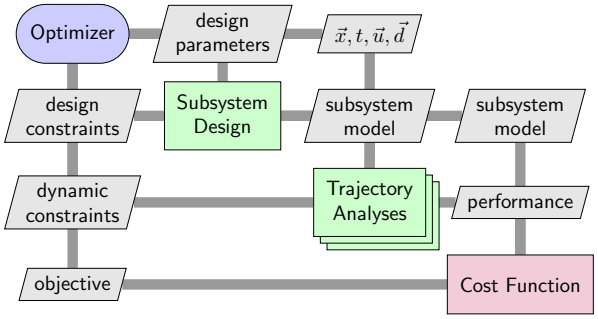


Fig. 1 A notional XDSM showing an optimization involving subsystem design, which feeds one or more trajectory optimizations, with some overall cost function.

B. Pseudospectral Optimal Control Transcriptions

The optimal control transcriptions employed in Dymos were chosen for two reasons. First, they provide a way of accurately posing an optimal control problem wherein the number of calculations performed in each iteration remains unchanged in each iteration. This is required for OpenMDAO's derivative calculation scheme to be able to assemble the total Jacobian from the partial derivatives of each constituent component. Secondly, the techniques chosen provide sparse Jacobian structures, which allows larger problems to be tackled more efficiently by reducing computation and memory requirements during evaluation of the total derivatives.

1. High Order Legendre-Gauss-Lobatto Transcription

The high-order Legendre-Gauss-Lobatto (LGL) method was developed by Herman and Conway and is primary method used in OTIS [11, 13]. It is a general extension of Hermite-Simpson collocation to higher orders [12]. The segments within a phase are discretized at the LGL nodes such that the total number of nodes within each segment is odd. Nodes with an even index (starting at an index of zero for the first node) are the *state discretization* nodes. Dynamic controls are provided at the so-called *control discretization nodes*. For LGL collocation, the control discretization subset includes all nodes within the phase. A single evaluation of an LGL Phase within Dymos proceeds as follows: First, the user or optimizer provides values of the design variables (t_0, t_p, $\bar{x}_d$, $\bar{u}_d$, and $\bar{d}$). The OpenMDAO system which provides the state dynamics is evaluated at the state discretization nodes, providing the state time derivatives at the state discretization nodes:

$$\dot{\bar{x}}_d = f_{ode}(t_d, \bar{x}_d, \bar{u}_d, \bar{d}) \tag{1}$$

The states and state-rates are then interpolated to the collocation nodes using Hermite-interpolation [16]:

$$\bar{x}_c = [A_i]\, x_d + \frac{t_{seg}}{2}\,[B_i]\,\dot{\bar{x}}_d \tag{2}$$

$$\bar{x}'_c = \frac{2}{t_{seg}}\,[A_d]\, x_d + [B_d]\,\dot{\bar{x}}_d \tag{3}$$

where t_{seg} is the duration of the segment in which each node lies. At this point the system providing the state dynamics is evaluated again, at the collocation nodes:

$$\dot{\bar{x}}_c = f_{ode}(t_c, \bar{x}_c, \bar{u}_c, d) \tag{4}$$

Now we have two independent values for the state rates: the "true" values as determined by the ODE function, and the approximate values obtained from the slope of the interpolating polynomials. The difference between these values, in normalized segment time space, constitutes the *collocation defects* (Δ) for each state in the phase. In the transcribed nonlinear programming problem, these values are constrained be zero.

$$\bar{\Delta} = \bar{x}'_{col} - \frac{\bar{t}_{seg}}{2}\dot{\bar{x}}_{col} = 0 \tag{5}$$

If the collocation *grid* (the number of segments in the phase, and the polynomial order along each segment) provides enough degrees of freedom, then the discretized time history of the states and controls will be capable of accurately representing the continuous time solution to the optimal control problem. Figure 2a depicts the state discretization and collocation nodes in for a simple grid.

2. Radau Pseudospectral Transcription

The Radau Pseudospectral Method (RPM) has been extensively developed and is the primary method used by the GPOPS trajectory optimization tool[14, 17]. The RPM uses the Legendre-Gauss-Radau (LGR) nodes to discretize the problem within each segment. For a segment with n LGR nodes, the states and controls are discretized such that values are provided at all n nodes. The first $n-1$ nodes within each segment are taken to be the collocation nodes. In order to maintain consistency with the LGL approach, whereby a segment with n^{th}-order state transcription has $(n-1)^{th}$-order control transcription, controls are discretized at the collocation nodes and interpolated to the end of the segment to allow enforcement of boundary constraints.

The execution of the LGR phase proceeds in a way similar to the LGL phase, with a few key differences. First, the equations of motion are evaluated at all nodes (a) simultaneously, since the collocation nodes are just a subset of the state discretization nodes.

$$\dot{\bar{x}}_a = f_{ode}(t_a, \bar{x}_a, \bar{u}_a, \bar{a}) \tag{6}$$

The state rates at all nodes are approximated using a Lagrange differentiation matrix ($[D]$) [18].

$$\bar{x}'_a = \frac{2}{t_{seg}}[D]\,\bar{x}_a \tag{7}$$

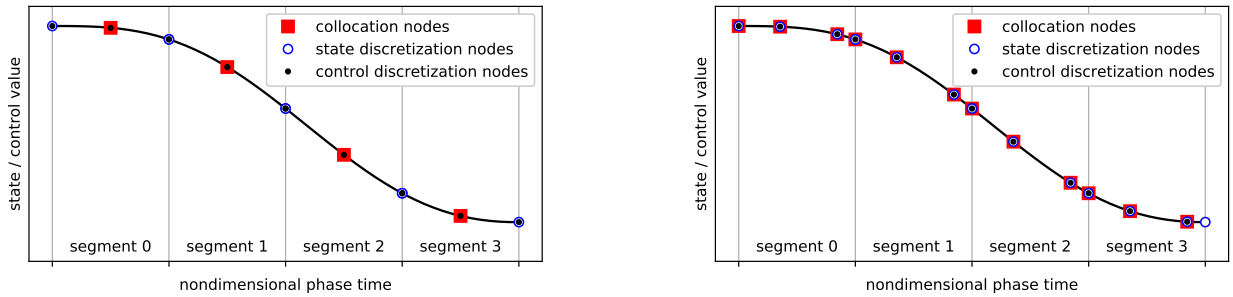
Equation (5) can then be evaluated by extracting the collocation subset values from Equation (7). Figure 2b depicts the state discretization and collocation node subsets in for a simple grid.

3. Comparisons of the two pseudospectral methods in Dymos

In general, the two pseudospectral methods used in Dymos have similar performance in many situations, but there are subtleties in the implementation that may make one method more suitable than the other for a specific problem. As Figure 2 depicts, for the same number of segments with the same transcription order, the Radau Pseudospectral method has more design variables and constraints than the equivalent problem using Legendre-Gauss-Lobatto transcription. However, the Legendre-Gauss-Lobatto implementation used in Dymos requires two separate evaluations of the ordinary differential equations governing the dynamics in each iteration. If there is excessive overhead associated with the evaluation of the ODE, then the RPM may provide better performance.

The interpolation step can also be problematic when solving implicit systems within the ODE. Because the LGL method interpolates the discretized states to the collocation nodes and then evaluates the ODE there, a poor initial guess of the dynamics may result in interpolated states that result in the inability to converge the underlying implicit systems.

Finally, since the RPM provides state values at all nodes as design variables, path-constraining them can be accomplished with simple bounds. In the LGL method, a nonlinear path constraint is required to ensure that interpolated values at the collocation nodes are constrained, since simple bounds constraints would only limit values at the state discretization nodes.



(a) Legendre-Gauss-Lobatto **(b) Radau-Pseudospectral**

Fig. 2 An example of the state interpolating polynomials for the Legendre-Gauss-Lobatto transcription (left) and Radau Pseudospectral transcription (right) with four equally-spaced, 3rd-order segments.

C. Model Derivative Calculations

With direct transcription, optimal control problems are solved via a nonlinear programming approach most often addressed using gradient based optimization. As the name implies, gradient based optimization makes use of gradient (also known as derivative) information to navigate very large design spaces. When using a gradient based optimization method, the majority of the computational cost comes from derivative calculations, and hence the efficiency of those calculations governs the overall efficiency of the method. There are two main approaches to computing derivatives for optimization: numerical approximations and analytic methods. Broadly speaking the numerical approximations are much easier to implement but the analytic methods offer far superior computational performance.

Numerical schemes for computing derivatives use either finite-difference or complex-step [19] techniques to compute approximations for derivatives of a model by examining the change in objective and constraint values given small increments to the design variable values. For the finite-difference method, the increments to the design variables are taken in the real plane. For example, a first-order forward difference approximation of the objective gradient, with error proportional to $\Delta\bar{x}$ is computed as:

$$\frac{dJ}{d\bar{x}} \approx \frac{f_{obj}(\bar{x}+h) - f_{obj}(\bar{x})}{h} \tag{8}$$

Finite-difference is the easiest method to use, but also the least accurate since the small h needed to drive the approximation error down also introduces so-called subtractive cancellation [19] that limits the accuracy. More accurate approximations can be achieved with the complex-step technique, which offers second order accuracy and does not suffer the subtractive cancellation problem:

$$\frac{dJ}{d\bar{x}} \approx \frac{Im\left[f_{obj}(\bar{x}+hi)\right]}{h} \tag{9}$$

The complex-step method offers error that scales with h^2 and the formula does not suffer from the limits of subtractive cancellation, so the step can be made arbitrarily small and the resulting derivative is effectively exact. The improved accuracy comes at the cost of increased implementation complexity, since your model now must accept complex inputs, and increased cost since complex arithmetic is more expensive than real arithmetic. Since both numerical methods require computing solutions with increments on the design variables, these methods generally have computational cost that scales linearly with the number of design variables — except in certain special cases that we will describe in more detail in Section III.C.

Analytic derivative methods offer greater computational efficiency than the numerical approach — and superior accuracy than the finite-difference method — by solving for the total derivatives needed by the optimizer using only relatively inexpensive and easy to compute partial derivatives of the internal calculations of the model. The partial derivatives assembled into a linear system that can be solved to compute the needed total derivatives analytically. There are two techniques for constructing this linear system: the direct (i.e. forward) method requires one linear solution per design variable — analogous to the numerical approximation methods – and the adjoint (i.e. reverse) methods requires one linear solution per objective and constraint. Since these methods are so central to the functionality of Dymos, their derivation is very briefly summarized here. Much more complete and detailed explanations of the analytic methods can be found in the work of Martins and Hwang [20, 21].

Consider the general optimization problem given in Table 1, with a set of design variables $(\bar{x}, t, \bar{u}, \bar{d})$ that we can concatenate into a single design variable vector: X. For any specific set of values for X, the objective and constraint functions can be an arbitrarily complex set of calculations that may include some internal implicit calculations — $R(X, Y) = 0$ — that must be converged in order to evaluate the quantity of interest. The total derivative needed by the optimizer must account for the change in in these internal variables, given a change in X. For example, the objective gradient, $d\mathrm{J}/dX$, can be computed as:

$$\frac{d\mathrm{J}}{dX} = \frac{\partial \mathrm{J}}{\partial X} + \frac{\partial \mathrm{J}}{\partial Y}\frac{dY}{dX} \tag{10}$$

Two of the terms in the Equation (10) require only cheap partial derivatives, but the dY/dX term is a total derivative that accounts for the change in the converged internal state variables with respect to changes in X. dY/dX is difficult and expensive to compute directly, but we can use the governing equations to compute it indirectly in a more efficient manner.

$$R(X, Y) = 0 \tag{11}$$

$$\frac{dR}{dX} = \frac{\partial R}{\partial X} + \frac{\partial R}{\partial Y}\frac{dY}{dX} = 0 \tag{12}$$

$$\frac{dY}{dX} = -\left[\frac{\partial R}{\partial Y}\right]^{-1}\frac{\partial R}{\partial X} \tag{13}$$

Equation (13) is a linear system that must be solved once for each variable in the X vector in order to compute dY/dX. Once that is known, it can be substituted back into Equation (10) to compute the total derivatives for optimization. Together Equations (10) and (13) comprise the direct method, and the linear system formed by Equation (13) causes the cost of the direct method to scale with the size of the design variable vector X. The adjoint method can be derived through some further manipulation of equations Equations (10) and (13) into a different form:

$$\frac{d\mathrm{J}}{dX} = \frac{\partial \mathrm{J}}{\partial X} - \underbrace{\frac{\partial \mathrm{J}}{\partial Y}\left[\frac{\partial R}{\partial Y}\right]^{-1}}_{\psi^T}\frac{\partial R}{\partial X} \tag{14}$$

$$\frac{d\mathrm{J}}{dX} = \frac{\partial \mathrm{J}}{\partial X} + \psi^T\frac{\partial R}{\partial X} \tag{15}$$

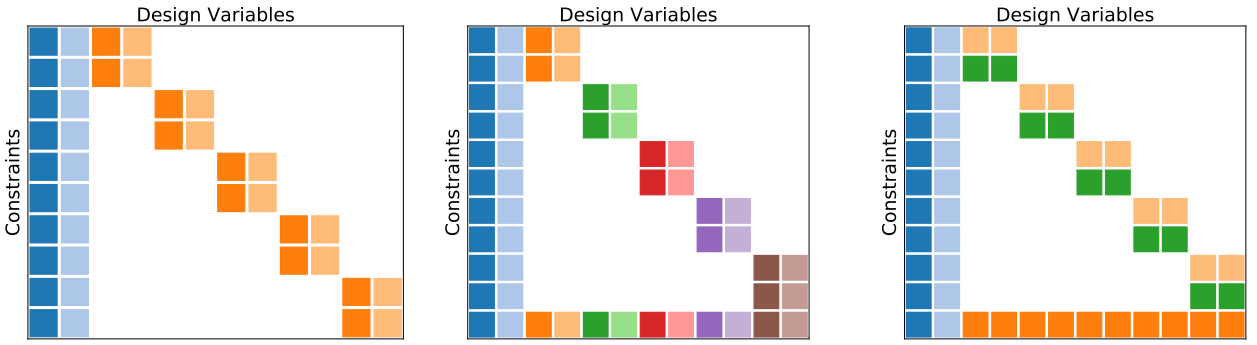
$$\psi = -\left[\frac{\partial R}{\partial Y}^T\right]^{-1}\left[\frac{\partial \mathrm{J}}{\partial Y}^T\right] \tag{16}$$

Equation (16) is a linear system that requires one solution per quantity of interest (objective or constraint). Each linear solution provides a specific ψ, termed the adjoint vector, which can be substituted back into Equation (15) to compute the total derivatives of one objective or constraint with respect to all design variables. Since Equation (16) is solved once per quantity of interest, the adjoint method scales with the number of constraints in an optimization problem.

Traditionally, application of direct or adjoint analytic methods for derivatives requires an ad-hoc and one-off implementation that can be somewhat labor intensive to build. Due to the effort required, a given implementation is typically uni-directional, supporting either the direct or adjoint methods, but not both. However, recent theoretical developments by Martins and Hwang [20, 21] have provided a more generalized approach that has been implemented in NASA's OpenMDAO framework, eliminating the need for one-off implementations of the analytic derivative methods. As-such the implementation barrier to adopting analytic derivative methods has been significantly lowered and it has also become possible to apply a combination of forward and adjoint methods on a single problem to create a bi-directional analytic derivative approach.

The pseudospectral methods described in Section II.B are well known to offer computational efficiency in a forward differentiation mode due to the structure of the problem. To understand why, consider a typical total-derivative Jacobian matrix for a pseudospectral optimal control problem in Figure 3a.

By decomposing each trajectory phase into multiple segments, the defect constraints within in each segment are only dependent upon time, state, control, and design parameter values which affect that particular segment. Because of this, in a forward method —either numerical or forward analytic — multiple design variables can be operated on simultaneously and hence multiple columns of the total-derivative Jacobian can be computed simultaneously. We group to each set of variables that can be worked with simultaneously into a "color". With a numerical approach, the number



(a) This total-Jacobian sparsity pattern corresponds to four colors in forward mode.

(b) This total-Jacobian sparsity pattern corresponds to twelve colors in forward mode.

(c) The total-Jacobian pattern from Figure 3b corresponds to only five colors in bi-directional mode.

Fig. 3 Notional total-derivative Jacobian matrices for a typical pseudospectral optimal control problem (3a) and a total-derivative Jacobian for a pseudospectral optimal control problem with a constraint on some post-processing analysis which is reliant on the entire trajectory (3b). Each color corresponds to one perturbation of design variables and constraint vector evaluation when finite differencing, or one linear solve when using the unified derivatives approach. Figure 3c shows sparsity pattern with a dense row that uses OpenMDAO's bi-directional coloring algorithm to significantly reduce the number of linear solves needed to compute the total-derivative Jacobian.

of total calls to the nonlinear model is equal to the number of colors. Similarly, with a forward analytic method number of colors gives the number of times that Equation (13) must be solved.

Generally speaking, coloring can also be applied in the reverse mode for analytic derivatives (note that there is no numerical analog to reverse mode). In reverse mode, each color represents a set of rows of the total derivative Jacobian that can be solved for simultaneously via a single solution to Equation (16). However, though the adjoint method has been shown to be useful in the application of shooting methods to optimal control[5, 6, 22], its application to pseudospectral problems is limited because of two properties. First, the large number of defect equality constraints mean that a pseudospectral problem will have far more constraints than an equivalent problem under a direct shooting approach, and because they are equality constraints they aren't well suited to aggregation methods. Second, the total-derivative Jacobian matrix of pseudospectral problems often includes dense columns. That is, there are a few variables in a pseudospectral problem that can potentially impact defect or path constraints in each segment — time and the design parameters.

In a typical psuedospectral approach, care must be taken to maintain a sparsity pattern that is separable under forward differentiation. A design goal of Dymos is to allow the use of optimal control within a multidisciplinary optimization environment, under a monolithic optimization approach. Ensuring all models in an analysis are compatible with the pseudospectral sparsity pattern is limiting. For instance, one might imagine a tool which takes the entirety of an aircraft state-time history and outputs a single scalar acoustic metric to be constrained. Rewriting such tools to accommodate the needs of the pseudospectral method is not likely to be feasible. In addition, approaches which have seen success in direct shooting approaches, such as path constraint aggregation via the Kreisselmeier-Steinhauser function [23], are also inadvisable in pseudospectral transcriptions due to the fact that they add dense rows to the total-derivative Jacobian matrix.

Figure 3b shows a notional total-derivative Jacobian for a pseudospectral problem with a dense row due to some "naive" post-processing analysis. The presence of one or more dense rows and dense columns would seemingly ruin the sparsity of pseudospectral methods here since the problem is not separable in purely forward nor reverse mode. However, the analytic derivative approach allows a bi-directional graph coloring scheme to be employed where some elements of the Jacobian matrix are computed in forward mode and others in reverse mode, as demonstrated by Figure 3c. The result can be significantly fewer colors than is achievable in a uni-directional approach. In Section III.C, we discuss how Dymos makes direct usage of the mixed direct (forward) and adjoint (reverse) analytic methods to achieve a significantly lower cost to compute analytic derivatives compared to more traditional single direction. Dymos leverages the bi-directional derivative functionality built into OpenMDAO, which provides both a bi-directional coloring algorithm and an automatic mechanism to implement the coloring for reduced total derivative computational cost.

III. Example Cases

Three example cases are used to demonstrate the performance of Dymos in various settings. The first is a traditional optimal control problem, which is used to demonstrate the performance of Dymos compared to the legacy OTIS software. Next, we introduce a problem that includes an iterative subprocess. These subprocesses are traditionally detrimental to performance since finite differencing across an iterative solver can result in inaccurate derivatives. The third example demonstrates a problem formulation wherein a single constraint depends on many design variables. This case is often encountered when coupling a pseudospectral optimization with a naive post-processing analysis that is not coded with the sparsity of the pseudospectral method in mind. The derivative assembly techniques used in OpenMDAO can minimize the performance impacts of such approaches.

A. Example 1: Performance in Traditional Optimal Control

The minimum-time climb problem was posed by Bryson, Desai, and Hoffman [24]. In this instance, we use a control parameterization in which angle of attack is the sole dynamic control. The optimal control problem is posed as:

$$
\begin{aligned}
&\text{minimize:} && J = t_f \\
&\text{subject to dynamics:} && \dot{v} = \frac{T\cos\alpha - D}{m} - g\sin\gamma && (17)\\
& && \dot{\gamma} = \frac{T\sin\alpha + L}{mv} - \frac{g\cos\gamma}{v} && (18)\\
& && \dot{h} = v\sin\gamma && (19)\\
& && \dot{r} = v\cos\gamma && (20)\\
& && \dot{m} = -\frac{T}{gI_{sp}} && (21)\\
&\text{and initial conditions:} && h_0 = 100\text{m} \\
& && M_0 = 0.3 \\
&\text{and final conditions:} && h_f = 20\text{km} \\
& && M_f = 1.0 \\
&\text{and path constraints:} && 100\text{m} \leq h(t) \leq 20\text{km}
\end{aligned}
$$

In this case, thrust is a function of vehicle altitude and airspeed, assuming that the aircraft is operating at full throttle, and is obtained from a table lookup function from the data provided by Bryson in [25]. The same reference also provides analytic fits for the vehicle aerodynamic coefficients as a function of Mach number. The XDSM of the ODE system is shown in Figure 4.

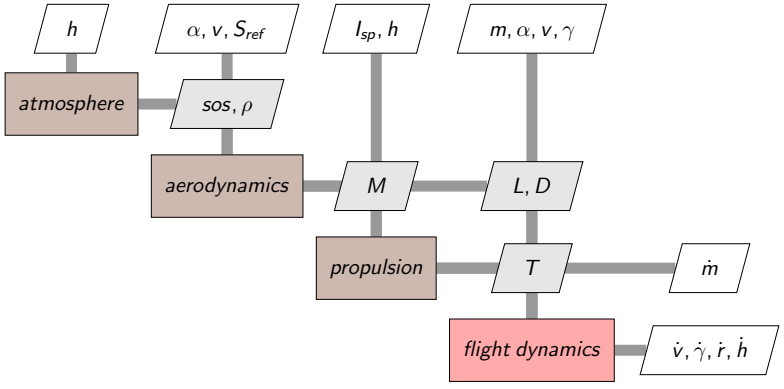


Fig. 4 The extended design structure matrix for the ODE of the supersonic minimum time-to-climb problem.

1. Results

Figure 5 shows a solution to the minimum time climb problem. The implicit collocation solution is plotted alongside an explicit time-history obtained by integrating the ODE from the initial conditions using the control profile found during the implicit optimization. These results were generated using Gauss-Lobatto collocation with 15 equally-spaced 3rd-order segments. The time required to perform the climb agrees to within one-second (0.3%) with the same case optimized using OTIS, and within 6 seconds (1.8%) of the results as provided by Bryson [25]. Discrepancies in the results are likely due to the difference is interpolations methods between this work and the interpolation given by Bryson.

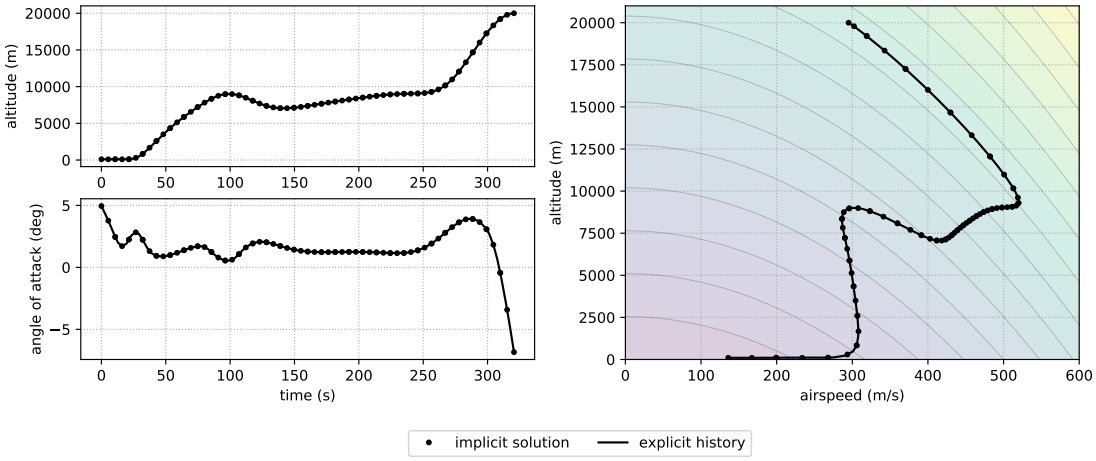


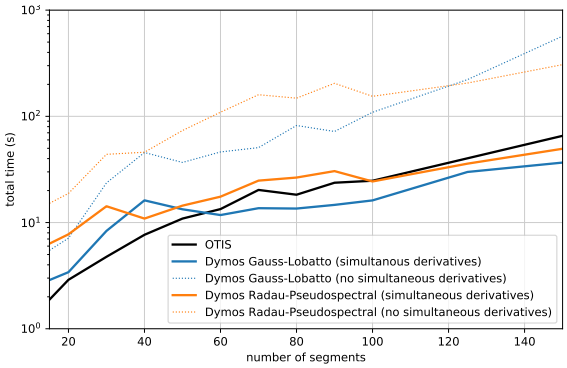
Fig. 5 The results of the minimum time to climb optimization from Dymos. The altitude vs. airspeed plot is superimposed on contours of constant specific energy.

The supersonic minimum time climb problem is a common example-case in optimal control software, and makes a useful case for performance comparison. In this section we compare the performance of Dymos to that of the legacy optimal control software OTIS4 for solving the minimum time climb problem. Where possible the authors have attempted to put the two pieces of software on equal footing, such as using the same settings for SNOPT, the same problem size, etc. OTIS utilizes a sparse finite-difference approach similar to that described by Betts [26]. Dymos can utilize analytic derivatives if provided by the user, but will assemble total derivatives even if some components provide the derivatives via finite-difference or complex-step. This approach gives the user the ability to implement models more quickly and then improve performance later by implementing analytic derivatives. Dymos is, to the authors' knowledge, the only general optimal control package to leverage the unified derivatives approach [20]. OpenMDAO allows the user to rearrange models from constituent components without the need to rederive the total derivatives for the optimizer.

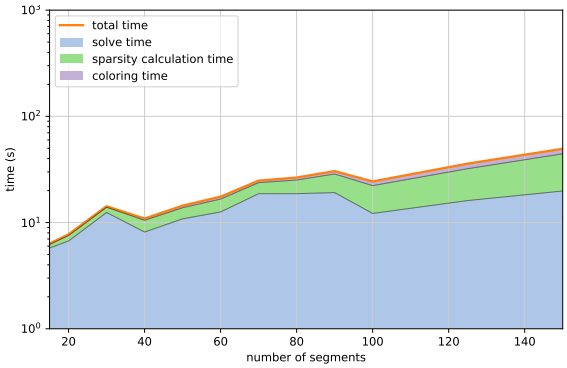
Figure 6a shows a performance comparison for Dymos and OTIS in solving the supersonic minimum time-to-climb problem. Dymos results are provided with and without simultaneous derivatives to show the performance benefit achieved by using the simultaneous derivatives capability of OpenMDAO. In this case, the performance of Legendre-Gauss-Lobatto collocation slightly outperforms the Radau Pseudospectral Method and the simultaneous derivative method reduces solution time by about an order of magnitude. Both methods are comparable in performance to OTIS and begin to outperform it as problem size increases.

Figure 6b provides a breakdown of the total time required to solve the problem in Dymos using Radau Pseudospectral transcription. As the number of segments used to solve the problem is increased, the time required to analyze the sparsity pattern and color the constraint Jacobian becomes significant. Note that figure 6a includes this time. Despite the extra time required to analyze the sparsity pattern, Dymos is still comparable to OTIS in terms of performance. In addition, if the structure of the optimization problem is known to remain the same between runs, the sparsity information can be cached to further improve performance.

There are a couple of reasons why Dymos is capable of showing performance comparable to OTIS despite the former being written in Python while the latter is in Fortran. First, because Dymos analyzes the sparsity of the total Jacobian at runtime using the sparsity pattern of each constituent calculation (component) as provided by the user, it is able to provide a more accurate total Jacobian pattern. Conversely OTIS, using finite-difference, doesn't have any



(a) Performance comparison between Dymos implicit techniques, with and without simultaneous derivatives, and OTIS



(b) Breakdown of time required to analyze the sparsity pattern, color the Jacobian, and solve the problem using Radau-Pseudospectral transcription.

Fig. 6 Dymos performance for the supersonic minimum time-to-climb problem.

knowledge of the particular dynamics in a given problem. Instead it conservatively assumes that every design variable in a given segment can *potentially* impact every constraint in the segment. This results in a block diagonal structure that is significantly more dense than that obtained using analytic derivatives with automated detection of the sparsity pattern.

B. Example 2: Optimal control with an implicit ODE

In this section we solve an optimal control problem in which the ODE system includes an iterative subprocess. Often times the dynamics of a problem rely on iterative solvers to find the value of some implicit variable. Examples typically seen include solving Kepler's equation and solving for geodetic latitude. Analytic derivatives are important in such instances because finite-differencing across the nonlinear solver has been demonstrated to produce incorrect derivatives and hinder progress of the optimization [27–29].

Another application of nonlinear solvers in the realm of optimal control is the removal of typical control variables from the optimization problem. The *differential inclusion* approach originally presented by Seywald [30] instead poses the state-time histories as "truth". The state dynamic equations of the system can then be used to solve for the corresponding control time-history. The trajectory is feasible when posed state time-history is in the reachable set of the controls. This approach to solving differential inclusion-based problems via pseudospectral methods was demonstrated by Fahroo and Ross [31].

OpenMDAO's application of nonlinear solvers is useful in this context. Rather than treating states as integrated quantities with a corresponding ordinary differential equation, they are provided as control variables. Dymos treats the provided time history of the state as "truth", and applies a nonlinear solver in the ODE system that solves for the traditional control variables. The traditional control variables can be bounded either in the nonlinear solver, in which case the optimizer is always presented with a valid control time history, or via path constraints to the optimizer, which allows the optimizer to search the problem outside of the feasible region. In general it is better to apply limits to the control via nonlinear path constraints in the optimizer, since an infeasible solution at the solver level will lead to nonconvergence of the solver. While OpenMDAO allows nonconvergence of the solver to trigger backtracking in the optimizer's line search, doing so significantly impairs performance.

To demonstrate an optimal control problem in which the dynamics includes an iterative solver, we again consider the supersonic minimum time-to-climb problem. In this instance, we use the energy state approximation employed by Bryson, Desai, and Hoffman [24]. Bryson, Desai, and Hoffman simplified that analysis by assuming quasi-steady flight, nearly level flight, and neglecting thrust normal to the flight path:

$$\dot{v} \approx 0 \tag{22}$$

$$\dot{\gamma} \approx 0 \tag{23}$$

$$\gamma \approx 0 \tag{24}$$

$$T \sin\alpha \approx 0 \tag{25}$$

With these assumptions, the system is reduced to two states, specific energy (E) and mass (m). Velocity is applied as a dynamic control. A nonlinear solver is used to iterate on α such that the generated lift is equal to weight:

$$\mathcal{R}(\alpha) = L(h, v, \alpha) - mg \tag{26}$$

The optimal control problem is posed as:

$$\begin{aligned}
&\text{minimize:} && J = t_f \\
&\text{subject to dynamics:} && \dot{E} = \frac{v(T-D)}{m} \\
& && \dot{m} = -\frac{T}{gI_{sp}} \\
&\text{and initial conditions:} && h_0 = 100\text{m} \\
& && M_0 = 0.3 \\
&\text{and final conditions:} && h_f = 20\text{km} \\
& && M_f = 1.0 \\
&\text{and path constraints:} && 100\text{m} \leq h(t) \leq 20\text{km} \\
& && 3\text{g} \leq \dot{v}(t) \leq 3\text{g}
\end{aligned}$$

The extended design structure matrix for the ODE system is pictured in Figure 7. Note the presence of the nonlinear Newton solver in the system which eliminates the residual from Equation (26).

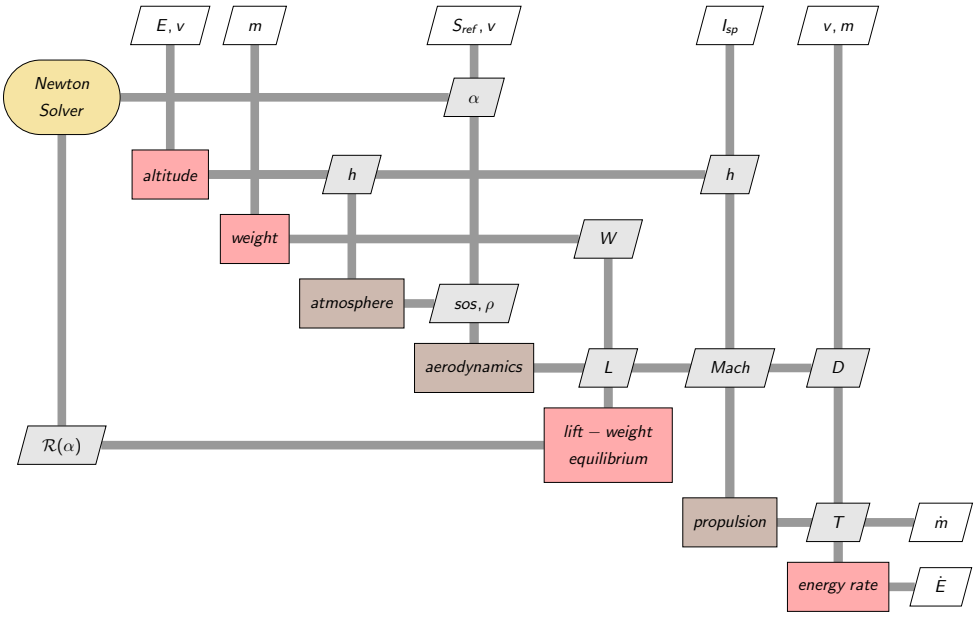


Fig. 7 The extended design structure matrix for the ODE of the supersonic minimum time-to-climb via the energy state approximation.

The solution for the problem is shown in Figure 8. The problem was solved with 50 3$^{\text{rd}}$-order segments using the Radau pseudospectral method. In their work, Bryson et al. claim a result of 277 seconds, which neglects the time spent in zoom dive and climb (moving along constant specific energy contours) [24]. The results here are well within one percent at 278.6 seconds. There is some noticeable transient behavior when velocity is nearly discontinuous at the

transitions into and out of zoom dive and climb. Adding a reasonable path constraint to the airspeed rate, computed via Equation (7), helps alleviate some of the ringing behavior. Automatic grid refinement, which would also help avoid ringing at discontinuities, is not yet implemented in Dymos.

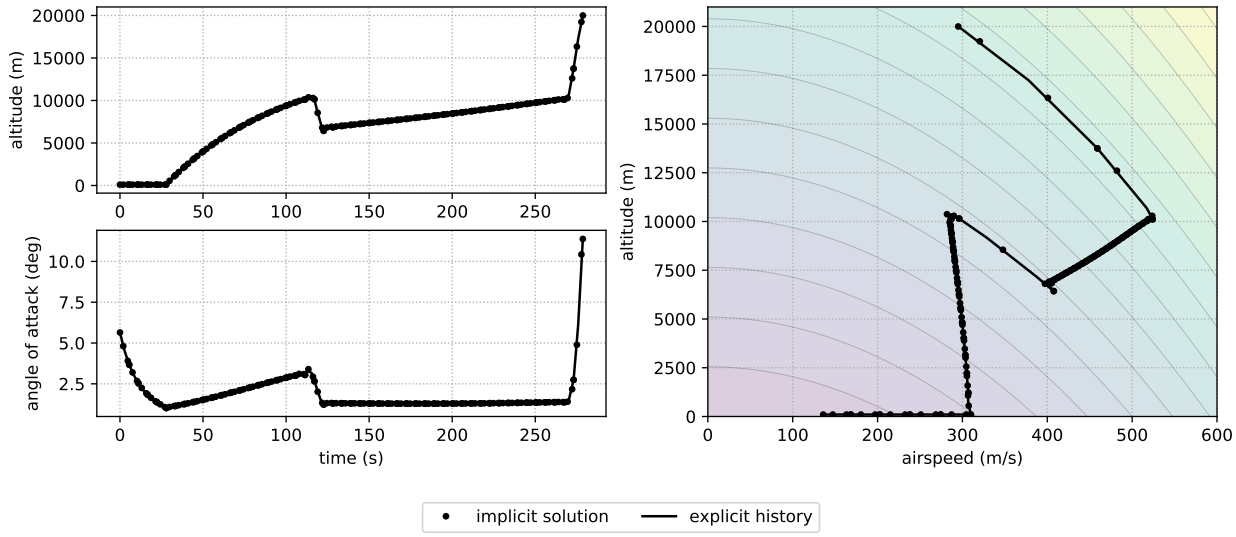


Fig. 8 The solution to the supersonic minimum time-to-climb problem via the energy state approximation. The altitude vs. airspeed plot is superimposed on contours of constant specific energy.

C. Example 3: Brachistochrone with a naive post-processing analysis

In this case we use a slight variation to the Brachistochrone problem originally posed by Johann Bernoulli[32]. In the original version, the objective is to shape a wire between two points such that the bead slides from the initial point to the final point in minimum time. Here we formulate the problem such that control angle θ is the angle between the wire and the nadir at any point along it.

minimize:	$J = t_f$
subject to dynamics:	$\dot{x} = v \sin\theta$
	$\dot{y} = -v \cos\theta$
	$\dot{v} = -g \cos\theta$
and initial conditions:	$x_0 = 0\text{m}$
	$y_0 = 10\text{m}$
	$v_0 = 0\text{m/s}$
and final conditions:	$x_f = 10\text{m}$
	$y_f = 5\text{m}$

In this example, we introduce a complication often faced in MDO analyses. A secondary analysis is introduced that is a function of the trajectory state-history, but it is implemented in a way that is not efficiently integrated with pseudospectral methods. Rather than simply minimizing the time required for the bead to travel down a frictionless wire, we treat the wire as a limited resource and constrain its maximum length. The length of the wire is:

$$S = \int_{x_0}^{x_f} \sqrt{1 + \frac{dy}{dx}} dx \tag{27}$$

$$\frac{dy}{dx} = \frac{1}{\tan\theta} \tag{28}$$

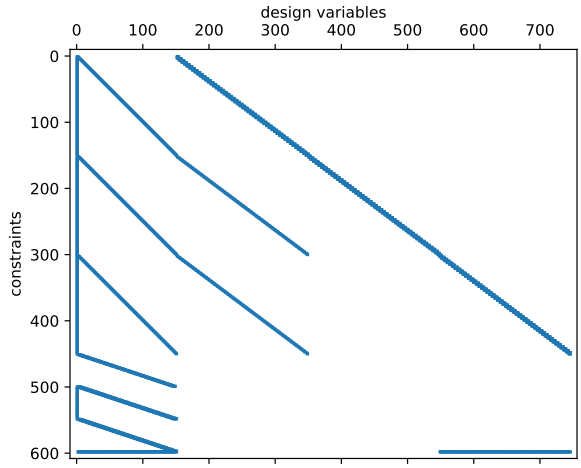
The most efficient way to approach this problem would be to transform the integral in Equation (27) into the time domain and treat the arc-length S as an integrated state variable. In this case, as is often the case in real-world MDO analyses, the implementation of our arc-length function is not integrated into our pseudospectral approach. Rather than rewrite an analysis tool to accommodate the pseudospectral approach, the arc-length analysis simply takes the result of the trajectory in its entirety and computes the arc-length constraint via the trapezoidal rule:

$$S = \frac{1}{2}\left(\sum_{i=1}^{N-1}\sqrt{1+\frac{1}{\tan\theta_{i-1}}}+\sqrt{1+\frac{1}{\tan\theta_i}}\right)(x_{i-1}-x_i) \tag{29}$$

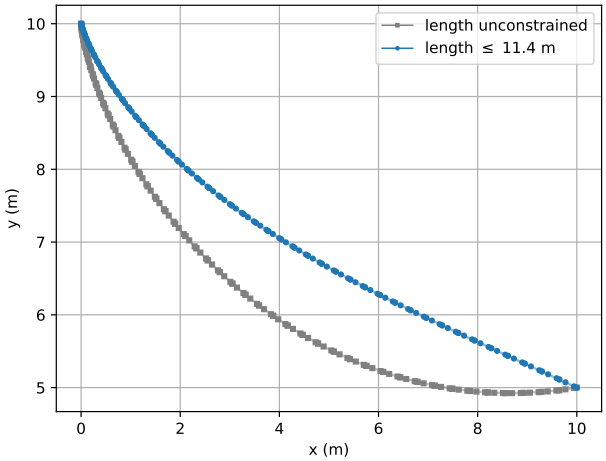
The problem as formulated above poses a challenge to efficient pseudospectral methods; a dense column due to the impact of the phase duration on nearly all constraints, and a dense row due the impact of the state and control variables on the arc-length constraint. We demonstrate the application of OpenMDAO's bi-directional coloring of the total Jacobian to solve many elements of the Jacobian simultaneously for improved efficiency. Figure 9a shows the sparsity of the total Jacobian for the problem. Performance comparisons using various approaches to computing the total derivatives are included below.

1. Results

This problem was solved in Dymos using the Radau pseudospectral method with 50 3$^{\text{rd}}$-order segments. A comparison of the solution vs. that of the unconstrained brachistochrone is shown in Figure 9b. The maximum allowable length of the wire is 11.4 m, only slightly longer than the straight-line distance of 11.18 m.



(a) Sparsity pattern for the length-constrained brachistochrone problem. The dense column on the left is due to the dependence of the defect constraints on the phase duration while the dense row at the bottom is due to the dependence of the arc-length constraint on control θ and state x.



(b) The solution of the arc-length-constrained brachistochrone problem compared to the unconstrained solution.

Fig. 9 Results of the length-constrained brachistochrone problem in Dymos.

Table 2 shows a comparison of performance in using the various derivative modes available in Dymos. Colors refers to the number of groupings of variables/constraints which can be perturbed simultaneously, and corresponds to the number of linear solves required to assemble the total Jacobian matrix. Given the sparsity structure in 9a this problem isn't particularly efficient in forward or reverse modes, and the bi-directional coloring approach is able to reduce the number of solves required from 599 in the reverse mode to only 14, with a significant reduction in the time required to compute the derivatives. It's also interesting to note that in this case the problem has more variables than constraints and is thus more efficient in reverse mode when a derivative coloring approach is not used. However, the separability of the total Jacobian matrix is such that reverse mode is the worst performing option when derivative coloring is employed.

In this case the performance improvement achieved through simultaneous derivative calculation is less significant than that observed in the minimum time-to-climb example from Section III.A. Here the solution time for the bi-directional approach (5.99 seconds) is roughly 60% that of the best case without simultaneous derivatives (10.24 seconds). The Radau Pseudospectral method from Section III.A with 50 segments saw solution time reduced from roughly 70 seconds to about 11 seconds. This is likely due to increased complexity of the ODE system for the minimum time-to-climb case, which contains multiple groups, each containing several components. That ODE also involves some components with more expensive calculations, like the interpolation of the aerodynamic coefficients. This suggests that the performance improvement achieved through simultaneous derivative calculation increases with increased complexity of the ODE system, since the overhead involved in evaluating more complex systems is avoided by computing the outputs and sensitivities of the system far fewer times.

Table 2 The impact of derivative mode on solution time for the length-constrained brachistochrone.

Derivative Mode	Colors	Solution Time (s)	Sensitivity Time (s)
Forward-Colored	349	7.36	3.55
Reverse-Colored	549	8.08	4.07
Bi-directional-Colored	14	5.99	1.93
Forward	746	11.31	5.68
Reverse	599	10.24	4.40

IV. Conclusions

In this paper we have presented a new software tool, Dymos, designed to enable more efficient use of optimal control techniques in multidisciplinary design optimization. By building Dymos on top of NASA's OpenMDAO software we leverage an efficient method for assembling the total-derivative matrix of the optimal control problem from various models, each of which may provide its partial derivatives via finite-difference, complex-step, or analytically. Because OpenMDAO's use of the unified derivatives approach allows it to accurately compute derivatives even in the presence of iterative solvers, we can apply these solvers to enable other solutions approaches to optimal control, such as the use of differential inclusion. The combination of more accurate derivatives and automatic coloring of the sparsity pattern make Python-based Dymos competitive with Fortran-based legacy optimal control software such as OTIS in terms of performance. The bi-directional coloring capability for computing derivatives allows a wider variety of algorithms to be brought to bear on a given problem without needing to consider the impact of various techniques on the sparsity pattern of the overall optimization problem.

To date, Dymos has been employed in several studies involving electric aircraft. The mission planning tool for NASA's all-electric X-57 aircraft is built using Dymos. It is used to plan flight profiles which meet mission objectives while observing constraints on the battery state-of-charge and the temperature of electrical components [33]. That work is an extension of an earlier demonstration of the ability of Dymos to optimize electric aircraft trajectories while observing subsystem thermal constraints [34]. The authors also used Dymos to demonstrate the ability to optimize the trajectory of a quad-rotor aircraft subject to a qualitative acoustic constraint [35]. Dymos remains under active development and has expanded the utility of OpenMDAO by allowing it to be more easily applied to the optimization of dynamic systems.

V. Acknowledgements

The authors would like to thank the NASA Transformational Tools and Technologies Project for supporting this work. We would like to thank Sydney Schnulo, Eliot Aretskin-Hariton, Jeffrey Chin, and Jeffryes Chapman of NASA Glenn Research Center for their extensive testing and helpful feedback in the development of Dymos.

References

[1] Tedford, N. P., and Martins, J. R. R. A., "Benchmarking Multidisciplinary Design Optimization Algorithms," *Optimization and Engineering*, Vol. 11, No. 1, 2010, p. 159–183. doi:10.1007/s11081-009-9082-6.

[2] Gray, J. S., Moore, K. T., Hearn, T. A., and Naylor, B. A., "Standard Platform for Benchmarking Multidisciplinary Design Analysis and Optimization Architectures," *AIAA Journal*, Vol. 51, No. 10, 2013, pp. 2380–2394.

[3] Gray, J. S., Chin, J., Hearn, T., Hendricks, E. S., Lavelle, T. M., and Martins, J., "Thermodynamics For Gas Turbine Cycles With Analytic Derivatives in OpenMDAO," *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-0669, URL `https://doi.org/10.2514/6.2016-0669`.

[4] Gray, J., Moore, K., and Naylor, B., "OpenMDAO: An open source framework for multidisciplinary analysis and optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010, p. 9101.

[5] Hwang, J. T., Lee, D. Y., Cutler, J. W., and Martins, J. R., "Large-scale multidisciplinary optimization of a small satellite's design and operation," *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, 2014, pp. 1648–1663.

[6] Kao, J., Hwang, J., Martins, J., Gray, J. S., and Moore, K. T., "A modular adjoint approach to aircraft mission analysis and optimization," *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2015, pp. 1–14. doi:10.2514/6.2015-0136, URL `http://arc.aiaa.org/doi/abs/10.2514/6.2015-0136`.

[7] Hwang, J. T., and Munster, D., "Solution of ordinary differential equations in gradient-based multidisciplinary design optimization," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2018. doi:10.2514/6.2018-1646, URL `https://doi.org/10.2514/6.2018-1646`.

[8] Botero, E. M., Wendorff, A., MacDonald, T., Variyar, A., Vegh, J. M., Lukaczyk, T. W., Alonso, J. J., Orra, T. H., and Ilario da Silva, C., "Suave: An open-source environment for conceptual vehicle design and optimization," *54th AIAA Aerospace Sciences Meeting*, 2016, p. 1275.

[9] MacDonald, T., Clarke, M., Botero, E. M., Vegh, J. M., and Alonso, J. J., "SUAVE: an open-source environment enabling multi-fidelity vehicle optimization," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 4437.

[10] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.

[11] Paris, S. W., Riehl, J. P., and Sjauw, W. K., "Enhanced Procedures for Direct Trajectory Optimization Using Nonlinear Programming and Implicit Integration," *Proceedings of the AIAA/AAS Astrodynamics Specialists Conference and Exhibit*, American Institute of Aeronautics and Astronautics, 2006.

[12] Hargraves, C. R., and Paris, S. W., "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, 1987, pp. 338–342.

[13] Herman, A. L., and Conway, B. A., "Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules," *{AIAA} Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 522–529. doi:10.2514/3.21662.

[14] Garg, D., Patterson, M., Darby, C., Francolin, C., Huntington, G., Hager, W., and Rao, A., *Direct Trajectory Optimization and Costate Estimation of General Optimal Control Problems Using a Radau Pseudospectral Method*, American Institute of Aeronautics and Astronautics, 2009.

[15] Lambe, A. B., and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

[16] Schubert, G. R., "Algorithm 211: Hermite Interpolation," *Commun. ACM*, Vol. 6, No. 10, 1963, pp. 617–. doi:10.1145/367651.367666, URL `http://doi.acm.org/10.1145/367651.367666`.

[17] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 41, No. 1, 2014, p. 1.

[18] Ross, I. M., and Fahroo, F., "Legendre pseudospectral approximations of optimal control problems," *New trends in nonlinear dynamics and control and their applications*, Springer, 2003, pp. 327–342.

[19] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003, pp. 245–262. doi:10.1145/838250.838251.

[20] Martins, J. R. R. A., and Hwang, J. T., "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2582–2599. doi:10.2514/1.J052184, URL http://arc.aiaa.org/doi/abs/10.2514/1.J052184.

[21] Hwang, J. T., and Martins, J. R. R. A., "A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives," *ACM Transactions on Mathematical Software*, Vol. 44, No. 4, 2018. doi:10.1145/3182393.

[22] Jeon, M., "Parallel optimal control with multiple shooting, constraints aggregation and adjoint methods," *Journal of Applied Math and Computing*, Vol. 19, No. 1-2, 2005, pp. 215–229. doi:DOI:10.1007/BF02935800.

[23] Martins, J. R. R. A., and Poon, N. M. K., "On Structural Optimization Using Constraint Aggregation," *Proceedings of 6th World Congress on Structural and Multidisciplinary Optimization*, , No. June, 2005, pp. 1–10.

[24] Bryson Jr, A. E., Desai, M. N., and Hoffman, W. C., "Energy-state approximation in performance optimization of supersonicaircraft," *Journal of Aircraft*, Vol. 6, No. 6, 1969, pp. 481–488.

[25] Bryson, A. E., *Dynamic optimization*, Addison Wesley Longman Menlo Park, CA, 1999.

[26] Betts, J., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed., Society for Industrial and Applied Mathematics, 2010. doi:10.1137/1.9780898718577, URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718577.

[27] Gray, J. S., Hearn, T. A., Moore, K. T., Hwang, J., Martins, J., and Ning, A., "Automatic Evaluation of Multidisciplinary Derivatives Using a Graph-Based Problem Formulation in OpenMDAO," *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2014. doi:10.2514/6.2014-2042, URL http://dx.doi.org/10.2514/6.2014-2042.

[28] Hearn, T. A., Hendricks, E., Chin, J., and Gray, J. S., "Optimization of Turbine Engine Cycle Analysis with Analytic Derivatives," *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-4297, URL https://doi.org/10.2514/6.2016-4297.

[29] Gray, J., Chin, J., Hearn, T., Hendricks, E., Lavelle, T., and Martins, J. R. R. A., "Chemical-Equilibrium Analysis with Adjoint Derivatives for Propulsion Cycle Analysis," *Journal of Propulsion and Power*, Vol. 33, No. 5, 2017, pp. 1041–1052. doi:10.2514/1.B36215, URL https://doi.org/10.2514/1.B36215.

[30] Seywald, H., "Trajectory optimization based on differential inclusion (Revised)," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, 1994, pp. 480–487. doi:10.2514/3.21224, URL https://doi.org/10.2514/3.21224.

[31] Fahroo, F., and Ross, I. M., "Second look at approximating differential inclusions," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 1, 2001, pp. 131–133.

[32] Bryson, A., and Ho, Y.-C., "Applied optimal control: Optimization, estimation, and control (revised edition)," *Levittown, Pennsylvania: Taylor & Francis*, 1975.

[33] Schnulo, S. L., Chin, J., Falck, R. D., Gray, J. S., Papathakis, K. V., Clarke, S. C., Reid, N., and Borer, N. K., "Development of a Multi-Segment Mission Planning Tool for SCEPTOR X-57," *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 3738.

[34] Falck, R. D., Chin, J., Schnulo, S. L., Burt, J. M., and Gray, J. S., "Trajectory optimization of electric aircraft subject to subsystem thermal constraints," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 4002.

[35] Falck, R. D., Ingraham, D., and Aretskin-Hariton, E., "Multidisciplinary Optimization of Urban-Air-Mobility Class Aircraft Trajectories with Acoustic Constraints," *2018 AIAA/IEEE Electric Aircraft Technologies Symposium*, 2018, p. 4985.