Proceedings of the ASME 2020
International Mechanical Engineering Congress and Exposition
IMECE2020
November 15–19, 2020, Portland, OR, USA

IMECE2020-23885

ON THE USES OF LINEAR-QUADRATIC METHODS IN SOLVING NONLINEAR DYNAMIC OPTIMIZATION PROBLEMS WITH DIRECT TRANSCRIPTION

Daniel R. Herber
Assistant Professor
Colorado State University
Department of Systems Engineering
Fort Collins, CO 80523
Email: daniel.herber@colostate.edu

Athul K. Sundarrajan
Graduate Student
Colorado State University
Department of Systems Engineering
Fort Collins, CO 80523
Email: athul.sundarrajan@colostate.edu

ABSTRACT

Solving nonlinear dynamic optimization (NLDO) and optimal control problems can be quite challenging, but the need for effective methods is ever increasing as more engineered systems become more dynamic and integrated. In this article, we will explore the various uses of linear-quadratic dynamic optimization (LQDO) in the direct transcription-based solution strategies for NLDO. Three general LQDO-based strategies are discussed, including direct incorporation, two-level optimization, and quasilinearization. Connections are made between a variety of existing approaches, including sequential quadratic programming. The case studies are solved with the various methods using a publicly available, MATLAB-based tool. Results indicate that the LQDO-based strategies can improve existing solvers and be effective solution strategies. However, there are robustness issues and problem derivative requirements that must be considered.

1 INTRODUCTION

Dynamic optimization (DO), or the optimization of systems with time-varying behavior, has proven to be instrumental to the advancement of many domains, including aerospace [1, 2], agricultural [3], chemical [4, 5], energy [6], financial [7], mechanical [8, 9], and medical [10] applications. Also frequently termed optimal control problems, the infinite-dimensional nature of DO necessitates specialized solution strategies. A variety of numerical approaches have been developed, including those based on the optimality conditions [1, 11–13] and direct methods based on transcribing the DO problem into a finite-dimensional optimization problem. A class of direct methods that has proven to be quite effective are the direct transcription (DT) methods [1, 4, 5, 13, 14], which involves a finite number optimization variables for the controls, states, and parameters, and are the methods considered here.

As is the case with the advancement of many numerical methods, exploiting specific mathematical problem structure can lead to developments that both increase the efficiency and robustness of the underlying techniques. This has undoubtedly been the case in DT research, such as with problem sparsity and derivatives [1, 5]. Here we will be exploring methods for exploiting a particular class of DO problems where the objective function is quadratic, and the constraints are linear, i.e., *linear-quadratic dynamic optimization* (LQDO). Because of the considerable degree of problem structure, there are many advantages to developing numerical methods that specifically exploit these patterns. This has historically included explicit matrix construction [15, 16], two-level [9, 17–19], quasilinearization [2, 12, 20–26], and sequential quadratic programming methods [1, 5, 24, 26, 27], which will be discussed in Sec. 4. Furthermore, many relevant DO problems contain linear or quadratic problem elements [10, 20, 22, 23, 28], such as is the case in model-predictive control [26, 28, 29]. Therefore, generalizing the uses of LQDO-based methods will allow for this structure to be exploited when needed and is the primary purpose of this article.

The remainder of this article is organized as follows. Section 2 presents the nonlinear dynamic optimization (NLDO) problem and a brief overview of DT. Section 3 focuses on defin-

ing the LQDO problem, a particular subclass of NLDO. Section 4 then presents several methods for solving NLDO problems using LQDO-based methods. Section 6 presents four case studies demonstrating the effectiveness and potential issues with the methods from Sec. 4. Finally, Sec. 7 provides the conclusions.

2 NONLINEAR DYNAMIC OPTIMIZATION

We begin by defining the NLDO problem class[1]:

$$\min_{\boldsymbol{x}=[\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}]} \quad o = \int_{t_0}^{t_f} \ell\left(t,\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}\right)dt + m\left(\boldsymbol{p},\boldsymbol{\xi}_0,\boldsymbol{\xi}_f\right) \tag{1a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}}(t) - \boldsymbol{f}\left(t,\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}\right) = \boldsymbol{0} \tag{1b}$$

$$\boldsymbol{h}\left(t,\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p},\boldsymbol{\xi}_0,\boldsymbol{\xi}_f\right) = \boldsymbol{0} \tag{1c}$$

$$\boldsymbol{g}\left(t,\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p},\boldsymbol{\xi}_0,\boldsymbol{\xi}_f\right) \leq \boldsymbol{0} \tag{1d}$$

$$\text{where:} \quad \boldsymbol{\xi}_0 = \boldsymbol{\xi}(t_0),\ \boldsymbol{\xi}_f = \boldsymbol{\xi}(t_f) \tag{1e}$$

where $t \in [t_0, t_f]$ is the fixed time horizon, $\{\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}\}$ are the collections of the problem's controls, states, and parameters, respectively; $o(\cdot)$ is the objective function composed of the Lagrange term denoted $\ell(\cdot)$, and Mayer term denoted $m(\cdot)$; $\boldsymbol{f}(\cdot)$ is the first-order explicit state derivative function; and $\boldsymbol{h}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are the additional equality and inequality constraints, respectively.

Problem (1) is an infinite-dimensional problem because constraints need to be satisfied at all points in t using the optimization variables that can be time varying [1, 5, 13]. Optimality conditions for this problem can be prescribed using Pontryagin's minimum principle with extensions for the various additional constraints and parameters [1, 11–13]. As previously mentioned, it can be quite challenging to find solutions to this problem using the optimality conditions, so a variety of parameterization methods have been developed to convert the infinite-dimensional problem into one that can be (approximately) solved using using finite-dimensional optimization, which will now be discussed.

2.1 Nonlinear Programming

The formulation of a general finite-dimensional nonlinear program (NLP) is [1, 5, 30]:

$$\min_{\mathbf{x}} \quad v(\mathbf{x}) \tag{2a}$$

$$\text{subject to:} \quad \mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{2b}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{2c}$$

where $\mathbf{x}$ are the NLP optimization variables, $v(\mathbf{x})$ is the scalar objective or value function, $\mathbf{h}(\mathbf{x})$ is the collection of equality constraints, and the $\mathbf{g}(\mathbf{x})$ is the collection of inequality constraints. A variety of optimization algorithms have been developed to solve NLPs, and some popular methods include sequential quadratic programming, interior-point, trust-region, and active-set methods [1, 5, 30–32].

[1]Many of statements in this article can be applied to NLDO problems with multiple phases, a variable time horizon, and general state differential equations.

2.2 Direct Transcription

Here we discuss the DT method for constructing an NLP in the form of Prob. (2) for the NLDO problem in Prob. (1) [1, 4, 5, 13, 14]. First, we define the various discretization matrices that will be used:

$$\boldsymbol{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_{n_t} \end{bmatrix},\ \boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_1 \\ \vdots \\ \boldsymbol{U}_{n_t} \end{bmatrix} = \begin{bmatrix} u_1(t_1) & \cdots & u_{n_u}(t_1) \\ \vdots & \ddots & \vdots \\ u_1(t_{n_t}) & \cdots & u_{n_u}(t_{n_t}) \end{bmatrix} \tag{3a}$$

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\Xi}_1 \\ \vdots \\ \boldsymbol{\Xi}_{n_t} \end{bmatrix} = \begin{bmatrix} \xi_1(t_1) & \cdots & \xi_{n_u}(t_1) \\ \vdots & \ddots & \vdots \\ \xi_1(t_{n_t}) & \cdots & \xi_{n_u}(t_{n_t}) \end{bmatrix},\ \boldsymbol{p} = \begin{bmatrix} p_1 & \cdots & p_{n_p} \end{bmatrix} \tag{3b}$$

where $\boldsymbol{t}$ is a discretized, strictly-increasing collection of time points, as termed the mesh; $\boldsymbol{U}$ and $\boldsymbol{\Xi}$ are the discretization matrices for the controls and states, respectively, where the entries are collocated parameterizations of these trajectories; and $\boldsymbol{p}$ is a vector of the static parameters.

Using this collection of discretized optimization variables, the DT NLP problem is:

$$\min_{\mathbf{x}=[\mathrm{vec}(\boldsymbol{U}),\mathrm{vec}(\boldsymbol{\Xi}),\boldsymbol{p}]} \quad v^{DT}\left(\boldsymbol{t},\boldsymbol{U},\boldsymbol{\Xi},\boldsymbol{p},\boldsymbol{\Xi}_1,\boldsymbol{\Xi}_{n_t}\right) \tag{4a}$$

$$\text{subject to:} \quad \mathbf{h}^{DT} = \begin{bmatrix} \mathbf{h}_{\boldsymbol{\zeta}}^{DT}\left(\boldsymbol{t},\boldsymbol{U},\boldsymbol{\Xi},\boldsymbol{p}\right) \\ \mathbf{h}_{\boldsymbol{h}}^{DT}\left(\boldsymbol{t},\boldsymbol{U},\boldsymbol{\Xi},\boldsymbol{p},\boldsymbol{\Xi}_1,\boldsymbol{\Xi}_{n_t}\right) \end{bmatrix} = \mathbf{0} \tag{4b}$$

$$\mathbf{g}^{DT}\left(\boldsymbol{t},\boldsymbol{U},\boldsymbol{\Xi},\boldsymbol{p},\boldsymbol{\Xi}_1,\boldsymbol{\Xi}_{n_t}\right) \leq \mathbf{0} \tag{4c}$$

where $\mathrm{vec}(\cdot)$ takes the matrix input and shapes it into a vector with the same elements. The specific constructions of the NLP problem elements $\left\{v^{DT}, \mathbf{h}_{\boldsymbol{\zeta}}^{DT}, \mathbf{h}_{\boldsymbol{h}}^{DT}, \mathbf{g}^{DT}\right\}$ are now discussed.

2.2.1 Dynamic constraints.

There are a variety of methods for approximating the state dynamic constraints in Eq. (1b). Here we will consider two different constructions.

First are the single-step (SS) methods that enforce an integral condition between adjacent states values with:

$$\mathbf{h}_{\boldsymbol{\zeta},k}^{DT} = \boldsymbol{\Xi}_{k+1} - \boldsymbol{\Xi}_k - \int_{t_k}^{t_{k+1}} \boldsymbol{f}\left(t,\tilde{\boldsymbol{u}},\tilde{\boldsymbol{\xi}},\boldsymbol{p}\right)dt, \quad k = 1,\ldots,n_t - 1 \tag{5}$$

where $\tilde{\boldsymbol{\xi}} = \tilde{\boldsymbol{\xi}}(t,\boldsymbol{t},\boldsymbol{\Xi})$ and $\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}(t,\boldsymbol{t},\boldsymbol{U})$ are interpolating functions constructed using the optimization variables, and these constraints capture the first fundamental theorem of calculus. Various SS methods are discussed in Refs. [1, 5, 13–15].

Second are the single-interval pseudospectral (PS) methods which construct global interpolating polynomials using $\boldsymbol{\Xi}$ with a specific mesh scheme and ensure that the derivatives of these polynomials are the same as the state derivative function:

$$\mathbf{h}_{\boldsymbol{\zeta},k}^{DT} = \boldsymbol{D}\boldsymbol{\Xi}_k - \boldsymbol{f}\left(t_k,\boldsymbol{U}_k,\boldsymbol{\Xi}_k,\boldsymbol{p}\right), \qquad k = 1,\ldots,n_t \tag{6}$$

which is in the same form as Eq. (1b). Please see Refs. [13, 33] for more details on PS methods.

2.2.2 Additional constraints.

For each of the additional constraints (both equality $\boldsymbol{h}$ and inequality $\boldsymbol{g}$), a single constraint

evaluated at each point in $\boldsymbol{t}$ is added to ensure that the constraint condition is satisfied on the selected mesh. Using the NLP optimization variables, these constraints have the following form:

$$\mathbf{h}^{DT}_{k} = \boldsymbol{h}(t_k, \boldsymbol{U}_k, \boldsymbol{\Xi}_k, \boldsymbol{p}, \boldsymbol{\Xi}_1, \boldsymbol{\Xi}_{n_t}) = \mathbf{0}, \qquad k = 1, \ldots, n_t \tag{7}$$

$$\mathbf{g}^{DT}_{k} = \boldsymbol{g}(t_k, \boldsymbol{U}_k, \boldsymbol{\Xi}_k, \boldsymbol{p}, \boldsymbol{\Xi}_1, \boldsymbol{\Xi}_{n_t}) \leq \mathbf{0}, \qquad k = 1, \ldots, n_t \tag{8}$$

For constraints with no dependence on $\{t_k, \boldsymbol{U}_k, \boldsymbol{\Xi}_k\}$, only a single constraint is needed because the constraint condition is the same at each time point. Such constraints are typically called *boundary* constraints, while the time-dependent ones are termed *path* constraints [1, 5, 14]. However, for presentation sake, we will assume this general path constraint form but the implementations will take into account if the constraint is a path or boundary constraint.

2.2.3 Objective function.

In order to evaluate the NLDO objective function in Eq. (1a), both the integral Lagrange and Mayer terms need to be evaluated using the NLP optimization variables. The value of the Mayer term can be computed exactly because it is represented by variables that are included in the DT NLP optimization variables. For the integral, numerical quadrature can be used. Some common quadrature schemes include composite trapezoidal [14], Gaussian [13, 33], and Clensaw-Curtis [33] methods. For these schemes, the DT objective function is computed with the following expression:

$$v^{DT} = \sum_{k=1}^{n_t} w_k \ell(t_k, \boldsymbol{U}_k, \boldsymbol{\Xi}_k, \boldsymbol{p}) + m(\boldsymbol{p}, \boldsymbol{\Xi}_1, \boldsymbol{\Xi}_{n_t}) \tag{9}$$

where $\boldsymbol{w}$ are suitable integration weights.

3 LINEAR-QUADRATIC DYNAMIC OPTIMIZATION

A linear-quadratic dynamic optimization (LQDO) problem is a subclass of Prob. (1) where the objective function is limited to quadratic terms and the constraints have only linear terms. To completely characterize the LQDO problem class, we first need to define the Jacobian matrix of a vector-valued function $\boldsymbol{e}$ of size m in variables $\boldsymbol{x}$ of size n:

$$\boldsymbol{J}^{\boldsymbol{e}}_{\boldsymbol{x}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial e_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial e_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial e_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \tag{10}$$

which is a matrix of all its first-order partial derivatives, and the Hessian matrix of a scalar-valued function e in variables $\boldsymbol{x}$ of size n:

$$\boldsymbol{H}^{e}_{\boldsymbol{x}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 e(\boldsymbol{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 e(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 e(\boldsymbol{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 e(\boldsymbol{x})}{\partial x_n^2} \end{bmatrix} \tag{11}$$

which is a square matrix of second-order partial derivatives.

Now consider some reference solution $\bar{\boldsymbol{x}}$ comprised of reference control trajectories $\bar{\boldsymbol{u}}$, state trajectories $\bar{\boldsymbol{\xi}}$, and parameter values $\bar{\boldsymbol{p}}$. Using the derivative matrices, we can construct the best local approximations of the NLDO problem elements about $\bar{\boldsymbol{x}}$. The best linear approximation of $\boldsymbol{e}$ around point $\bar{\boldsymbol{x}}$ is:

$$\boldsymbol{e}(\boldsymbol{x}) \approx \boldsymbol{e}^{L}(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \boldsymbol{e}(\bar{\boldsymbol{x}}) + \boldsymbol{J}^{\boldsymbol{e}}_{\boldsymbol{x}}(\bar{\boldsymbol{x}})[\boldsymbol{x} - \bar{\boldsymbol{x}}] \tag{12}$$

which is the first-order multivariate Taylor expansion of $\boldsymbol{e}$. The best quadratic approximation of e around $\bar{\boldsymbol{x}}$ is:

$$e(\boldsymbol{x}) \approx e^{Q}(\boldsymbol{x}, \bar{\boldsymbol{x}}) = e(\bar{\boldsymbol{x}}) + [\boldsymbol{J}^{e}_{\boldsymbol{x}}(\bar{\boldsymbol{x}})]^{\mathsf{T}}[\boldsymbol{x} - \bar{\boldsymbol{x}}] + \cdots \tag{13a}$$

$$+ \frac{1}{2}[\boldsymbol{x} - \bar{\boldsymbol{x}}]^{\mathsf{T}} \boldsymbol{H}^{e}_{\boldsymbol{x}}(\bar{\boldsymbol{x}})[\boldsymbol{x} - \bar{\boldsymbol{x}}] \tag{13b}$$

which is the second-order Taylor polynomial.

Now, we apply these approximations on the appropriate problem elements of Prob. (1), resulting in the following DO problem:

$$\min_{\boldsymbol{x}=[\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}]} \quad o^{Q}\left(t, \boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{p}, \boldsymbol{\xi}_0, \boldsymbol{\xi}_f, \bar{\boldsymbol{x}}\right) \tag{14a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}}(t) - \boldsymbol{f}^{L}(t, \boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{p}, \bar{\boldsymbol{x}}) = \mathbf{0} \tag{14b}$$

$$\boldsymbol{h}^{L}\left(t, \boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{p}, \boldsymbol{\xi}_0, \boldsymbol{\xi}_f, \bar{\boldsymbol{x}}\right) = \mathbf{0} \tag{14c}$$

$$\boldsymbol{g}^{L}\left(t, \boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{p}, \boldsymbol{\xi}_0, \boldsymbol{\xi}_f, \bar{\boldsymbol{x}}\right) \leq \mathbf{0} \tag{14d}$$

$$\text{where:} \quad \boldsymbol{\xi}_0 = \boldsymbol{\xi}(t_0),\ \boldsymbol{\xi}_f = \boldsymbol{\xi}(t_f) \tag{14e}$$

which is an LQDO problem by construction because the highest-order terms in the objective function are quadratic and the highest-order terms in the constraints are linear[2].

For the linearized constraints, it is common to express the resulting expression stratified by optimization variable type. For example, $\boldsymbol{f}^{L}$ is a linear time-varying (LTV) dynamic system about the reference trajectory $\bar{\boldsymbol{x}}$:

$$\boldsymbol{f}^{L}(t, \boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{p}, \bar{\boldsymbol{x}}) = \boldsymbol{A}(t, \bar{\boldsymbol{x}})\boldsymbol{\xi} + \boldsymbol{B}(t, \bar{\boldsymbol{x}})\boldsymbol{u} + \cdots \tag{15a}$$

$$+ \boldsymbol{G}(t, \bar{\boldsymbol{x}})\boldsymbol{p} + \boldsymbol{d}(t, \bar{\boldsymbol{x}}) \tag{15b}$$

$$\text{where:} \quad \boldsymbol{A}(t, \bar{\boldsymbol{x}}) = \boldsymbol{J}^{\boldsymbol{f}}_{\boldsymbol{\xi}}(\bar{\boldsymbol{x}}) \tag{15c}$$

$$\boldsymbol{B}(t, \bar{\boldsymbol{x}}) = \boldsymbol{J}^{\boldsymbol{f}}_{\boldsymbol{u}}(\bar{\boldsymbol{x}}) \tag{15d}$$

$$\boldsymbol{G}(t, \bar{\boldsymbol{x}}) = \boldsymbol{J}^{\boldsymbol{f}}_{\boldsymbol{p}}(\bar{\boldsymbol{x}}) \tag{15e}$$

$$\boldsymbol{d}(t, \bar{\boldsymbol{x}}) = \boldsymbol{f}\left(t, \bar{\boldsymbol{u}}, \bar{\boldsymbol{\xi}}, \bar{\boldsymbol{p}}\right) - \boldsymbol{J}^{\boldsymbol{f}}_{\boldsymbol{x}}(\bar{\boldsymbol{x}})\bar{\boldsymbol{x}} \tag{15f}$$

Similar representations can be made for the additional path and boundary constraints, as well as the objective function.

As an example, consider one of the dynamic equations for the Van der Pol Oscillator problem (cf. Eq. (25b)):

$$e(\boldsymbol{x}) = -p_1\xi_1 + \xi_2\left[1 - p_2\xi_1^2\right] + u_1 \tag{16}$$

Now we have the LTV dynamic system defined in Eq. (15):

$$e^{L}(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \begin{bmatrix} -\bar{p}_1 - 2\bar{p}_2\bar{\xi}_1\bar{\xi}_2 \\ 1 + \bar{p}_2\bar{\xi}_1^2 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + u_1 - \begin{bmatrix} \bar{\xi}_1 \\ \bar{\xi}_1^2\bar{\xi}_2 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \ldots$$

$$+ \bar{p}_1\bar{\xi}_1 + 3\bar{p}_2\bar{\xi}_1^2\bar{\xi}_2 \tag{17}$$

[2] The dynamic constraint could be written in the same form as $\boldsymbol{h}^{L}$ and $\boldsymbol{g}^{L}$ but this form is equivalent because $[\dot{\boldsymbol{\xi}}(t)]_{\bar{\boldsymbol{x}}} = \dot{\boldsymbol{\xi}}(t)$ and $\boldsymbol{J}^{\dot{\boldsymbol{\xi}}}_{\boldsymbol{x}}(\bar{\boldsymbol{x}}) = \mathbf{0}$.

3.1 Ordinary vs. Reference LQDO Problems

The current definition of an LQDO problem depends on the reference trajectory $\bar{x}$, now termed a reference LQDO (RLQDO) problem.

It is quite common to have quadratic and linear problem elements in the NLDO problem; thus, the derivative matrices for those elements are constant and do not depend on $\bar{x}$. These are termed ordinary linear-quadratic (OLQ) elements. Common problem elements such as linear dynamic systems, simple state bounds, initial and final conditions, quadratic Lagrange terms, and linear Mayer terms are all OLQ elements. Here we define an OLQ problem as one where there are only OLQ elements, and Prob. (1) and Prob. (14) are equivalent.

For example, these four constraints are all OLQ elements:

$$\dot{\xi}_1 + 4\xi_1 - 2u_1 + \sin(t) = 0, \quad \xi_1(0) = 0$$
$$\xi_1 - 1 \leq 0, \quad tu_1 - 2p_1 + 1 \leq 0$$

noting that linear time-varying quantities are admissible. However, this objective function term is not an OLQ element even though it contains a quadratic term (however it can be partitioned as is discussed in Sec. 4.1):

$$\int_{t_0}^{t_f} \left[\xi_2^2 + \sin(u_1)\right] dt$$

There are some standard LQDO problems such as the finite-horizon linear-quadratic regulator [11].

3.2 Quadratic Programming

One of the key reasons to study and use LQDO problem formulations is that the DT optimization problem in Prob. (4) is a quadratic program (QP). A QP is a special subclass of NLPs where the objective function is quadratic and the constraints are linear:

$$\min_{\mathbf{x}} \quad \mathbf{f}^\mathsf{T}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{x} \tag{18a}$$
$$\text{subject to:} \quad \mathbf{A_h}\mathbf{x} - \mathbf{b_h} = \mathbf{0} \tag{18b}$$
$$\mathbf{A_g}\mathbf{x} - \mathbf{b_g} \leq \mathbf{0} \tag{18c}$$

where $\left\{\mathbf{f}, \mathbf{H}, \mathbf{A_h}, \mathbf{b_h}, \mathbf{A_g}, \mathbf{b_g}\right\}$ are real-valued matrices/vectors of the appropriate size. These matrices are typically large and sparse (contain many zero entries), and have a predicable pattern that permits efficient construction [1, 5, 15]. There are many efficient algorithms for solving QPs including interior-point, active-set, augmented Lagrangian, conjugate gradient, and alternating direction method of multipliers [5, 28, 30, 32]. For large-scale sparse QPs, certain algorithms can utilize sparse linear algebra to efficiently find optimal solutions [4, 24, 32].

In this work, we will be using *DTQP*, which is a *MATLAB*-based tool for efficiently constructing the matrices/vectors for all LQDO problem elements using user-friendly, structure-based definition [16]. While there are a variety of other general DT-based tools [34–36], the initial focus only on LQDO problems in the *DTQP* has led to a number of focused advancements. The sparse matrices are constructed using vectorized computations and direct creation of the row, column, and value sequences [37] that define the required matrix resulting in good scalability for both small and large meshes. A variety of defect constraint methods (including several SS and PS schemes) as well as the common quadrature schemes discussed in Sec. 2.2.3 are available. The main algorithms are described in Ref. [15].

4 METHODS FOR SOLVING NLDO PROBLEMS USING LQDO PROBLEM ELEMENTS

In this section, we will discuss several methods for effectively utilizing LQDO problems and OLQ elements to solve NLDO problems.

4.1 Direct Incorporation of OLQ Elements

The first method is perhaps the most self-evident. For a given NLDO problem, we can treat the OLQ elements differently than the general nonlinear elements. In many DT-based software implementations, most of the problem elements are assumed to be nonlinear, and there is no way to specify if a particular constraint or objective term is an OLQ element (with the primary exception being linear constraints for the initial and final values of the trajectories and simple box constraints).

Consider the following partitioning of the NLDO problem in Prob. (1):

$$\min_{\boldsymbol{x}=[\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}]} \quad \int_{t_0}^{t_f} \left[\ell^{OL}(\cdot) + \ell^{NL}(\cdot)\right] dt + m^{OL}(\cdot) + m^{NL}(\cdot) \tag{19a}$$
$$\text{subject to:} \quad \dot{\boldsymbol{\xi}}(t) - \boldsymbol{f}(\cdot) = \begin{bmatrix} \dot{\boldsymbol{\xi}}^{OL}(t) - \boldsymbol{f}^{OL}(\cdot) \\ \dot{\boldsymbol{\xi}}^{NL}(t) - \boldsymbol{f}^{NL}(\cdot) \end{bmatrix} = \mathbf{0} \tag{19b}$$
$$\boldsymbol{h}(\cdot) = \begin{bmatrix} \boldsymbol{h}^{OL}(\cdot) \\ \boldsymbol{h}^{NL}(\cdot) \end{bmatrix} = \mathbf{0} \tag{19c}$$
$$\boldsymbol{g}(\cdot) = \begin{bmatrix} \boldsymbol{g}^{OL}(\cdot) \\ \boldsymbol{g}^{NL}(\cdot) \end{bmatrix} \leq \mathbf{0} \tag{19d}$$

where the superscript OL indicates an OLQ element and superscript NL indicates a non-OLQ element. There are a few methods for determining the correct partitioning. First is a manual specification by the user, assuming it is supported by the selected tool. Second are the automated methods using either exact methods, such as symbolic or automatic differentiation, or a careful numerical procedure that checks the finite-difference values at several diverse points (because the result for OLQ elements would be the same for every point).

There are several potential advantages to this approach. First, most gradient-based NLP solvers require the computation of the problem's derivatives, which can be expensive. Running these computations for the OLQ elements within the NLP solver is an additional computational burden because results are independent of the current iteration. Therefore, computing the matrices once before starting the NLP solver will reduce the overall computational cost of computing the derivatives. If analytic

derivatives are provided, the difference might be relatively small. If finite-difference methods are being utilized for computation, the gap can be much more pronounced, even if a sparsity pattern is provided. Additionally, because of the numerical errors introduced through finite-difference schemes, the derivatives for the OLQ elements may be more accurate.

Second, many NLP algorithms have special methods for handling linear constraints that may be more numerically accurate and computationally efficient. For example, *MATLAB*'s `fmincon` directly allows for the inclusion of the linear constraints and simple upper and lower bounds [38]. Derivatives are then known for these problem elements, and certain options can be enabled such as "honor bounds". More specifically, many NLP solvers utilize the Lagrangian function:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\sigma}) = v(\mathbf{x}) + \boldsymbol{\lambda} \cdot \mathbf{h}(\mathbf{x}) + \boldsymbol{\sigma} \cdot \mathbf{g}(\mathbf{x}) \tag{20}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ are the multipliers for the equality and inequality constraints, respectively [30]. Second derivative computations for $\boldsymbol{H}^{\mathcal{L}}_{\mathbf{x}}(\mathbf{x})$ may be required. Direct incorporation of the $\boldsymbol{h}^{OL}$ constraints will result in a linear system $\mathbf{h}^{OL}(\mathbf{x}) = \mathbf{A}_{\mathbf{h}^{OL}}\mathbf{x} - \mathbf{b}_{\mathbf{h}^{OL}}$. Now, observing only the contribution of equality constraint terms:

$$\boldsymbol{H}^{\mathcal{L}}_{\mathbf{x}}(\mathbf{x}) = \sum_j \boldsymbol{\lambda}^{OL}_j \cdot \cancelto{\mathbf{0}}{\left[\boldsymbol{H}^{\mathbf{h}^{OL}_j}_{\mathbf{x}}(\mathbf{x})\right]} + \sum_j \boldsymbol{\lambda}^{NL}_j \cdot \left[\boldsymbol{H}^{\mathbf{h}^{NL}_j}_{\mathbf{x}}(\mathbf{x})\right] + \cdots \tag{21}$$

Therefore, we see that no Hessian computations are required for linear equality constraints. This is also true for the other linear OLQ constraints.

Finally, both linear variable scaling and constraint row scaling can be efficiently implemented for the OLQ matrices. The extent of the advantages of direct incorporation of the OLQ elements can depend on the number and type of OLQ elements and mesh size, but generally, there are few disadvantages.

4.2 Two-level Methods

Two-level optimization problems consider some partitioning of the optimization variables into upper $\boldsymbol{x}^u$ and lower $\boldsymbol{x}^l$ variables (sometimes termed outer and inner variables) [39]. These problems are represented as:

$$\min_{\boldsymbol{x}^u} \quad o\left(\boldsymbol{x}^u, \boldsymbol{x}^l\right) \tag{22a}$$

$$\text{subject to:} \quad \boldsymbol{x}^u \in \Omega \tag{22b}$$

$$\boldsymbol{x}^l \in L(\boldsymbol{x}^u) \tag{22c}$$

where $\boldsymbol{\Omega}$ is the feasibility region of $\boldsymbol{x}^u$, and $L(\boldsymbol{x}^u)$ is a set defined by the solution to an appropriate optimization problem. Determining a feasible $\boldsymbol{x}^l$ for a particular candidate $\boldsymbol{x}^u$ is solving the *lower-level problem* (LLP), while Prob. (22) is termed the *upper-level problem* (ULP). The two-level method is only one type of decomposition-based optimization strategy. Alternative decomposition-based formulations and solvers may also utilize the properties and methods for LQDO problems and OLQ elements [28, 40–42]. Here we will discuss two common partitioning schemes for NLDO problems with DT that incorporate LQDO LLPs.

4.2.1 Shooting. The first partitioning scheme considers only $\boldsymbol{\xi}$ in the LLP, while $\boldsymbol{u}$ and $\boldsymbol{p}$ are optimized in the ULP. This is a version of the direct shooting approach [5] where feasible dynamics are determined in the LLP. Reducing the state dynamic analyses to a sparse system of linear equations can be computationally attractive, permit computation of ULP derivatives, and still leverage the global stability and robustness properties of DT methods that are lacking in forward simulation-based methods [4]. Additionally, it is straightforward to include a low-dimensional parameterization of $\boldsymbol{u}$. In many cases, the LLP is effectively a feasibility problem. However, care must be taken to ensure that there exists a solution to LLP or some coordination/penalty method is utilized.

4.2.2 Fixed parameters. The second scheme is applicable to the class of NLDO problems where for fixed values of $\boldsymbol{p}$, the resulting subproblem is a convex LQDO problem. The two-level architecture is constructed with only the parameters and associated problem elements in ULP, while the LLP with respect to $\boldsymbol{u}$ and $\boldsymbol{\xi}$ is an LQDO problem. This partitioning allows for the LLP to utilize LQDO-based tools and QP solvers. Typically the set of parameters and associated ULP constraints are relatively small compared to the size of the mesh-dependent control and state DT variables, as well as the number of defect and path constraints. Furthermore, while general NLDO problems are non-convex, many common LQDO formulations are convex. Therefore, we can implement efficient solution strategies in the LLP and focus global search procedures in the ULP which has a much smaller design space to consider. However, this approach may still have issues with feasibility of the LLP, derivative computations in the ULP, and may be less efficient computationally depending of the particular problem [19].

This approach has been studied previously under the term nested (control) co-design [9, 17–19], and the aforementioned class of problems have been termed *LQDO-amendable co-design problems* [9]. References [6, 19] include comparisons showing that the nested method as described here had lower computational expense than a simultaneous approach, i.e., Prob. (1). There are several co-design studies that have effectively utilized the nested approach including attitude control of a distributed actuation system on a satellite [43], active vehicle suspensions [9], and horizontal-axis wind turbines [6].

4.3 Quasilinearization Methods

The methods described in the previous two sections did not utilize the RLQDO problem discussed in Sec. 3. Here we will discuss a class of methods, frequently termed *quasilinearization* [12, 20, 22, 23, 44], that utilizes reference trajectories to solve

ALGORITHM 1: Quasilinearization.

Input : $\mathbf{x}^0$ – initial reference solution
$k_{\max}$ – maximum number of iterations
ϵ – convergence tolerance
Output: $\mathbf{x}^*$ – final (optimal) solution

$k \leftarrow 0$ // initialize iteration counter
$v\left(\mathbf{x}^{k-1}\right) \leftarrow \infty$ // initialize objective value
while $\left|v\left(\mathbf{x}^k\right) - v\left(\mathbf{x}^{k-1}\right)\right| \leq \epsilon$ or $k \leq k_{\max}$ **do**
  $k \leftarrow k+1$ // update iteration counter
  $\{\mathbf{f}, \mathbf{H}, \mathbf{A_h}, \mathbf{b_h}, \mathbf{A_g}, \mathbf{b_g}\} \leftarrow$ Construct QP matrices using the RLQDO problem and selected DT method with $\mathbf{x}^{k-1}$
  $\mathbf{x}^k \leftarrow$ Solve QP defined by $\{\mathbf{f}, \mathbf{H}, \mathbf{A_h}, \mathbf{b_h}, \mathbf{A_g}, \mathbf{b_g}\}$
end
$\mathbf{x}^* \leftarrow \mathbf{x}^k$ // final solution (optimal if $k < k_{\max}$)

the original NLDO problem. Generally, these methods construct RLQDO subproblems that are successively solved where the previous iteration's solution is the reference trajectory for the next iteration. Under certain conditions, convergence of the solutions of the RLQDO subproblems implies that the result is good approximation of a feasible minimizer in both RLQDO and original NLDO problems [2,12]. A basic quasilinearization algorithm for NLDO is shown in Alg. 1.

In general, a good initial reference solution is preferred, otherwise, there may be slow convergence, convergence to an undesirable local minima, or divergence [2, 12, 22, 26]. Because of the LQDO-nature of the subproblems, sparse Hessians and Jacobians are readily determined [22]. Quasilinearization can be implemented for a variety of defect constraint and quadrature schemes [2, 23]. Typically, parameters $\boldsymbol{p}$ are not directly studied, but their inclusion is fairly straightforward with the primary difference being a reference value rather than a reference trajectory. Now four different variations of quasilinearization will be discussed.

4.3.1 Dynamics only. Perhaps the most commonly used form of quasilinearization is when only the dynamics are linearized as in Eq. (15). In many NLDO problems, the dynamics are the most challenging problem element because of their non-convex structure [2]. Frequently, assumptions are made so that other problem elements are OLQ [20]. In these cases, the OLQ elements can be directly incorporated in the same manner as Sec. 4.1, while LQDO-based methods for constructing the defect constraints can be used at each iteration to efficiently update the matrices based on the previous iteration's solution. This successive linear approximation of the dynamics results in a particularly advantageous constraint form that can be effectively utilized in specialized solvers, such a sequential convex programming [2] and Riccati difference equations [21].

4.3.2 All constraints. In this case, all constraints are linearized with Eqs. (14b)–(14d), including terminal state constraints [22] and state-control path constraints [23]. Generally, the same methods can be applied as the dynamics-only case, but an issue of inconsistencies of the RLQDO constraints can be exacerbated. Constraint relaxation and trust-region methods could be utilized to prevent this failure condition (and many require the solution of additional QPs) [1, 32].

4.3.3 Feasible point. Here we ignore the objective function in order to find a point $\mathbf{x}$ that satisfies all constraints in the NLDO problem [24]. In this form, we are attempting to solve a sequence of linear systems potentially with inequalities. However, these are frequently under-determined equations so additional measures should be taken to ensure convergence and uniqueness [1]. The result can be used as the initial point for the basic quasilinearization algorithm or any general NLP solver.

4.3.4 Entire NLDO problem. The RLQDO in Prob. (14) constructed an LQDO problem with respect to the reference solution $\bar{\boldsymbol{x}}$. Therefore, a solution strategy for the entire NLDO problem is Alg. 1. However, in addition the potential inconsistencies in the constraints, the RLQDO problem might result in a non-convex QP. Quasi-Newton methods can be alternative strategy, and defines a different QP problem than the one created from Prob. (14) [1, 5]. Alternative linear approximations other than a Taylor series can be used [25]. If only linear approximations are used in the objective function, the problem is a linear program.

All the previously discussed implementations of quasilinearization have ignored an important fact that because we are solving a constrained finite-dimensional optimization problem when using DT methods, the optimality conditions are the Karush-Kuhn-Tucker conditions [30], which include the Lagrange multipliers as shown in Eq. (20). Utilizing the Lagrangian instead of $v(\cdot)$ provides curvature information from both objective and constraints [32]. Sequential quadratic programming (SQP) is a popular family of methods that directly incorporate the Lagrangian and previous solution estimates for $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ [1, 5, 32, 45]. The SQP subproblem for Prob. (2) is:

$$\min_{\mathbf{s}} \quad v(\bar{\mathbf{x}}) + \left[\boldsymbol{J}_{\mathbf{x}}^{v}(\bar{\mathbf{x}})\right]^{\mathsf{T}}\mathbf{s} + \frac{1}{2}\mathbf{s}^{\mathsf{T}}\boldsymbol{H}_{\mathbf{x}}^{\mathcal{L}}\left(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\sigma}}\right)\mathbf{s} \tag{23a}$$

$$\text{subject to:} \quad \mathbf{h}(\bar{\mathbf{x}}) + \left[\boldsymbol{J}_{\mathbf{x}}^{\mathbf{h}}(\bar{\mathbf{x}})\right]^{\mathsf{T}}\mathbf{s} = \mathbf{0} \tag{23b}$$

$$\mathbf{g}(\bar{\mathbf{x}}) + \left[\boldsymbol{J}_{\mathbf{x}}^{\mathbf{g}}(\bar{\mathbf{x}})\right]^{\mathsf{T}}\mathbf{s} \leq \mathbf{0} \tag{23c}$$

$$\text{where:} \quad \mathbf{s} = \mathbf{x} - \bar{\mathbf{x}} \tag{23d}$$

SQP methods have been long studied in the context of DT methods [1, 5, 24, 26, 27].

Now an interesting connection can be made between the QP generated for a RLQDO problem and the SQP subproblem for the original NLDO problem in Prob. (2). Consider the case when the discretization matrices in Eq. (3) are the same for $\bar{\mathbf{x}}$ and an appropriately interpolated $\bar{\boldsymbol{x}}$. If we introduce a linear transformation $\boldsymbol{s} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ to the RLQO problem, then the resulting QP

will have the same constraints as the SQP subproblem for both the SS and PS defect constraints in Eqs. (5) and (6), respectively. This is due to the linear summation structure of the defect constraints. This implies that SQP methods on DT-based NLPs are really quasilinearization methods based on a RLQDO problem that utilize additional Hessian information from the constraints (as well as the other techniques that make SQP methods so effective). Because of this, LQDO-based methods for defining the problem elements for the SQP subproblems can be directly used. Exact computations for the additional Hessian information can also be incorporated with modifications to the LQDO discretization methods (considering the additional terms have the same sparsity structure as the original Hessian) [26, 46]. Full implementation of SQP methods using *DTQP* is left as future work.

5 METHOD OPTIONS

In this section, we summarize the methods that we implemented and compared. The four option fragments are now described.

1. *NLDO algorithm*. IP denotes use of the interior-point method in `fmincon` to solve the NLDO problem. QLIN denotes the quasilinearization method presented in Alg. 1. The RLQDO problem is automatically created using symbolic differentiation. TLFP denotes a two-level fixed parameter method in Sec. 4.2.2 was used to solve the NLDO problem (i.e., the nested control co-design solution strategy). The LQDO LLP is created and solved using *DTQP*, while the ULP is solved using `fmincon` with the interior-point algorithm. Finite-difference methods are used to obtain derivatives in ULP.
2. *OLQ elements*. OLQ denotes the OLQ elements are directly incorporated as discussed in Sec. 4.1. NL denotes all problem elements were treated as nonlinear except the simple bounds.
3. *Derivatives*. SD denotes symbolic derivatives are provided. Problem element derivatives are automatically determined using symbolic differentiation and used to construct the required differentiation matrices for the select DT method. FD denotes a forward finite-difference scheme was used to compute any required derivatives.
4. *DT methods*. TR denotes defect constraints constructed using the trapezoidal rule (a SS DT method), and the composite trapezoidal rule was used for quadrature. PS denotes defect constraints constructed using the single-interval Legendre pseudospectral method on a Lagrange-Gauss-Lobatto mesh and Gaussian quadrature weights [33]. For both methods, the catenated number indicates the number of points in the mesh.

These four options are combined to create a complete NLDO solution strategy. For example, IP-SD-OLQ-TR200 indicates that the NLP problem is constructed using the trapezoidal rule with 200 time points and symbolic derivatives with direct incorporation of OLQ elements and solved using an interior-point method. The computational cost is measured in several parts. T_{sym} indicates the time spent determining the symbolic derivatives. The total time is denoted T, which the combination of the initialization time T_{int} (which includes the time to create the OLQ element matrices) and optimizer time T_{opt}. The symbolic computations are not included in the total time so a fair comparison can be made between the methods. All the method and case study details are available in the free and open-source *DTQP* software tool [16]. The computer architecture used for all case study results was a desktop workstation with an AMD 3970X CPU at 3.7 GHz, 128 GB DDR4 2666 MHz RAM, *MATLAB* 2020a update 4, and *Windows* 10 build 17763.1397.

6 CASE STUDIES

In this section, four different case studies are presented to demonstrate how various LQDO-based methods can be incorporated to solve NLDO problems.

6.1 Container Crane

6.1.1 Problem description. In this problem, we want to optimally transfer the states of a container crane subject to linear path constraints. Please see Ref. [47] and is also known as `cran` in Ref. [10]. The NLDO problem is:

$$\min_{\boldsymbol{u},\boldsymbol{\xi}} \quad \frac{1}{2}\int_0^{t_f}\left[\xi_3^2+\xi_6^2+\rho\left[u_1^2+u_2^2\right]\right]dt \tag{24a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}} = \begin{bmatrix} \xi_4 \\ \xi_5 \\ \xi_6 \\ u_1 + c_4\xi_3 \\ u_2 \\ -\left[u_1 + c_5\xi_3 + 2\xi_5\xi_6\right]/\xi_2 \end{bmatrix} \tag{24b}$$

$$\boldsymbol{\xi}(0) = (0, 22, 0, 0, -1, 0) \tag{24c}$$

$$\boldsymbol{\xi}(t_f) = (10, 14, 0, 2.5, 0, 0) \tag{24d}$$

$$-c_6 \le \xi_4 \le c_6, \quad c_7 \le \xi_5 \le c_7 \tag{24e}$$

$$-c_1 \le u_1 \le c_1, \quad c_2 \le u_2 \le c_3 \tag{24f}$$

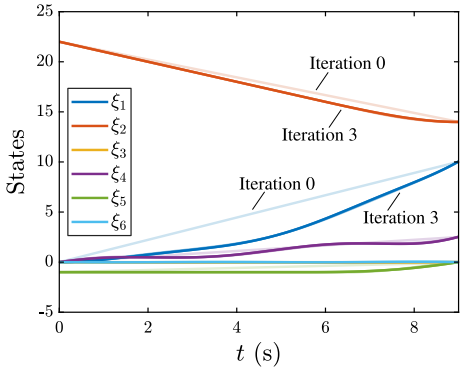
Note that all parts except for $\dot{\xi}_6$ are OLQ elements. The constants are $c_1 = 2.83374, c_2 = -0.80865, c_3 = 0.71265, c_4 = 17.2656, c_5 = 27.0756, c_6 = 2.5, c_7 = 1, t_f = 9$, and $\rho = 0.01$. The initial guess used for all methods was a linear solution between the specified initial and final states values and maximum and minimum control values (see Fig. 1).

6.1.2 Results. The results for this case study are summarized in Table 1.

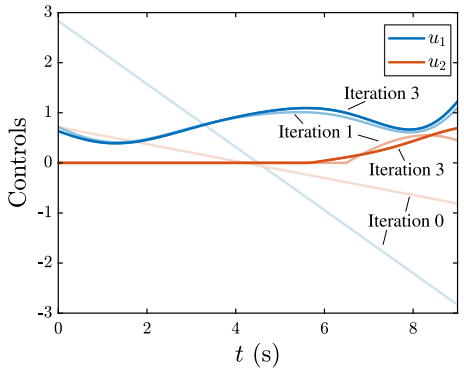
We will first focus on the effectiveness of the quasilinearization method. In most of the cases, QLIN had a lower computational cost than the best IP method while converging to nearly the same solution. For example, QLIN was 2.6× faster for the TR20 and 1.9× faster for the PS40 variant. There was some additional cost to create the OLQ matrices, but the overall solver time was faster. However, when the number of time points was large in the TR2000 variant, the QLIN method was 1.2× slower

TABLE 1: Results for the Container Crane case study.

Method	v	Iter.	T_{sym}	T_{int}	T_{opt}	T
QLIN-TR20	0.0380	3	0.57	0.007	0.03	0.03
IP-SD-OLQ-TR20	0.0380	20	1.42	0.002	0.08	0.08
IP-SD-NL-TR20	0.0380	20	2.17	0.003	0.16	0.16
IP-FD-OLQ-TR20	0.0380	22	0.00	0.003	0.22	0.22
IP-FD-NL-TR20	0.0380	22	0.00	0.001	0.24	0.24
QLIN-TR200	0.0375	3	0.56	0.010	0.31	0.32
IP-SD-OLQ-TR200	0.0375	29	1.41	0.002	0.41	0.41
IP-SD-NL-TR200	0.0375	29	2.17	0.003	0.58	0.59
IP-FD-OLQ-TR200	0.0375	30	0.00	0.003	0.88	0.89
IP-FD-NL-TR200	0.0375	29	0.00	0.002	0.95	0.95
QLIN-TR2000	0.0375	3	0.56	0.033	8.23	8.26
IP-SD-OLQ-TR2000	0.0375	35	1.42	0.008	7.21	7.21
IP-SD-NL-TR2000	0.0375	35	2.18	0.004	8.25	8.26
IP-FD-OLQ-TR2000	0.0375	39	0.00	0.003	13.92	13.92
IP-FD-NL-TR2000	0.0375	35	0.00	0.003	13.95	13.95
QLIN-PS10	0.0376	4	0.55	0.009	0.03	0.04
IP-SD-OLQ-PS10	0.0376	24	1.42	0.002	0.08	0.08
IP-SD-NL-PS10	0.0376	28	2.17	0.003	0.21	0.21
IP-FD-OLQ-PS10	0.0376	29	0.00	0.003	0.25	0.26
IP-FD-NL-PS10	0.0376	31	0.00	0.002	0.31	0.31
QLIN-PS40	0.0375	3	0.56	0.008	0.18	0.18
IP-SD-OLQ-PS40	0.0375	29	1.42	0.002	0.34	0.34
IP-SD-NL-PS40	0.0375	29	2.18	0.002	0.46	0.46
IP-FD-OLQ-PS40	0.0375	29	0.00	0.002	0.53	0.53
IP-FD-NL-PS40	0.0375	31	0.00	0.002	0.62	0.62



(a) States.



(b) Controls.

FIGURE 1: Quasilinearization results for the Container Crane problem (QLIN-TR2000).

(but still competitive with all other variants). This indicates that further studies are needed to determine what the best uses are, especially since this problem was a prime candidate for the QLIN method with many OLQ elements. The different QLIN-TR2000 iterations are shown in Fig. 1 where the darker lines indicate later iterations. We note that the initial guess was quite poor for the controls, but reasonable for the states. However, only after a single iteration, all trajectories were nominally similar to their final convergence counterparts.

Now let us turn our focus to the impact of the direct incorporation of OLQ elements. As was anticipated, in all cases, the computational cost was lower when the OLQ matrices were included for the appropriate problem elements, and each variant converged to the same solution with the same DT method specification. In some cases, the iterations were different, potentially due to the differences in the calculated derivatives. The benefit of the OLQ option with the symbolic derivatives provided was between 15–163%, while for the tested finite-difference variants, it was between 0–19% faster. The wide range with SD can be attributed to the fact that the optimization algorithm computational cost starts to dominate when the problem size grows; thus, the marginal gains recorded with TR2000. The minor benefit for the FD variants is because there is generally a high fixed cost with computing the finite differences for this relatively small problem. This illustrates the point that there are clear advantageous to allowing the user of a DT-based tool to define the OLQ elements directly and create the underlying methods that leverage these elements.

A final point of emphasis is the fact that computational costs of direct incorporation of the OLQ elements were minimal. All values of T_{int} were quite small when compared to the other computational costs. Even the QLIN methods, which had to create large matrices with time-varying elements, had a much lower cost than the actual optimization algorithms and symbolic operations. Therefore, there are limited disadvantages in OLQ when properly implemented.

6.2 Van der Pol Oscillator

6.2.1 Problem description. The Van der Pol oscillator is a common test problem for many numerical optimal control methods (see `vpol` in Ref. [10]), including quasilinearization [20,22]. Here we consider two variations: V1 $= \{t_f = 5, c_1 = -\infty, c_2 = -0.3, c_3 = 1, c_4 = 1, c_5 = 1\}$ which has fixed parameter values [48],

TABLE 2: Results for the Van der Pol Oscillator V2 case study.

Method	v	Iter.	T_{sym}	T_{int}	T_{opt}	T
IP-SD-OLQ-TR20	1.9809	15	0.85	0.001	0.04	0.04
IP-SD-NL-TR20	1.9809	15	1.14	0.001	0.06	0.06
IP-FD-OLQ-TR20	1.9809	21	0.00	0.001	0.10	0.10
IP-FD-NL-TR20	1.9809	18	0.00	0.001	0.09	0.09
IP-SD-OLQ-TR200	1.9652	23	0.86	0.002	0.12	0.12
IP-SD-NL-TR200	1.9652	23	1.22	0.001	0.16	0.16
IP-FD-OLQ-TR200	1.9652	26	0.00	0.001	0.18	0.18
IP-FD-NL-TR200	1.9652	27	0.00	0.001	0.25	0.25
IP-SD-OLQ-TR2000	1.9650	26	0.86	0.003	1.10	1.10
IP-SD-NL-TR2000	1.9650	26	1.17	0.002	1.21	1.21
IP-FD-OLQ-TR2000	1.9650	27	0.00	0.002	1.52	1.52
IP-FD-NL-TR2000	1.9650	28	0.00	0.002	2.22	2.22
IP-SD-OLQ-PS10	1.9649	17	0.85	0.002	0.04	0.04
IP-SD-NL-PS10	1.9649	17	1.15	0.001	0.06	0.06
IP-FD-OLQ-PS10	1.9649	22	0.00	0.001	0.07	0.07
IP-FD-NL-PS10	1.9649	29	0.00	0.001	0.12	0.12
IP-SD-OLQ-PS40	1.9650	16	0.86	0.002	0.07	0.07
IP-SD-NL-PS40	1.9650	16	1.15	0.001	0.09	0.09
IP-FD-OLQ-PS40	1.9650	17	0.00	0.001	0.09	0.09
IP-FD-NL-PS40	1.9650	19	0.00	0.001	0.13	0.13
QLIN-TRX	—	did not converge				
QLIN-PSX	—	did not converge				

and V2 $= \{t_f = 5, c_1 = -0.4, c_2 = -0.5, c_3 = 1, c_4 = 0.1, c_5 = 5\}$ which must determine two parameter values [49]. The NLDO problem is:

$$\min_{\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}} \quad \int_0^{t_f} \left[\xi_1^2 + \xi_2^2 + u_1^2\right] dt \tag{25a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}} = \begin{bmatrix} \xi_2 \\ -p_1\xi_1 + \xi_2\left[1 - p_2\xi_1^2\right] + u_1 \end{bmatrix} \tag{25b}$$

$$\boldsymbol{\xi}(0) = (1, 0) \tag{25c}$$

$$c_1 \le \xi_2, \quad c_2 \le u(t) \le c_3 \tag{25d}$$

$$c_4 \le p_1 \le c_5, \quad c_4 \le p_2 \le c_5 \tag{25e}$$

Note that all problem elements except for $\dot{\xi}_2$ are OLQ elements. The linearized form for $\dot{\xi}_2$ was shown in Eq. (17). The initial guess for both variations and all methods can be seen in Fig. 3.

6.2.2 Results. For V1, similar results to Container Crane are observed. This problem has been well studied as a convergent quasilinearization example [20, 22]. More interesting observations can be made for V2, and the results are summarized in Table 2. A very accurate solution is obtained using IP-SD-OLQ-PS40 in only 0.07 s, and is shown in Fig. 2 with optimal parameter values of 0.15206 and 1.5374, respectively.

Here, the basic quasilinearization method *fails to converge* (and was tested with many different initial guesses). The iteration behavior is shown in Fig. 3 where after a few iterations, the solution oscillates between two trajectories. Both state and path constraints enter and exit activity as well as the parameters' simple bound constraints. The inclusion of more advanced techniques such as line searches and trust regions could improve convergence, but, as was discussed in Sec. 4.3.4, the modified approach would be similar to existing methods for solving NLPs.

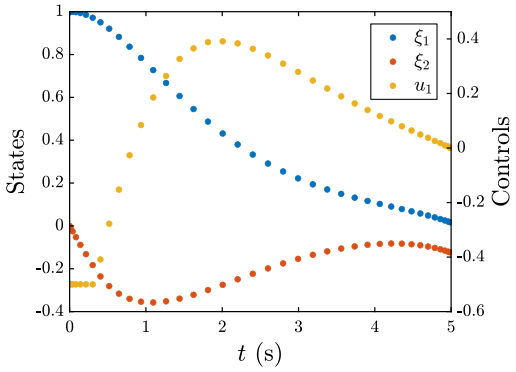


FIGURE 2: Results for the Van der Pol Oscillator problem V2 (IP-SD-OLQ-PS40).

Again, the use of the OLQ option resulted in faster overall solving times in all cases. Computational costs were decreased between 10–50% with SD and between −10% and 71% with FD. The small negative performance hit was due to the increased iterations needed by the OLQ option vs. NL, again indicating that the derivatives are different between the two.

6.3 Co-design Transfer

6.3.1 Problem description. This is a control co-design test problem with a known solution [19]. It has been previously used to study the TLFP method. The NLDO problem is:

$$\min_{\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{p}} \quad \int_0^{t_f} u_1^2 dt \tag{26a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}} = \begin{bmatrix} \xi_2 \\ -p_1\xi_1 + u_1 \end{bmatrix} \tag{26b}$$

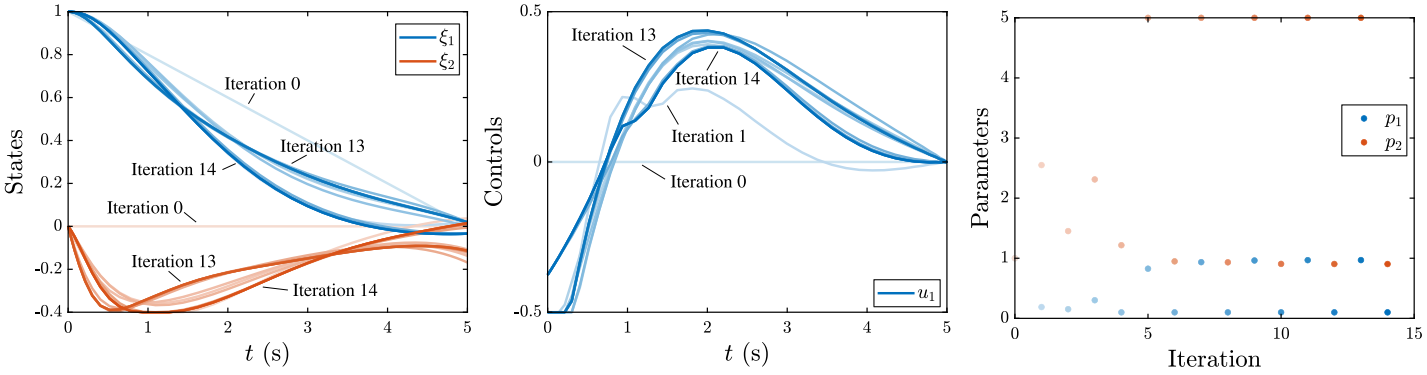
$$\boldsymbol{\xi}(0) = (x_0, v_0) \tag{26c}$$

$$\boldsymbol{\xi}(t_f) = (0, 0) \tag{26d}$$

$$0 \le p_1 \tag{26e}$$

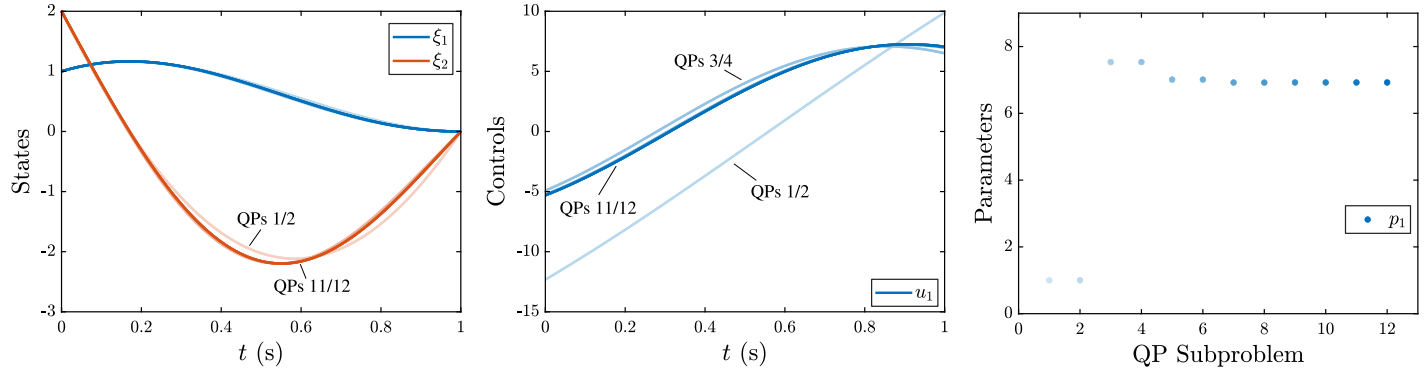
Note that all parts except for $\dot{\xi}_2$ are OLQ elements. The constants are $t_f = 1, x_0 = 1$, and $v_0 = 2$. A poor initial guess of one was used for all optimization variables.

6.3.2 Results. The results for this case study are summarized in Table 3. This is the first of the case study problems that is an LQDO-amenable co-design problem; thus, TLFP is applicable. In Table 3, this strategy is slower than QLIN and IP-SD-OLQ. However, it can be argued that comparisons to FD are more appropriate because TLFP does not require any symbolic opera-



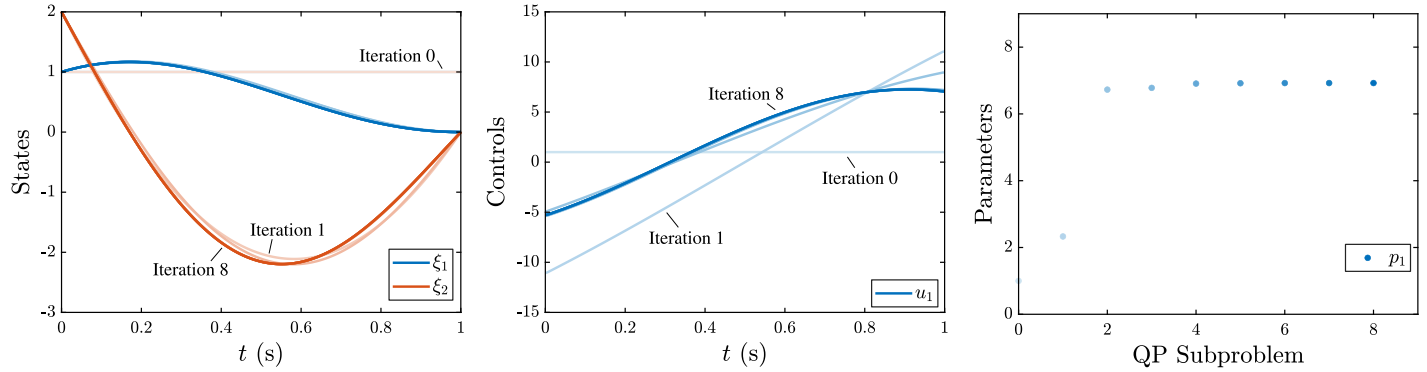
(a) States. (b) Controls. (c) Parameters.

FIGURE 3: Quasilinearization results for the Van der Pol Oscillator problem V2 (QLIN-PS40).



(a) States. (b) Controls. (c) Parameters.

FIGURE 4: Two-level fixed parameter results for the Co-design Transfer problem (TLFP-TR2000).



(a) States. (b) Controls. (c) Parameters.

FIGURE 5: Quasilinearization results for the Co-design Transfer problem (QLIN-TR2000).

tions. Therefore, comparing TLFP and IP-FD-OLQ, we see that the two-level method is faster or only slightly slower. Even if all OLQ elements are being directly incorporated, there are still some problem elements with nonlinear behavior. If a user identifies that their problem is appropriate for the TLFP method, then there still might be some place for the nested co-design method when analytic derivatives are unavailable. This will be further studied in the next example.

TABLE 3: Results for the Co-design Transfer case study.

Method	v	Iter.	T_{sym}	T_{int}	T_{opt}	T
TLFP-TR200	23.6256	6	—	—	—	0.09
QLIN-TR200	23.6256	8	0.25	0.015	0.03	0.05
IP-SD-OLQ-TR200	23.6256	6	0.31	0.001	0.03	0.03
IP-SD-NL-TR200	23.6256	6	0.47	0.001	0.03	0.04
IP-FD-OLQ-TR200	23.6256	9	0.00	0.001	0.11	0.11
IP-FD-NL-TR200	23.6256	8	0.00	0.001	0.11	0.11
TLFP-TR2000	23.6228	6	—	—	—	0.31
QLIN-TR2000	23.6228	8	0.25	0.031	0.17	0.20
IP-SD-OLQ-TR2000	23.6228	6	0.31	0.002	0.14	0.14
IP-SD-NL-TR2000	23.6228	6	0.47	0.001	0.16	0.16
IP-FD-OLQ-TR2000	23.6228	8	0.00	0.003	0.58	0.58
IP-FD-NL-TR2000	23.6228	8	0.00	0.003	0.61	0.61
TLFP-PS10	23.6228	6	—	—	—	0.07
QLIN-PS10	23.6228	8	0.24	0.013	0.02	0.04
IP-SD-OLQ-PS10	23.6228	12	0.31	0.001	0.04	0.04
IP-SD-NL-PS10	23.6228	9	0.47	0.001	0.05	0.05
IP-FD-OLQ-PS10	23.6228	17	0.00	0.001	0.09	0.09
IP-FD-NL-PS10	23.6228	14	0.00	0.001	0.08	0.08

TABLE 4: Results for the Vehicle Suspension Co-design case study.

Method	v	Iter.	T_{sym}	T_{int}	T_{opt}	T
TLFP-TR200	1.977	17	—	—	—	1.10
IP-SD-OLQ-TR200	1.977	12	6.46	0.003	0.3	0.31
IP-SD-NL-TR200	1.977	12	6.47	0.003	0.3	0.33
IP-FD-OLQ-TR200	2.048	598	0.00	0.003	21.8	21.80
IP-FD-NL-TR200	2.048	598	0.00	0.003	21.9	21.92
TLFP-TR2000	1.996	22	—	—	—	9.18
IP-SD-OLQ-TR2000	1.996	13	6.49	0.005	1.6	1.63
IP-SD-NL-TR2000	1.996	13	6.51	0.004	1.7	1.74
IP-FD-OLQ-TR2000	1.997	599	0.00	0.006	216.7	216.68
IP-FD-NL-TR2000	1.997	599	0.00	0.004	216.4	216.43
QLIN-TRX	—	did not converge				

The iteration sequences of TLFP-TR2000 and QLIN-TR2000 are shown in Figs. 4 and 5, respectively. Each method requires the solution of a different sequence of QP subproblems (LQDO problems). Note that each iteration of TLFP has feasible dynamics with respect to the original problem, while this is not necessarily true for QLIN. The finite-difference nature of the TLFP method can also be observed.

6.4 Vehicle Suspension Co-design

6.4.1 Problem description. This is a simplified quarter-car vehicle suspension problem, illustrated in Fig. 6a. Similar versions of this problem have been studied in Refs. [8, 9, 50]. The four states represent: 1) the difference in the unsprung mass and road elevation z_0, 2) velocity of the unsprung mass, 3) difference in the mass positions, and 4) velocity of the spring mass. The single open-loop control variable represents the active force actuator in the suspension, and the two parameters are for the suspension damper and spring constants, respectively. The NLDO problem is:

$$\min_{u,\boldsymbol{\xi},\boldsymbol{p}} \quad \int_0^{t_f} \left[w_1 \xi_1^2 + w_2 \dot{\xi}_4^2 + w_3 u^2 \right] dt \tag{27a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-k_t}{m_{us/4}} & \frac{-[p_1+c_t]}{m_{us/4}} & \frac{p_2}{m_{us/4}} & \frac{p_1}{m_{us/4}} \\ 0 & -1 & 0 & 1 \\ 0 & \frac{p_1}{m_{s/4}} & \frac{-p_2}{m_{s/4}} & \frac{-p_1}{m_{s/4}} \end{bmatrix} \boldsymbol{\xi}(t) + \cdots \tag{27b}$$

$$+ \begin{bmatrix} 0 \\ \frac{-1}{m_{us/4}} \\ 0 \\ \frac{1}{m_{s/4}} \end{bmatrix} u(t) + \begin{bmatrix} -1 \\ \frac{c_t}{m_{us/4}} \\ 0 \\ 0 \end{bmatrix} \dot{z}_0(t)$$

$$\boldsymbol{\xi}(0) = \mathbf{0} \tag{27c}$$

$$|\xi_3(t)| \leq r_{\max} \tag{27d}$$

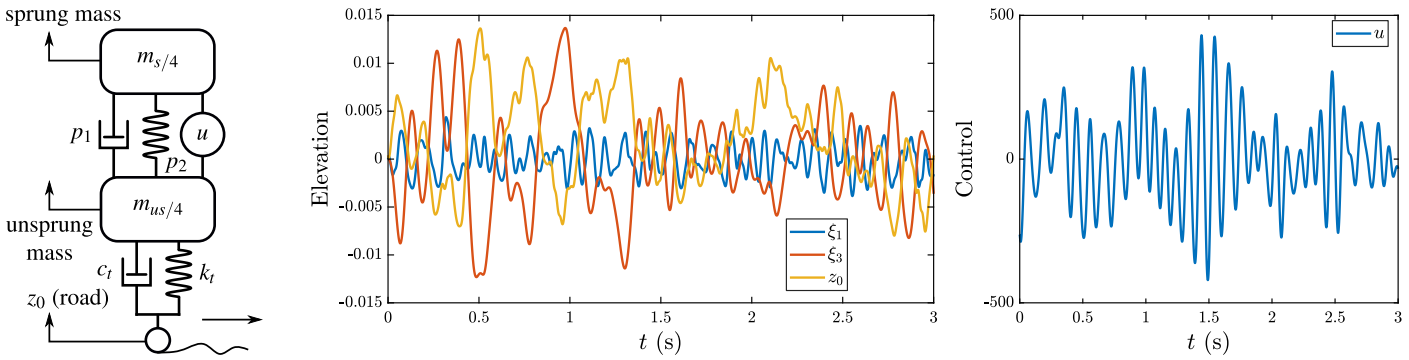
$$\boldsymbol{p}_{\min} \leq \boldsymbol{p} \leq \boldsymbol{p}_{\max} \tag{27e}$$

where objective in Eq. (27a) is a weighted sum of handling, comfort, and control effort metrics, and Eq. (27d) represents a rattle space path constraint. The constants are $w_1 = 10^5, w_2 = 0.5, w_3 = 10^{-5}, k_t = 232500, c_t = 0, m_{us/4} = 65, m_{s/4} = 325, r_{\max} = 0.04, \boldsymbol{p}_{\min} = \left[10^2, 10^2\right]$, and $\boldsymbol{p}_{\max} = \left[10^5, 10^6\right]$. The rough road profile used is from Refs. [8, 9] and is shown in Fig. 6b. The TLFP method is applicable [9].

6.4.2 Results. The results for this case study are summarized in Table 4. The optimal solution using TLFP-TR2000 for select states and the control are shown in Figs. 6b and 6c, respectively. Optimal parameter values were found to be 101.12 and 21558. Because a large mesh is needed to accurately solve this problem, the single-interval PS methods were not tested. Similar to Van der Pol Oscillator V2, this problem did not converge when using QLIN.

The main takeaway from this example is the relative effectiveness of the TLFP method. The cases using the symbolic derivatives were significantly faster (3.5–5.6×) than TLFP, but as was stated in Sec. 6.3, a fairer comparison is to the FD methods. Now the two-level approach is 20–24× faster! Furthermore, the FD methods results have noticeably poorer final solutions, especially for TR200, as well as much slower convergence rates. Therefore, for certain LQDO-amendable co-design problems where the symbolic derivatives are not feasible, the TLFP approach might perform better than the simultaneous approach [9, 19].

Finally, the difference between OLQ and NL was relatively small. In one case it was 7% faster, while in other cases the



(a) Quarter-car vehicle suspension. (b) Select states and road elevation profile. (c) Control.

FIGURE 6: Results for the Vehicle Suspension Co-design problem (TLFP-TR2000).

effect within the margin of error. Only two of the state derivative functions were OLQ elements, and the objective function was nonlinear, so there was little difference between the two options.

7 CONCLUSION

In this article, we explored the various uses of linear-quadratic dynamic optimization (LQDO) and ordinary linear-quadratic (OLQ) elements in the context of nonlinear dynamic optimization (NLDO) and optimal control methods based on direct transcription (DT). OLQ elements are either quadratic objective terms or linear constraints with respect to the DO problem's controls, states, and parameters. Three general classes of strategies for the use of LQDO were discussed.

The first was the direct incorporation of OLQ elements into DT-based NLDO tools with the appropriate matrices. LQDO-based tools such as *DTQP* [16] are tailored to create the appropriate matrices for OLQ elements efficiently. The case studies demonstrated that there is some computational benefit of their direct incorporation using either symbolic derivatives or finite-difference methods. The benefit was observed to be up to 163% but generally in the range of 5–30%. There was generally decreasing improvements as the problem's mesh size increased and select instances where performance was slightly worse because of the differences in the derivative calculations.

Additionally, different two-level optimization problems that have LQDO subproblems were discussed, including the popular nested control co-design method. This particular nested scheme is applicable to the class of NLDO problems where for fixed values of $\boldsymbol{p}$, the resulting subproblem is a convex LQDO problem and was investigated in the final two case studies. Generally, the methods that used symbolic derivatives were slightly faster. However, since the nested co-design method generally does not use any symbolic derivatives, a fair comparison can be made to the finite-difference methods where as two-level optimization was 1.1 to 24× faster with significant advantages on the more challenging vehicle suspension problem.

Finally, the quasilinearization method was presented where the linearization of the constraints and quadratization of the objective function are performed with respect to some reference trajectory, creating a reference LQDO problem. The historical uses of quasilinearization were discussed, including the connection to sequential quadratic programming in DT. In the first case study, the quasilinearization method was shown to be about 2× faster than the interior-point NLP method for smaller mesh sizes. However, this approach did not converge for some of the case study problems, which included parameters. Therefore, care must be taken when utilizing the quasilinearization for solving NLDO problems.

There are a variety of potential future work items. The inclusion of the various LQDO-based methods from Sec. 4 in other DT-based tools could lead to improved outcomes. Furthermore, investigating methods that improve the convergence properties of the quasilinearization without sacrificing the observed benefits could lead to more efficient and robust NLDO tools. The impetus of the methods discussed in this article is to further exploit the specific mathematical problem structure of NLDO problems with limited or no disadvantages.

REFERENCES

[1] Betts, J. T., 2010. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, Jan. doi: 10.1137/1.9780898718577

[2] Foust, R., Chung, S.-J., and Hadaegh, F. Y., 2019. "Solving optimal control with nonlinear dynamics using sequential convex programming". In AIAA Scitech Forum. doi: 10.2514/6.2019-0652

[3] van Straten, G., van Willigenburg, G., van Henten, E., and van Ooteghem, R., 2011. *Optimal Control of Greenhouse Cultivation*. CRC Press, Nov. doi: 10.1201/b10321

[4] Biegler, L. T., 2007. "An overview of simultaneous strategies for dynamic optimization". *Chem. Eng. Process. Process Intensif.*, ***46***(11), Nov., pp. 1043–1053. doi: 10.1016/j.cep.2006.06.021

[5] Biegler, L. T., 2010. *Nonlinear Programming*. Society for Industrial and Applied Mathematics, Jan. doi: 10.1137/1.9780898719383

[6] Deshmukh, A. P., and Allison, J. T., 2015. "Multidisciplinary dynamic optimization of horizontal axis wind turbine design". *Struct. Multidiscip. Optim.,* ***53***(1), Aug., pp. 15–27. doi: 10.1007/s00158-015-1308-y

[7] Chen, P., and Islam, S. M. N., 2005. *Optimal Control Models in Finance*. Kluwer Academic Publishers. doi: 10.1007/b101888

[8] Allison, J. T., Guo, T., and Han, Z., 2014. "Co-design of an active suspension using simultaneous dynamic optimization". *J. Mech. Des.,* ***136***(8), June. doi: 10.1115/1.4027335

[9] Herber, D. R., and Allison, J. T., 2019. "A problem class with combined architecture, plant, and control design applied to vehicle suspensions". *J. Mech. Des.,* ***141***(10), May. doi: 10.1115/1.4043312

[10] Betts, J. T., 2015. A collection of optimal control test problems. Tech. rep., Applied Mathematical Analysis, LLC.

[11] Bryson, Jr., A. E., and Ho, Y.-C., 1975. *Applied Optimal Control*. Taylor & Francis.

[12] Lee, E. S., 1968. *Quasilinearization and Invariant Imbedding: With Applications to Chemical Engineering and Adaptive Control*. Academic Press.

[13] Rao, A. V., 2010. "A survey of numerical methods for optimal control". *Advances in the Astronautical Sciences,* ***135***(1), pp. 497–528.

[14] Kelly, M., 2017. "An introduction to trajectory optimization: how to do your own direct collocation". *SIAM Rev.,* ***59***(4), jan, pp. 849–904. doi: 10.1137/16M1062569

[15] Herber, D. R., 2017. "Advances in combined architecture, plant, and control design". Ph.D. Dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, USA, Dec.

[16] The DTQP project. [Online]. url: https://github.com/danielrherber/dt-qp-project

[17] Fathy, H. K., Reyer, J. A., Papalambros, P. Y., and Ulsoy, A. G., 2001. "On the coupling between the plant and controller optimization problems". In American Control Conference. doi: 10.1109/ACC.2001.946008

[18] Fathy, H. K., Papalambros, P. Y., Ulsoy, A. G., and Hrovat, D., 2003. "Nested plant/controller optimization with application to combined passive/active automotive suspensions". In American Control Conference. doi: 10.1109/ACC.2003.1244053

[19] Herber, D. R., and Allison, J. T., 2018. "Nested and simultaneous solution strategies for general combined plant and control design problems". *J. Mech. Des.,* ***141***(1), Oct. doi: 10.1115/1.4040705

[20] Jaddu, H., and Vlach, M., 2014. "Closed form solution of nonlinear-quadratic optimal control problem by state-control parameterization using chebyshev polynomials". *Int. J. Comput. Appl.,* ***91***(10), Apr., pp. 1–7. doi: 10.5120/15914-5281

[21] Sideris, A., and Bobrow, J. E., 2005. "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems". In American Control Conference. doi: 10.1109/ACC.2005.1470308

[22] Jaddu, H., 2002. "Direct solution of nonlinear optimal control problems using quasilinearization and Chebyshev polynomials". *J. Franklin Inst.,* ***339***(4-5), July, pp. 479–498. doi: 10.1016/S0016-0032(02)00028-5

[23] Li, M., and Peng, H., 2016. "Solutions of nonlinear constrained optimal control problems using quasilinearization and variational pseudospectral methods". *ISA Trans.,* ***62***, May, pp. 177–192. doi: 10.1016/j.isatra.2016.02.007

[24] Betts, J. T., and Huffman, W. P., 1993. "Path-constrained trajectory optimization using sparse sequential quadratic programming". *J. Guid. Control Dyn.,* ***16***(1), Jan., pp. 59–68. doi: 10.2514/3.11428

[25] Liu, X., Lu, P., and Pan, B., 2017. "Survey of convex optimization for aerospace applications". *Astrodynamics,* ***1***(1), Sept., pp. 23–40. doi: 10.1007/s42064-017-0003-8

[26] Tenny, M. J., Wright, S. J., and Rawlings, J. B., 2004. "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming". *Comput. Optim. Appl.,* ***28***(1), Apr., pp. 87–121. doi: 10.1023/B:COAP.0000018880.63497.eb

[27] Biegler, L. T., 1984. "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation". *Comput. Chem. Eng.,* ***8***(3-4), pp. 243–247. doi: 10.1016/0098-1354(84)87012-X

[28] Raghunathan, A. U., and Cairano, S. D., 2014. "Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control". In International Symposium on Mathematical Theory of Networks and Systems, pp. 807–814.

[29] Lu, L., 2015. "Separable nonlinear model predictive control via sequential quadratic programming for large-scale systems". *IFAC-PapersOnLine,* ***48***(23), pp. 495–500. doi: 10.1016/j.ifacol.2015.11.327

[30] Papalambros, P. Y., and Wilde, D. J., 2017. *Principles of Optimal Design*, 3rd ed. Cambridge University Press.

[31] Byrd, R. H., Schnabel, R. B., and Shultz, G. A., 1987. "A trust region algorithm for nonlinearly constrained optimization". *SIAM J. Numer. Anal.,* ***24***(5), Oct., pp. 1152–1170. doi: 10.1137/0724076

[32] Nocedal, J., and Wright, S. J., 2006. *Numerical Optimization*, 2nd ed. Springer. doi: 10.1007/978-0-387-40065-5

[33] Herber, D. R., 2015. Basic implementation of multiple-interval pseudospectral methods to solve optimal control problems. Tech. Report UIUC-ESDL-2015-01, Engineering System Design Lab, Urbana, IL, USA, June. url: http://hdl.handle.net/2142/77888

[34] Åkesson, J., 2008. "Optimica—an extension of modelica supporting dynamic optimization". In International Modelica Conference.

[35] Patterson, M. A., and Rao, A. V., 2014. "GPOPS–II: a matlab software for solving multiple-phase optimalcontrol problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming". *ACM Trans. Math. Software,* ***41***(1), Oct., pp. 1–37. doi: 10.1145/2558904

[36] Becerra, V. M., 2019. PSOPT optimal control solver user manual. User Manual Release 4.0.0 build 2019.02.24.

[37] The Mathworks. sparse. [Online]. url: https://www.mathworks.com/help/matlab/ref/sparse.html

[38] The MathWorks. fmincon. [Online]. url: https://www.mathworks.com/help/optim/ug/fmincon.html

[39] Bard, J. F., 1988. "Convex two-level optimization". *Math. Program.,* ***40-40***(1-3), Jan., pp. 15–27. doi: 10.1007/BF01580720

[40] Behtash, M., and Alexander-Ramos, M. J., 2020. "A decomposition-based optimization algorithm for combined plant and control design of interconnected dynamic systems". *J. Mech. Des.,* ***142***(6), Mar. doi: 10.1115/1.4046240

[41] Liu, T., Azarm, S., and Chopra, N., 2020. "Decentralized multi-subsystem co-design optimization using direct collocation and decomposition-based methods". *J. Mech. Des.,* ***142***(9), Feb., pp. 1–12. doi: 10.1115/1.4046438

[42] Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M., 2015. "Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems". *IEEE Trans. Autom. Control,* ***60***(3), Mar., pp. 644–658. doi: 10.1109/TAC.2014.2354892

[43] Chilan, C. M., Herber, D. R., Nakka, Y. K., Chung, S.-J., Allison, J. T., Aldrich, J. B., and Alvarez-Salazar, O. S., 2017. "Co-design of strain-actuated solar arrays for spacecraft precision pointing and jitter reduction". *AIAA J.,* ***55***(9), Sept., pp. 3180–3195. doi: 10.2514/1.J055748

[44] Bellman, R., and Kalaba, R., 1965. *Quasilinearization and Nonlinear Boundary Value Problems*. Elsevier.

[45] Boggs, P. T., and Tolle, J. W., 1995. "Sequential quadratic programming". *Acta Numerica,* ***4***, Jan., pp. 1–51. doi: 10.1017/S0962492900002518

[46] Quirynen, R., Houska, B., Vallerio, M., Telen, D., Logist, F., Impe, J. V., and Diehl, M., 2014. "Symmetric algorithmic differ-

entiation based exact hessian SQP method and software for economic MPC". In IEEE Conference on Decision and Control. doi: 10.1109/CDC.2014.7039811

[47] Augustin, D., and Maurer, H., 2001. "Sensitivity analysis and real-time control of a container crane under state constraints". In *Online Optimization of Large Scale Systems*. Springer, pp. 69–82. doi: 10.1007/978-3-662-04331-8_4

[48] Canto, E. B., Banga, J. R., Alonso, A. A., and Vassiliadis, V. S., 2002. "Restricted second order information for the solution of optimal control problems using control vector parameterization". *J. Process Control,* ***12***(2), Feb., pp. 243–255. doi: 10.1016/S0959-1524(01)00008-7

[49] Azad, S., and Alexander-Ramos, M. J., 2019. "Reliability-based MDSDO for co-design of stochastic dynamic systems". In International Mechanical Engineering Congress and Exposition. doi: 10.1115/IMECE2019-10632

[50] Clarizia, G., 2019. "Co-design optimization of a tethered multi drone system". M.S. Thesis, Politecnico di Milano, Milano, Italy.