Adjoint- and Hybrid-Based Hessians for Optimization Problems in System Identification

Souransu Nandi
Department of Mechanical and Aerospace Engineering,
University at Buffalo,
Buffalo, NY 14260
e-mail: souransu@buffalo.edu

Tarunraj Singh
Professor
Department of Mechanical and Aerospace Engineering,
University at Buffalo,
Buffalo, NY 14260
e-mail: tsingh@buffalo.edu

An adjoint sensitivity-based approach to determine the gradient and Hessian of cost functions for system identification of dynamical systems is presented. The motivation is the development of a computationally efficient approach relative to the direct differentiation (DD) technique and which overcomes the challenges of the step-size selection in finite difference (FD) approaches. An optimization framework is used to determine the parameters of a dynamical system which minimizes a summation of a scalar cost function evaluated at the discrete measurement instants. The discrete time measurements result in discontinuities in the Lagrange multipliers. Two approaches labeled as the Adjoint and the Hybrid are developed for the calculation of the gradient and Hessian for gradient-based optimization algorithms. The proposed approach is illustrated on the Lorenz 63 model where part of the initial conditions and model parameters are estimated using synthetic data. Examples of identifying model parameters of light curves of type 1a supernovae and a two-tank dynamic model using publicly available data are also included.
[DOI: 10.1115/1.4040072]

1 Introduction

In data assimilation and system identification of dynamic systems (among other fields), it is often required to solve optimization problems where cost functions are functions of the state variables at specific time instants. When gradient-based optimization techniques are used to solve these problems, gradients of the cost with respect to the optimization variables are needed. However, the first-order methods may sometimes perform poorly especially if the condition number of the Hessian (the second derivative of the cost function with respect to the decision variables) is large [1], the system is highly nonlinear or there is a strong interaction between parameters of the model [2]. In these cases, for faster convergence, Hessians of the cost with respect to the optimization variables are desired. In applications, when Hessians are readily available, the Newton method is chosen as it provides excellent local quadratic convergence [3]. In other scenarios, Trust-region algorithms are used where Hessians are usually used to approximate the cost by local quadratic functions [4]. Several other methods (like the Davidon-Fletcher-Powell (DFP), Broyden-Fletcher-Goldfarb-Shanno (BFGS), and other quasi-Newton methods) approximate the Hessian (when computing them becomes numerically intractable) as part of the optimization algorithm.

However, for dynamic systems, calculating the Hessian of a cost function which is state dependent is computationally very expensive since it requires solving matrix and tensor differential equations. As a result, methods to efficiently evaluate these derivatives become paramount. Numerous researchers have addressed the issue of determining gradients and Hessians of the cost using adjoint-based techniques for several different class of problems. For example, Refs. [5] and [6] studied efficient adjoint sensitivities for systems defined by differential algebraic equations, Sengupta et al. in Ref. [7] informally derived the adjoint gradient equations (for a system defined by the first-order ordinary differential equations) and compared its performance to finite difference (FD) and direct differentiation (DD). Raffard et al. [8,9] proposed an adjoint-based approach to determine the gradients of a cost function for parameter identification of protein regulatory networks and planar cell polarity signaling, respectively. The Hessian, however, was approximated by the finite difference of the gradients in both of these papers. Suwartadi et al. [10] investigated adjoint-based gradients for optimization problems with state constraints as well as a summation cost. Raffard and Tomlin [1] proposed the second-order adjoint-based algorithms to deal with partial differential equations in optimization problems and illustrated it on a problem of air traffic flow. Extensive literature ([2,11,12] and references therein) is also present to determine Hessian-vector products cheaply using adjoint-based methods. Several other papers (for example, Refs. [13–15]) studied the use of adjoint-based algorithms for optimal control or sensitivity analysis where the cost metric is an integral function.

This paper focuses on determining adjoint-based second-order derivatives of cost functions which are summations of a state and/or measurement dependent scalar function, reflecting the availability of measurements at discrete time instants. This results in discontinuities in the time evolution of the Lagrange multipliers, necessitating additional boundary conditions on the co-states at the discrete time instants. Two separate algorithms (the adjoint and the hybrid methods) are explored in this framework inspired by the literature review presented previously. The proposed algorithms develop novel expressions for Hessians which do not affect codes necessary for integrations. Hence, any existing numerical integrators can be used to implement them, thereby employing a "differentiate then discretize" strategy (a strategy similar to Ref. [2]) as opposed to "discretize then differentiate." As a result, the algorithms can be used on stiff as well as nonstiff systems as long as appropriate numerical integrators are employed. The efficiency of the adjoint-based approaches is compared to the existing approaches of FD and DD to illustrate the computational benefits.

The paper has been structured as follows: Section 2 formally defines the problem statement and the necessary notation required throughout the document. Section 3 presents the methods of finite difference and illustrates the effect of perturbation values on the derivative errors. Section 4 reviews the existing method of direct differentiation. Sections 5 and 6 present the adjoint and the hybrid method, respectively, emphasizing the novel way to evaluate

Contributed by the Dynamic Systems Division of ASME for publication in the Journal of Dynamic Systems, Measurement, and Control. Manuscript received June 30, 2017; final manuscript received April 19, 2018; published online May 22, 2018. Assoc. Editor: Jongeun Choi.

Hessians. Finally, the document ends with three examples of system identification problems including a chaotic dynamic system, light curve of type 1a supernovae, and a two-tank model in Sec. 8 along with concluding remarks in Sec. 9.

2 Problem Statement

Dynamic models considered in this work are of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}, t) \quad \text{with} \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector of the system, $\boldsymbol{x}(t_0)$ is the initial condition of the states, and $\boldsymbol{p} \in \mathbb{R}^p$ is the parameter vector on which the system model depends.

Consider a generic scalar cost function of the form

$$J(\boldsymbol{x}_0, \boldsymbol{p}) = \sum_{i=0}^{N} g(\boldsymbol{y}(t_i), \boldsymbol{x}(t_i), \boldsymbol{p}) \tag{2}$$

where $g(\boldsymbol{y}(t_i), \boldsymbol{x}(t_i), \boldsymbol{p})$ represents the value of a scalar function $g(\boldsymbol{y}, \boldsymbol{x}(t), \boldsymbol{p})$ (C^2 continuous in $\boldsymbol{x}$ and $\boldsymbol{p}$) at the ith time-step (t_i). In the context of a system identification problem (where the initial conditions and the parameters of the system are being estimated), $g(\boldsymbol{y}(t_i), \boldsymbol{x}(t_i), \boldsymbol{p})$ could be the squared residual between the model output and the observation $\boldsymbol{y}(t_i)$ of the true plant.

For simplicity, in all subsequent equations, $J(\boldsymbol{x}_0, \boldsymbol{p})$ is written as J. Similarly, $g(\boldsymbol{y}(t_i), \boldsymbol{x}(t_i), \boldsymbol{p})$ and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}, t)$ have also been simplified to just g_i and $\boldsymbol{f}$, respectively.

From an optimization point of view, it is required to find a $\boldsymbol{p}$ and an $\boldsymbol{x}_0$ which minimizes J. The objective of this work is to facilitate the use of all derivative-based optimization techniques to solve the problem by developing efficient ways to determine the gradients and Hessians of the cost function with respect to the optimization variables (i.e., $\boldsymbol{p}$ and $\boldsymbol{x}_0$). If the optimization variables are grouped as $\boldsymbol{q} = [\boldsymbol{p}^{\mathrm{T}}, \boldsymbol{x}_0^{\mathrm{T}}]^{\mathrm{T}}$ where $\boldsymbol{p} = [p_1, \cdots, p_p]^{\mathrm{T}}$ and $\boldsymbol{x}_0 = [x_{01}, \ldots, x_{0n}]^{\mathrm{T}}$, then the goal is to evaluate

$$\underbrace{\frac{dJ}{d\boldsymbol{q}}}_{(p+n)\times 1} = \begin{bmatrix} \frac{dJ}{dp_1} \\ \vdots \\ \frac{dJ}{dp_p} \\ \frac{dJ}{dx_{01}} \\ \vdots \\ \frac{dJ}{dx_{0n}} \end{bmatrix} \quad \text{and} \quad \underbrace{\frac{d^2J}{d\boldsymbol{q}^2}}_{(p+n)\times(p+n)} = \begin{bmatrix} \frac{d^2J}{d\boldsymbol{p}^2} & \frac{d^2J}{d\boldsymbol{p}d\boldsymbol{x}_0} \\ \frac{d^2J}{d\boldsymbol{x}_0 d\boldsymbol{p}} & \frac{d^2J}{d\boldsymbol{x}_0^2} \end{bmatrix} \tag{3}$$

where we define the second derivative of any scalar function $((\cdot))$ with respect to any two vectors ($\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^p$) by

$$\underbrace{\frac{d^2(\cdot)}{d\boldsymbol{a}d\boldsymbol{b}}}_{n\times p} = \begin{bmatrix} \frac{d^2(\cdot)}{da_1 db_1} & \cdots & \frac{d^2(\cdot)}{da_1 db_p} \\ \vdots & \ddots & \vdots \\ \frac{d^2(\cdot)}{da_n db_1} & \cdots & \frac{d^2(\cdot)}{da_n db_p} \end{bmatrix} \tag{4}$$

It should be noted that, since $\boldsymbol{x}$ is a function of $\boldsymbol{x}_0$ and $\boldsymbol{p}$, J can be expressed as $J(\boldsymbol{q})$.

In this work, comparison of the adjoint and the hybrid method to other existing approaches has been realized on the basis of the number of scalar integrations required in each algorithm. As more integrations require a higher computational effort, this number (referred to as $N_s^{(\text{method})}$ in the document) has been used to provide an indirect comparison of computational efficiency of each method. $N_s^{(\text{method})}$ is derived in terms of two variables, namely, the number of states (n) and the number of parameters (p) in the model equation (Eq. (1)).

Realizing that no two integrations are the same, for example, in a variable step-size integrator, the local error for every step is a function of the form of the differential equation, the comparison of the number of integrations while not being ideal is a reasonable metric for comparison of computational cost. The number of iterations to convergence might be a metric, but it might not reasonably reflect the computational cost per iteration. Efficiency refers to the time required to reach the optimum and is a reflection of computational efficiency. Effectiveness refers to the ability of the algorithm to converge to the optimal solution. Effectiveness does not reflect computational cost which is why we will refer to both efficiency and effectiveness in comparing various algorithms in the numerical examples section of the paper.

3 Finite Difference

The central idea in FD to calculate the gradients is to observe the fractional change in cost when each optimization variable is sequentially perturbed. The gradient can be calculated using either the forward, backward, or central difference approaches. The forward difference formula is given by

$$\frac{dJ}{dq_i}(\boldsymbol{q}) = \frac{J^{(+q_i)}(\boldsymbol{q}) - J(\boldsymbol{q})}{\delta q_i} \tag{5}$$

where

$$J^{(\pm q_i)}(\boldsymbol{q}) = J(q_1, \ldots, q_i \pm \delta q, \ldots, q_{n+p}) \tag{6}$$

will be considered to illustrate the challenges of using finite difference approaches to estimate gradients. A derivation of the forward-difference formula is now presented. The derivation starts from a Taylor series expansion of the cost function. Although, the derivation is shown only for a system with one parameter, it can be generalized to a system with multiple parameters.

The cost function can be written as

$$J(q + \delta q) = J(q) + \frac{dJ}{dq}(q)\delta q + \cdots \tag{7}$$

$$\Rightarrow \frac{dJ}{dq}(q) = \frac{J(q + \delta q) - J(q)}{\delta q} - \cdots \tag{8}$$

where the ellipsis represents the higher-order terms (HOTs). If the HOTs are ignored in Eq. (8), it leads to Eq. (5). Similar perturbation-based strategies can also be developed to determine Hessians.

It should be noted that FD only provides an approximation and is greatly dependent on the perturbation size δq_i. For the purpose of error analysis, consider the following equations:

$$\underbrace{\frac{dJ}{dq}(q)}_{\text{True}} = \underbrace{\frac{J(q + \delta q) - J(q)}{\delta q}}_{\text{FD}} + e_{\text{num}}(\delta q) - \underbrace{\frac{1}{2}\frac{d^2J}{dq^2}(q)\delta q - \cdots}_{\text{HOT}} \tag{9}$$

and

$$\texttt{TotalFDError} = |\mathrm{e}_{\text{num}} - \frac{1}{2}\frac{\mathrm{d}^2\mathrm{J}}{\mathrm{dq}^2}(\mathrm{q})\delta\mathrm{q} - \cdots| \tag{10}$$

e_{num} is the term introduced to denote all numerical errors (like round off or numerical integration errors). `TotalFDError` represents the total error in the derivative that is unaccounted for while calculating the gradient using the forward difference approach (which is the sum of the numerical errors and truncation error).

To illustrate the impact of the magnitude of perturbation on the FD error, consider the scalar dynamical system

$$\dot{x} = -ax^2 \tag{11}$$

This equation can be solved in closed-form

$$x(t) = \frac{x_0}{1 + x_0 at} \tag{12}$$

where $x_0 = x(0)$ is the initial condition of the state $x(t)$. Let the cost function be

$$J = \sum_{i=0}^{N} g(x(t_i), a) \quad \text{where } g(x(t), a) = \frac{1}{x(t)^2} \tag{13}$$

This leads to

$$J = \sum_{i=0}^{N} \frac{1}{x_0^2}\left(1 + 2x_0 a t_i + x_0^2 a^2 t_i^2\right) = 109.585 \tag{14}$$

for the parameter values, initial conditions and final time given by $a = 1, x_0 = 2$ and $t_{N=100} = 1$. Let the uniform time interval be 0.01, i.e., $t_{i+1} - t_i = 0.01$. For illustration of FD behavior on derivatives, sensitivities of the cost only with respect to the parameter p are presented. The analytical gradient and the Hessian are, therefore, given by

$$\begin{aligned} \frac{dJ}{dp} &= \sum_{i=0}^{N} \frac{2}{x_0^2}\left(x_0 t_i + x_0^2 a t_i^2\right) = 118.17 \text{ and} \\ \frac{d^2J}{dp^2} &= \sum_{i=0}^{N} 2t_i^2 = 67.67 \end{aligned} \tag{15}$$

respectively.

The plot of the variation of the total FD error and its components (calculated using the analytical expressions in Eq. (15)) in Fig. 1 for the example problem allows for a number of interesting observations. Three different cases with varying integration tolerances were considered here. It is seen that the numerical error (e_{num}) keeps decreasing initially with increasing perturbation values before saturating. The levels of saturation are seen to be dependent on their respective integration tolerances. Furthermore, the error term originating from ignoring the HOT in the Taylor series begins dominating the total FD error beyond a certain point on the δp axis, and thus, causes the `TotalFDerror` to increase.

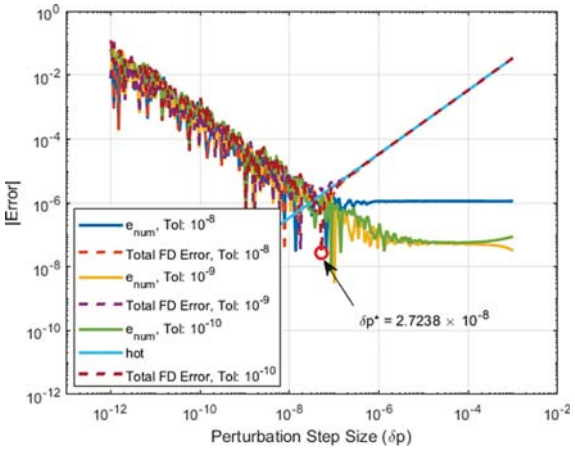

Fig. 1 Forward difference error analysis for the gradient of J

Hence, in summary, the total error from forward difference during gradient computation first decreases with increasing perturbation value due to decreasing numerical errors and then increases due to the increasing influence of the HOT in the Taylor series expansion. This motivates the appropriate selection of δp such that the `TotalFDerror` is at the minima. In the current example, the case with a tolerance of 10^{-10} in Fig. 1 has minima at $\delta p^* = 2.7238 \times 10^{-8}$. However, in a generic problem, it is infeasible to know what the optimal value of δp is. Note that although we have used the forward difference to illustrate the impact of δp on the error in estimated gradient, we will use the central difference approach to estimate the gradient and Hessian for the numerical examples in view of its improved accuracy.

4 Direct Differentiation

As the name suggests, direct differentiation directly takes the derivative of the cost function J with respect to the optimization variables ($\boldsymbol{q}$) (a similar development can be found in Ref. [2]). The method is rather straightforward and is expounded on in Secs. 4.1 and 4.3.

4.1 Gradient. The cost function of interest is

$$J(\boldsymbol{q}) = \sum_{i=0}^{N} g_i \tag{16}$$

After defining the following notation:

$$\begin{aligned} \underbrace{\frac{d\boldsymbol{a}}{d\boldsymbol{b}}}_{(p\times n)} &= \begin{bmatrix} \frac{da_1}{db_1} & \cdots & \frac{da_n}{db_1} \\ \vdots & \ddots & \vdots \\ \frac{da_1}{db_p} & \cdots & \frac{da_n}{db_p} \end{bmatrix}; \quad \underbrace{\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}}}_{(p\times n)} = \begin{bmatrix} \frac{\partial a_1}{\partial b_1} & \cdots & \frac{\partial a_n}{\partial b_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_1}{\partial b_p} & \cdots & \frac{\partial a_n}{\partial b_p} \end{bmatrix}; \\ \underbrace{\frac{\partial(\cdot)}{\partial \boldsymbol{a}}}_{n\times 1} &= \begin{bmatrix} \frac{\partial(\cdot)}{\partial a_1} & \cdots & \frac{\partial(\cdot)}{\partial a_n} \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{17}$$

where $\boldsymbol{a} = [a_1, \ldots, a_n]^{\mathrm{T}}$, $\boldsymbol{b} = [b_1, \ldots, b_p]^{\mathrm{T}}$, and $(\cdot)$ is a scalar quantity, the derivative of the cost function is

$$\frac{dJ}{d\boldsymbol{q}} = \begin{bmatrix} \frac{dJ}{d\boldsymbol{p}} \\ \frac{dJ}{d\boldsymbol{x}_0} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} \left(\frac{\partial g}{\partial p} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{x}}\right)_i \\ \sum_{i=0}^{N} \left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial g}{\partial \boldsymbol{x}}\right)_i \end{bmatrix} \tag{18}$$

$(\cdot)_i$ represents $(\cdot)$ evaluated at time t_i. Except the sensitivity of the states to the parameters and initial conditions (i.e., $(d\boldsymbol{x}/d\boldsymbol{p})$ and $(d\boldsymbol{x}/d\boldsymbol{x}_0)$), all other terms are known since the analytical form of g is known. $(d\boldsymbol{x}/d\boldsymbol{p})$ and $(d\boldsymbol{x}/d\boldsymbol{x}_0)$ can be found from the derivatives of the dynamic model equation (Eq. (1)) with respect to $\boldsymbol{p}$ and $\boldsymbol{x}_0$, respectively. These equations are

$$\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{q}} = \begin{bmatrix} \frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}} \\ \frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{x}_0} \end{bmatrix} = \begin{bmatrix} \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \\ \frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \end{bmatrix} \tag{19}$$

Integrating Eq. (19) allows one to determine $(d\boldsymbol{x}/d\boldsymbol{p})$ and $(d\boldsymbol{x}/d\boldsymbol{x}_0)$ over time. Once they are known in time, they can be substituted in Eq. (18) to evaluate the desired gradient. It should be noted that Eq. (19) is a matrix differential equation which needs $(p + n) \times n$ simultaneous scalar integrations.

4.2 Tensor-Matrix Operators. Before the derivation of the Hessian via DD is presented, three operators are defined which operate on the second-order tensors (or three-dimensional (3D) matrices) and two-dimensional (2D) matrices. These operators can then be used to express the Hessian in a concise manner.

4.2.1 Operator 1. If T is a 3D matrix of dimensions $(a \times b \times c)$ and M is a 2D matrix of dimension $(c \times d)$, then

$$(T^{\rightarrow M})_{i,j,l} = \sum_{k=1}^{c} T_{i,j,k} M_{k,l} \tag{20}$$

where $T^{\rightarrow M} \in \mathbb{R}^{(a \times b \times d)}$. Operator 1 is developed to write the following derivative concisely:

$$\frac{d}{d\underbrace{\boldsymbol{b}}_{b\times 1}}\left(\underbrace{A(\boldsymbol{b})}_{a\times c}\underbrace{M}_{c\times d}\right) = \left(\underbrace{\frac{dA(\boldsymbol{b})}{d\boldsymbol{b}}}_{T}\right)^{\rightarrow M} \tag{21}$$

4.2.2 Operator 2. If T is a 3D matrix of dimensions $(b \times c \times d)$ and M is a 2D matrix of dimension $(a \times b)$, then

$$({}^{M\rightarrow}T)_{i,l,k} = \sum_{j=1}^{b} T_{j,l,k} M_{i,j} \tag{22}$$

where ${}^{M\rightarrow}T \in \mathbb{R}^{(a \times c \times d)}$. Operator 2 is developed to write the following derivative concisely:

$$\frac{d}{d\underbrace{\boldsymbol{c}}_{c\times 1}}\left(\underbrace{M}_{a\times b}\underbrace{A(\boldsymbol{c})}_{b\times d}\right) = {}^{M\rightarrow}\underbrace{\frac{dA(\boldsymbol{c})}{d\boldsymbol{c}}}_{T} \tag{23}$$

4.2.3 Operator 3. If T is a 3D matrix of dimensions $(a \times b \times c)$ and M is a 2D matrix of dimension $(b \times d)$, then

$$(T \rightarrow M)_{i,l,k} = \sum_{j=1}^{b} T_{i,j,k} M_{j,l} \tag{24}$$

where $T \rightarrow M \in \mathbb{R}^{(a \times d \times c)}$. Operator 3 is developed to write the following derivative concisely:

$$\frac{d}{d\underbrace{\boldsymbol{d}}_{d\times 1}}\left(\underbrace{A(\boldsymbol{x}(\boldsymbol{d}))}_{a\times c}\right) = \underbrace{\frac{dA(\boldsymbol{x})}{d\boldsymbol{x}}}_{T} \rightarrow \underbrace{\frac{d\boldsymbol{x}}{d\boldsymbol{d}}}_{b\times d} \tag{25}$$

Operators 1–3 have been used to represent tensor–matrix products henceforth.

4.3 Hessian. Similar to the method used to find the gradient, the Hessian of the cost function via DD can be evaluated by directly differentiating the gradient $(dJ/d\boldsymbol{q})$ with respect to the $\boldsymbol{q}$ vector.

Therefore, on differentiating Eq. (18), we get

$$\frac{d^2J}{d\boldsymbol{q}^2} = \begin{bmatrix} \dfrac{d^2J}{d\boldsymbol{p}^2} & \dfrac{d^2J}{d\boldsymbol{p}d\boldsymbol{x}_0} \\ \dfrac{d^2J}{d\boldsymbol{x}_0 d\boldsymbol{p}} & \dfrac{d^2J}{d\boldsymbol{x}_0^2} \end{bmatrix} \tag{26}$$

where

$$\frac{d^2J}{d\boldsymbol{p}^2} = \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\frac{\partial g}{\partial \boldsymbol{x}}}\right)_i \tag{27}$$

$$\frac{d^2J}{d\boldsymbol{x}_0^2} = \sum_{i=0}^{N}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}^{\mathrm{T}} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{x}_0^2}^{\rightarrow\frac{\partial g}{\partial \boldsymbol{x}}}\right)_i \text{ and} \tag{28}$$

$$\frac{d^2J}{d\boldsymbol{x}_0 d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{x}_0 d\boldsymbol{p}}^{\rightarrow\frac{\partial g}{\partial \boldsymbol{x}}}\right)_i \tag{29}$$

The known quantities $((\partial^2 g/\partial \boldsymbol{x}^2), (\partial^2 g/\partial \boldsymbol{x}\partial \boldsymbol{p}), (\partial^2 g/\partial \boldsymbol{p}\partial \boldsymbol{x})$, and $(\partial^2 g/\partial \boldsymbol{p}^2))$ can be defined in a manner similar to Eq. (4).

The unknown quantities in Eqs. (27)–(29) are the tensors $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$, $(d^2\boldsymbol{x}/d\boldsymbol{x}_0 d\boldsymbol{p})$, and $(d^2\boldsymbol{x}/d\boldsymbol{x}_0^2)$. These quantities need to be calculated dynamically from a tensor differential equation derived from differentiating equation (19) with respect to $\boldsymbol{p}$ and $\boldsymbol{x}_0$. On doing so, we get three independent equations

$$\begin{aligned}\frac{d^2\dot{\boldsymbol{x}}}{d\boldsymbol{p}^2} = \frac{\partial^2 f}{\partial \boldsymbol{p}^2} + \left(\frac{\partial^2 f}{\partial \boldsymbol{p}\partial \boldsymbol{x}} \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right) + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right) \\ + \left[\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right] + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\frac{\partial f}{\partial \boldsymbol{x}}}\end{aligned} \tag{30}$$

$$\begin{aligned}\frac{d^2\dot{\boldsymbol{x}}}{d\boldsymbol{x}_0 d\boldsymbol{p}} = \left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right) \\ + \left[\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right] + \frac{d^2\boldsymbol{x}}{d\boldsymbol{x}_0 d\boldsymbol{p}}^{\rightarrow\frac{\partial f}{\partial \boldsymbol{x}}} \text{ and}\end{aligned} \tag{31}$$

$$\frac{d^2\dot{\boldsymbol{x}}}{d\boldsymbol{x}_0^2} = \left[\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}^{\mathrm{T}}\right] + \frac{d^2\boldsymbol{x}}{d\boldsymbol{x}_0^2}^{\rightarrow\frac{\partial f}{\partial \boldsymbol{x}}} \tag{32}$$

where the tensors can be defined via the notation in Fig. 2.

Since Eqs. (30)–(32) are tensor differential equations of $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$, $(d^2\boldsymbol{x}/d\boldsymbol{x}_0 d\boldsymbol{p})$, and $(d^2\boldsymbol{x}/d\boldsymbol{x}_0^2)$, they need $(p^2 n)$, (pn^2), and (n^3) scalar integrations to compute, respectively. Once, $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$, $(d^2\boldsymbol{x}/d\boldsymbol{x}_0 d\boldsymbol{p})$, and $(d^2\boldsymbol{x}/d\boldsymbol{x}_0^2)$are known over time, they can be substituted in Eq. (26) to evaluate the Hessian.

5 Adjoint Method

The direct differentiation provides a fairly straightforward method to obtaining the gradients and the Hessians. However, in DD, while calculating the gradient $((dJ/d\boldsymbol{q}))$, the first step was to determine the sensitivity of the states $((d\boldsymbol{x}/d\boldsymbol{q}))$. Similarly, while calculating the Hessian $((d^2J/d\boldsymbol{q}^2))$, the second derivative of the states to the variables $(d^2\boldsymbol{x}/d\boldsymbol{q}^2)$ was needed. Determination of $(d\boldsymbol{x}/d\boldsymbol{q})$ and $(d^2\boldsymbol{x}/d\boldsymbol{q}^2)$ requires the solution to a matrix and a tensor differential equation, respectively; both of which are relatively expensive.

To avoid those expensive calculations, an alternative method of computing gradients and Hessians can be devised called the adjoint method [16]. It provides an efficient way to determine gradients without having to calculate the sensitivity of the states $((d\boldsymbol{x}/d\boldsymbol{q}))$ and a way to determine Hessians without computing the expensive $(d^2\boldsymbol{x}/d\boldsymbol{q}^2)$, thereby improving the computational efficiency.

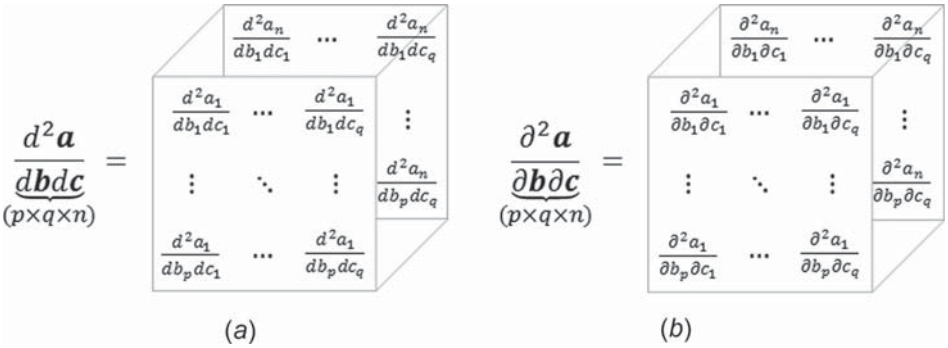

Fig. 2 Visualization of the second derivative tensors where $\boldsymbol{a}\in\mathbb{R}^n$, $\boldsymbol{b}\in\mathbb{R}^p$, and $\boldsymbol{c}\in\mathbb{R}^q$: (*a*) ($d^2\boldsymbol{a}/d\boldsymbol{b}d\boldsymbol{c}$) and (*b*) ($\partial^2\boldsymbol{a}/\partial\boldsymbol{b}\partial\boldsymbol{c}$)

5.1 Gradient. The derivation starts with an augmented cost function of interest (L_1) (of the form of a Lagrangian) as

$$L_1(\boldsymbol{q}) = \sum_{i=0}^{N} g_i + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-} (\dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x},\boldsymbol{p},t))^{\mathrm{T}}\boldsymbol{\lambda}dt\right) \tag{33}$$

where $\boldsymbol{\lambda}\in\mathbb{R}^n$ is a new set of introduced variables (also called the co-states and are analogous to Lagrange multipliers). $\boldsymbol{\lambda}$ is assumed to be a discontinuous variable being nondifferentiable at time points t_i (i.e., the value of $\boldsymbol{\lambda}$ jumps at the edges of the intervals). This is why the total integral over time has been broken into N integrals over N time intervals. When the state equations are satisfied, the second summation term in L_1 (Eq. (33)) becomes 0, making $L_1 = J$. In that case, $(dL_1/d\boldsymbol{q}) = (dJ/d\boldsymbol{q})$ also holds true. Since, the states ($\boldsymbol{x}$) are always determined by solving the state equation, the gradient of the cost function $(dJ/d\boldsymbol{q})$ is always equal to the gradient of the augmented cost function $(dL_1/d\boldsymbol{q})$. Using this property, the gradient of the augmented cost is ultimately evaluated.

An expression for the gradient can be determined by differentiating Eq. (33) with respect to $\boldsymbol{q}$ to get $dL_1/d\boldsymbol{p} = [dL_1/d\boldsymbol{p}^T,\ dL_1/d\boldsymbol{x}_0{}^T]^T$. The development of only $(dL_1/d\boldsymbol{p})$ is presented here since deriving $(dL_1/d\boldsymbol{x}_0)$ is almost identical.

An expression for $(dL_1/d\boldsymbol{p})$ is obtained from L_1 as

$$\frac{dL_1}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}} - \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\right)\boldsymbol{\lambda}dt\right) \tag{34}$$

Now, using the properties of integration by parts on the term $(d\dot{\boldsymbol{x}}/d\boldsymbol{p})$, we get

$$\int_{t_i^+}^{t_{(i+1)}^-}\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}}\boldsymbol{\lambda}dt = \left[\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\boldsymbol{\lambda}\right]_{t_i^+}^{t_{(i+1)}^-} - \int_{t_i^+}^{t_{(i+1)}^-}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\dot{\boldsymbol{\lambda}}\right)dt \tag{35}$$

With the substitution of Eq. (35), Eq. (34) simplifies to

$$\begin{aligned}\frac{dL_1}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \sum_{i=1}^{N-1}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_i\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_i + \boldsymbol{\lambda}_{i^-} - \boldsymbol{\lambda}_{i^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_0\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_0 - \boldsymbol{\lambda}_{0^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_N\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_N + \boldsymbol{\lambda}_{N^-}\right]\\ + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\left(-\dot{\boldsymbol{\lambda}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\lambda}\right)dt\right) + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\boldsymbol{\lambda}\right)dt\right)\end{aligned} \tag{36}$$

It should be noted that $(d\boldsymbol{x}/d\boldsymbol{p})$ and $(\partial g/\partial \boldsymbol{x})$ are continuous functions, i.e., $(d\boldsymbol{x}/d\boldsymbol{p})_i = (d\boldsymbol{x}/d\boldsymbol{p})_{i^-} = (d\boldsymbol{x}/d\boldsymbol{p})_{i^+}$ and $(\partial g/\partial \boldsymbol{x})_i = (\partial g/\partial \boldsymbol{x})_{i^-} = (\partial g/\partial \boldsymbol{x})_{i^+}$.

Equation (36) is still dependent on $(d\boldsymbol{x}/d\boldsymbol{p})$. However, the whole purpose of the Adjoint method was to avoid calculating $(d\boldsymbol{x}/d\boldsymbol{p})$. To make this possible, all the terms that are associated with $(d\boldsymbol{x}/d\boldsymbol{p})$ need to be eliminated (shown in gray in Eq. (36)). The first step is to solve the co-state differential equation

$$-\dot{\boldsymbol{\lambda}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\lambda} = 0 \tag{37}$$

Once Eq. (37) is solved and $\boldsymbol{\lambda}$ is known over time, Eq. (36) simplifies to

$$\frac{dL_1}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \sum_{i=1}^{N-1}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_i\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_i + \boldsymbol{\lambda}_{i^-} - \boldsymbol{\lambda}_{i^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_0\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_0 - \boldsymbol{\lambda}_{0^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_N\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_N + \boldsymbol{\lambda}_{N^-}\right] + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\boldsymbol{\lambda}\right)dt\right) \tag{38}$$

Equation (37) needs to be solved separately over each time interval (since $\boldsymbol{\lambda}$ is discontinuous at the edges of the intervals). A set of boundary conditions for each of those intervals is also necessary. A smart selection of these boundary conditions can be used to eliminate some of the other terms containing $(d\boldsymbol{x}/d\boldsymbol{p})$ in Eq. (38). Assuming $\boldsymbol{\lambda}_{N^-} = -(\partial g/\partial \boldsymbol{x})_N$, Eq. (37) can be solved by integrating it

backward in time from t_{N^-} to t_{N-1^+}. This eliminates the need for the term $(d\boldsymbol{x}/d\boldsymbol{p})_N$ from Eq. (38) since it now multiplies 0. The next step is to evaluate $\boldsymbol{\lambda}$ in the time intervals between t_{i+1^-} and t_{i^+}. This is again done by solving Eq. (37) by integrating it back in the respective time intervals. However, this time, the boundary conditions $\boldsymbol{\lambda}_{i+1^-}$ for each interval are calculated by solving the algebraic equation

$$\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_i + \boldsymbol{\lambda}_{i^-} - \boldsymbol{\lambda}_{i^+} = 0 \tag{39}$$

where $\boldsymbol{\lambda}_{i^+}$ is known: as it is the terminal $\boldsymbol{\lambda}$ from the solution of Eq. (37) in the previous time interval. Such selection of boundary conditions removes the need for evaluating $(d\boldsymbol{x}/d\boldsymbol{p})_i$.

Therefore, in summary, $\boldsymbol{\lambda}$ needs to be solved separately over each time interval (using Eq. (37)). At the end of each integration, the boundary value of $\boldsymbol{\lambda}$ for the next integration (or time interval) is determined (using Eq. (39)).

Solving Eq. (39) causes the second term in Eq. (38) to be 0 and simplifies it to

$$\frac{dL_1}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_0\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_0 - \boldsymbol{\lambda}_{0^+}\right] + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\boldsymbol{\lambda}\right)dt\right) \tag{40}$$

Moreover, since the initial value of the states is independent of the parameters, $(d\boldsymbol{x}/d\boldsymbol{p})_0 = 0$. Therefore, the second term in Eq. (40) vanishes and Eq. (40) simplifies to

$$\frac{dJ}{d\boldsymbol{p}} = \frac{dL_1}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\boldsymbol{\lambda}\right)dt\right) \tag{41}$$

Equation (37) is also called the adjoint equation (and hence the name adjoint method). Equation (41) represents the final equation that needs to be evaluated to calculate the gradient.

Similarly, a gradient equation for the initial conditions can also be obtained. It can be shown that

$$\frac{dJ}{d\boldsymbol{x}_0} = \frac{dL_1}{d\boldsymbol{x}_0} = \left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_0 - \boldsymbol{\lambda}_{0^+} \tag{42}$$

where the co-states $\boldsymbol{\lambda}$ are solved in an identical fashion.

5.2 Hessian. A critical drawback of the direct differentiation method for calculating the Hessian was the need to calculate $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$, $(d^2\boldsymbol{x}/d\boldsymbol{x}_0 d\boldsymbol{p})$, and $(d^2\boldsymbol{x}/d\boldsymbol{x}_0^2)$. To solve for them over time, one needs to solve three tensor differential equations involving $(p^2n + n^2p + n^3)$ scalar integrations, which can get computationally expensive very quickly if the number of parameters and states are large. The adjoint method once again allows us to bypass those integrations and evaluate the Hessian in an alternative manner, thus improving the computational efficiency.

Once again, only the development of $(d^2J/d\boldsymbol{p}^2)$ is presented since the other parts of the Hessian $((d^2J/d\boldsymbol{x}_0^2)$ and $(d^2J/d\boldsymbol{x}_0 d\boldsymbol{p}))$ can be derived in the same way. Similar to the derivation of the gradient, first an augmented cost function (L_2) is written (in the form of a Lagrangian) as

$$\boldsymbol{L}_2 = \frac{dL_1}{d\boldsymbol{p}} + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\eta(\dot{\boldsymbol{x}} - \boldsymbol{f})dt + \int_{t_i^+}^{t_{(i+1)}^-}\gamma\left(-\dot{\boldsymbol{\lambda}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\lambda}\right)dt\right) \tag{43}$$

where $\eta \in \mathbb{R}^{(p\times n)}$ and $\gamma \in \mathbb{R}^{(p\times n)}$ are Lagrangian multipliers.

With a similar argument as the one used for L_1, when the state and the co-state equations are satisfied, the integral terms in Eq. (43) become 0 making $\boldsymbol{L}_2 = (dL_1/d\boldsymbol{p}) = (dJ/d\boldsymbol{p})$. In that case, $(dL_2/d\boldsymbol{p}) = (d^2J/d\boldsymbol{p}^2)$ also holds true. Since, the states $\boldsymbol{x}$ are always determined by solving the state equation, and the co-states $\boldsymbol{\lambda}$ are also always determined using the co-state dynamic equation: the Hessian of the cost function $(d^2J/d\boldsymbol{p}^2)$ is always equal to the first derivative of the augmented cost function $(dL_2/d\boldsymbol{p})$. Using this property, the derivative of the augmented cost L_2 is ultimately evaluated.

Now, substituting $(dL_1/d\boldsymbol{p})$ from Eq. (38), we get

$$\begin{aligned}\boldsymbol{L}_2 = {} & \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}}\right)_i + \sum_{i=1}^{N-1}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_i\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_i + \boldsymbol{\lambda}_{i^-} - \boldsymbol{\lambda}_{i^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_0\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_0 - \boldsymbol{\lambda}_{0^+}\right] + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_N\left[\left(\frac{\partial g}{\partial \boldsymbol{x}}\right)_N + \boldsymbol{\lambda}_{N^-}\right] \\ & + \sum_{i=0}^{N-1}\left(\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\boldsymbol{\lambda}\right)dt + \int_{t_i^+}^{t_{(i+1)}^-}\eta(\dot{\boldsymbol{x}} - \boldsymbol{f})dt + \int_{t_i^+}^{t_{(i+1)}^-}\gamma\left(-\dot{\boldsymbol{\lambda}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\lambda}\right)dt\right)\end{aligned} \tag{44}$$

To evaluate $(dL_2/d\boldsymbol{p})$, Eq. (44) is differentiated with respect to $\boldsymbol{p}$ to yield

$$\frac{dL_2}{dp}=\sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial p^2}+\frac{\partial^2 g}{\partial p\partial x}\frac{dx^{\mathrm{T}}}{dp}\right)_i+\sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial f}{\partial p}\frac{d\lambda^T}{dp}-\frac{\partial^2 f}{\partial p\partial x}^{\rightarrow\lambda}\frac{dx^{\mathrm{T}}}{dp}-\frac{\partial^2 f}{\partial p^2}^{\rightarrow\lambda}\right)dt\right.$$
$$+\int_{t_i^+}^{t_{(i+1)}^-}\left(\eta\frac{d\dot{x}^{\mathrm{T}}}{dp}-\eta\frac{\partial f^{\mathrm{T}}}{\partial p}-\eta\frac{\partial f^{\mathrm{T}}}{\partial x}\frac{dx^{\mathrm{T}}}{dp}\right)dt+\int_{t_i^+}^{t_{(i+1)}^-}\left(-\gamma\frac{d\dot{\lambda}^T}{dp}-\gamma\frac{\partial^2 f}{\partial x^2}^{\rightarrow\lambda}\frac{dx^{\mathrm{T}}}{dp}-\gamma\frac{\partial^2 f}{\partial x\partial p}^{\rightarrow\lambda}-\gamma\frac{\partial f}{\partial x}\frac{d\lambda^T}{dp}\right)dt\Bigg]+\sum_{i=1}^{N-1}\left[\left(\frac{d^2x}{dp^2}\right)_i^{\rightarrow\left[\left(\frac{\partial g}{\partial x}\right)_i+\lambda_{i^-}-\lambda_{i^+}\right]}\right.$$
$$\left.+\left(\frac{dx}{dp}\frac{\partial^2 g}{\partial x\partial p}+\frac{dx}{dp}\frac{\partial^2 g}{\partial x^2}\frac{dx^T}{dp}\right)_i+\left(\frac{dx}{dp}\right)_i\left(\frac{d\lambda^T}{dp_{i^-}}-\frac{d\lambda^T}{dp_{i^+}}\right)\right]+\left(\frac{d^2x}{dp^2}\right)_0^{\rightarrow\left[\left(\frac{\partial g}{\partial x}\right)_0-\lambda_{0^+}\right]}$$
$$+\left(\frac{dx}{dp}\right)_0\left(\frac{\partial^2 g}{\partial x\partial p}+\frac{\partial^2 g}{\partial x^2}\frac{dx^{\mathrm{T}}}{dp}\right)_0-\left(\frac{dx}{dp}\right)_0\frac{d\lambda}{dp}_{0^+}+\left(\frac{d^2x}{dp^2}\right)_N^{\rightarrow\left[\left(\frac{\partial g}{\partial x}\right)_N+\lambda_{N^-}\right]}+\left(\frac{dx}{dp}\right)_N\left(\frac{\partial^2 g}{\partial x\partial p}+\frac{\partial^2 g}{\partial x^2}\frac{dx^{\mathrm{T}}}{dp}\right)_N+\left(\frac{dx}{dp}\right)_N\frac{d\lambda}{dp}_{N^-} \tag{45}$$

Equation (45) can be further simplified by recognizing that some of the terms in the equation are inherently zero. For example, since the initial value of the states does not depend on the value of the parameters, we have $(d^2x/dp^2)_0=0$ and $(dx/dp)_0=0$. Moreover, since the terms multiplying the boundary conditions for λ are satisfied when evaluating λ, we can also eliminate them. Therefore, Eq. (45) simplifies to

$$\frac{dL_2}{dp}=\sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial p^2}\right)_i+\sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial f}{\partial p}\frac{d\lambda^T}{dp}-\frac{\partial^2 f}{\partial p^2}^{\rightarrow\lambda}-\frac{\partial^2 f}{\partial p\partial x}^{\rightarrow\lambda}\frac{dx^{\mathrm{T}}}{dp}\right)dt+\int_{t_i^+}^{t_{(i+1)}^-}\left(\eta\frac{d\dot{x}^{\mathrm{T}}}{dp}-\eta\frac{\partial f^{\mathrm{T}}}{\partial p}-\eta\frac{\partial f^{\mathrm{T}}}{\partial x}\frac{dx^{\mathrm{T}}}{dp}\right)dt\right.$$
$$\left.+\int_{t_i^+}^{t_{(i+1)}^-}\left(-\gamma\frac{d\dot{\lambda}^T}{dp}-\gamma\frac{\partial^2 f}{\partial x^2}^{\rightarrow\lambda}\frac{dx^{\mathrm{T}}}{dp}-\gamma\frac{\partial^2 f}{\partial x\partial p}^{\rightarrow\lambda}-\gamma\frac{\partial f}{\partial x}\frac{d\lambda^T}{dp}\right)dt\right]$$
$$+\sum_{i=1}^{N}\left[\left(\frac{dx}{dp}\frac{\partial^2 g}{\partial x\partial p}+\frac{dx}{dp}\frac{\partial^2 g}{\partial x^2}\frac{dx^{\mathrm{T}}}{dp}+\frac{\partial^2 g}{\partial p\partial x}\frac{dx^{\mathrm{T}}}{dp}\right)_i+\left(\frac{dx}{dp}\right)_i\frac{d\lambda^T}{dp}_{i^-}\right]-\sum_{i=1}^{N-1}\left(\frac{dx}{dp}\right)_i\frac{d\lambda^T}{dp}_{i^+} \tag{46}$$

Using the properties of integration by parts, we can use

$$\int_{t_i^+}^{t_{(i+1)}^-}\gamma\frac{d\dot{\lambda}^T}{dp}dt=\left[\gamma\frac{d\lambda^T}{dp}\right]_{t_i^+}^{t_{(i+1)}^-}-\int_{t_i^+}^{t_{(i+1)}^-}\dot{\gamma}\frac{d\lambda^T}{dp}dt \tag{47}$$

and

$$\int_{t_i^+}^{t_{(i+1)}^-}\eta\frac{d\dot{x}^{\mathrm{T}}}{dp}dt=\left[\eta\frac{dx^{\mathrm{T}}}{dp}\right]_{t_i^+}^{t_{(i+1)}^-}-\int_{t_i^+}^{t_{(i+1)}^-}\dot{\eta}\frac{dx^{\mathrm{T}}}{dp}dt \tag{48}$$

to rewrite Eq. (46) as

$$\frac{dL_2}{dp}=\sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 f}{\partial p^2}^{\rightarrow\lambda}-\gamma\frac{\partial^2 f}{\partial x\partial p}^{\rightarrow\lambda}-\eta\frac{\partial f^{\mathrm{T}}}{\partial p}\right)dt+\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 f}{\partial p\partial x}^{\rightarrow\lambda}-\gamma\frac{\partial^2 f}{\partial x^2}^{\rightarrow\lambda}-\eta\frac{\partial f^T}{\partial x}-\dot{\eta}\right)\frac{dx^T}{dp}dt\right.$$
$$\left.+\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial f}{\partial p}-\gamma\frac{\partial f}{\partial x}+\dot{\gamma}\right)\frac{d\lambda^T}{dp}dt\right]+\sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial p^2}\right)_i+\sum_{i=1}^{N}\left[\left(\frac{dx}{dp}\frac{\partial^2 g}{\partial x\partial p}+\frac{dx}{dp}\frac{\partial^2 g}{\partial x^2}\frac{dx^T}{dp}+\frac{\partial^2 g}{\partial p\partial x}\frac{dx^T}{dp}\right)_i\right.$$
$$\left.+\left(\frac{dx}{dp}\right)_i\frac{d\lambda^T}{dp}_{i^-}-\gamma_i\frac{d\lambda^T}{dp}_{i^-}\right]+\sum_{i=1}^{N-1}\left[\gamma_i\frac{d\lambda^T}{dp}_{i^+}-\left(\frac{dx}{dp}\right)_i\frac{d\lambda^T}{dp}_{i^+}\right]+\gamma_0\frac{d\lambda^T}{dp}_{0^+}\sum_{i=1}^{N-1}(\eta_{i^-}-\eta_{i^+})\left(\frac{dx^T}{dp}\right)_i+\eta\frac{dx^T}{dp}_{N^-}-\eta\frac{dx^T}{dp}_{0^+} \tag{49}$$

Equation (49) is the expression to calculate the Hessian. However, this expression is still dependent on some terms which are unknown to us (for example, $(d\lambda/dp)$, (dx/dp)). To solve this problem, all terms associated with them need to be eliminated (shown in gray in Eq. (49)). An approach similar to the adjoint equation during the gradient calculation is exercised.

The third integral term of Eq. (49) is eliminated by solving

$$-\frac{\partial f}{\partial p}-\gamma\frac{\partial f}{\partial x}+\dot{\gamma}=0,\ \text{with}\ \gamma(0)=0 \tag{50}$$

The second integral term of Eq. (49) is eliminated by solving the differential equation in η

$$-\frac{\partial^2 f}{\partial p\partial x}^{\rightarrow\lambda}-\eta\frac{\partial f^{\mathrm{T}}}{\partial x}-\dot{\eta}-\gamma\frac{\partial^2 f}{\partial x^2}^{\rightarrow\lambda}=0 \tag{51}$$

with a terminal boundary condition $\eta(T) = 0$ and other interval boundary conditions derived from $\eta_{i^-} - \eta_{i^+} = 0$. It should be noted that on doing so, the term $\eta(d\boldsymbol{x}/d\boldsymbol{p})_{N^-}^{\mathrm{T}}$ also equates to 0. The method to solve for η is similar to that of $\boldsymbol{\lambda}$ (i.e., it requires separate integrations for each interval with boundary conditions calculated for each of those intervals separately). Furthermore, recognizing that $(d\boldsymbol{x}/d\boldsymbol{p})_0 = 0$, Eq. (49) simplifies to

$$\frac{dL_2}{d\boldsymbol{p}} = \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{p}^2} - \gamma\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{x}\partial \boldsymbol{p}} - \eta\frac{\partial \boldsymbol{f}^{\mathrm{T}}}{\partial \boldsymbol{p}}\right)dt\right] + \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2}\right)_i + \sum_{i=1}^{N}\left[\left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)_i + \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_i \frac{d\boldsymbol{\lambda}^T}{d\boldsymbol{p}}_{i^-} - \gamma_i \frac{d\boldsymbol{\lambda}^T}{d\boldsymbol{p}}_{i^-}\right] + \sum_{i=1}^{N-1}\left[\gamma_i \frac{d\boldsymbol{\lambda}^T}{d\boldsymbol{p}}_{i^+} - \left(\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right)_i \frac{d\boldsymbol{\lambda}^T}{d\boldsymbol{p}}_{i^+}\right] \tag{52}$$

It appears that the intermittent values of $(d\boldsymbol{x}/d\boldsymbol{p})$ are needed to evaluate Eq. (52). However, on observing the problem carefully, one can see that the γ equation (Eq. (50)) is in fact essentially the same as the $(d\boldsymbol{x}/d\boldsymbol{p})$ equation (in Eq. (19)): justifying the selection of the initial condition $\gamma(0) = 0$ in Eq. (50). Therefore, it turns out that when calculating the Hessian of the cost function, one has to evaluate the sensitivity of the states to the parameters over time, whether directly as in DD or indirectly as in the adjoint method to determine γ. Since $(d\boldsymbol{x}/d\boldsymbol{p}) = \gamma$, it can be substituted in Eq. (52).

Hence, the final value of the Hessian is given by

$$\frac{d^2J}{d\boldsymbol{p}^2} = \frac{dL_2}{d\boldsymbol{p}} = \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{p}^2} - \gamma\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{x}\partial \boldsymbol{p}} - \eta\frac{\partial \boldsymbol{f}^{\mathrm{T}}}{\partial \boldsymbol{p}}\right)dt\right] + \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2}\right)_i + \sum_{i=1}^{N}\left[\left(\gamma\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \gamma\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\gamma^{\mathrm{T}} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\gamma^{\mathrm{T}}\right)_i\right. \tag{53}$$

Similarly, it can be shown that

$$\frac{d^2J}{d\boldsymbol{x}_0^2} = \sum_{i=0}^{N}\left(\gamma_2\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\gamma_2^{\mathrm{T}}\right)_i - (\eta_2)_{0^+} \tag{54}$$

where $\gamma_2 \in \mathbb{R}^{(n\times n)}$, $\eta_2 \in \mathbb{R}^{(n\times n)}$

$$-\gamma_2\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \dot{\gamma_2} = 0, \quad \text{with}\ \gamma_2(0) = I \quad \text{and} \tag{55}$$

$$-\eta_2\frac{\partial \boldsymbol{f}^{\mathrm{T}}}{\partial \boldsymbol{x}} - \dot{\eta_2} - \gamma_2\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{x}^2} = 0 \tag{56}$$

with $\eta_2(T) = 0$ and $(\eta_2)_{i^-} - (\eta_2)_{i^+} = 0$.

It can also be shown that

$$\frac{d^2J}{d\boldsymbol{x}_0 d\boldsymbol{p}} = \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\gamma_2\frac{\partial^2 \boldsymbol{f}^{\rightarrow\boldsymbol{\lambda}}}{\partial \boldsymbol{x}\partial \boldsymbol{p}} - \eta_2\frac{\partial \boldsymbol{f}^{\mathrm{T}}}{\partial \boldsymbol{p}}\right)dt\right] + \sum_{i=0}^{N}\left(\gamma_2\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\gamma^{\mathrm{T}} + \gamma_2\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)_i \tag{57}$$

This concludes the presentation of all the results for the gradient and the Hessian calculations using the adjoint method.

6 Hybrid Method

The computational efficiency can be further reduced by using a technique that combines concepts from Secs. 4 and 5. This method is referred to as the hybrid method.

In the hybrid method, the gradient is evaluated in a manner identical to that of DD. However, the concept of an augmented cost function is used while evaluating the Hessian. This reduces the number of additional states introduced in the derivation when compared to the adjoint method. Moreover, using the augmented cost function approach also allows one to avoid the determination of $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$ over time.

The derivation of the gradient is identical to Sec. 4.1 and therefore is omitted. Only the detailed derivation of the Hessian is presented.

6.1 Hessian. Similar to Sec. 5.2, the derivation of $(d^2J/d\boldsymbol{p}^2)$ is shown in detail while only the final results for $(d^2J/d\boldsymbol{x}_0^2)$ and $(d^2J/d\boldsymbol{x}_0 d\boldsymbol{p})$ are mentioned since the derivation is almost identical.

The derivation starts with the augmented cost function (Eq. (34))

$$\boldsymbol{L}_2 = \sum_{i=0}^{N}\left(\frac{\partial g}{\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial g}{\partial \boldsymbol{x}}\right)_i + \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} - \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)\boldsymbol{\gamma}dt\right] \tag{58}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^n$ are introduced variables (analogous to Lagrangian multipliers). When Eq. (19) is satisfied, the integrand in Eq. (58) becomes 0 making $\boldsymbol{L}_2 = (d\boldsymbol{J}/d\boldsymbol{p})$. Under these conditions, the derivative of the augmented cost becomes equal to the Hessian of the cost function, i.e., $(d\boldsymbol{L}_2/d\boldsymbol{p}) = (d^2J/d\boldsymbol{p}^2)$. This relation always holds true as long as the states and their sensitivities to parameters are calculated using Eqs. (1) and (19), respectively. Therefore, Eq. (58) is differentiated with respect to $\boldsymbol{p}$ to get

$$\frac{dL_2}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\frac{d\boldsymbol{x}^T}{d\boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)_i + \sum_{i=0}^{N}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\frac{\partial g}{\partial \boldsymbol{x}}}\right)_i$$
$$+\sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(\frac{d^2\dot{\boldsymbol{x}}}{d\boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}} - \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}} - \left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}\partial \boldsymbol{x}} \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} - \left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} - \left(\left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^T}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} - \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\left(\frac{\partial f}{\partial x}\boldsymbol{\gamma}\right)}\right)dt\right] \tag{59}$$

However, we know from the properties of integration by parts that

$$\int_{t_i^+}^{t_{(i+1)}^-}\frac{d^2\dot{\boldsymbol{x}}}{d\boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}}dt = \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}}\Big|_{t_{(i+1)}^-} - \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}}\Big|_{t_i^+} - \int_{t_i^+}^{t_{(i+1)}^-}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\dot{\boldsymbol{\gamma}}}\right)dt \tag{60}$$

On substituting Eq. (60) in Eq. (59) and collecting appropriate terms, we get

$$\frac{dL_2}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)_i + \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}} - \left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}\partial \boldsymbol{x}} \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} - \left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}}\right.\right.$$
$$\left.\left. - \left(\left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}^{\rightarrow\left(-\dot{\boldsymbol{\gamma}}-\frac{\partial f}{\partial x}\boldsymbol{\gamma}\right)}\right)dt\right] + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}_0^{\rightarrow\left(\frac{\partial g}{\partial \boldsymbol{x}_0}-\boldsymbol{\gamma}_0^+\right)} + \sum_{i=1}^{N-1}\left(\frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}\right)_i^{\rightarrow\left(\frac{\partial g}{\partial \boldsymbol{x}_i}+\boldsymbol{\gamma}_i^- -\boldsymbol{\gamma}_i^+\right)} + \frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}_N^{\rightarrow\left(\frac{\partial g}{\partial \boldsymbol{x}_N}+\boldsymbol{\gamma}_N^-\right)} \tag{61}$$

Equation (61) is now the expression to calculate $(d^2J/d\boldsymbol{p}^2)$. However, we can see that it is still dependent on $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$ which we want to avoid evaluating. Hence (similar to the Adjoint method), all coefficients associated with that term are forced to 0 (shown by the shaded regions of Eq. (61)). This leads to solving the matrix differential equation

$$-\dot{\boldsymbol{\gamma}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\gamma} = 0 \tag{62}$$

in every time interval between two observations (i.e., $[t_i^+, t_{i+1}^-]$). The boundary conditions for solving Eq. (62) in each of the intervals can be derived from the equations

$$\frac{\partial g}{\partial \boldsymbol{x}_i} + \boldsymbol{\gamma}_i^- - \boldsymbol{\gamma}_i^+ = 0 \quad \text{and} \quad \frac{\partial g}{\partial \boldsymbol{x}_N} + \boldsymbol{\gamma}_N^- = 0 \tag{63}$$

It should also be noted that $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)_0 = 0$. The final expression for $(d^2J/d\boldsymbol{p}^2)$ can thus be written as

$$\frac{d^2J}{d\boldsymbol{p}^2} = \frac{dL_2}{d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{\partial^2 g}{\partial \boldsymbol{p}^2} + \frac{\partial^2 g}{\partial \boldsymbol{p}\partial \boldsymbol{x}}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{p}}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)_i + \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(-\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}^2}^{\rightarrow\boldsymbol{\gamma}} - \left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{p}\partial \boldsymbol{x}} \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}}\right.\right.$$
$$\left.\left. - \left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}} - \left(\left(\frac{dx}{dp}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right)^{\rightarrow\boldsymbol{\gamma}}\right)dt\right] \tag{64}$$

Similarly, it can be shown that

$$\frac{d^2J}{d\boldsymbol{x}_0^2} = \sum_{i=0}^{N}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{x}_0}\right)_i - \sum_{i=0}^{N-1}\int_{t_i^+}^{t_{(i+1)}^-}\left(\left(\frac{dx}{dx_0}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{x}_0}\right)^{\rightarrow\boldsymbol{\gamma}}dt \tag{65}$$

where γ and $(d\boldsymbol{x}/d\boldsymbol{x}_0)$ are calculated using Eqs. (19) and (62), respectively. It can also be shown that

$$\frac{d^2J}{d\boldsymbol{x}_0 d\boldsymbol{p}} = \sum_{i=0}^{N}\left(\frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}^2}\frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}} + \frac{d\boldsymbol{x}}{d\boldsymbol{x}_0}\frac{\partial^2 g}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)_i - \sum_{i=0}^{N-1}\left[\int_{t_i^+}^{t_{(i+1)}^-}\left(\left(\left(\frac{dx}{dx_0}\rightarrow \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\right) \rightarrow \frac{d\boldsymbol{x}^{\mathrm{T}}}{d\boldsymbol{p}}\right) + \left(\frac{dx}{dx_0}\rightarrow \frac{\partial^2 f}{\partial \boldsymbol{x}\partial \boldsymbol{p}}\right)\right)^{\rightarrow\boldsymbol{\gamma}}dt\right] \tag{66}$$

This concludes the presentation of all the results for the gradient and the Hessian calculations using the hybrid method. The next section presents a brief summary of the computational efficiencies of each method.

7 Computational Efficiency

It is already established that function evaluation (which involves state integrations for dynamic systems) based gradients and Hessians incur more computational expense than evaluating cheap adjoint-based gradients and Hessians [2,17] suggesting that FD should be more expensive than the adjoint and hybrid algorithms.

After profiling each method for the three numerical example problems discussed in Sec. 8, it can be observed that the majority of the program time is spent inside the integrator function. The exact percentage(s) of total program run time (one iteration) have been listed in Table 1. Note that the interpolation is done as part of the integration and as illustrated in Table 1 makes up a large fraction of the

Table 1 Profiling results showing the percentage(s) of total program run time (first iteration)

Problem	Function	DD	Adjoint	Hybrid
1 Lorenz 63	Integrator	99.95	99.98	99.98
	Interpolator	—	80.02	77.94
2 (Supernova 1999dq)	Integrator	99.45	99.85	72.32
	Interpolator	—	79.78	50.27
3 Two-tank model	Integrator	99.07	99.95	99.93
	Interpolator	—	62.26	57.28

Note: The integration subsumes the interpolation time.

Table 2 Comparison of computational expense

Algorithm	Gradient	Gradient + Hessian
$N_s^{(\mathrm{FD})}$	$2np + 2n^2$	$n^3 + 2n^2p + p^2n + n^2 + pn + n$
$N_s^{(\mathrm{DD})}$	$n + pn + n^2$	$n^2 + n + pn + p^2n + n^3 + n^2p$
$N_s^{(\mathrm{Adjoint})}$	$2n + p$	$p + p^2 + 2n + 2n^2 + 3pn$
$N_s^{(\mathrm{Hybrid})}$	$n + pn + n^2$	$p^2 + 2n + 2n^2 + 2pn$

Note: n: # of states and p: # of parameters.

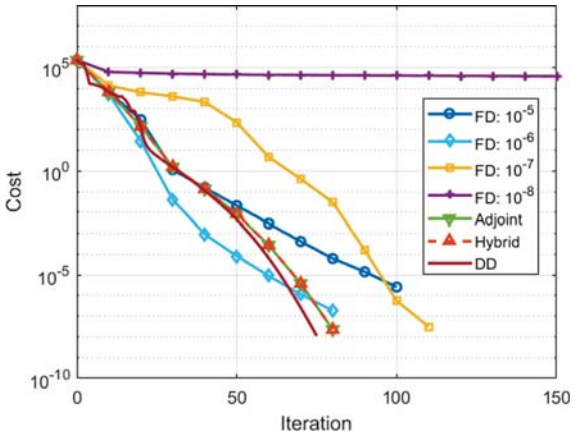

Fig. 3 Convergence of different algorithms

integration time. Therefore, it makes sense to compare objectively the number of scalar integrations being executed in each algorithm as a measure of computational efficiency.

Comparisons have been made for when only the gradient is calculated and when both the gradient and Hessian are calculated. More details can be found in the Appendix.

Table 2 compares the computational cost of the FD, DD, adjoint, and hybrid approaches clearly illustrating the benefit of the adjoint and hybrid algorithms (For details regarding the calculation of N_s, refer to the Appendix). Column three from Table 2 suggests that the growth of N_s is given by the third-order polynomials for FD as well as DD, while it is only a second-order polynomial for the adjoint and the hybrid. It should also be noted that the adjoint method is preferable over the hybrid method when calculating only gradients, while the hybrid method is less expensive when calculating the gradient as well as the Hessian. This is because the additional co-states in the adjoint method (η and η_2) are matrices, while the co-state in the hybrid method (γ) is only a vector.

The DD, adjoint, and hybrid values of N_s that have been quoted in the third column of Table 2 do not consider the symmetry of the Hessian matrix. Hence, the counts for N_s provided are conservative.

It is also interesting to note that a significant amount of time is also spent in the interpolator function (refer Table 1) and is responsible for lengthening the program run time significantly. The interpolator function is used to interpolate several values of states and costates when solving numerous two point boundary value problems that show up in the adjoint and the hybrid method. In this work, efforts have not been made to improve on interpolations of independent variables, however, there is literature that looks into the issue ([18] and references therein).

8 System Identification Problems

Several system identification problems can be posed with the structure of the cost function being investigated in this work. The adjoint method of gradients and Hessians, therefore, now provides another tool to solve all such problems. The results from three test problems are presented. The first example uses synthetic data to illustrate the algorithm and the following two examples use publicly available data to identify the parameters of the dynamic models.

8.1 Problem 1. The objective is to identify the parameters and initial conditions of the popular Lorenz System (also known as the Lorenz-63 model). The dynamics of the system are given by

$$\dot{x}_1 = -p_1(x_1 - x_2) \tag{67}$$

$$\dot{x}_2 = x_1(p_2 - x_3) - x_2 \tag{68}$$

$$\dot{x}_3 = x_1x_2 - p_3x_3 \tag{69}$$

This particular system is chosen since it is known to be chaotic and has been used as a system identification example to illustrate effectiveness of algorithms in the literature ([19,20] and

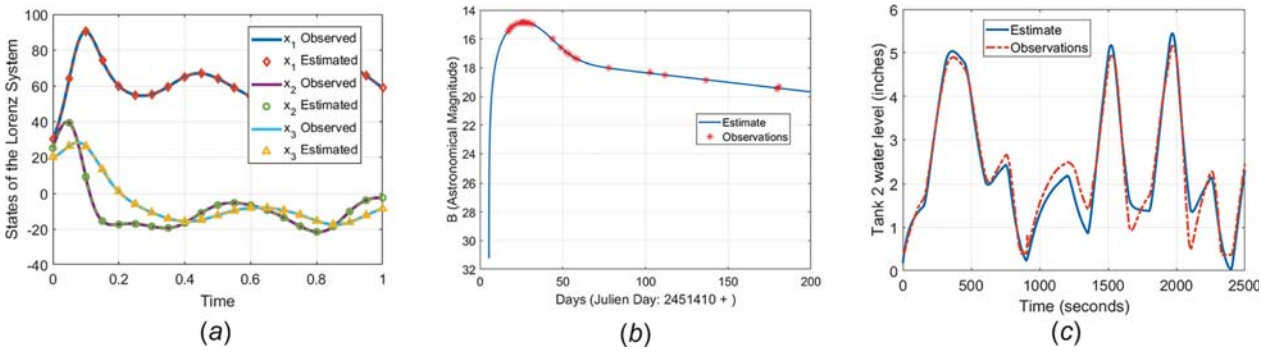

Fig. 4 Illustration of system dynamics for parameters estimated using the hybrid method for the example problems: (*a*) Lorenz-63 model, (*b*) Supernova 1999dq luminosity, and (*c*) two-tank model

Table 3 Supernovae dataset

Supernova	1999dq	1998aq	1990n
B_{ref}	14.5	12.0	12.0
N	25	52	34
$\boldsymbol{q}^{\text{iter}=0}$	$10[1,1,1,1,1,1]^{\text{T}}$	$10[1,1,1,1,1,1]^{\text{T}}$	$5[1,1,1,1,1,1]^{\text{T}}$

references therein). It is highly sensitive to initial conditions and parameters making it an ideal choice to illustrate the accuracy and the feasibility of the adjoint and the hybrid method.

It is assumed that all the parameters p_1, p_2, p_3 and the initial conditions for x_2 and x_3 are unknown. For system identification, it is also assumed that a time series of observations (at intervals of $\Delta t = 0.01$ up to $t = 1$) is available for x_1, x_2, and x_3 similar to Ref. [20]. The observations are grouped as $[y_1(t_i), y_2(t_i), y_3(t_i)]^T = [x_1(t_i), x_2(t_i), x_3(t_i)]^T$. The time series is generated from the following values: $p_1 = 10, p_2 = 60, p_3 = 8/3, x_1(0) = 20$, $x_2(0) = 25$, and $x_3(0) = 30$ (these values were selected from Ref. [19]). The intention is to estimate $\boldsymbol{q} = [p_1, p_2, p_3, x_2(0), x_3(0)]^T$ correctly.

To identify the optimal $\boldsymbol{q}$, an optimization problem is posed as

$$\begin{aligned} &\underset{\boldsymbol{q}}{\text{minimize}} \quad J \\ &\text{subject to} \quad \text{Equations (67) through (69)} \end{aligned} \tag{70}$$

where the cost function J is defined to be a sum of the squared errors between the observed states and the estimated states

$$J = \sum_{i=0}^{N} \sum_{j=1}^{3} (y_j(t_i) - \hat{x}_j(t_i))^2 \tag{71}$$

The problem is solved iteratively. The initial guesses are chosen to be $\hat{p}_1 = 20, \hat{p}_2 = 75, \hat{p}_3 = 10, \hat{x}_2(0) = 10$, and $\hat{x}_3(0) = 15$ for all the methods.

The gradients and the Hessians of the cost function J with respect to $\boldsymbol{q}$ are evaluated at every iteration. These quantities are then used to employ a gradient-based algorithm in MATLAB to converge to a solution. This is done for each of the methods discussed in the document under identical optimization environments. Figure 3 shows a comparative plot of convergence between different techniques. The only difference while simulating them was the source of gradients and Hessians that were fed to the optimizer.

Four distinct simulations were made for the FD method with each simulation having a distinct step size to calculate the gradients and Hessians. The step sizes have been listed in Fig. 3. Curves corresponding to step sizes 10^{-5} and 10^{-7} show comparatively slow convergences. FD with a step size of 10^{-6} performs very well, while FD with a step size of 10^{-8} fails to converge. These results illustrate the variability in the accuracy of FD. Since it is impossible to know the magnitude of the optimal step size beforehand, a method independent of step size is motivated. The DD and the adjoint method have comparable performance although the DD appears to be the most accurate algorithm in this example having converged to the lowest terminal cost with the fewest iterations. However, it must be noted that the adjoint and the hybrid approach are far less expensive as compared to FD as well as DD.

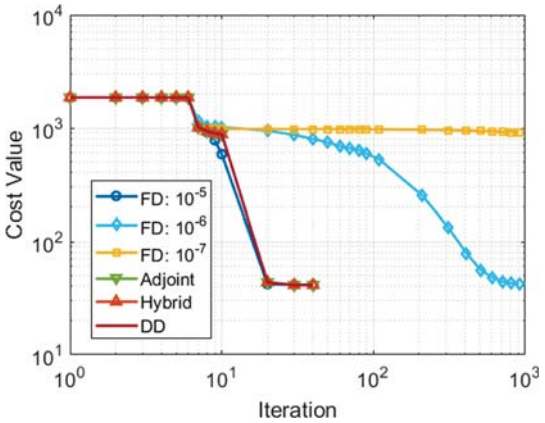

Fig. 6 Convergence of different algorithms

Time response of the states (determined from parameters identified via the hybrid method) along with observations has been shown in Fig. 4(*a*) for reference.

8.2 Problem 2. The second example presented is that of parameter identification of dynamic type 1a supernovae models using real observed luminosity data. This particular example is chosen since it has already been shown to be a difficult data fitting problem [21]. The details regarding the physics of the phenomenon and the model can be found in Ref. [21]. However, before presenting results from three different supernovae, a brief description of the problem statement is provided.

The governing system dynamics are given by the equations

$$\dot{x}_1 = W(t, p_1, p_2, p_3) - x_1/8.764p_4 \tag{72}$$

$$\dot{x}_2 = x_1/8.764p_4 - x_2/111.42p_4 \tag{73}$$

$$\dot{x}_3 = x_2/111.42p_4 \tag{74}$$

where $W(t, p_1, p_2, p_3)$ is the Weibull distribution and is given by the following equation:

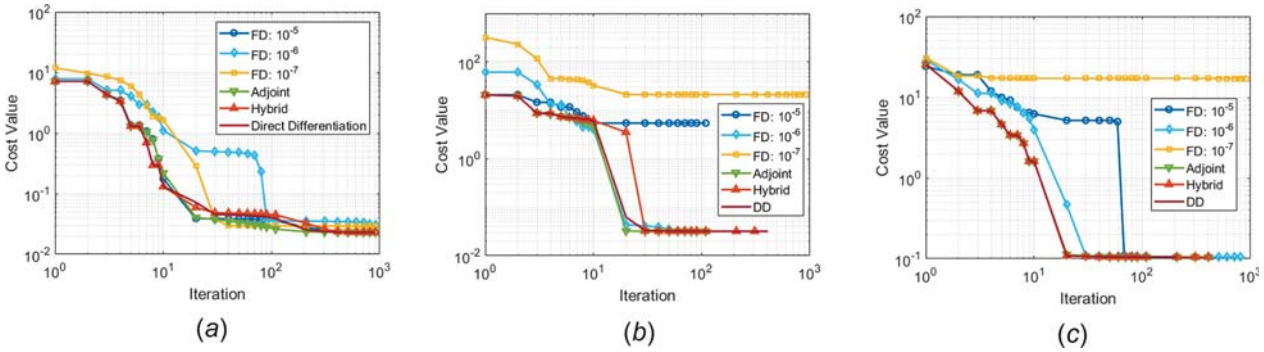

Fig. 5 Convergence for different algorithms in supernova system identification: (*a*) Supernova 1999dq, (*b*) supernova 1998aq, and (*c*) supernova 1990n

Table 4 Supernovae system ID results

Supernova	Parameters	FD:1×10^{-5}	FD:1×10^{-6}	FD:1×10^{-7}	DD	Adjoint	Hybrid
1999dq	p_1	5.387346	9.492225	9.455105	5.381856	5.345180	5.403384
	p_2	2.375212	1.906856	1.911192	2.375813	2.379884	2.373393
	p_3	19.396321	15.199001	15.241670	19.401609	19.438616	19.379808
	p_4	0.726794	0.728809	0.728346	0.726823	0.726809	0.726818
	p_5	16.754936	16.152228	16.158109	16.755841	16.760413	16.752827
	p_6	5.593438	5.544746	5.545865	5.593443	5.593827	5.593384
	Cost	0.023091	0.029096	0.028948	0.023091	0.023090	0.023091
	R^2	0.998692	0.998344	0.998352	0.998692	0.998692	0.998692
1998aq	p_1	—	14.703285	—	14.703516	14.703373	14.703456
	p_2	—	2.094165	—	2.094131	2.094152	2.094139
	p_3	—	15.325454	—	15.325210	15.325349	15.325273
	p_4	—	0.655404	—	0.655405	0.655405	0.655405
	p_5	—	14.743421	—	14.743401	14.743408	14.743411
	p_6	—	4.367701	—	4.367689	4.367697	4.367690
	Cost	—	0.031963	—	0.031963	0.031963	0.031963
	R^2	—	0.999144	—	0.999144	0.999144	0.999144
1990n	p_1	13.574933	13.539436	—	13.559120	13.559156	13.559125
	p_2	2.337681	2.342851	—	2.339985	2.339980	2.339984
	p_3	17.639979	17.676908	—	17.656383	17.656346	17.656378
	p_4	0.690074	0.690061	—	0.690075	0.690075	0.690075
	p_5	10.746664	10.747313	—	10.746995	10.746994	10.746995
	p_6	4.052780	4.053178	—	4.052910	4.052910	4.052911
	Cost	0.104936	0.104935	—	0.104935	0.104935	0.104935
	R^2	0.995074	0.995073	—	0.995074	0.995074	0.995074

$$W(t,p_1,p_2,p_3) = \begin{cases} \dfrac{p_2}{p_3}\left(\dfrac{t-p_1}{p_3}\right)^{p_2-1} e^{-\left(\frac{t-p_1}{p_3}\right)^{p_2}} & t \geq p_1 \\ 0 & t < p_1 \end{cases} \tag{75}$$

The initial conditions for all the states are known and are equal to zero, i.e., $x_1(0) = 0, x_2(0) = 0$, and $x_3(0) = 0$. The output model is a linear function of the states and is given by

$$\hat{y}(t) = p_5 \frac{x_1}{8.764 p_4} + p_6 \frac{x_2}{111.42 p_4} \tag{76}$$

where y represents the total observed luminosity. The observations are, however, made in terms of astronomical magnitudes ($B(t_i)$), where t_i represents the time of the ith observation. To pose a weighted least squares cost function, the observations are first transformed to the *total observed luminosity* space using the relation

$$y(t_i) = 10^{-0.4(B(t_i)-B_{\text{ref}})} \tag{77}$$

where B_{ref} is a known reference magnitude constant specific to each supernovae (refer Table 3). The final optimization problem can be posed as

$$\begin{aligned} &\text{minimize}_{\boldsymbol{q}} \quad J = \sum_{i=1}^{N} (w_i((y(t_i) - \hat{y}(t_i)))^2 \\ &\text{subject to} \quad \text{Equations (72) through (74)} \end{aligned} \tag{78}$$

where $\boldsymbol{q} = [p_1, p_2, p_3, p_4, p_5, p_6]^{\mathrm{T}}$, and w_i are weights associated with each observation and are given by $w_i = 1/y(t_i)$. N represents the number of observations available for each of the supernovae (shown in Table 3).

Three distinct simulations were made for the FD method for each of the supernovae corresponding to distinct step sizes. The convergence of all the algorithms for the three supernovae have been shown in Figs. 5(*a*)–5(*c*). It is once again evident that the adjoint and the hybrid methods are consistently as effective as the DD although they are computationally more efficient. In the FD method, as can be seen from the figures, it is impossible to determine beforehand the step size magnitude which would give the best performance. For example, in supernova 1999dq, all the FD step sizes work reasonably well with 1×10^{-5} being most effective. In supernova 1998aq, FD with the step size of 1×10^{-6} is the only one which converges. In supernova 1990n, step sizes 1×10^{-6} and 1×10^{-7} converge with 1×10^{-6} converging faster. This variability again makes a strong case for the proposed algorithms especially because the Adjoint as well as the Hybrid is cheaper than FD. In fact, for the Supernova 1990n, all finite difference algorithms are outperformed by the DD, Adjoint, and the Hybrid in efficiency.

The gradients and the Hessians are evaluated at every iteration and are then fed to the MATLAB optimizer under identical optimization environments. The initial conditions used were identical for all the algorithms and have been listed in Table 3 (row 3).

The identified parameter values are listed in Table 4. Algorithms which did not converge have been replaced by "–." The final values of the cost, as well as R^2, the coefficient of determination, for each algorithm have also been listed. R^2 was calculated

Table 5 Two-tank system ID results

Parameters	FD:1×10^{-5}	FD:1×10^{-6}	FD:1×10^{-7}	DD	Adjoint	Hybrid
p_1	0.038652	0.042732	—	0.041798	0.041796	0.041799
p_2	0.020728	0.027374	—	0.023773	0.023772	0.023774
p_3	0.021493	0.022253	—	0.021545	0.021546	0.021544
p_4	0.062638	0.054184	—	0.059154	0.059155	0.059152
$x_1(0)$	0.256533	0.253653	—	0.420376	0.419485	0.420434
$x_2(0)$	0.252388	0.252185	—	0.178079	0.178487	0.178367

in a manner identical to [21] and is seen to be comparable (for the algorithms that converged) to the values reported there.

The estimated time response of $B(t)$ of the supernova 1999dq (determined from parameters estimated via the hybrid method) along with the observations is shown in Fig. 4(*b*).

8.3 Problem 3. The final illustrative example chosen is that of the extremely popular benchmark nonlinear system identification two-tank problem that has been used for testing system identification algorithms extensively in the literature [22]. The system dynamics are given by the equations

$$\dot{x_1} = -p_1\sqrt{x_1} + p_2 u \tag{79}$$

$$\dot{x_2} = -p_3\sqrt{x_2} + p_4\sqrt{x_1} \tag{80}$$

where x_1 and x_2 represent the water level in tanks 1 and 2, respectively. The objective is to identify the parameters as well as the initial conditions using data collected from a real experiment. The data are publicly available and can be found from Ref. [22]. For identification, it is assumed that only the second state is available as measurement, i.e., $y(t_i) = x_2(t_i)$.

To ensure that the value of the states always remains greater than 0, the transformation $x_1 = z_1^2$ and $x_2 = z_2^2$ is used to rewrite the model equations as

$$\dot{z_1} = -p_1/2 + p_2 u/2z_1 \tag{81}$$

$$\dot{z_2} = -p_3/2 + p_4 z_1/2z_2 \tag{82}$$

The final identification problem is solved in the $[z_1, z_2]^{\mathrm{T}}$ space. Similar to the previous sections, the final optimization problem can be posed as

$$\begin{aligned} &\text{minimize}_{\boldsymbol{q}} \quad J = \sum_{i=1}^{N} (y(t_i) - z_2(t_i)^2)^2 \\ &\text{subject to} \quad \text{Equations (81) and (82)} \end{aligned} \tag{83}$$

where $\boldsymbol{q} = [p_1, p_2, p_3, p_4, z_1(0), z_2(0)]^{\mathrm{T}}$. In this work, only the first 501 observations were used for identification, i.e., $N = 501$. Consecutive observations differ by a time of 5 s.

The convergence of different algorithms for this problem is shown in Fig. 6. It can be seen that the adjoint, hybrid, and the DD are equally effective, while the adjoint and hybrid approaches are more efficient relative to the DD. The identified parameter values are listed in Table 5.

The estimated time response of $x_2(t)$ for this example (determined from parameters estimated via the hybrid method) along with the observations are shown in Fig. 4(*c*).

9 Conclusion

Stimulated by the variability in finite difference estimates of gradients and Hessians and extensive computational requirements for the direct differentiation method, this paper presents a detailed development of the adjoint and the hybrid approaches for the determination of the exact Hessian for system identification problems where the measurements are available at discrete times.

The adjoint method uses a Lagrange multiplier technique to eliminate the need to evaluate $(d\boldsymbol{x}/d\boldsymbol{p})$ for gradients and $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$ for Hessians of the cost function. On the other hand, the hybrid method evaluates $(d\boldsymbol{x}/d\boldsymbol{p})$ using the DD method while calculating the gradient but uses the adjoint technique to avoid evaluating $(d^2\boldsymbol{x}/d\boldsymbol{p}^2)$ while calculating the Hessian.

The proposed approaches are then illustrated on identifying the Lorenz System, the type 1a Supernovae as well as the benchmark two-tank system. It is seen from the results that the Hessian derived from the adjoint as well as the hybrid method is as accurate as direct differentiation, more consistent and reliable than finite difference and definitely requires fewer integrations than both finite difference and direct differentiation. We realize that tests on a few numerical examples do not provide insight into the relative benefits of the proposed approaches which are agnostic of application, however, they provide a strong motivation in favor of using the adjoint and hybrid algorithms for system identification.

Acknowledgment

This material is based upon work supported through National Science Foundation (NSF) under Award No. CMMI-1537210. All results and opinions expressed in this paper are those of the authors and do not reflect opinions of NSF.

Appendix

Since all implementation was done in MATLAB, in-built functions such as `interp1` and `ode45` were used to execute interpolation and integration operations, respectively. After the code was written to implement all the algorithms, the profiling tool in MATLAB (`profile`) was used to determine the relative time that was spent in each function.

The percentages quoted in Table 1 were calculated as follows: For integrator, we have $(t_{\text{ode45}}/t_{\text{iter=1}}) \times 100$, and for the interpolator, we have $(t_{\text{interp1}}/t_{\text{iter=1}}) \times 100$, where t_{ode45} is the time spent inside the function `ode45` when running the algorithms for 1 iteration, t_{interp1} is the time spent inside the function `interp1` when running the algorithms for 1 iteration, and $t_{\text{iter=1}}$ is the total run time of the algorithms for 1 iteration.

Values determined for different algorithms have been listed in Table 1.

Table 2 summarizes the number of scalar integrations that are necessary to evaluate the gradient as well as the Hessian. A detailed breakdown for the determination of the expressions is presented next.

A.1 Finite Difference. It should be noted that each model evaluation takes n scalar integrations since $\boldsymbol{x} \in \mathbb{R}^n$. To evaluate the gradient using the central-difference approach, each element requires two model evaluations (since $dJ/da = (J(a+\delta a) - J(a-\delta a)/2\delta a)$). Therefore, to evaluate each element of $(dJ/d\boldsymbol{q})$, $2n$ scalar integrations are needed. Since there are $(n+p)$ elements in $(dJ/d\boldsymbol{q})$, a total of $N_s^{(\text{FD})} = 2n^2 + 2np$ scalar integrations are needed to evaluate $(dJ/d\boldsymbol{q})$.

When the Hessian is also desired, the method in which calculations from the gradient can be used to evaluate elements of the Hessian is used to make the computations cheaper.

The basic formulae to determine Hessians for a two variable $((x, y))$ cost function $(f(x, y))$ are given by equations $d^2f/Jdx^2 = (f(x+\delta x, y) - 2f(x, y) + f(x-\delta x, y))/(\delta x^2)$ and

$$\begin{aligned} \frac{d^2f(x,y)}{dxdy} = &(f(x+\delta, y+\delta y) - f(x+\delta x, y) - f(x, y+\delta y) \\ &+2f(x, y) - f(x-\delta x, y) - f(x, y-\delta y) \\ &+f(x-\delta, y-\delta y))/(2\delta x \delta y). \end{aligned}$$

For $(d^2J/d\boldsymbol{q}^2)$, the diagonal elements do not need any additional integrations apart from one evaluation of the states (i.e., n integrations). The nondiagonal elements, however, each need additional two state evaluations (i.e., $2n$ integrations as per the above formula). Since the total number of unique nondiagonal elements in $(d^2J/d\boldsymbol{q}^2)$ is $(n+p)(n+p-1)/2$ (because of symmetry), the number of integrations necessary for the nondiagonal elements is $n^3 + 2n^2p - n^2 + p^2n - pn$. Therefore, the total number of necessary integrations for $(d^2J/d\boldsymbol{q}^2)$ is $(n^3 + 2n^2p - n^2 + p^2n - pn + n)$.

Table 6 DD integration breakdown

Variables	# of integrations	Variables	# of integrations
$\boldsymbol{x}(t)$	n	$\frac{d^2\boldsymbol{x}}{d\boldsymbol{p}^2}$	np^2
$\frac{d\boldsymbol{x}}{d\boldsymbol{q}}$	$n(p+n)$	$\frac{d^2\boldsymbol{x}}{d\boldsymbol{x}_0^2}$	n^3
		$\frac{d^2\boldsymbol{x}}{d\boldsymbol{p}d\boldsymbol{x}_0}$	n^2p

Table 7 Adjoint integration breakdown

Variables	# of integrations	Variables	# of integrations
$\boldsymbol{x}(t)$	n	γ	np
$\boldsymbol{\lambda}$	n	η_2	n^2
$\frac{dJ}{d\boldsymbol{p}}$	p	γ_2	n^2
η	np	$\frac{d^2J}{d\boldsymbol{p}^2}$	p^2
		$\frac{d^2J}{d\boldsymbol{p}d\boldsymbol{x}_0}$	np

Table 8 Hybrid integration breakdown

Variables	# of integrations	Variables	# of integrations
$\boldsymbol{x}(t)$	n	$\frac{d^2J}{d\boldsymbol{p}^2}$	p^2
$\frac{d\boldsymbol{x}}{d\boldsymbol{q}}$	$n(p+n)$	$\frac{d^2J}{d\boldsymbol{x}_0^2}$	n^2
γ	n	$\frac{d^2J}{d\boldsymbol{p}d\boldsymbol{x}_0}$	np

When the number of integrations from the gradient computation is added, we get $N_s^{(\mathrm{FD})} = n^3 + 2n^2p + p^2n + n^2 + pn + n$.

A.2 Direct Differentiation. The number of integrations necessary in this method can be derived by observing the equations that need to be solved. A breakdown in the form of a table has been presented in Table 6. The first and third columns list all the variables that need evaluation. The second and fourth columns list the scalar integrations needed to evaluate the corresponding variables in columns 1 and 3, respectively.

For only gradient evaluations, $N_s^{(\mathrm{DD})}$ is just the sum of the rows in the second column, i.e., $N_s^{(\mathrm{DD})} = n + pn + n^2$. However, when both the gradient and the Hessian are of interest, we get $N_s^{(\mathrm{DD})} = n^2 + n + pn + p^2n + n^3 + n^2p$.

A.3 Adjoint Method. Table 7 presents the integration breakdown for the adjoint method. Hence, we have for gradient: $N_s^{(\mathrm{Adjoint})} = 2n + p$ (sum of first three rows, second column) and for gradient + Hessian: $N_s^{(\mathrm{Adjoint})} = p + p^2 + 2n + 2n^2 + 3pn$.

A.4 Hybrid Method. Table 8 presents the integration breakdown for the hybrid method.

Hence, we have for gradient: $N_s^{(\mathrm{Hybrid})} = n + pn + n^2$ (sum of first two rows, second column) and for gradient + Hessian: $N_s^{(\mathrm{Hybrid})} = p^2 + 2n + 2n^2 + 2pn$.

References

[1] Raffard, R. L., and Tomlin, C. J., 2005, "Second Order Adjoint-Based Optimization of Ordinary and Partial Differential Equations With Application to Air Traffic Flow," American Control Conference (ACC), Portland, OR, June 8–10, pp. 798–803.

[2] Özyurt, D. B., and Barton, P. I., 2005, "Cheap Second Order Directional Derivatives of Stiff Ode Embedded Functionals," SIAM J. Sci. Comput., **26**(5), pp. 1725–1743.

[3] Nocedal, J., and Wright, S., 2006, *Numerical Optimization*, Springer Science & Business Media, New York.

[4] Kelley, C. T., 1999, *Iterative Methods for Optimization*, Vol. 18, SIAM, Philadelphia, PA.

[5] Cao, Y., Li, S., Petzold, L., and Serban, R., 2003, "Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint Dae System and Its Numerical Solution," SIAM J. Sci. Comput., **24**(3), pp. 1076–1089.

[6] Cao, Y., Li, S., and Petzold, L., 2002, "Adjoint Sensitivity Analysis for Differential-Algebraic Equations: Algorithms and Software," J. Comput. Appl. Math., **149**(1), pp. 171–191.

[7] Sengupta, B., Friston, K. J., and Penny, W. D., 2014, "Efficient Gradient Computation for Dynamical Models," NeuroImage, **98**, pp. 521–527.

[8] Raffard, R. L., Amonlirdviman, K., Axelrod, J. D., and Tomlin, C. J., 2006, "Parameter Identification Via the Adjoint Method: Application to Protein Regulatory Networks," IFAC Proc., **39**(2), pp. 475–482.

[9] Raffard, R. L., Amonlirdviman, K., Axelrod, J. D., and Tomlin, C. J., 2008, "An Adjoint-Based Parameter Identification Algorithm Applied to Planar Cell Polarity Signaling," IEEE Trans. Autom. Control, **53** (Special Issue), pp. 109–121.

[10] Suwartadi, E., Krogstad, S., and Foss, B., 2009, "On State Constraints of Adjoint Optimization in Oil Reservoir Water-Flooding," SPE/EAGE Reservoir Characterization & Simulation Conference, Abu Dhabi, United Arab Emirates, Oct. 19–21.

[11] Le Dimet, F.-X., Navon, I. M., and Daescu, D. N., 2002, "Second-Order Information in Data Assimilation," Mon. Weather Rev., **130**(3), pp. 629–648.

[12] Wang, Z., Navon, I. M., Le Dimet, F., and Zou, X., 1992, "The Second Order Adjoint Analysis: Theory and Applications," Meteorol. Atmos. Phys., **50**(1–3), pp. 3–20.

[13] Sandu, A., Daescu, D. N., and Carmichael, G. R., 2003, "Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems With Kpp—Part I: Theory and Software Tools," Atmos. Environ., **37**(36), pp. 5083–5096.

[14] Shichitake, M., and Kawahara, M., 2008, "Optimal Control Applied to Water Flow Using Second Order Adjoint Method," Int. J. Comput. Fluid Dyn., **22**(5), pp. 351–365.

[15] Liu, S., and Bewley, T. R., 2003, "Adjoint-Based System Identification and Feedforward Control Optimization in Automotive Powertrain Subsystems," American Control Conference, (ACC), Denver, CO, June 4–6, pp. 2566–2571.

[16] Nandi, S., and Singh, T., 2017, "Adjoint Based Hessians for Optimization Problems in System Identification," IEEE Conference on Control Technology and Applications (CCTA), Mauna Lani, HI, Aug. 27–30, pp. 626–631.

[17] Griewank, A., and Walther, A., 2008, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, PA.

[18] Alexe, M., and Sandu, A., 2009, "Forward and Adjoint Sensitivity Analysis With Continuous Explicit Runge–Kutta Schemes," Appl. Math. Comput., **208**(2), pp. 328–346.

[19] Lu, F., Xu, D., and Wen, G., 2004, "Estimation of Initial Conditions and Parameters of a Chaotic Evolution Process From a Short Time Series," Chaos: Interdiscip. J. Nonlinear Sci., **14**(4), pp. 1050–1055.

[20] Lazzús, J. A., Rivera, M., and López-Caraballo, C. H., 2016, "Parameter Estimation of Lorenz Chaotic System Using a Hybrid Swarm Intelligence Algorithm," Phys. Lett. A, **380**(11–12), pp. 1164–1171.

[21] Rust, B. W., Oleary, D. P., and Mullen, K. M., 2010, "Modelling Type 1a Supernova Light Curves," *Exponential Data Fitting and Its Applications*, Bentham Science Publishers, Emirate of Sharjah, United Arab Emirates, pp. 169–186.

[22] Wigren, T., and Schoukens, J., 2013, "Three Free Data Sets for Development and Benchmarking in Nonlinear System Identification," European Control Conference, Zurich, Switzerland, July 17–19, pp. 2933–2938.