Second-Order Adjoint Method for Quantum Optimal Control

Harish S. Bhat

***Abstract*— We derive and implement a second-order adjoint method to compute exact gradients and Hessians for a prototypical quantum optimal control problem, that of solving for the minimal energy applied electric field that drives a molecule from a given initial state to a desired target state. For small to moderately sized systems, we demonstrate a vectorized GPU implementation of a second-order adjoint method that computes both Hessians and gradients with wall times only marginally more than those required to compute gradients via commonly used first-order adjoint methods. Pairing our second-order adjoint method with a trust region optimizer (a type of Newton method), we show that it outperforms a first-order method, requiring significantly fewer iterations and wall time to find optimal controls for four molecular systems.**

I. INTRODUCTION

In quantum optimal control, unless our problem admits a solution that we can derive by hand, we compute solutions using numerical optimization. Within the space of gradient-based optimization methods, there is a broad distinction between (i) first-order methods that require only first derivatives of the objective and (ii) second-order methods that require first and second derivatives. Many if not most numerical studies in quantum optimal control use first-order methods, which we take to include quasi-Newton methods that use gradients to update an approximate (inverse) Hessian. In this paper, we derive both first- and second-order adjoint methods for a prototypical quantum optimal control problem. The second-order adjoint method gives us an algorithm to compute exact Hessians, which enables use of an exact trust region solver, a second-order Newton method, to compute optimal controls. We show that the second-order method yields more rapid convergence than the first-order method, both in terms of iteration count and wall clock time.

We consider the dynamics of electrons in molecules with nuclei held fixed as part of the Born-Oppenheimer approximation. The resulting dynamics are governed by the time-dependent Schrödinger equation (TDSE) with a core Hamiltonian consisting of a sum of kinetic operators (for the electrons), electron-nuclear potentials, and electron-electron potentials. To this core Hamiltonian we add time-dependent terms that model applied electric fields within the dipole approximation. In general, for a many-electron system, one cannot solve the resulting TDSE for the full time-dependent wave function without making approximations. Here we employ a method equivalent to full configuration interaction, explained in Section II, through which one can derive from the TDSE a tractable system of ordinary differential equations (ODE). This ODE system is generic and arises whenever one expresses the TDSE's operators and wave function in a finite basis.

For a given molecule, we seek the optimal electric field that one should apply to transfer the molecule from a given initial state to a desired target state. We relax this problem slightly and use a cost that combines the squared norm of the control effort (i.e., the energy of the applied field) with the squared error between achieved and target states at the final time. This allows for the possibility that the desired target is unreachable. The aim in such problems is to reach a final state that is sufficiently close to the desired target.

Though the mathematical form of our quantum optimal control problem can be found throughout the literature [2]–[5], relatively few prior studies use exact Hessians and Newton-type methods. Among those that do, one strand incorporates exact Hessians into the GRAPE (gradient ascent pulse engineering) optimal control algorithm [6], using either an auxiliary matrix exponential [7]–[9] or exact diagonalization [10] to compute the required second derivatives of the matrix exponential. Another strand employs Krylov-Newton methods [3, §6.3], including a matrix-free, semismooth Newton approach in which the whole Hessian matrix need not be stored [11]. As in these works, we solve a quantum optimal control problem using an adjoint method that enforces the equations of motion (as equality constraints) via time-dependent Lagrange multipliers. Alternatively, one can enforce these constraints via projection; [12] combines such an approach with a Hessian-based Newton method.

Our work adds to this literature in two ways. First, throughout all prior work of which we are aware, the second-order adjoint method has not been used to derive algorithms to compute Hessians for quantum optimal control problems. This is in contrast to optimal control, data assimilation, and inverse problems in other areas of science, for which second-order adjoint methods have proven useful, even for large-scale problems [13]. Second, we implement our second-order adjoint method in JAX [14], taking advantage of GPU

This research was sponsored by the Office of Naval Research (Grant Number W911NF-23-1-0153). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This research used the Delta advanced computing and data resource which is supported by the U.S. National Science Foundation (award OAC 2005572) and the State of Illinois. Delta is a joint effort of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications. This work used the Delta system at NCSA through allocation MTH230003 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants 2138259, 2138286, 2138307, 2137603, and 2138296 [1].

H. S. Bhat is with the Department of Applied Mathematics, University of California, Merced, Merced, CA 95343 USA hbhat@ucmerced.edu

vectorization. As we show, this leads to near-linear scaling of Hessian wall clock time as a function of the number of time steps, mitigating a common concern that Hessians take too long to compute. All code developed for this paper is available here: https://github.com/hbhat4000/hessQOC.

II. PROBLEM FORMULATION

For the electron dynamics problem described in Section I, we apply the CASSCF method [15] with a large enough active space such that it is equivalent to full configuration interaction (CI), a standard method in computational chemistry [16]–[18]. For our purposes, full CI yields an orthonormal basis $\{\Psi_q^{\mathrm{CI}}(\mathbf{X})\}_{q=1}^N$, in terms of which we expand the TDSE wave function: $\Psi(\mathbf{X},t) = \sum_{q=1}^N a_q(t)\Psi_q^{\mathrm{CI}}(\mathbf{X})$. Here $a_q(t)$ is the q-th component of the coefficient vector $\mathbf{a}(t)$. Using this in the TDSE (in atomic units with $\hbar = 1$), we obtain

$$i\frac{d}{dt}\mathbf{a}(t) = \left(H_0 + \sum_{k=1}^3 f^k(t)M_k\right)\mathbf{a}(t). \tag{1}$$

Here H_0 is the core Hamiltonian matrix, diagonalized by the CI basis. In the x, y, and z directions, respectively, we have dipole moment matrices $\{M_1, M_2, M_3\}$ and applied electric field strengths $\mathbf{f}(t) = (f^1(t), f^2(t), f^3(t))$. The M_k matrices express the dipole moment operators in the full CI basis; in general, M_k does not commute with H_0. Because we have used all nontrivial Slater determinants, the only error incurred by replacing the TDSE with the finite-dimensional system (1) is due to the finiteness of our atomic orbital basis set. As the size of this basis set goes to infinity, we recover the exact solution of the TDSE with electronic Born-Oppenheimer Hamiltonian [19].

The field strengths $\mathbf{f}(t)$ are the open-loop controls that we solve for in this work. We can now formulate our problem in continuous time: given H_0, $\{M_k\}_{k=1}^3$, a cost balance parameter $\rho > 0$, a final time $T > 0$, an initial state $\boldsymbol{\alpha}$, and a target state $\boldsymbol{\beta}$, solve for $\mathbf{f} : [0,T] \to \mathbb{R}^3$ that minimizes

$$C[f] = \frac{1}{2}\sum_{k=1}^3 \int_0^T f^k(t)^2\,dt + \frac{\rho}{2}\|\mathbf{a}(T) - \boldsymbol{\beta}\|_2^2 \tag{2}$$

subject to the equation of motion (1) and initial condition $\mathbf{a}(0) = \boldsymbol{\alpha}$. Note that we have incorporated the target state into the cost; unlike a hard constraint of the form "$\mathbf{a}(T) = \boldsymbol{\beta}$," we allow for the possibility that it may not be possible to exactly reach $\boldsymbol{\beta}$ in time T starting from $\boldsymbol{\alpha}$.

To avoid gradient/Hessian inconsistencies that may arise in an optimize-then-discretize approach [3, §6.3.3], we take a discretize-then-optimize approach. Using fixed time step $\Delta t > 0$ such that $J\Delta t = T$, we discretize (1) via

$$Z_j = -i\left(H_0 + \sum_{k=1}^3 f^k(j\Delta t;\boldsymbol{\theta})M_k\right)\Delta t \tag{3a}$$

$$\mathbf{a}_{j+1} = \exp(Z_j)\mathbf{a}_j \tag{3b}$$

Here $\mathbf{a}_j$ approximates $\mathbf{a}(j\Delta t)$ and exp denotes the matrix exponential. We have introduced the *model* $\mathbf{f}(j\Delta t;\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^{N_p}$, which we take to be our decision variables. Aside from sufficient smoothness to compute required derivatives, we impose no other requirements on this model. In this work, for a $3 \times J$ parameter matrix $\boldsymbol{\theta}$, we set

$$f^k(j\Delta t;\boldsymbol{\theta}) = \theta_{k,j}. \tag{4}$$

This yields a model in which we optimize directly over all individual values of the discrete-time controls $f^k(j\Delta t)$, equivalent to the piecewise constant model employed in prior work [8], [10]. This model maximizes flexibility: any discrete-time control signal $\mathbf{f}(j\Delta t;\boldsymbol{\theta})$ is achievable under this model. Hence $N_p = 3J$ is an upper bound on the number of model parameters. We explore the maximal model (4) in several tests below.

In future work, we will demonstrate the use of neural network models that take t as input and produce as output $\mathbf{f}(t;\boldsymbol{\theta})$. Here $\boldsymbol{\theta}$ is a concatenation of all neural network parameters (e.g., weights and biases). By choosing the network architecture carefully, we can ensure $N_p = |\boldsymbol{\theta}| \ll 3J$, potentially making such models attractive when J is large.

Given $\{H_0, \{M_k\}_{k=1}^3, \rho, J, \Delta t, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, the discrete-time optimal control problem is to find $\boldsymbol{\theta} \in \mathbb{R}^{N_p}$ that minimizes

$$\mathscr{C}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=1}^3\sum_{j=0}^{J-1} f^k(j\Delta t;\boldsymbol{\theta})^2 + \frac{\rho}{2}\|\mathbf{a}_J - \boldsymbol{\beta}\|_2^2 \tag{5}$$

subject to $\mathbf{a}_0 = \boldsymbol{\alpha}$ and (3) for $j = 0, \ldots, J-1$.

In this paper, we focus on the two-electron molecules H_2 and HeH^+ treated using both the STO-3G [16, §3.6.2] and 6-31G [20] basis sets. For these molecules, full CI in STO-3G and 6-31G results in problem sizes of $N = 4$ and $N = 16$, respectively. Also, for these molecules, the dipole moment matrices in the x and y directions vanish; the remaining control is $f_3(t) \in \mathbb{R}$.

III. SOLUTION METHODS

A. First-Order Adjoint Method

The method begins with a Lagrangian that incorporates the discrete-time cost (5) and equations of motion (3):

$$\mathscr{L}(A,\Lambda,\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=1}^3\sum_{j=0}^{J-1} f_j^k(\boldsymbol{\theta})^2 + \frac{\rho}{2}\|\mathbf{a}_J - \boldsymbol{\beta}\|_2^2 - \Re\sum_{j=0}^{J-1}\boldsymbol{\lambda}_{j+1}^\dagger(\mathbf{a}_{j+1} - \exp(Z_j)\mathbf{a}_j), \tag{6}$$

where Z_j was defined in (3a) and $f_j^k(\boldsymbol{\theta}) = f^k(j\Delta t;\boldsymbol{\theta})$. Here $A = \{\mathbf{a}_j\}_{j=1}^J$ and $\Lambda = \{\boldsymbol{\lambda}_j\}_{j=1}^J$ denote the collections of states and costates, respectively. In (6), we set $\mathbf{a}_0 = \boldsymbol{\alpha}$. Our goal is to find a critical point of $\mathscr{L}$, i.e., to satisfy a necessary condition for the solution of the discrete-time optimal control problem. Hence we compute gradients of (6) with respect to A, Λ and $\boldsymbol{\theta}$. As $\nabla_\Lambda\mathscr{L} = \mathbf{0}$ will reproduce (3b), we focus on the remaining two gradients. Treating $\mathbf{a}_k$ and $\mathbf{a}_k^*$ as separate variables, we compute and set $\nabla_{\mathbf{a}_k}\mathscr{L} = 0$, resulting in

$$\boldsymbol{\lambda}_J = \mathbf{a}_J - \boldsymbol{\beta} \tag{7a}$$

$$\boldsymbol{\lambda}_k^\dagger = \boldsymbol{\lambda}_{k+1}^\dagger \exp(Z_k). \tag{7b}$$

Algorithm 1 First-order adjoint method to solve the discrete-time optimal control problem from Section II

Require: $\{H_0, \{M_k\}_{k=1}^3, \rho, J, \Delta t, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, $\boldsymbol{\theta}^{(0)}$, and $m = 0$.
1: $\mathbf{a}_0 \leftarrow \boldsymbol{\alpha}$
2: **for** $j = 0, \ldots, J-1$ **do**
3: Compute and store the decomposition $V_j D_j V_j^\dagger = Z_j$
4: $\mathbf{a}_{j+1} \leftarrow V_j \exp(D_j) V_j^\dagger \mathbf{a}_j$
5: **end for**
6: $\boldsymbol{\lambda}_J \leftarrow \mathbf{a}_J - \boldsymbol{\beta}$
7: **for** $k = J-1, \ldots, 1$ **do**
8: $\boldsymbol{\lambda}_k^\dagger \leftarrow \boldsymbol{\lambda}_{k+1}^\dagger V_k \exp(D_k) V_k^\dagger$
9: **end for**
10: Use the stored eigendecomposition of Z_k to compute and store $d(\exp Z)/dZ \bullet M$, evaluated at each $Z = Z_k$.
11: Using $\{\mathbf{a}_j\}_{j=0}^{J-1}$, $\{\boldsymbol{\lambda}_j\}_{j=1}^{J}$, the derivatives computed in line 10, and $\nabla_{\boldsymbol{\theta}} \mathbf{f}$, compute $\nabla_{\boldsymbol{\theta}} \mathscr{L}$ via (8). Use $\nabla_{\boldsymbol{\theta}} \mathscr{L}$ in a gradient-based optimization method to compute the next iterate $\boldsymbol{\theta}^{(m+1)}$.
12: If $\|\boldsymbol{\theta}^{(m+1)} - \boldsymbol{\theta}^{(m)}\| < \delta$ or $\|\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}^{(m+1)})\| < \epsilon$, exit; else $m \leftarrow m+1$ and return to line 1.

This system determines $\boldsymbol{\lambda}$: we start with the final condition (7a) and iterate (7b) backwards in time for $k = J-1, \ldots, 1$. Next, we have for $k = 0, \ldots, J-1$:

$$\frac{\partial \mathscr{L}}{\partial \theta_\ell} = \sum_{k=1}^{3} \sum_{j=0}^{J-1} f_j^k(\boldsymbol{\theta}) \frac{\partial f_j^k}{\partial \theta_\ell} - \Re \sum_{j=0}^{J-1} \boldsymbol{\lambda}_{j+1}^\dagger \left[i\Delta t \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k \frac{\partial f_j^k}{\partial \theta_\ell} \right] \mathbf{a}_j. \quad (8)$$

We assume that $\nabla_{\boldsymbol{\theta}} \mathbf{f}$ can be computed efficiently either by hand or by automatic differentiation. Prior work has shown how to compute the directional derivative $d(\exp Z)/dZ \bullet W$ for Hermitian or anti-Hermitian matrices Z [7], [10], [21]. We have our own derivations, omitted due to space constraints, that result in efficient implementations. For small to moderately sized Z, we compute $\exp(Z)$ by first computing the eigendecomposition $Z = VDV^\dagger$, which yields $\exp(Z) = V \exp(D) V^\dagger$. Once we have computed V and D, we compute $d(\exp Z)/dZ \bullet W$ with low additional cost. We summarize the first-order adjoint method with exact gradients (8) in Algorithm 1.

B. Second-Order Adjoint Method

In order to compute the required Hessian, we take a *total* derivative of (8) with respect to $\boldsymbol{\theta}$, resulting in

$$\begin{aligned}
\frac{d}{d\theta_m} \frac{\partial \mathscr{L}}{\partial \theta_\ell} &= \sum_{k=1}^{3} \sum_{j=0}^{J-1} f_j^k(\boldsymbol{\theta}) \frac{\partial^2 f_j^k}{\partial \theta_\ell \partial \theta_m} + \frac{\partial f_j^k}{\partial \theta_\ell} \frac{\partial f_j^k}{\partial \theta_m} \\
&+ \Re \sum_{j=0}^{J-1} \left\{ \frac{d\boldsymbol{\lambda}_{j+1}^\dagger}{d\theta_m} \left[i\Delta t \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k \frac{\partial f_j^k}{\partial \theta_\ell} \right] \mathbf{a}_j \right. \\
&+ \boldsymbol{\lambda}_{j+1}^\dagger \left[(\Delta t)^2 \sum_{k,n=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet (M_k \otimes M_n) \frac{\partial f_j^k}{\partial \theta_\ell} \frac{\partial f_j^n}{\partial \theta_m} \right] \mathbf{a}_j \\
&+ \boldsymbol{\lambda}_{j+1}^\dagger \left[i\Delta t \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k \frac{\partial^2 f_j^k}{\partial \theta_\ell \partial \theta_m} \right] \mathbf{a}_j \\
&\left. + \boldsymbol{\lambda}_{j+1}^\dagger \left[i\Delta t \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k \frac{\partial f_j^k}{\partial \theta_\ell} \right] \frac{d\mathbf{a}_j}{d\theta_m} \right\}. \quad (9)
\end{aligned}$$

Algorithm 2 Second-order adjoint method to solve the discrete-time optimal control problem from Section II

Require: $\{H_0, \{M_k\}_{k=1}^3, \rho, J, \Delta t, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, $\boldsymbol{\theta}^{(0)}$, and $m = 0$.
1: $\mathbf{a}_0 \leftarrow \boldsymbol{\alpha}$
2: **for** $j = 0, \ldots, J-1$ **do**
3: Compute and store the decomposition $V_j D_j V_j^\dagger = Z_j$
4: $\mathbf{a}_{j+1} \leftarrow V_j \exp(D_j) V_j^\dagger \mathbf{a}_j$
5: **end for**
6: $\boldsymbol{\lambda}_J \leftarrow \mathbf{a}_J - \boldsymbol{\beta}$
7: **for** $j = J-1, \ldots, 1$ **do**
8: $\boldsymbol{\lambda}_j^\dagger \leftarrow \boldsymbol{\lambda}_{j+1}^\dagger V_j \exp(D_j) V_j^\dagger$
9: **end for**
10: **for** $j = 0, \ldots, J-1$ **do**
11: Use the eigendecomposition of Z_j to compute and store all derivatives $d(\exp Z)/dZ \bullet M_k$ and $d^2(\exp Z)/dZ^2 \bullet (M_k \otimes M_n)$, evaluated at $Z = Z_j$.
12: **end for**
13: **for** $\ell = 1, \ldots, N_p$ **do** ▷ Parallelize/vectorize over ℓ
14: $d\mathbf{a}_0/d\theta_\ell \leftarrow \mathbf{0}$
15: **for** $j = 0, \ldots, J-1$ **do**
16: $d\mathbf{a}_{j+1}/d\theta_\ell \leftarrow V_j \exp(D_j) V_j^\dagger d\mathbf{a}_j/d\theta_\ell$
17: $-i\Delta t \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k (\partial f_j^k / \partial \theta_\ell) \mathbf{a}_j$
18: **end for**
19: **end for**
20: **for** $\ell = 1, \ldots, N_p$ **do** ▷ Parallelize/vectorize over ℓ
21: $\boldsymbol{\mu}_{J,\ell} \leftarrow \rho(d\mathbf{a}_J/df_\ell)$
22: **for** $j = J-1, \ldots, 1$ **do**
23: $\boldsymbol{\mu}_{j,\ell}^\dagger \leftarrow \boldsymbol{\mu}_{j+1,\ell}^\dagger V_j \exp(D_j) V_j^\dagger$
24: $-i\Delta t \boldsymbol{\lambda}_{j+1}^\dagger \sum_{k=1}^{3} \frac{d}{dZ} \exp(Z) \Big|_{Z=Z_j} \bullet M_k (\partial f_j^k / \partial \theta_\ell)$
25: **end for**
26: **end for**
27: Using $\{\mathbf{a}_j\}_{j=0}^{J-1}$, $\{\boldsymbol{\lambda}_j\}_{j=1}^{J}$, $\{\nabla_{\boldsymbol{\theta}} \mathbf{a}_j\}_{j=0}^{J-1}$, $\{\boldsymbol{\mu}_j\}_{j=1}^{J}$ and derivatives of matrix exponential, compute $\nabla_{\boldsymbol{\theta}} \mathscr{L}$ via (8) and $\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathscr{L}$ via (9); use these in a second-order optimization method to compute next iterate $\boldsymbol{\theta}^{(m+1)}$.
28: If $\|\boldsymbol{\theta}^{(m+1)} - \boldsymbol{\theta}^{(m)}\| < \delta$ or $\|\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}^{(m+1)})\| < \epsilon$, exit; else $m \leftarrow m+1$ and return to line 1.

We see $\nabla_{\boldsymbol{\theta}} \mathbf{f}$ and $\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathbf{f}$ throughout; we assume both can be computed efficiently either by hand or by automatic differentiation. In the remaining lines, we see three new objects. One new object is the second derivative $d^2(\exp Z)/dZ^2 \bullet (W_1 \otimes W_2)$. As before, we have derived expressions for this that can be implemented efficiently assuming we have precomputed $Z = VDV^\dagger$. The remaining new objects in (9) are $d\mathbf{a}_j/d\theta_\ell$ and $\boldsymbol{\mu}_{j,\ell} = d\boldsymbol{\lambda}_j/d\theta_\ell$. To compute these, we begin with the total derivative of the Lagrangian: $\mathscr{L}'_\ell := d\mathscr{L}/d\theta_\ell$, which we regard as a function of $A = \{\mathbf{a}_j\}_{j=1}^J$, $\Lambda = \{\boldsymbol{\lambda}_j\}_{j=1}^J$ and $\nabla_{\boldsymbol{\theta}} \Lambda = \{\boldsymbol{\mu}_j = \nabla_{\boldsymbol{\theta}} \boldsymbol{\lambda}_j\}_{j=1}^J$. As before, we seek a critical point of $\mathscr{L}'$. Note that $\nabla_{\boldsymbol{\mu}_k} \mathscr{L}' = \mathbf{0}$ and $\nabla_{\boldsymbol{\lambda}_k} \mathscr{L}' = \mathbf{0}$ will reproduce both (3b) and the total derivative of (3b) with respect to $\boldsymbol{\theta}$. This latter derivative naturally yields lines 13-19 in Algorithm 2, a scheme to compute $d\mathbf{a}_j/d\theta_\ell$. We then focus on variations of $\mathscr{L}'$ with respect to $\mathbf{a}$; requiring that $\delta\mathscr{L}'$ vanishes for all variations $\delta\mathbf{a}_j$ and $\nabla_{\boldsymbol{\theta}} \delta\mathbf{a}_j$, we derive both the first-order adjoint system (7) and lines 20-26 in Algorithm 2, a scheme to compute $\boldsymbol{\mu}_{j,\ell}$. For full derivations, see [22]. In Algorithm 2, we give an end-to-end procedure to iteratively solve for θ (and thereby solve for $\mathbf{f}$) using exact Hessians and gradients.

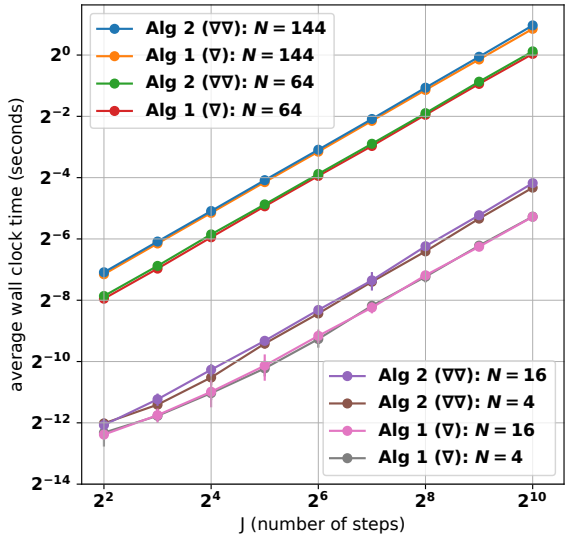

Fig. 1. We give a log-log plot of wall clock time results for Algorithms 1 and 2. Here N represents problem size, while J is the number of time steps required to go from $t = 0$ to $t = T$. For each fixed choice of parameters (N, J), we ran each algorithm 1000 times; we plot the mean (circular dot) and error bar (plus/minus twice the standard deviation) of these runs. *The results show that both algorithms' scaling as a function of J is close to linear—see Section IV-A in the main text for details.* The upshot: for the quantum optimal control problem we studied, Hessians and gradients can be computed at similar cost to gradients alone.

C. Operational Difference

The first 9 lines of both algorithms are identical. The differences between the two algorithms are that Alg. 2 requires (i) computation of second derivatives of the matrix exponential, (ii) two nested loops, from lines 13-19 and lines 20-26, and (iii) computation of the Hessian via (9). Regarding (i), let us assume that if one has already diagonalized each Z_j, then the cost of computing second derivatives of $\exp(Z)$ at Z_j is a constant multiple of the cost of computing first derivatives of $\exp(Z)$ at Z_j.

As for (ii), in the two nested loops, the inner loops are very similar, each requiring a total of $4J$ matrix-vector multiplications and J matrix-scalar multiplications. The outer loops over the ℓ variable can be parallelized easily. Here ℓ goes from 1 to N_p, the length of the $\boldsymbol{\theta}$ vector, the total number of parameters in the model. When we examine (iii), we see that computation of both the Hessian (9) and the gradient (8) can also be parallelized over the ℓ variable. In the above, we use *parallelize* to include either shared-memory multiprocessing (e.g., via OpenMP) or vectorization. Our implementations, designed to run on graphical processing units (GPUs), utilize vectorization via `vmap` constructs in JAX [14]. With effective parallelization or vectorization, we hypothesize that one pass of either Alg. 1 or 2 will have a wall clock time that is nearly linear (if not linear) in J, the total number of time steps required to go from $t = 0$ to $t = T$. Below we will test this empirically.

IV. RESULTS

A. Scaling of Gradient and Hessian Algorithms

With the maximal model (4), the outer double for loops (over the ℓ variable) in Alg. 2 go up to $N_p \geq J$. Thus a naïve implementation of Alg. 2 might be expected to have a run time that scales quadratically in J. On the other hand, vectorization of this outer loop (described in Section III-C) should yield closer-to-linear scaling in J.

We test this on systems with four values of N, the dimension of the Hamiltonian and dipole moment matrices. The $N = 4$ and $N = 16$ systems correspond to full CI for the molecule HeH^+ in, respectively, the STO-3G and 6-31G basis sets. For this molecular system, only the dipole moment matrix in the z direction is nonzero; hence the control consists of scalars $f_j = f(j\Delta t; \boldsymbol{\theta})$, and $\boldsymbol{\theta}$ is simply a vector of length J. For $N = 64$ and $N = 144$, we created random diagonal core Hamiltonian matrices H_0 and random Hermitian dipole moment matrices M. For all systems, we take the initial state to be $\boldsymbol{\alpha} = [1, 0, \ldots, 0]$ and the target state to be $\boldsymbol{\beta} = [0, \ldots, 0, 1]$.

For a given molecular system, we fix the number of time steps J, sample an initial $\boldsymbol{\theta}^{(0)}$ with each $\theta_j^{(0)} \sim \mathcal{N}(0, 1)$ (standard normal), and check the wall clock time to run each algorithm (in turn) on precisely the same set of inputs $\boldsymbol{\theta}^{(0)}$ and $\{H_0, \{M_k\}_{k=1}^3, \rho, J, \Delta t, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$. We take care to force JIT compilation of our JAX implementations of both algorithms before checking wall clock times.

Fig. 1 shows a log-log plot of the means and error bars (plus/minus two standard deviations) across 1000 trials of the above experiment, conducted for each $J = 2^\kappa$ with $\kappa = 2, \ldots, 10$. All tests were conducted on Nvidia A100 (40 GB RAM) GPU nodes on NCSA's Delta GPU cluster [23]. The results indicate that Alg. 2's wall clock time has nearly linear scaling. For Alg. 1, the slopes of the OLS (ordinary least squares) regression lines fit to the log-log data for $N = 4, 16, 64, 144$ are, respectively, 0.9082, 0.9067, 0.9999, and 1.001; for Alg. 2, the slopes are 0.9871, 0.9938, 0.9975, and 1.006. Slopes $m \approx 1$ indicate near-linear scaling.

In prior work that did not use second-order adjoints, with a parallel CPU implementation (for $N = 144$), we learn that "Given the same computing resources, a Hessian calculation takes approximately ten times longer than a gradient calculation" [8]. Our results above (esp. for $N = 64$ and $N = 144$) show that the wall clock time required to obtain the Hessian *and* the gradient (via Alg. 2) is practically identical to that required to obtain only the gradient (via Alg. 1). This shows the advantages of the second-order adjoint method and our vectorized GPU implementation.

B. Optimal Control Results

Continuing with the maximal model (4), we consider the question of whether using gradients *and* Hessians improves our ability to solve the optimal control problem, relative to using gradients alone. We answer this question for both HeH^+ and H_2 in each of two basis sets, STO-3G and 6-31G. As with HeH^+ (described above), the control consists of an applied electric field in the z-direction only; the dipole moment matrices in the x- and y- directions vanish. Hence $|\boldsymbol{\theta}| = J$. We retain the same initial and target states mentioned above: $\boldsymbol{\alpha} = [1, 0, \ldots, 0]$ and $\boldsymbol{\beta} = [0, \ldots, 0, 1]$. For all systems, we use time step $\Delta t = 0.1$, $J = 200$ steps,

molecule	basis set	algorithm	# iterations	wall time	final cost	final $\|\text{grad}\|$	$\|\text{target viol}\|$
H_2	STO-3G	Alg 1	5748.12	73.4884	**12.7697**	0.882974	1.604×10^{-5}
		Alg 2	**1421.67**	**21.3366**	12.7795	$\mathbf{6.2616\times10^{-5}}$	$\mathbf{1.57277\times10^{-5}}$
H_2	6-31G	Alg 1	5313.17	92.6164	14.5274	1.28098	$\mathbf{2.41829\times10^{-5}}$
		Alg 2	**1335.56**	**21.41**	**14.5221**	$\mathbf{4.75845\times10^{-5}}$	2.52149×10^{-5}
HeH^+	STO-3G	Alg 1	7707.69	92.0181	21.0352	2.38341	3.23744×10^{-5}
		Alg 2	**2729.95**	**42.2176**	**20.8101**	$\mathbf{5.45808\times10^{-2}}$	$\mathbf{3.22242\times10^{-5}}$
HeH^+	6-31G	Alg 1	6967.62	105.222	16.1364	3.25426	1.93098×10^{-5}
		Alg 2	**1997.94**	**31.8765**	**15.8801**	$\mathbf{8.06679\times10^{-3}}$	$\mathbf{1.77365\times10^{-5}}$

TABLE I

MEAN PERFORMANCE OF EACH ALGORITHM ACROSS 1000 TRIALS USING THE MAXIMAL MODEL (4). THE RESULTS SHOW THAT ALG. 2 OUTPERFORMS ALG. 1, CONSISTENTLY REQUIRING FEWER ITERATIONS *and* LESS WALL CLOCK TIME TO ACHIEVE SOLUTIONS OF THE SAME OR BETTER QUALITY. FOR DETAILS, SEE SECTION IV-B IN THE MAIN TEXT. FOR EACH MOLECULAR SYSTEM, WE BOLDFACE THE BEST RESULTS.

molecule	basis set	# iterations	wall time	final cost	final $\|\text{grad}\|$	$\|\text{target viol}\|$
H_2	STO-3G	5.08 (2.12, 9.88)	4.44 (1.72, 9.06)	1.00 (0.83, 1.17)	5.00×10^{6} (1.62×10^{-1}, 4.11×10^{4})	1.03 (0.83, 1.24)
H_2	6-31G	4.95 (1.66, 10.27)	5.46 (1.58, 13.74)	1.03 (0.68, 1.49)	3.53×10^{6} (2.84×10^{-1}, 1.20×10^{5})	1.05 (0.47, 1.98)
HeH^+	STO-3G	3.33 (1.67, 6.05)	2.58 (1.31, 4.81)	1.02 (0.81, 1.24)	1.80×10^{7} (2.87×10^{-1}, 1.72×10^{7})	1.03 (0.66, 1.52)
HeH^+	6-31G	4.33 (1.56, 8.80)	4.15 (1.36, 9.14)	1.03 (0.80, 1.29)	1.64×10^{7} (3.83×10^{-1}, 5.85×10^{6})	1.12 (0.71, 1.79)

TABLE II

RATIOS INDICATING THE RELATIVE PERFORMANCE OF ALG. 1 TO ALG. 2 USING THE MAXIMAL MODEL (4). NOTE THAT RATIOS ARE COMPUTED FIRST (IN EACH OF 1000 TRIALS), AFTER WHICH WE COMPUTE MEANS AND $(0.05, 0.95)$ QUANTILES, REPORTED IN PARENTHESES. FOR DETAILS, SEE SECTION IV-B IN THE MAIN TEXT. THE RESULTS SHOW THAT USING EXACT HESSIANS ALLOWS FOR MORE RAPID CONVERGENCE TO SOLUTIONS OF SUBSTANTIALLY THE SAME QUALITY IN TERMS OF FINAL COST AND NORM DISCREPANCY BETWEEN ACHIEVED AND DESIRED TARGETS.

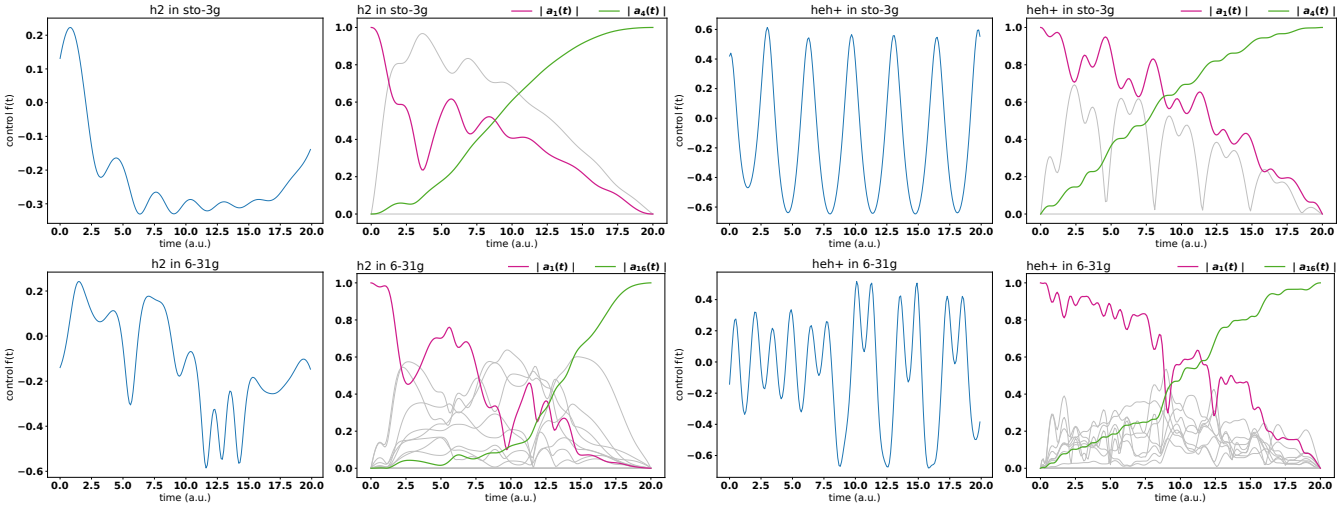

Fig. 2. From the pool of 1000 **maximal model** (4) solutions whose statistics were reported in Tables I-II, the plotted solutions achieved the lowest cost. For each molecular system (choice of molecule plus basis set), we plot both the optimal field strength $f(t)$ and the magnitudes $|a_j(t)|$ of the controlled trajectory. Here $J = 200$ and $\Delta t = 0.1$. Gray components of the controlled trajectory have initial and target values of 0; colored components' initial values differ from their targets.

and cost-balance parameter $\rho = 10^6$. We set the termination criteria tolerances δ and ϵ (in Algorithms 1 and 2) to 10^{-10}. We also set the maximum number of iterations to be 10^4.

Each experiment starts by drawing a random initial guess for $\boldsymbol{\theta}$, with each entry normally distributed with mean zero and variance one. With this initial guess, we solve the optimal control problem using Alg. 1; with the same initial guess, we solve the problem again using Alg. 2. We carried out this experiment 1000 times, recording data on algorithm performance and solution quality. Optimization always terminated due to satisfaction of either the δ or ϵ criteria; the iteration limit was never reached. In both cases, we couple our algorithms with the trust-region optimizer *trust-constr* in the *scipy.optimize* package [24]. When we use Alg. 1 to supply only gradients to the trust-region optimizer, it uses these gradients to compute approximate inverse Hessians via BFGS updates. When we use Alg. 2, the trust-region optimizer uses exact gradients and Hessians.

We measure both algorithms' performance in terms of the iteration count and wall time required to achieve termination

criteria. We measure solution quality in terms of the final cost, the final norm of the gradient (final $\|\text{grad}\|$), and the final norm difference between achieved and desired targets ($\|\text{target viol}\|$). For each molecule, basis set, and algorithm, we present in Table I the mean results across 1000 trials. The results show that Alg. 2 requires fewer iterations *and* less wall clock time to achieve solutions whose quality is no worse than those achieved by Alg. 1. Based on what we learned from Fig. 1, the per-iteration cost of Alg. 2 is nearly identical to that of Alg. 1, so fewer iterations should imply less wall clock time. In Table I, in each block of two rows (same molecule, same basis set), we highlight in bold the best results. Whenever Alg. 1's result is boldfaced, it is practically equivalent to Alg. 2's result.

For each of the 1000 trials, we also compute the *ratio* of Alg. 1's iteration count, wall time, final cost, final $\|\text{grad}\|$, and $\|\text{target viol}\|$ to those of Alg. 2. We report in Table II the means and, in parentheses, the 0.05 and 0.95 quantiles of the 1000 computed ratios in each category. Let us illustrate how to read Table II concretely: for HeH^+ in STO-3G, Table II reports that Alg. 1 will require (on average across 1000 trials) 5.08 times as many iterations than Alg. 2 to achieve the termination criteria. For this molecular system, there is an empirical probability of 0.90 that Alg. 1 required between 2.12 and 9.88 times more iterations than Alg. 2 to achieve the termination criteria.

The means and quantile intervals in Table II confirm that for most choices of the initialization $\boldsymbol{\theta}^{(0)}$, Alg. 2 converges more rapidly to solutions of substantially the same quality as Alg. 1. While there do exist bad initializations $\boldsymbol{\theta}^{(0)}$ that lead to Alg. 2 producing a worse final solution than Alg. 1 (in terms of cost and/or $\|\text{target viol}\|$), this is uncommon. At least 90% of the time, both algorithms' final values of cost and $\|\text{target viol}\|$ are within an order of magnitude.

For each of our four molecular systems, we use the final value of the optimized cost to select the best solution $\boldsymbol{\theta}$ obtained across the 1000 trials. In Fig. 2, we plot these optimal control signals $f(t) = f_3(t)$ (left panels) and the magnitudes $|a_j(t)|$ of the corresponding controlled trajectories (right panels). As we have used the maximal model (4), what we actually plot here (in the left panels) is $f(j\Delta t; \boldsymbol{\theta}) = \theta_j$ as a function of j. Without any regularization to enforce smoothness in time of the control $f(t)$, the resulting solutions are all smooth, with a visual resemblance to sums of sinusoids. In all cases, the controlled trajectories achieve their targets with an error on the order of 10^{-5}.

Consider again the best maximal model solutions produced by Alg. 2 and plotted in Fig. 2; for each molecular system, the plotted $f(t)$ and $|a_j(t)|$ curves correspond to a particular $\boldsymbol{\theta}$ vector. When we examine the eigenvalues of the Hessians corresponding to these four theta vectors (one per molecular system), we find that all eigenvalues are positive. Together with the near-vanishing gradients for Alg. 2 reported in Table I, we conclude that with the maximal model (4), our methods have succeeded in finding locally optimal controls.

References

[1] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns, "ACCESS: Advancing innovation: NSF's advanced cyberinfrastructure coordination ecosystem: Services & support," in *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, ser. PEARC '23. New York, NY, USA: ACM, 2023, pp. 173–176.

[2] J. Werschnik and E. K. U. Gross, "Quantum optimal control theory," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 40, no. 18, p. R175, sep 2007.

[3] A. Borzì, G. Ciaramella, and M. Sprengel, *Formulation and Numerical Solution of Quantum Control Problems*. Philadelphia: Society for Industrial and Applied Mathematics, 2017.

[4] U. Boscain, M. Sigalotti, and D. Sugny, "Introduction to the Pontryagin Maximum Principle for Quantum Optimal Control," *PRX Quantum*, vol. 2, no. 3, p. 030203, Sept. 2021.

[5] D. D'Alessandro, *Introduction to Quantum Control and Dynamics*, 2nd ed., ser. Chapman & Hall/CRC Applied Mathematics & Nonlinear Science. Boca Raton, FL: CRC Press, 2022.

[6] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbruggen, and S. J. Glaser, "Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms," *Journal of Magnetic Resonance*, vol. 172, no. 2, pp. 296–305, 2005.

[7] D. L. Goodwin and I. Kuprov, "Auxiliary matrix formalism for interaction representation transformations, optimal control, and spin relaxation theories," *The Journal of Chemical Physics*, vol. 143, no. 8, p. 084113, 08 2015.

[8] ——, "Modified Newton-Raphson GRAPE methods for optimal control of spin systems," *The Journal of Chemical Physics*, vol. 144, no. 20, p. 204107, 05 2016.

[9] D. L. Goodwin and M. S. Vinding, "Accelerated Newton-Raphson GRAPE methods for optimal control," *Physical Review Research*, vol. 5, no. 1, p. L012042, Mar. 2023.

[10] M. Dalgaard, F. Motzoi, J. H. M. Jensen, and J. Sherson, "Hessian-based optimization of constrained quantum control," *Phys. Rev. A*, vol. 102, p. 042612, Oct 2020.

[11] G. Ciaramella, A. Borzì, G. Dirr, and D. Wachsmuth, "Newton Methods for the Optimal Control of Closed Quantum Spin Systems," *SIAM J. Sci. Comp.*, vol. 37, no. 1, pp. A319–A346, 2015.

[12] J. Shao, M. Naris, J. Hauser, and M. M. Nicotra, "Solving quantum optimal control problems using projection-operator-based Newton steps," *Phys. Rev. A*, vol. 109, p. 012609, Jan 2024.

[13] N. Petra and E. W. Sachs, "Second order adjoints in optimization," in *Numerical Analysis and Optimization*, M. Al-Baali, A. Purnama, and L. Grandinetti, Eds. Cham: Springer, 2021, pp. 209–230.

[14] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, "JAX: composable transformations of Python+NumPy programs," 2018, http://github.com/google/jax.

[15] B. O. Roos, P. R. Taylor, and P. E. Sigbahn, "A complete active space SCF method (CASSCF) using a density matrix formulated super-CI approach," *Chemical Physics*, vol. 48, no. 2, pp. 157–173, 1980.

[16] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. NY: McGraw-Hill, 1989.

[17] R. McWeeny, *Methods of Molecular Quantum Mechanics*, 2nd ed. San Diego: Academic Press, 1989.

[18] H. S. Bhat, H. Bassi, K. Ranka, and C. M. Isborn, "Incorporating memory into propagation of 1-electron reduced density matrices," *Journal of Mathematical Physics*, vol. 66, no. 2, p. 023503, 02 2025.

[19] J. Townsend, J. K. Kirkland, and K. D. Vogiatzis, "Post-Hartree-Fock methods: configuration interaction, many-body perturbation theory, coupled-cluster theory," in *Mathematical Physics in Theoretical Chemistry*. Elsevier, 2019, pp. 63–117.

[20] W. J. Hehre, R. Ditchfield, and J. A. Pople, "Self-consistent molecular orbital methods. XII. Further extensions of Gaussian-type basis sets for use in molecular orbital studies of organic molecules," *The Journal of Chemical Physics*, vol. 56, no. 5, pp. 2257–2261, 09 1972.

[21] A. S. Lewis and H. S. Sendov, "Twice differentiable spectral functions," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 2, pp. 368–386, 2001.

[22] https://faculty.ucmerced.edu/hbhat/qocHessian7.pdf.

[23] "NCSA Delta System Architecture." [Online]. Available: https://docs.ncsa.illinois.edu/systems/delta/en/latest/user_guide/architecture.html

[24] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Philadelphia, PA: SIAM, 2000.