Automatic differentiation of dominant eigensolver and its applications in quantum physics

Hao Xie,1,2 Jin-Guo Liu,1 and Lei Wang1,3,*

1Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China
2University of Chinese Academy of Sciences, Beijing 100049, China
3Songshan Lake Materials Laboratory, Dongguan, Guangdong 523808, China

(Dated: January 14, 2020)

We investigate the automatic differentiation of dominant eigensolver where only a small proportion of eigenvalues and corresponding eigenvectors are obtained. Backpropagation through the dominant eigensolver involves solving certain low-rank linear systems without direct access to the full spectrum of the problem. Furthermore, the backward pass can be conveniently differentiated again, which implies that in principle one can obtain arbitrarily higher order derivatives of the dominant eigen-decomposition process. These results allow for the construction of an efficient dominant eigensolver primitive, which has wide applications in quantum physics. As a demonstration, we compute second order derivative of the ground state energy of the same model in the thermodynamic limit by performing gradient-based optimization of uniform matrix product states. By programming these computation tasks in a fully differentiable way, one can efficiently handle the dominant eigen-decomposition of very large matrices while still sharing various advantages of differentiable programming paradigm, notably the generic nature of the implementation and free of tedious human efforts of deriving gradients analytically.

I. INTRODUCTION

Automatic differentiation is a technique of numerically evaluating the exact derivatives of a computation process expressed as a program [1]. Basically, the derivatives are obtained by traversing through the computation graph from end to end and iteratively applying the chain rule. Compared to numerical differentiation, it can compute the value of derivatives to machine precision. Automatic differentiation is the computation engine underlying modern deep learning applications [2–4], and has been realized in different ways in a large variety of modern deep learning libraries, such as TensorFlow [5], autograd [6], PyTorch [7], Jax [8] and Zygote [9]. This fact has triggered applications of automatic differentiation in much broader research areas, such as quantum optimal control [10, 11], various electron structure methods in quantum chemistry [12–14] and tensor network approach of studying statistical physics and quantum many-body problems [15].

One of the most important characteristics of automatic differentiation is its modular nature. More specifically, this means that the programmer can control the granularity of the computation process as he wants by grouping many “elementary” computation steps in a single unit. Such a unit often has higher-level mathematical meanings and is called a primitive. For example, it turns out that many subroutines in scientific computing can be differentiated as a whole unit. Typical examples include solving ordinary differential equations [16], various linear algebra manipulations, including both simple operations such as matrix multiplication, matrix inverse and more sophisticated ones such as full eigen-decomposition, Singular Value Decomposition (SVD), QR decomposition, etc [17, 18]. In view of this perspective, a new programming paradigm called differentiable programming has emerged, which lays emphasize on assembling relative simple differentiable components (i.e., primitives) together and differentiating through them by applying the chain rule iteratively. By formulating a computation task in this way, one is able to combine the domain specific knowledge and the flexibility of modern machine learning techniques.

There can be two kinds of schemes of performing automatic differentiation of a computation process, namely the forward mode and the reverse mode. The difference lies on the order of evaluating and passing gradients through the computation graph using the chain rule. The forward mode automatic differentiation computes the gradients along with the objective output in a single forward pass, whereas in the reverse mode version, one needs an extra backward pass in which the gradient message are passed from the output back to the input via a series of vector-Jacobian product. This approach is usually referred to as the back-propagation algorithm [3] in the context of deep learning. It is also favored and more commonly adopted in applications of physics and deep learning than the forward mode, due to the fact that the dimension of the output is often much smaller than that of the input. For the same reason, we will almost exclusively focus on reverse mode automatic differentiation in this paper.

Since primitives are the building blocks of a differentiable program, the central task of differentiable programming lies on the automatic differentiation of the primitives. To be more specific, in a typical computation process, one starts from some input parameters, say, \( \theta \), and computes a series of intermediate results following the topological order characterized by the computation graph, until reaching the final outcome \( L \), which is usually assumed to be a scalar valued loss function. Consider a certain primitive, which can be generally described by a function \( O = O(I) \), where \( O \) and \( I \) denote the outputs and inputs, respectively. It is convenient to introduce the \textit{adjoint} of a certain variable \( T \) as \( \overline{T} \equiv \frac{\partial O}{\partial I} \). Then in reverse mode
automatic differentiation, one is typically concerned with the
adjoint $\tilde{I}$ of inputs as a function of the adjoint $\tilde{O}$ of outputs.
This is sometimes referred to as the adjoint relation of the
primitive. This adjoint relation can be written in an abstract
mathematical form as follows:

$$\tilde{I} = \tilde{I}(\tilde{O}; I, O) = \frac{\partial O}{\partial I}.$$  \hspace{1cm} (1)

Notice that this function depends linearly on the argument
$\tilde{O}$, as indicated by the linear approximation nature of deriva-
tives. Once this function is determined for all the primitives
involved in a computation process, one can use the chain rule to
“glue” them together and compute the desired gradients $\frac{\partial }{\partial I}$
by traversing through the computation graph in the reverse di-
rection.

In this paper, we will concentrate on the automatic differen-
tiation of dominant eigensolver, which is essentially the
process of eigen-decomposition, except that only a small
number of eigenvalues and eigenvectors are desired. Eigen-
decomposition plays a fundamental role in quantum physics
and chemistry, and is related to many practical methods such as
exact diagonalization, full configuration interaction and
Hartree-Fock method [12]; it has also been used in conjunc-
tion with neural network architectures in various deep learn-
ing algorithms [19–23]; furthermore, it has close and intrin-
sic relation with SVD, and they have been widely used in
various tensor network algorithms [15, 24–26]. In many of
these applications, usually only a small number of eigenval-
ues and eigenvectors are of practical interest, despite the fact
that the dimension of the vector space involved is possibly
quite large. In the context of quantum physics and chemistry,
for example, this typically means that one is concerned only
about the ground state or several low-lying excited states. The
situation is also similar in many other settings, including deep
learning applications [22, 23] and tensor network algorithms
[24, 25]. In this respect, since the computation cost of full
eigen-decomposition is fairly high when the matrix dimension
is large, one would usually resort to more efficient numerical
algorithms of dominant eigen-decomposition, such as power
iteration or Lanczos method. These algorithms are particu-
larly useful when the matrix to be diagonalized has certain
inner structures (e.g., sparse), which is often the case in prac-
tical applications.

However, there emerges an additional difficulty when trying to
implement the dominant eigen-decomposition process in a
derifferentiable way. To get an intuitive understanding, consider
a quantum system described by a Hamiltonian $H$, which de-
pends on a certain parameter $\lambda$. To obtain the derivative of the
ground state $|\psi_0\rangle$ with respect to $\lambda$, the first order perturbation
theory gives

$$\frac{\partial \psi_0}{\partial \lambda} = \sum_{n \neq 0} \left( \frac{\partial H}{\partial \lambda} \left| \psi_0 \right\rangle \right) \left( \frac{E_0 - E_n}{E_0 - E_n} \right) |\psi_n\rangle.$$  \hspace{1cm} (2)

where $E_n, |\psi_n\rangle$ are the energy eigenvalues and eigenstates,
respectively. Due to explicit presence of the full eigen-spectrum
in Eq. (2), the computation of full eigen-decomposition is in-
evitable in this direct approach, which is inefficient and even

intractable when the dimension of the Hilbert space is large.
One way around this problem is proposed in Ref. [23], which
is based on the power iteration algorithm of dominant eigens-
 solver. Since the operations involved in the power iteration
procedure are very simple (mainly matrix multiplications), it
can be easily differentiated without any reference to the full
spectrum. However, this approach also has some drawbacks.
The convergence rate of the power iteration can be a problem
in practice and have to be analyzed case by case. In addition,
although possible in principle, it is often tedious and imprac-
tical to obtain good estimate of other eigenvalues and eigen-
 vectors than the dominant one through power iteration, which
makes the approach inflexible to various user needs.

Basically, we need a way to effectively separate the infor-
mation about the desired eigenvalues and eigenvectors out of
the full spectrum. In this paper, this problem is tackled by
two different methods, which allow us to construct a high-
level primitive that correctly handles the automatic differenti-
ation of dominant eigensolver without direct access to the full
spectrum. The first one, called the adjoint method, can yield
the relevant formulas straightforwardly in a full-spectrum-
free form. On the other hand, the second method reflects the
modular nature of differentiable programming paradigm
mentioned above, by wrapping the process of full eigen-
decomposition within the dominant one and utilizing the re-
uts of the former in the latter. Typically, the results obtained
by this method still have explicit dependence on the full spec-
trum of the matrix. Nevertheless, these two methods are to-
tally equivalent, and by making a careful contrast between
them, one can get a clear understanding of how the goal of
separating the desired information out of the full spectrum is
achieved behind the scene. Even more ideally, it turns out
that the obtained dominant eigensolver primitive could be dif-
ferentiated again in a convenient way, which in turn makes
it support in principle arbitrarily higher order derivatives of
the dominant eigen-decomposition process. These results are
very useful in practice, since one can take advantage of the ef-
ficiency of dominant eigen-decomposition algorithms, while
also sharing various advantages of the differentiable pro-
gramming paradigm discussed above at the same time.

The organization of this paper is as follows. In Sec. II, the
automatic differentiation of dominant eigensolver is studied in
a typical setting. The mechanisms that effectively separate the
desired information out of the full spectrum as well as support
taking arbitrarily order of derivatives are carefully explained.
For the reasons discussed in the third paragraph above, we will
mainly focus on reverse mode automatic differentiation unless
otherwise stated. In Sec. III, we demonstrate application of
the techniques through study of the ground state properties of
1D transverse field Ising model via two different approaches,
namely exact diagonalization and gradient-based optimization
of uniform matrix product states. The concluding remarks are
given in Sec. IV. Our code implementation is publicly avail-
able [27].
II. FORMULATIONS

For the sake of simplicity and clarity, let \( A \) be an \( N \)-dimensional real square matrix, and we are concerned with only one certain eigenvalue \( \lambda \) and corresponding left and right eigenvector \( l, r \) of \( A \), respectively. In other words, we have

\[
\vec{V}A = \lambda \vec{V}, \quad Ar = \lambda r, \quad \vec{V}^T r = 1.
\]

(3)

where we have imposed the conventional normalization condition. Note here that \( A \) is generally non-symmetric, and we only assume that \( A \) is diagonalizable and the desired eigenvalue \( \lambda \) is non-degenerate. Since the set of non-diagonalizable matrices has measure zero, these are not strong restrictions in practice. In reverse mode automatic differentiation, what we need is the adjoint relation \( \overline{A} = \overline{A}(\lambda, l, r) \), as discussed in Sec. 1. Below we will adopt two different approaches to this task, namely the adjoint method and the more “traditional” approach based on the full eigen-decomposition process, and explain the intimate relation between them.

A. The adjoint method

The adjoint method [28] is a general way of deriving backward pass of various computation processes. To demonstrate the basic ideas, consider a simple yet generic setting. Let \( \theta = (\theta_1, \cdots, \theta_p) \) be a \( p \)-dimensional input vector of parameters to be differentiated, and the output \( x = (x_1, \cdots, x_M)^T \) is an \( M \)-dimensional column vector. \( x \) is implicitly dependent on \( \theta \) through \( M \) (generally nonlinear) equations of the form \( f_i(x, \theta) = 0 \), where \( i \) ranges from 1 to \( M \).

To derive the adjoint relation in the framework of reverse mode automatic differentiation, one needs to compute the following vector-Jacobian product:

\[
\overline{\partial \mu} = \overline{x}^T \frac{\partial x}{\partial \theta_\mu}, \quad \forall \mu = 1, \cdots, P.
\]

(4)

where the \( M \)-dimensional vector \( \frac{\partial x}{\partial \theta_\mu} \) is determined by the set of equations

\[
\frac{\partial f}{\partial \theta_\mu} + \frac{\partial f}{\partial x} \frac{\partial x}{\partial \theta_\mu} = 0.
\]

(5)

where

\[
\frac{\partial f}{\partial \theta_\mu} = \begin{pmatrix} \frac{\partial f_1}{\partial \theta_\mu} \\ \vdots \\ \frac{\partial f_M}{\partial \theta_\mu} \end{pmatrix}, \quad \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x} \\ \vdots \\ \frac{\partial f_M}{\partial x} \end{pmatrix}.
\]

Assuming the matrix \( \frac{\partial f}{\partial x} \) is invertible, one can solve for \( \frac{\partial x}{\partial \theta_\mu} \) directly from Eq. (5), then substitute it back to Eq. (4) to obtain the adjoint relation:

\[
\overline{\partial \mu} = -\overline{x}^T \left( \frac{\partial f}{\partial x} \right)^{-1} \frac{\partial f}{\partial \theta_\mu}.
\]

(7)

where in the second line we have defined a column vector \( \eta \) determined by the so-called adjoint equation:

\[
\left( \frac{\partial f}{\partial x} \right)^T \eta = \overline{x}.
\]

(8)

In this way, we have also rearranged the order of matrix multiplication to avoid explicitly solving \( \frac{\partial x}{\partial \theta_\mu} \) appearing in Eq. (4). This rearrangement is the core idea of the adjoint method.

Specifically, in the settings of the dominant eigen-decomposition described above, the \( N \)-dimensional matrix \( A = A(\theta) \) depends on some parameters \( \theta \), and the output vector is effectively \( (2N + 1) \)-dimensional, including both the left/right eigenvectors \( l, r \) and the scalar eigenvalue \( \lambda \). The \( 2N + 1 \) equations \( f_i(l, r, \lambda, \theta) = 0 \) connecting the inputs and outputs are given by

\[
f_i(l, r, \lambda, \theta) = \begin{cases} (A - \lambda I)^T I, & i = 1, \cdots, N. \\ (A^T - \lambda I)^T r, & i = N + 1, \cdots, 2N. \\ r^T - 1, & i = 0. \end{cases}
\]

(9)

where the subscript \( i \) in an expression \( M \), denotes the \( i \)th column of the matrix \( M \), and the equation \( f_0(l, r, \lambda, \theta) = 0 \) imposes the normalization constraint.

Making use of Eq. (9), one can solve for \( \eta \) in Eq. (8), then substitute it back to Eq. (7) to obtain the desired expression of \( \overline{\theta}_l \) for the dominant eigensolver. The derivation is fairly straightforward, and we refer the reader to Appendix A for details. The final results are:

\[
\overline{\theta}_l = \overline{\lambda}^T \frac{\partial A}{\partial \theta}, \quad \frac{\partial A}{\partial \theta}, \quad \xi_l - \xi_r \frac{\partial A}{\partial \theta} r.
\]

(10)

where the vectors \( \xi_l \) and \( \xi_r \) satisfy the linear systems

\[
(A - \lambda I) \xi_l = (1 - r^T I), \quad r^T \xi_l = 0.
\]

(11a)

\[
(A^T - \lambda I) \xi_r = (1 - r^T \overline{r}), \quad r^T \xi_r = 0.
\]

(11b)

respectively. These linear systems are low-rank, in the sense that the coefficient matrices \( A - \lambda I \) and \( A^T - \lambda I \) are singular. Specifically, under our assumption that the eigenvalue \( \lambda \) is non-degenerate, they have rank \( N - 1 \). Nevertheless, the solution for \( \xi_l, \xi_r \) is unique, because the singular matrix \( A - \lambda I(A^T - \lambda I) \), when represented in the \( (N - 1) \)-dimensional subspace spanned by the \( N - 1 \) right(left) eigenvectors other than \( r(I) \), is effectively non-singular. See also the discussions in Appendix A.

Eq. (10) can be further simplified. In fact, we can “strip” the parameter \( \theta \) out of the primitive and obtain the neater expression for \( \overline{A} \) by taking account of the fact that \( \overline{\theta}_l = Tr(A^T \frac{\partial A}{\partial \theta}) \). This way, we finally write the adjoint relation of the dominant eigensolver as follows:

\[
\overline{A} = \overline{\lambda} lr^T - l \xi_l^T - \xi_r^T r^T.
\]

(12)

Fairly simple.

From Eqs. (11) and (12), one can see that the adjoint of \( A \) needs only the desired eigenvalue \( \lambda \) and corresponding eigenvectors \( l, r \) without explicit reference to the full spectrum. In
other words, we have successfully stripped out the information we want in the backward pass of dominant eigensolver, at the price of solving two somewhat nontrivial low-rank linear systems shown in Eq. (11). In a typical implementation, the forward pass can be accomplished by using Lanczos or other dominant eigen-decomposition algorithms, while the low-rank linear systems (11) involved in the backward pass can be solved efficiently using Krylov-based iterative algorithms such as biconjugate gradient and generalized minimal residual methods, among others. It’s worth noting that both the dominant eigensolver algorithms in the forward pass and the iterative linear system solvers in the backward pass do not need to know each individual entries of the matrix $A$; they only require the computation of matrix-vector products $Av$ with an arbitrary vector $v$. In many applications, this computation can be fairly efficient with the help of certain inner structures of $A$, even though the size of $A$ can be quite large.

**B. Special case: $A$ is symmetric**

In this section, we will briefly discuss the special and important case where the real matrix $A$ to be diagonalized is symmetric. This case is particularly relevant to many practical applications in quantum physics. There, all physical observables, including the Hamiltonian, are represented by Hermitian operators, thus also symmetric when all the matrix elements involved are real.

When $A$ is real symmetric, the desired left eigenvector is equal to the corresponding right eigenvector, that is, $l = r \equiv v$. To obtain the adjoint of $A$ as a function of $\lambda$ and $\bar{v}$ in this special case, one can imitate the derivation in Sec. II A based on the adjoint method and obtain

$$\bar{A} = (\bar{A}v - \xi)v^T, \quad \text{where } \xi \text{ satisfies}$$

$$(A - \lambda I)\xi = (1 - vv^T)\bar{v}, \quad v^T\xi = 0. \quad (13)$$

This result can also be easily obtained from the general formulas (11) and (12). To do this, simply let the adjoint $\bar{l}$ be zero, and $\bar{r}$ be equal to $\bar{v}$. The reason is that only the right eigenvector $r$ is needed for downstream calculations, while the left eigenvector $l$, which is equal to $r$ in this case, acts as a piece of redundant information that doesn’t affect the downstream results at all. It is then easy to see that the vector $\xi$ in Eq. (11a) vanishes, and the general formula (12) immediately reduces to the special form (13).

It’s instructive to furthermore inspect the physical implications of the adjoint relation (13). Let again the matrix $A$ depend on one certain parameter, say, $\theta$. Then the adjoint of $\theta$ reads [17]

$$\bar{\theta} \equiv \text{Tr}\left(A^T\frac{\partial A}{\partial \theta}\right)$$

$$= \bar{A}v^T\frac{\partial A}{\partial \theta}v - \xi^T\frac{\partial A}{\partial \theta}v. \quad (14)$$

Note the two terms correspond to dependence of the eigenvalue $\lambda$ and eigenvector $v$ on the parameter $\theta$, respectively.

In particular, if only the eigenvalue $\lambda$ is used for downstream computations, then the second term vanishes. For clarity, one could just consider the case where the loss $\mathcal{L} \equiv \lambda$. Thus we have $\bar{\lambda} = 1$, and Eq. (14) reduces to

$$\bar{\theta} \equiv \frac{\partial \lambda}{\partial \theta} = v^T\frac{\partial A}{\partial \theta}v. \quad (15)$$

This is the celebrated Hellmann-Feynman theorem [29], which is equivalent to the result of first order energy correction in perturbation theory. However, in the general case where the eigenvector also has nontrivial effect on the computation process, the second term in Eq. (14) is nonzero, and the formulation presented above turns out to be very useful.

**C. Relation with the full eigen-decomposition approach**

In Sec. II A, the automatic differentiation of dominant eigensolver has been presented straightforwardly in a full-spectrum-free form through the adjoint method. To figure out how this is achieved, it is instructive to change to another perspective by studying the relation between the adjoint method described above and the traditional approach based on full eigen-decomposition. The point is that we can wrap the process of full eigen-decomposition within the dominant one and utilize the results of the former formulation in the latter, as illustrated in Fig. 1.

![FIG. 1. Studying the automatic differentiation of dominant eigensolver for a general real matrix $A$ by wrapping within it the corresponding full eigen-decomposition process. The internal data nodes $D$, $U$ and $V$, which are the outputs of the full eigen-decomposition process, act as a link for deriving desired results from already known ones. See text for more details, especially Eq. (18) for how this is achieved in reverse mode automatic differentiation. Nevertheless, these inner structures are invisible if one treats the dominant eigensolver as a whole unit, which reflects the flexibility of the modular nature of differentiable programming paradigm.](image-url)

For clarity and without loss of generality, let $\lambda$ and $l$, $r$ be the “first” eigenvalue and corresponding left/right eigenvector.
of the $N$-dimensional matrix $A$, respectively. Recall the assumption that $A$ is diagonalizable, we can write $V^T A U = D$, where
\[
D = \begin{pmatrix}
\lambda_2 & & \\
& \ddots & \\
& & \lambda_N \\
\end{pmatrix},
\quad
U = \begin{pmatrix}
| & | & | \\
1 & r_2 & \cdots & r_N \\
| & | & | \\
\end{pmatrix}.
\]

Expanding the vectors actually they are identically the same
\[
V^T \equiv U^{-1} = \begin{pmatrix}
-f^1 & -f^2 & \cdots & -f^N \\
\end{pmatrix}.
\] (16)

That is, the columns of $U$ and rows of $U^{-1}$ correspond to the basis consisting of the $N$ right eigenvectors $(r, r_2, \cdots, r_N)$ and left eigenvectors $(l, l_2, \cdots, l_N)$, respectively.

In the framework of reverse mode automatic differentiation, the adjoint relation of the full eigen-decomposition process is pretty standard [17] and reads
\[
\bar{A} = V \left[ D \circ I + (U^T U - V^T V) \circ F \right] U^T, \tag{17}
\]
where $F$ is an anti-symmetric matrix with off-diagonal elements $F_{ij} = (\lambda_j - \lambda_i)^{-1}$ and $\circ$ denotes the Hadamard element-wise product. Here comes the key point. Since only $\lambda$, $l$ and $r$ will be used for downstream computations, the procedure of wrapping the process of full eigen-decomposition within the dominant one means that the adjoints of $D$, $U$ and $V$ should take the following form:
\[
\bar{D} \circ I = \begin{pmatrix}
\bar{A} & \\
0 & \\
& \ddots & \\
& & 0 \\
\end{pmatrix},
\quad
\bar{U} = \begin{pmatrix}
| & | & | \\
1 & r & \cdots & r \\
| & | & | \\
\end{pmatrix},
\quad
\bar{V} = \begin{pmatrix}
| & | & | \\
1 & l & \cdots & l \\
| & | & | \\
\end{pmatrix}. \tag{18}
\]

Substituting Eq. (18) into Eq. (17) yields
\[
\bar{A} = \bar{\lambda} r^T - \sum_{i=2}^{N} c_i^{(r)} l_i r^T - \sum_{i=2}^{N} c_i^{(l)} l_i r^T, \tag{19}
\]
where we have introduced the quantities $c_i^{(r)} \equiv \frac{1}{\lambda_i - \lambda_j} r_i^T$ and $c_i^{(l)} \equiv \frac{1}{\lambda_i - \lambda_j} l_i^T l_j, \forall i = 2, \cdots, N$. This formula looks quite similar to the earlier result (12) obtained by the adjoint method. Actually they are identically the same. This can be seen by expanding the vectors $\xi_l$ and $\xi_r$ in Eq. (11) in the complete basis $(r, r_2, \cdots, r_N)$ and $(l, l_2, \cdots, l_N)$, respectively. One can easily see that the quantities $c_i^{(l)}$ and $c_i^{(r)}$ defined above are exactly the linear combination coefficients of $\xi_l$ and $\xi_r$ in these two basis. In other words, we have
\[
\xi_l = \sum_{i=2}^{N} c_i^{(l)} r_i, \quad \xi_r = \sum_{i=2}^{N} c_i^{(r)} l_i. \tag{20}
\]

Plugging these relations back into Eq. (19) clearly reproduces the earlier result (12).

The observation above truly reveals the way in which the full spectrum information appearing explicitly in the original full eigen-decomposition approach can be effectively eliminated and replaced by the vectors $\xi_l$, $\xi_r$ characterized in Eq. (11). The fact that the final results of the two approaches are identically the same is not surprising, but the “native” representations are indeed different from a practical point of view. Specifically, the formulation based on the adjoint method clearly reveals the feasibility of constructing a valid dominant eigensolver primitive without any access to the full spectrum, while the approach based on full eigen-decomposition helps to furthermore clarify how this is achieved behind the scene.

D. Towards higher order derivatives

In this section, we study the possibility of performing higher order derivatives of the dominant eigensolver primitive. To do this, we have to investigate the backward pass of the low-rank linear system solvers described in Eq. (11), since this is the only non-trivial part in the backward pass of the primitive. It’s instructive to study the simpler full-rank case first, where the coefficient matrix is non-singular. Specifically, Let $x$ be the unique solution to the full-rank linear system $Ax = b$, where $A$ is a non-singular matrix and $b$ is an arbitrarily chosen vector. Since $A^{-1}$ exists, the derivation of the backward pass (i.e., the adjoint relation) is fairly straightforward. The final results are [17]:
\[
\bar{b} \text{ satisfies } \bar{A} \bar{b} = \bar{x}, \tag{21a}
\quad
\bar{A} = -b x^T. \tag{21b}
\]

One can see that the backward pass of full-rank linear system solver involves solving another full-rank linear system. This observation is insightful, and as we will see, the similar conclusion can be drawn for the low-rank case.

The derivation for the backward pass of low-rank linear system solver is more subtle. For current purposes, it suffices to consider the following settings. Let $A$ be an $N$-dimensional real (diagonalizable) matrix of rank $N-1$. This indicates that $A$ has $N-1$ (right) eigenvectors $v_2, \cdots, v_N$ of nonzero eigenvalues $\lambda_2, \cdots, \lambda_N$, respectively, other than a single (right) eigenvector $v$ with eigenvalue zero. For simplicity, we will restrict ourselves to the case where $A$ is symmetric, hence the left and right eigenvectors coincide. The derivation for the general case is pretty similar. Let $b$ be an arbitrary vector lying in the $(N-1)$-dimensional subspace spanned by $v_2, \cdots, v_N$, the goal of the computation process is the unique solution for $x$ of the following equations:
\[
A x = b, \quad v^T x = 0. \tag{22}
\]

These settings can fit properly into the backward pass of the low-rank linear system appearing in, say, Eq. (13), under the correspondence $A \rightarrow A - I l, x \rightarrow \xi_r, b \rightarrow (1 - vv^T) \bar{r}, v \rightarrow v$.

Rigorously speaking, the information about the eigenvector $v$ of eigenvalue zero is contained in the matrix $A$. However
this information is somewhat hard to extract directly, and in practice one finds it more convenient to treat $v$ as an independent input to the process. To derive the adjoint relations for the low-rank linear system solver under these settings, one way is to manually perform decomposition of relevant quantities into the two orthogonal subspaces spanned by $v$ alone and other $N-1$ eigenvectors of nonzero eigenvalues, respectively. This makes it more convenient to take advantage of the fact that $A$ is effectively invertible in the latter subspace. For more details, see Appendix B. The final results are:

\[ \tilde{b} \] satisfies
\[ A\tilde{b} = (1-vv^T)\tilde{x}, \quad vv^T\tilde{b} = 0. \tag{23a} \]
\[ \tilde{A} = -\tilde{b}x^T. \tag{23b} \]
\[ \tilde{v} = -xv^T\tilde{x}. \tag{23c} \]

Notice the high similarity between Eq. (23) and the corresponding results (21) for the full-rank case. Just as in the full-rank case, the backward pass of the low-rank linear system solver involves solving another low-rank linear system (23a) of the same kind. This observation is crucial and satisfying for the purposes of this paper. It implies that the backward pass of dominant eigensolver, which involves solving a low-rank linear system, can be conveniently differentiated itself by solving another low-rank linear system of the same kind, which in turn can be differentiated again, and so on. In other words, the formulation presented above allows us to compute principle arbitrarily higher order derivatives of the dominant eigen-decomposition process in the framework of (reverse mode) automatic differentiation.

III. APPLICATIONS

In this section, we demonstrate the use of the dominant eigensolver primitive in action by two examples. Our code implementation [27] is based on PyTorch [7], a deep learning library that supports reverse mode automatic differentiation through dynamic construction of computation graphs [30]. In addition, note that PyTorch supports computing high order derivatives and, thanks to the modular nature of differentiable programming, has good flexibility and extensibility by allowing users to customize their own primitives. These features turns out to be very convenient for the purposes of this work.

Both of the examples are concentrated on the spin-$\frac{1}{2}$ Transverse Field Ising Model (TFIM) on a 1-dimensional lattice. The Hamiltonian reads:

\[ H = -\sum_{i=0}^{N-1} (g\sigma_i^z + \sigma_i^x\sigma_{i+1}^x). \tag{24} \]

where $g$ is a non-negative parameter characterizing the strength of the transverse magnetic field. When $g = 0$, the model reduces to the “Ising limit”, and the spins in the ground state are perfectly aligned along the $z$ direction. When $g > 0$, on the other hand, the transverse field term will disrupt the magnetic order by introducing flipping of the spins. This model is a well-known prototype of the study of quantum phase transitions [31]. Specifically, there is a transition point at $g = 1$ in thermodynamic limit, where the energy gap between ground state and the lowest excited state, which characterizes the energy scale of fluctuations at zero temperature for a gapped Hamiltonian like Eq. (24), vanishes through a power law. Furthermore, this behavior can be characterized by a critical exponent, which usually turns out to be universal, that is, independent of most of the microscopic details of the system.

Below we give a brief study of the model through the approach of exact diagonalization and variational optimization of matrix product states, respectively, using the formulations developed in Sec. II.

A. Identify the transition point by differentiating through exact diagonalization

We first study the ground state properties of the model through exact diagonalization, specifically the behavior near the transition point $g = 1$. As indicated above, one of the main characterization of quantum phase transition (of gapped systems) is the vanishing of the gap between ground state and the lowest excited state in thermodynamic limit. There has been various kind of quantities proposed in practice to indicate the emergence of such behavior, and we have chosen two of them for the purpose of demonstration.

1. 2nd order derivative of the ground state energy

Computing the second order derivative of ground state energy (per site) $\frac{\partial^2 E_0}{\partial g^2}$ with respect to the parameter $g$ is a convenient way to characterize the quantum phase transition. In fact, the vanishing gap at the transition point implies the divergence of this quantity, which can be easily seen from the expression of 2nd order perturbation theory as follows:

\[ \frac{\partial^2 E_0}{\partial g^2} = \sum_{n \neq 0} \frac{|\langle \psi_0 | h' | \psi_0 \rangle|^2}{E_0 - E_n}. \tag{25} \]

where

\[ h' \equiv \frac{1}{N} \frac{\partial H(g)}{\partial g} = -\frac{1}{N} \sum_{i=0}^{N-1} \sigma_i^z. \tag{26} \]

is the “perturbation Hamiltonian (per site)”, and $H(g) | \psi_0(g) \rangle = NE_0(g) | \psi_0(g) \rangle$, with $n = 0$ corresponding to the ground state.

Note that as discussed in Sec. II B, the computation of the 1st order derivative of $E_0$ is more or less trivial and essentially equivalent to the Hellmann-Feynman theorem. Specifically, we have in current case

\[ \frac{\partial E_0}{\partial g} = \langle \psi_0 | h' | \psi_0 \rangle. \tag{27} \]

On the other hand, it is the computation of the 2nd (and even higher) order derivative of $E_0$ that truly reveals the value of
the machinery of automatic differentiation developed above. In fact, the 1st order derivative of $E_0$ has explicit dependence on the eigenvector $|\psi_0\rangle$ as shown in Eq. (27), and this implies that the 2nd order derivative of $E_0$ has to be computed through direct differentiation onto $|\psi_0\rangle$. This is exactly when the formulation based on automatic differentiation could help to avoid the costly full eigen-decomposition as explicitly desired in Eq. (25). See also the discussion in the last paragraph of Sec. II B.

![Graph showing the 2nd derivative respect to parameter $g$ of the ground state energy per site $E_0$ of 1D TFIM, for three values of the lattice size number $N$, calculated through automatic differentiation of the exact (dominant) diagonalization.](image)

**Fig. 2.** The 2nd derivative respect to parameter $g$ of the ground state energy per site $E_0$ of 1D TFIM, for three values of the lattice size $N$, calculated through automatic differentiation of the exact (dominant) diagonalization.

Fig. 2 shows the 2nd derivative $\frac{\partial^2 E_0}{\partial g^2}$ for three distinct lattice sizes $N$. Note that when $N = 20$, the dimension of the Hilbert space involved is $2^{20} \approx 1000000$, and the full eigen-decomposition of the Hamiltonian has become extremely challenging in practice [32]. One can see that the 2nd derivative of ground state energy is negative, which is a well-known fact in perturbation theory (See, e.g., Eq. (25)). In addition, the peak near the transition point $g = 1$ becomes more and more evident as $N$ increases, which agrees with the physical characterization of the phase transition described above.

### 2. Fidelity susceptibility

Another indicator of the quantum phase transition is the fidelity susceptibility $\chi_F$ [33], whose origin can be traced back to the field of quantum information science [34]. To motivate this concept, note that there has emerged various kinds of classical and quantum phase transitions that go beyond the traditional Ginzburg-Landau-Wilson formulations based on the existence of some local order parameters. To name an example, topological phase transitions [35–37] do not have any local order parameter on either side of the phase transition. In view of this, new ideas and theoretical tools are needed to characterize these exotic phases and transitions among them, and various concepts in other fields, such as the quantum fidelity [38] and entanglement entropy [39] in quantum information science, has been borrowed and proved useful.

For current purposes, the concept of quantum fidelity is defined as the overlap of ground states of the Hamiltonian for two different parameters. More specifically, we have

$$F(g, \epsilon) = \langle \psi_0(g) | \psi_0(g + \epsilon) \rangle.$$  \hspace{1cm} (28)

Suppose the value of the parameter $g$ is fixed. When $\epsilon = 0$, the two ground states coincide, and the quantum fidelity as a function of the “distance” $\epsilon$ clearly reaches the maximum value 1. Previous work [40] has suggested that two ground states lying at different sides of a phase transition point is qualitatively different, thus have significantly smaller overlap. This means that when $g$ is near the transition point, the quantum fidelity as the function of $\epsilon$ has more drastic changes at the maximum $\epsilon = 0$.

The concept of fidelity susceptibility $\chi_F$ is then proposed as a quantitative measure of this rate of change at $\epsilon = 0$ for various values of the parameter $g$. Specifically, it is defined as

$$\chi_F = \frac{\partial^2 \ln F(g, \epsilon)}{\partial \epsilon^2} \bigg|_{\epsilon=0}. $$  \hspace{1cm} (29)

It can be seen from the argument above that $\chi_F$ may exhibit a maximum or even diverge in thermodynamic limit at the transition point, which has been demonstrated by various works [41–44] and furthermore used for the detection and characterization of topological [45–47] and other kinds of phase transitions.

Despite its high theoretical values, the practical calculation of the fidelity susceptibility has become a difficult task in many situations, and many previous studies have thus been restricted to the case where the accurate ground state overlap can be computed via analytic results, exact diagonalization or Density Matrix Renormalization Group (DMRG) method. Even within the framework of exact diagonalization, the accurate computation of the fidelity susceptibility is still annoying, which is largely due to the appearance of the 2nd derivative in Eq. (29). To make this statement clearer, one can do some simple manipulation on the original definition (29) of $\chi_F$ and obtain an equivalent expression as follows:

$$\chi_F = \left[ \frac{\partial \psi_0}{\partial g} \right] \frac{\partial^2 \psi_0}{\partial g^2} - \left( \frac{\partial \psi_0}{\partial g} \right)^2 \langle \psi_0 | \frac{\partial^2 \psi_0}{\partial g^2} | \psi_0 \rangle. $$  \hspace{1cm} (30)

where the ground state $|\psi_0(g)\rangle$ has assumed to be normalized. The differential $\left( \frac{\partial \psi_0}{\partial g} \right)$ is difficult to handle properly, as already indicated by Eq. (2) and the discussions therein. The most natural approach is certainly through the perturbation theory, which yields

$$\chi_F = \sum_{n \neq 0} \frac{|\langle \psi_n | h' | \psi_0 \rangle|^2}{(E_0 - E_n)^2}.$$  \hspace{1cm} (31)

Note that the second term in Eq. (30) vanishes identically due to the orthogonality of $|\psi_0\rangle$ and $\left( \frac{\partial \psi_0}{\partial g} \right)$ from the result of perturbation theory. Eq. (31) can be used as a benchmark against other methods based on certain approximations. However,
it becomes intractable fairly quickly as the lattice size $N$ increases, due to the need of the full spectrum through highly expensive computation of full eigen-decomposition.

Comparing Eq. (31) with (25) as well as the discussions therein, it is evident that the difficulties encountered here are pretty similar to those when attempting to compute the 2nd derivative of the ground state energy in Sec. III A 1. Again, the formulation based on automatic differentiation provides a unifying and satisfactory solution. Specifically, one can directly implement the 2nd order derivative in the original definition (29) in the framework of backward mode automatic differentiation using PyTorch, by transforming it to a slightly different form:

$$\chi_F = -\frac{\partial^2}{\partial g'^2} \ln \left| \langle \psi_0 (g') | \psi_0 (g) \rangle \right|_{g=g}$$  \hspace{1cm} (32)

Fig. 3 shows the computation graph of this process. This graph has to be differentiated twice to obtain the fidelity susceptibility as indicated in Eq. (32). Note the partial derivative in (32) operates on only one of the two ground state vectors involved in the overlap. This implies that a detached duplicate of $|\psi_0\rangle$ has to be created, in the sense that the detached data node in a computation process is no longer treated as dependent on the inputs, as demonstrated by the dashed arrow in Fig. 3. This mechanism of detaching is well supported by PyTorch, which makes the implementation fairly straightforward and easy.

![Fig. 3. The graph of computing the fidelity susceptibility $\chi_F$ in the framework of backward mode automatic differentiation, using Eq. (32). The color indicates the part of various data nodes that should be regarded as dependent on the input $g$ and differentiated upon when backward through the computation process. Note the presence of a detached duplicate of the ground state $|\psi_0\rangle$, which is necessary to obtain the desired result.](image)

Since the output node appearing in Fig. 3 has explicit dependence on the eigenvector $|\psi_0\rangle$, the 1st differentiation of the process involves solving low-rank linear systems of the kind in, say, Eq. (13). This in turn makes the 2nd differentiation involve the backward pass of this linear system solver, which typically involves the same kind of linear system solver again, as shown in Sec. II D. To sum up, this example makes full use of the machinery developed in Sec. II, including the mechanism of computing higher order derivatives.

![Fig. 4. The fidelity susceptibility of 1D TFIM, for three values of the lattice size number $N$, calculated through automatic differentiation of the exact (dominant) diagonalization.](image)

Matrix Product States (MPS) originates from the celebrated DMRG method [48, 49] and acts as the underlying variational ansatz of the formalism. MPS is a typical and well-known category of the rich family of Tensor Network (TN) states, which encode the correlation and entanglement of many-body states by virtual bonds connecting the microscopic degrees of freedom living on different sites. Based on such structures, TN states can provide a suitable parameterization of the low-energy states of various quantum many-body systems, with the theoretical guarantee relevant to the area law scaling of the entanglement entropy [39]. In particular, the class of MPS has proved to be very useful for studying the ground state of 1D strongly correlated systems with local interactions [50, 51].

There has exists several schemes for the variational optimization of MPS states both for the finite and infinite, uniform lattice sites. Typical examples include various variations of the original DMRG algorithm [25], infinite time evolving block decimation (iTEBD) based on Trotter decomposition of the evolution operator [52] and the recent variational uniform MPS (VUMPS) algorithm based on the concept of MPS tangent space [53, 54]. Despite the maturity and successful applications of these methods in 1D systems, they usually lack generality and extensibility in some sense. For example, the
formulation of VUMPS algorithm relies heavily on specific properties characteristic of the MPS state, and the analytic derivation of the gradients is rather cumbersome and error prone due to complicated and highly nonlinear dependence of the variational energy on input parameters. Maybe the most obvious consequence regarding this perspective is the well-known difficulty of generalization of these methods to two or higher dimensional systems.

The differentiable programming paradigm provides a natural solution in this respect. Owning to the various inherent advantages mentioned in Sec. I, differentiable programming serves as a suitable framework for various tensor network applications, and the practical implementation is usually more generic and free of specialized details relevant to certain settings. Recently [15], the technique of differentiable programming has been successfully used for studying ground state of the 2D square lattice antiferromagnetic Heisenberg model by gradient-base optimization of infinite projected entangled pair state (iPEPS). This clearly reveals its capability beyond other traditional tensor network methods.

Motivated by these arguments, we will calculate the ground state energy of 1D TFIM in thermodynamic limit by gradient-based variational optimization of uniform MPS, using the formulations developed in Sec. II. The variational ansatz of the ground state reads

$$|\psi_0\rangle = \cdots A A A A \cdots .$$

(33)

The parameter $A$ of the MPS is a rank-3 tensor of shape $d \times D \times D$, where $d$ is the dimension of the local, physical Hilbert space at each site (two in current case), and $D$ is the virtual bond dimension. Note the model involves only nearest-neighbor interactions, which implies that the Hamiltonian (24) can be written in the form

$$H = \sum_{i=0}^{N-1} h_{i,i+1}.$$  

(34)

where $h_{i,j}$ is a “local Hamiltonian” that acts only on the two spin degrees of freedom at site $i$ and $j$. In current case, for example, $h$ can be chosen to have the following symmetric form:

$$h_{i,j} = -\frac{g}{2} (\sigma_i^x + \sigma_j^x) - \sigma_i^z \sigma_j^z.$$  

(35)

By simple algebraic manipulations, one can express the energy expectation value per site as follows:

$$\frac{1}{N} \langle \psi_0 | H | \psi_0 \rangle = \frac{1}{\lambda^2}.$$  

(36)

where $h_{i,j} = \langle s_i^r s_j^l | h_{i,j} | s_i s_j \rangle$ is the tensor representation of $h_{i,j}$. $\lambda$ is the eigenvalue of largest amplitude of the transfer matrix $A$, and $l$ and $r$ are the corresponding left and right eigenvectors, respectively. More concretely, we have

$$A A A A = \lambda r l A A A A.$$  

(37)

Notice that the expectation energy Eq. (36) has nonlinear dependence on the parameter tensor $A$, both explicitly and implicitly via the eigenvalue $\lambda$ and eigenvectors $l$, $r$. We will compute the gradient of Eq. (36) with respect to $A$ using automatic differentiation, which can automatically take care of all the complicated ways of dependence without any laborious human efforts of deriving gradients analytically.

Since the computation process involves the dominant eigen-decomposition (37) of the transfer matrix and, notably, the output clearly has nontrivial dependence on the eigenvectors, the formulation developed in Sec. II can thus be efficiently exploited. For current purposes, we doesn’t impose any additional constraints on $A$ except assuming that it is real. Thus, the $D^2 \times D^2$ transfer matrix is generally not symmetric. Nevertheless, it is well-known that its largest-amplitude eigenvalue $\lambda$ is always real, positive and non-degenerate [55], which meets the presupposition of the formulations in Sec. II.

In practice, the parameter tensor $A$ is initialized with random Gaussian variables, and the optimization can be accomplished using a quasi-Newton L-BFGS algorithm [56] with automatically computed gradients. Fig. 5 shows the error of the ground state energy (per site) $E_0$ for several values of the parameter $g$ near the transition point $g = 1$ and various bond dimensions $D$, relative to the analytic result obtained through Jordan-Wigner transformation [31]:

$$E_0(g) = -\frac{1}{2\pi} \int_0^{2\pi} dk \sqrt{g^2 - 2g \cos k + 1}.$$  

(38)

For each point in the figure, the convergence can be quickly reached after several hundreds of forward and backward pass, and the result is fairly accurate with a relative error of at most $10^{-5}$. Due to high complexity of the optimization landscape, it is generally very hard for a given optimization algorithm to reach machine precision (i.e., $\sim 10^{-16}$) in a limited number of iteration steps. It is for the same reason that Fig. 5 looks imperfect and full of bumps and hollows. Nevertheless, the results shown here are still satisfactory for the purpose of
demonstration. Notice that as one approaches the transition point \( g = 1 \), it becomes more difficult to reach a certain level of accuracy. This phenomenon is typical and also arises in many other kinds of computation approaches, such as evaluating the integral Eq. (38) and various quantum Monte Carlo methods.

![Figure 5](image)

**FIG. 5.** The relative error of the ground state energy per site \( E_0 \) of 1D TFIM for different values of the parameter \( g \) and bond dimension \( D \), compared to the analytic result Eq. (38).

In Fig. 5, we perform the optimization for bond dimension \( D \) up to 100, in which case the transfer matrix is of size \( 10000 \times 10000 \) and the approach of full eigen-decomposition has become very slow. We also note that the differentiable approach adopted in this section is essentially equivalent to the VUMPS algorithm developed in Refs. [53, 54]. However, the approach here is clearly more straightforward, generic and free of problem-specific stuffs such as various properties characteristic of the MPS states. It also removes the need of manually deriving gradients, which has become a rather tedious task in this application.

**IV. DISCUSSIONS**

In this paper, the (reverse mode) automatic differentiation of dominant eigensolver is illustrated through two different yet equivalent approaches, namely the adjoint method and one based on the full eigen-decomposition process. In particular, the mechanism that effectively strips the desire information out of the full spectrum is carefully explained. In this respect, the former approach yields full-spectrum-free formulas more directly, while the results of the latter approach still have explicit dependence on the full spectrum. On the other hand, the latter approach reveals the modular nature of differentiable programming paradigm and is more routinely, while the former approach typically requires some specialized mathematical understanding of the primitive. In view of these arguments, the two approaches are complementary to each other and can be used for double checking in studying the automatic differentiation of certain computation processes.

In Sec. II, we have taken into account only one eigenvalue and corresponding eigenvectors. The results presented there can be very easily generalized to the case of multiple eigenvalues and eigenvectors. In fact, the eigenvalues and corresponding eigenvectors are determined by the matrix \( A \) in a totally independent way, implying that the adjoint of \( A \) can be obtained simply by adding the contributions from them together, each of which has the same form shown in Eqs. (11) and (12). The case of multiple eigenvalues and eigenvectors can be useful in many problems, such as Hamiltonian engineering to reproduce given low-energy spectrum [57] and various tensor network applications [15].

Furthermore, the formulations presented in this work can be readily extended to other similar computation processes. One typical and important example is the truncated SVD, in which, as opposed to full SVD, only a small proportion of singular values and corresponding singular vectors are desired. In practice, truncated SVD is widely adopted in various tensor network calculations [24, 25], such as the tensor renormalization group [58] and corner transfer matrix renormalization group methods [59]. From the mathematical point of view, on the other hand, the truncated SVD of a real matrix \( A \) has intimate relation with the dominant eigen-decomposition of the symmetric and positive semi-definite matrices \( AA^T \) and \( A^TA \). With this in mind, it turns out that one can exploits both of the approaches presented in Sec. II in a similar way, and derive the automatic differentiation of truncated SVD in a full-spectrum-free form. This will bring significant performance improvement over the traditional approach involving full SVD, such as the one implemented in Ref. [15]. Note that similar to the discussions in the last paragraph, one can derive the relevant backpropagation formulas assuming only one singular value and corresponding singular vectors are desired. The generalization to the multiple case is then pretty straightforward.

**ACKNOWLEDGMENTS**

The authors thank Shuo-Hui Li, Yue-Shui Zhang and Hai-Jun Liao for useful discussions. This work is supported by the National Natural Science Foundation of China under the Grant No. 11774398 and the Ministry of Science and Technology of China under the Grant No. 2016YFA0300603 and 2016YFA0302400.

**Appendix A: Derivation of the backward pass of dominant eigensolver using the adjoint method**

In this appendix, we present the derivation details of the backward pass of dominant eigensolver using the adjoint method introduced in Sec. II A. Under the settings of the dominant eigen-decomposition process described therein, one can make the correspondence with the generic notations appearing in Eqs. (7) and (8) as follows:
where the concrete form (9) of the various equations $f_i(t,r,\lambda,\theta) = 0$ has been used.

The original adjoint equation (8) reduces to the following set of equations:

\[
\begin{align*}
(A - \lambda I) \eta_l & + \eta_r \mathbf{r} = \tilde{\mathbf{I}}. \\
(A^T - \lambda I) \eta_r & + \eta_l \mathbf{I} = \tilde{\mathbf{r}}.
\end{align*}
\]  

(A2a)

(A2b)

As the final step, the generic equation (7) of the adjoint $\overline{\phi}_\mu$ of the parameters reduces to

\[
\overline{\phi}_\mu = -T^T \frac{\partial A}{\partial \mu} \eta_l - \eta_r^T \frac{\partial A}{\partial \mu} \mathbf{r} = A I - \mathbf{L} \mathbf{A}^T \mathbf{r} - I^T \frac{\partial A}{\partial \mu} \xi_l - \xi_r^T \frac{\partial A}{\partial \mu} \mathbf{r}. \tag{A8}
\]

where we have used the correspondence (A1) and Eq. (A7). This is precisely the result shown in Eq. (10).

### Appendix B: Derivation of the backward pass of low-rank symmetric linear system solver

In this appendix, we present the derivation details of the backward pass of low-rank symmetric linear system solver (22), under the setting described therein. Recall that $(v_2, \cdots, v_N)$ constitutes the $N-1$ eigenvectors of the real symmetric matrix $A$ with nonzero eigenvalues $\lambda_2, \cdots, \lambda_N$, respectively, other than a single eigenvector $v$ with eigenvalue zero. We introduce the following notations:

\[
D \equiv \begin{pmatrix} \lambda_2 \\ \vdots \\ \lambda_N \end{pmatrix}, \quad U \equiv \begin{pmatrix} v_2 & \cdots & v_N \end{pmatrix}. \tag{B1}
\]

Note that $D$ is non-singular, since all of its diagonal elements are nonzero. It’s not hard to see that they obey the following relations:

\[
A = UDU^T, \quad \tag{B2a}
\]

\[
U^TU = I_{N-1}, \quad \tag{B2b}
\]

\[
UU^T = I_N - vv^T. \quad \tag{B2c}
\]

From the conditions (22), we have

\[
Adx = db - dA \mathbf{x}, \quad \tag{B3a}
\]

\[
dv^T \mathbf{x} + v^T d\mathbf{x} = 0. \quad \tag{B3b}
\]

Making use of the completeness relation (B2c), one could expand $d\mathbf{x}$ in the complete basis $(v, v_2, \cdots, v_N)$ and get

\[
d\mathbf{x} = UU^T d\mathbf{x} + vv^T d\mathbf{x}. \quad \tag{B4}
\]
Basically, the conditions (B3a) and (B3b) completely determines the components of $dx$ in the two orthogonal subspaces spanned by $(v_2, \cdots, v_N)$ and $v$ alone, respectively. Making use of the various relations in Eq. (B2), it is easy to obtain

$$dx = UD^{-1}U^T(db - dA)x - vRx^Tdv. \quad (B5)$$

Comparing this result with the standard formula

$$dL = \bar{x}^Tdx = Tr\left(\bar{A}^TdA\right) + \bar{b}^Tdb + \bar{v}^Tdv. \quad (B6)$$

one can immediately obtain the adjoint relations as follows:

$$\bar{b} = UD^{-1}U^T\bar{x}, \quad (B7a)$$

$$\bar{A} = -UD^{-1}U^T\bar{x}x^T, \quad (B7b)$$

$$\bar{v} = -vx^T\bar{x}. \quad (B7c)$$

Notice the explicit presence of the matrices $D$ and $U$, which contain the unknown and not desired information about the full spectrum of $A$. This is the price we have paid in order to conveniently make use of the fact that $A$ is effectively invertible in the subspace spanned by $(v_2, \cdots, v_N)$, which is implied by the presence of $D^{-1}$ in the equations above.

To get rid of $D$ and $U$, note that the current situation is pretty similar to that we have encountered in Sec. II C when deriving the backward pass of dominant eigensolver by using the results of full eigen-decomposition. Inspired by the discussions therein, we can just multiply both sides of Eq. (B7a) by $A$ and get

$$A\bar{b} = (1 - vv^T)\bar{x}. \quad (B8)$$

where we have used the relations (B2) again. This equation is not completely equivalent to the original condition (B7a) in the sense that it just characterizes the components of $\bar{b}$ in the $(N - 1)$-dimensional subspace spanned by $(v_2, \cdots, v_N)$. To remedy this, just note that $\bar{b}$ actually has zero component along the direction of the eigenvector $v$, which can be seen from the right-hand side of (B7a) directly. All that being said, we thus obtain the final adjoint relations for the low-rank symmetric linear system solver as follows, in a full-spectrum-free form:

$$\bar{b}$$ satisfies $A\bar{b} = (1 - vv^T)\bar{x}, \quad vv^T\bar{b} = 0. \quad (B9a)$

$$\bar{A} = -\bar{b}x^T. \quad (B9b)$$

$$\bar{v} = -vx^T\bar{x}. \quad (B9c)$$

This is precisely the results shown in Eq. (23).

[1] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, Automatic differentiation of algorithms, Journal of Computational and Applied Mathematics 124, 171 (2000), numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[2] D. Maclaurin, D. Duvenaud, and R. Adams, Gradient-based hyperparameter optimization through reversible learning, in International Conference on Machine Learning (2015) pp. 2113–2122.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, nature 323, 533 (1986).

[4] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of machine learning research 18 (2018).

[5] See https://github.com/tensorflow/tensorflow.

[6] See https://github.com/HIPS/autograd.

[7] See https://github.com/pytorch/pytorch.

[8] See https://github.com/google/jax.

[9] See https://github.com/FluxML/Zygote.jl.

[10] H. Jirari, Optimal control approach to dynamical suppression of decoherence of a qubit, EPL (Europhysics Letters) 87, 40003 (2009).

[11] N. Leung, M. Abdelhafez, J. Koch, and D. Schuster, Speedup for quantum optimal control from automatic differentiation based on graphics processing units, Phys. Rev. A 95, 042318 (2017).

[12] T. Tamayo-Mendoza, C. Kreisbeck, R. Lindh, and A. Aspuru-Guzik, Automatic differentiation in quantum chemistry with applications to fully variational hartreefock, ACS Central Science 4, 559 (2018), https://doi.org/10.1021/acscentsci.7b00586.

[13] U. Ekstrom, L. Visscher, R. Bast, A. J. Thorvaldsen, and K. Ruud, Arbitrary-order density functional response theory from automatic differentiation, Journal of chemical theory and computation 6, 1971 (2010).

[14] R. Steiger, C. H. Bischof, B. Lang, and W. Thiel, Using automatic differentiation to compute derivatives for a quantum-chemical computer program, Future Generation Computer Systems 21, 1324 (2005).

[15] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, Differentiable programming tensor networks, Phys. Rev. X 9, 031041 (2019).

[16] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, Neural ordinary differential equations, Advances in Neural Information Processing Systems (2018).

[17] M. Giles, An extended collection of matrix derivative results for forward and reverse mode automatic differentiation, (2008).

[18] M. W. Seeger, A. Hetzel, Z. Dai, and N. D. Lawrence, Auto-differentiating linear algebra, ArXiv abs/1710.08717 (2017).

[19] L. Huang, D. Yang, B. Lang, and J. Deng, Decorrelated batch normalization, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018) pp. 791–800.

[20] C. Ionescu, O. Vantzos, and C. Sminchisescu, Training deep networks with structured layers by matrix backpropagation, arXiv preprint arXiv:1509.07838 (2015).

[21] Z. Dang, K. Moo Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann, Eigendecomposition-free training of deep networks with zero eigenvalue-based losses, in Proceedings of the European Conference on Computer Vision (ECCV) (2018) pp. 768–783.

[22] A. Zanfir and C. Sminchisescu, Deep learning of graph matching, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018).

[23] W. Wang, Z. Dang, Y. Hu, P. Fua, and M. Salzmann,
