Article

Quantum Perturbation Theory Using Tensor Cores and a Deep Neural Network

Joshua Finkelstein,* Emanuel H. Rubensson,* Susan M. Mniszewski, Christian F. A. Negre, and Anders M. N. Niklasson*

Cite This: *J. Chem. Theory Comput.* 2022, 18, 4255−4268

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Time-independent quantum response calculations are performed using Tensor cores. This is achieved by mapping density matrix perturbation theory onto the computational structure of a deep neural network. The main computational cost of each deep layer is dominated by tensor contractions, i.e., dense matrix−matrix multiplications, in mixed-precision arithmetics, which achieves close to peak performance. Quantum response calculations are demonstrated and analyzed using self-consistent charge density-functional tight-binding theory as well as coupled-perturbed Hartree−Fock theory. For linear response calculations, a novel parameter-free convergence criterion is presented that is well-suited for numerically noisy low-precision floating point operations and we demonstrate a peak performance of almost 200 Tflops using the Tensor cores of two Nvidia A100 GPUs.
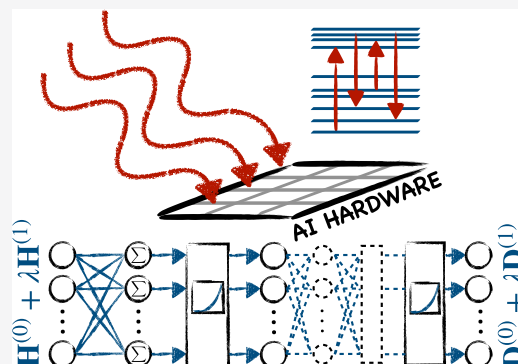

1. INTRODUCTION

A material's properties are in general characterized by its response to perturbations. Quantum perturbation theory and response calculations are therefore critical to our understanding of materials. Rayleigh−Schrödinger perturbation theory and Green's function approaches are commonly used in response calculations,[1−10] but they require explicit knowledge of the eigenstates or complex contour evaluations over residues, which can be a major hindrance for fast calculations. A less frequently used method is density matrix perturbation theory (DMPT), which is well-suited for time-independent response calculations. DMPT was pioneered by McWeeny in the 1960s[11] but was reintroduced in a new form based on recursive Fermi-operator expansion methods in the early 2000s.[12−19] This new form of DMPT represents a surprisingly simple methodology for calculating time-independent quantum response properties at any order. Costly summations and evaluations of eigenstates, as in Rayleigh−Schrödinger perturbation theory, or the summations and evaluations of complex residues, as in Green's function methods, are replaced by matrix−matrix multiplications of density matrix response terms and perturbations in the Hamiltonian, leading to rapidly converging recursive expansions,[12,18−20] which are well-adapted for modern hybrid and high-performance computing.

In this article, we explore the use of Nvidia Tensor cores[21] for increasing the performance of time-independent DMPT calculations. Tensor cores, and the related Tensor Processing Units (TPUs)[22] by Google and Matrix cores[23] from AMD, are a relatively new type of accelerated hardware developed to address the explosion in popularity of machine learning and

artificial intelligence (AI) algorithms. These specialized hardware units are designed to be extraordinarily efficient at dense tensor contractions, i.e., matrix−matrix multiplications, in low floating point (mixed) precision, which is the dominating operation required in deep neural network (DNN) models. To tap into the raw computational power of Tensor core units and other similarly accelerated hardware, for more general scientific applications, we must therefore develop stable computational algorithms that can take advantage of fast matrix−matrix multiplications and that are robust to low precision floating point operations. We achieve this by mapping the recursive DMPT-based response calculations onto the computational structure of a generalized deep neural network (DNN). This deep neural network density matrix perturbation theory (DNN-DMPT) framework is ideally suited for Tensor core computation. Matrix−matrix multiplications, which appear in the activation functions of our generalized DMPT network, dominate the cost of the DMPT-based response calculations and are naturally carried out on the Tensor core hardware. The multiplications can be performed in mixed-precision arithmetics using a dual half-precision matrix representation to enhance the numerical accuracy.[24] We also derive a robust stopping criterion that determines at what
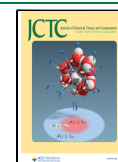
deep layer the approximate density matrix response has converged under the numerically noisy conditions of the low precision.

DMPT and related approaches to time-independent quantum response theory can be designed to take advantage of matrix sparsity. This makes it possible to achieve a reduced complexity scaling of the computational cost in quantum response calculations as a function of system size,[13,15,16,18,25,26] which is suitable for the study of large systems. However, in this article, we describe a dense matrix algebra approach, extending previous work,[12,17,24] which can utilize the substantial computing power offered by the specialized Tensor core platform. In this way, we show how to achieve high-performance response calculations of 120 Tflops on the Tensor cores of a single Nvidia A100 GPU. In this article, a combined multiplication and addition are counted as a single floating-point operation. Furthermore, by taking advantage of the computational structure of our DNN-DMPT framework, we also develop a multi-GPU-based approach. We demonstrate a performance of almost 200 Tflops using two sets of Tensor cores on two separate Nvidia A100 GPUs for a first-order density matrix response calculation.

The development of accelerated quantum response calculations presented in this article continues a currently ongoing theme of using Tensor cores (or other similar machine learning inspired hardware) for more general scientific applications.[24,27−34] This work is similar in spirit to the transition from CPUs to GPUs for scientific computations that started over a decade ago[35−47] and, in some sense, represents the next phase of a Darwinian-like computational evolution driven by new hardware environments. Currently, seven of the ten most powerful computers in the world utilize chips built with Tensor core accelerators, and GPU-accelerated architectures are common among the top 500 supercomputers.[48] Although this article is focused on response calculations using Tensor cores, our approach is general and should also apply to other specialized AI hardware platforms that are designed for high-performance matrix−matrix multiplications with low precision floating point operations, such as AMD's Matrix cores or Google's TPUs.

In our presentation of time-independent quantum response calculations with Tensor cores, we first (in Section 2) discuss some background on Tensor cores, electronic structure theory, and the density matrix formalism and how the unperturbed density matrix can be calculated with a deep neural network.[24] In Section 3, we present our approach to time-independent density matrix perturbation theory, and in Section 4, how we can formulate the response calculations through a deep neural network. In Section 4, we also derive a robust parameter-free stopping criterion for the layered network, which is of critical importance for the numerically noisy environment. In Section 5, we show some examples of density matrix linear response calculations using our deep neural network. We analyze the numerical accuracy and the performance. These examples highlight the utility and efficiency of our framework. We conclude with a brief summary and some final remarks.

2. DEEP NEURAL NETWORK APPROXIMATION OF THE ELECTRONIC STRUCTURE USING TENSOR CORES

Tensor cores can be used to calculate the electronic structure of a molecular system using the computational framework of a convolutional deep neural network, where the electronic structure, as in Hartree−Fock or density functional theory,[9,49−53] is represented by an effective single-particle density matrix.[54] This technique, described in the subsections below, forms the basis of our approach to quantum response calculations using DMPT. These response calculations can then similarly be formulated within the framework of a deep neural network.

2.1. Tensor Cores. A Tensor core is a specialized compute unit designed by Nvidia that exclusively performs a single dense matrix−matrix multiplication each GPU clock cycle. A schematic drawing of a GPU including four Tensor core units is shown in Figure 1. In our local cluster environment, an A100
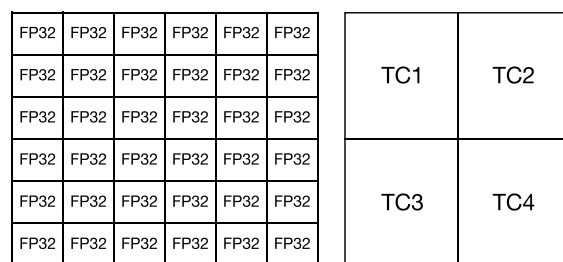

A100 GPU SM

FP32	FP32	FP32	FP32	FP32	FP32			
FP32	FP32	FP32	FP32	FP32	FP32		TC1	TC2
FP32	FP32	FP32	FP32	FP32	FP32			
FP32	FP32	FP32	FP32	FP32	FP32			
FP32	FP32	FP32	FP32	FP32	FP32		TC3	TC4
FP32	FP32	FP32	FP32	FP32	FP32			

Figure 1. Schematic drawing of an Nvidia A100 GPU streaming multiprocessor (SM) with 4 Tensor core (TC) units and 64 FP32 CUDA cores. Each A100 GPU on our local cluster has 108 SM's. INT and FP64 cores are not displayed.

GPU has 432 Tensor cores and each A100 Tensor core can compute a 4×8 matrix times an 8×8 matrix per clock cycle. Nvidia makes Tensor cores available through low-level CUDA API commands to individual blocks of 16×16 matrices and additionally through more high-level optimized CUDA libraries such as cuBLAS,[55] which is what is used in this article. Since each A100 GPU has a maximum clock rate of 1.41 GHz, this equates to a peak theoretical flop rate of approximately 156 Tflops, where we count a combined multiplication and addition as a single floating point operation. Compared to the standard single-precision A100 GPU compute units, this is a more than order-of-magnitude increase in performance. Although impressive, this gain in performance is offset by a reduction in accuracy and under the condition that the Tensor cores only operate at peak performance when utilizing half-precision (16-bit) multiplications with single-precision accumulations. It is therefore paramount that any numerical algorithms used with the Tensor core hardware are made robust to this introduction of lower precision arithmetic. We have previously shown how this is possible for quantum-based molecular dynamics simulations and density-matrix electronic structure calculations using the computational framework of a generalized convolutional deep neural network.[24,27]

2.2. Density Matrix Construction Using a Deep Neural Network. In Kohn−Sham density functional theory,[51−53] the electron density, $\rho(\mathbf{r})$, is given as the sum of densities from occupied effective single-particle orbitals, $\{\psi_i\}$, i.e., $\rho(\mathbf{r}) = \sum_{i \in \text{occ}} |\psi_i(\mathbf{r})|^2$. The single-particle molecular orbitals that represent the electronic ground-state density are obtained from the solution of the non-linear Kohn−Sham eigenvalue equation,

$$\left(-\frac{1}{2}\nabla^2 + V_{KS}[\mathbf{R}, \rho](\mathbf{r})\right)\psi_i(\mathbf{r}) = \varepsilon_i\psi_i(\mathbf{r}) \tag{1}$$

Here, $V_{KS}[\mathbf{R}, \rho](\mathbf{r})$ is the density-dependent effective single-particle Kohn–Sham potential (see the Appendix for details). The single-particle orbitals, $\{\psi_i(\mathbf{r})\}$, can be expanded in some finite basis set

$$\psi_i(\mathbf{r}) = \sum_l C_{i,l}\phi_l(\mathbf{r}) \tag{2}$$

so that the ground-state density is given by

$$\rho(\mathbf{r}) = \sum_i |\psi_i|^2 = \sum_i \sum_{ll'} C_{i,l}^*C_{i,l'}\phi_l^*(r)\phi_{l'}(r) \tag{3}$$

$$\equiv \sum_{l,l'} D_{l,l'}\phi_l^*(r)\phi_{l'}(\mathbf{r}) \tag{4}$$

which defines the density matrix, D. In this density matrix formulation, the non-linear Kohn–Sham eigenvalue equation is replaced by the construction of the density matrix (see the Appendix for details), which is given implicitly by the Heaviside matrix step function,

$$D^\perp = \theta(\mu I - H^\perp[D]). \tag{5}$$

Here, $H^\perp[D] \equiv H^\perp[\rho]$ is the orthogonalized matrix representation, $H^\perp = Z^T H Z$, of the density-matrix-dependent or density-dependent Kohn–Sham Hamiltonian matrix,

$$H_{ij}[\rho] = \int \phi_i^*(\mathbf{r})\left(-\frac{1}{2}\nabla^2 + V_{KS}[\mathbf{R}, \rho](\mathbf{r})\right)\phi_j(\mathbf{r})d\mathbf{r} \tag{6}$$

and Z is an inverse factor of the overlap matrix, $O_{ij} = \int \phi_i^*(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$, such that $Z^T O Z = I$. There are a number of efficient techniques to calculate Z.[56−60] A favorite of ours is based on a matrix recursion technique that is also well-suited for Tensor core multiplications (see the Appendix for details). The Hamiltonian matrix, $H[\rho] \equiv H[D]$, depends on the density matrix, D, through the density, $\rho(r)$ in the Kohn–Sham potential, $V_{KS}[\mathbf{R}, \rho](\mathbf{r})$. The solution of the electronic structure problem is therefore equivalent to a direct self-consistent calculation of the Heaviside matrix step function of $H^\perp[D]$, with the step formed at the chemical potential μ, where μ is set such that the trace of the density matrix equals the number of occupied states, i.e., $\text{Tr}[D^\perp] = N_{occ}$.

A practical way of calculating a matrix step function is by expressing θ as a recursive expansion.[12,18,61−63] A particularly simple and efficient technique for constructing such an expansion is the recursive second-order spectral projection scheme (SP2).[64−66] In this method, repeated compositions of the polynomials $g(x) = x^2$ and $g(x) = 2x - x^2$, applied to a re-scaled Hamiltonian matrix (so that the spectrum is inside $[0,1]$), lead to an approximation of the Heaviside step function in eq 5. In each iteration, the polynomial is chosen based on which polynomial gives a matrix trace closest to the true occupation number. In this way, the step is formed automatically at the correct chemical potential.

The SP2 scheme can easily be mapped onto the computational structure of a convolutional deep neural network (DNN),[24] where the orthogonalized density matrix, given by the step function of $H^\perp$, is generated through a recursive DNN expansion,

$$D_\perp = f(...f(W_1 f(W_0 X_0 + B_0) + B_1)...) \tag{7}$$

Here, $\{W_n\}$ and $\{B_n\}$ are the weight and bias values of the network, with the matrix square activation function, $f(S) = S^2$. This DNN formulation of the SP2 scheme is formally the same scheme as the original SP2 method, though it has some conceptual advantages.[24] The DNN formulation may also appear more familiar to many readers thanks to the widespread popularity of machine learning methods using neural networks. In general, we can expect Tensor cores to accelerate any recursive expansion that is dominated by matrix–matrix multiplications. This includes matrix inversions using Schulz iterations[67] or the inverse factorization of the overlap matrix[57] as well as higher-order recursive Fermi-operator expansion methods.[62,68] However, in most cases, these schemes do not naturally fit into a DNN structure.

After an initial normalization layer, the orthogonalized density matrix approximation at the nth deep layer of the DNN-SP2 scheme is constructed recursively by the network and is given by S_n, where

$$S_n = W_n X_n + B_n \tag{8}$$

$$X_n = f(S_{n-1}) = S_{n-1}^2 \tag{9}$$

$$W_n = \sigma_n I \text{ and } \sigma_n = \pm 1 \tag{10}$$

$$B_n = (1 - \sigma_n)S_n \tag{11}$$

A schematic diagram illustrating the deep neural network structure is given in Figure 2.
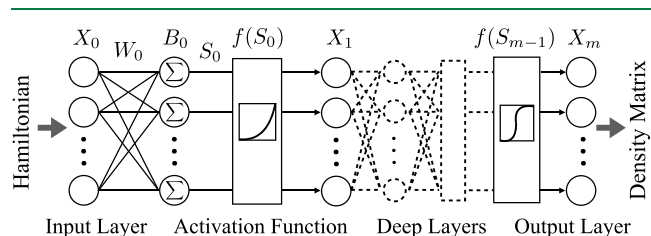

Figure 2. DNN-SP2 electronic structure method represented as a deep neural network. Given an orthogonalized Hamiltonian matrix, $H^\perp$, as the input layer, the neural network outputs the (orthogonalized) density matrix, $D^\perp$, given by the Heaviside matrix step function $D^\perp = \theta(\mu - H^\perp)$, with step formed at the Fermi level, μ. The dominating computational cost for the network is the activation function, $f(S_{n-1}) = S_{n-1}^2$, a matrix square operation, which Tensor cores are optimized for. After an initial normalization layer, the weights (W_n), biases (B_n), and layers (X_n) are given by $W_n = \sigma_n I$, $B_n = (1 - \sigma_n)S_n$, $S_n = W_n X_n + B_n$, and $X_n = f(S_{n-1}) = S_{n-1}^2$ for the deep layers. The number σ_n is chosen to be ± 1, and S_n represents the nth density matrix approximation.

The recursive deep neural network procedure in eq 7 starts with an input layer, X_0, which is given by the orthogonalized Hamiltonian, $X_0 = H^\perp$. Next, specific weight and bias values, W_0 and B_0, are chosen so that the eigenvalue spectrum of $H^\perp$ is brought to within the interval $[0,1]$ in reverse order, i.e., with the lowest lying eigenvalues closer to 1 and the highest closer to 0. This initial normalization step requires some approximate upper and lower bounds of the eigenvalue spectrum of $H^\perp$, which can be estimated, for example, by Gershgorin circles. Application of the activation function (the matrix square) to this linearly transformed initial layer moves us to the next layer, X_1. This sequence of steps represents the initial step $f(W_0 X_0 + B_0)$ in eq 7. After this first layer, at each deep layer of the DNN-SP2 network in eq 8, the weights are chosen as $W_n = \sigma_n I$

and biases as $B_n = (1 - \sigma_n)S_{n-1}$, with $\sigma_n = \pm 1$. The signs are selected such that the density matrix converges to the correct occupation by projecting the eigenvalues of the occupied states to 1 and the unoccupied states to 0.[62,64] The layers are repeated until all eigenvalues of the output matrix are sufficiently close to 1 or 0, i.e., $f(S_{n-1}) \approx S_{n-1}$, in which case we have an idempotent approximation of the density matrix, i.e., $D^\perp = D^\perp D^\perp$ or $D = DOD$. This idempotency condition can be used to check the convergence of the DNN-SP2 scheme.[24,62,69]

2.3. The Activation Function and Its Dual Matrix Representation.

To make full use of the Tensor core hardware, the DNN-SP2 algorithm needs to be re-purposed for low precision arithmetic, as the Tensor cores are at peak performance only when multiplying matrices in half-precision (FP16) and accumulating the products in single-precision (FP32). This can be accomplished by approximating a given matrix X as a sum of two low precision parts,[24,70,71]

$$X \approx X_{\text{high}} + X_{\text{low}} \tag{12}$$

where, in the case of half-precision representations, the two matrices are given by

$$X_{\text{low}} = \text{FP16}[X - X_{\text{high}}]$$
$$X_{\text{low}} = \text{FP16}[X - X_{\text{high}}] \tag{13}$$

Here, "FP16[X]" denotes the half-precision (16 bit) floating point (FP) representation of X. Because we represent the error between X and its half-precision representation, X_{high}, also in half-precision, we expect the overall approximation in eq 12 to have an accuracy of about 10^{-6}. Machine epsilon for FP16 arithmetic is approximately 10^{-3}. The matrix square activation function, f, can then be approximated by two separate half-precision Tensor core matrix–matrix multiplications with their product accumulated in single-precision (32 bit), that is

$$f(X) \approx \text{FP32}[(X_{\text{high}} + X_{\text{low}})^2]$$
$$\approx \text{FP32}[\mathcal{A} + \mathcal{B} + \mathcal{B}^T] \tag{14}$$

where

$$\mathcal{A} = X_{\text{high}} \times X_{\text{high}} (\text{Tensor core mult.}) \tag{15}$$

$$\mathcal{B} = X_{\text{high}} \times X_{\text{low}} (\text{Tensor core mult.}) \tag{16}$$

and the $X_{\text{low}} \times X_{\text{low}}$ has been discarded. To further enhance accuracy, a refinement, or purification layer, may be used as the final output layer of the network in eq 7, where all matrix algebra is carried out in double-precision, FP64.[24,27] This purification step is equivalent to a final double flip, e.g., $\sigma = 1$ followed by $\sigma = -1$ or vice versa, and was shown to improve accuracy by second order for ground-state density matrix calculations.[24] However, experience has shown this additional refinement step to be unnecessary in DMPT and other certain cases, e.g., for QMD simulations,[27] and we will not consider any refinement layers in this article. In the case of DMPT, this ultimately stems from there being two main sources of error: the idempotency error and the commutation error. In a variational total energy expression, the energy error is first order in idempotency and second order in commutation. This is not true for the response calculations. The additional refinement therefore does not help.

3. DENSITY MATRIX PERTURBATION THEORY AS A DEEP NEURAL NETWORK

3.1. Density Matrix Perturbation Theory.

In density functional theory, a time-independent external perturbation, e.g., from a static electric or a magnetic field, can be introduced to the electronic structure through the Kohn–Sham Hamiltonian matrix, H. If λ is the magnitude of the perturbation, we expand the Hamiltonian in powers of λ as

$$H(\lambda) = H^{(0)} + \lambda H^{(1)} + \lambda^2 H^{(2)} + \cdots \tag{17}$$

where $H^{(1)}$, $H^{(2)}$, ... are the perturbations to various order in λ. An example might be where $H^{(1)}$ is the dipole moment operator that couples the system to an external field in some direction, and λ is the field strength in that direction. The corresponding expansion of $H(\lambda)$ in an orthogonalized representation is given by

$$H^\perp(\lambda) = H_\perp^{(0)} + \lambda H_\perp^{(1)} + \lambda^2 H_\perp^{(2)} + \cdots \tag{18}$$

where $H_\perp^{(k)} = Z^T H^{(k)} Z$.

The expression in eq 17 will induce a similar λ expansion in the density matrix, D, i.e., describing the response in the electronic structure due to the perturbation. In general (as described in Section 2.2), this is a non-linear matrix function of the Hamiltonian and the perturbation, where the density matrix and its response $D^\perp(H(\rho, \lambda)) = \theta(\mu I - H^\perp(\rho, \lambda))$ have to be calculated self-consistently, both with respect to the unperturbed ground-state density (or density matrix) and its response to the original perturbation in H. This DMPT approach is a density matrix formulation of what is usually referred to as the coupled-perturbed self-consistent field or density functional perturbation theory.[5−8,72]

Expanding D in λ,

$$D(\lambda) = D^{(0)} + \lambda D^{(1)} + \lambda^2 D^{(2)} + \cdots \tag{19}$$

we can identify $D^{(k)}$ from

$$D_\perp^{(k)} = \frac{1}{k!} \frac{\partial^k \theta(\mu I - H^\perp(\rho, \lambda))}{\partial \lambda^k} \bigg|_{\lambda=0} \tag{20}$$

through a Taylor series expansion for $D(\lambda)$, where $D^{(k)} = Z D_\perp^{(k)} Z^T$. This response in the density matrix leads to the corresponding response in the electron density

$$\rho(\mathbf{r}) = \rho^{(0)}(\mathbf{r}) + \lambda \rho^{(1)}(\mathbf{r}) + \lambda^2 \rho^{(2)}(\mathbf{r}) + \ldots \tag{21}$$

where

$$\rho^{(k)}(\mathbf{r}) = \sum_{l,l'} D_{l,l'}^{(k)} \phi_l(\mathbf{r}) \phi_{l'}(\mathbf{r}) \tag{22}$$

The response in the density matrix using a recursive expansion of the matrix step function forms the basis of DMPT and together with the electron density couples to the perturbation in the Hamiltonian. In general, the response equations therefore have to be solved self-consistently. In its formulation above, the DMPT is only valid for static, time-independent perturbations of non-degenerate systems at zero electronic temperatures.

The recursive nature of the DNN-SP2 algorithm allows us to map DMPT onto the computational structure of a DNN. In the same way as in ref 12, we can determine how perturbations of the Hamiltonian of each order in λ propagate, layer by layer, through the DNN-SP2 network. An approximate density

matrix and its response to various orders in λ are then given as the final output layer. To understand how this works, we can first look at the non-perturbative case in eqs 8 and 9. The nth layer of the DNN-SP2 algorithm, X_n, is given by $X_n = f(S_{n-1}) = S_{n-1}^2$, and the nth density matrix approximation, S_n, is given by

$$S_n = W_n X_n + B_n = \sigma_n S_{n-1}^2 + (1 - \sigma_n) S_{n-1} \tag{23}$$

If we use this update rule in eq 24 with an approximate density matrix expanded up to some order in λ at layer n, i.e.

$$S_n(\lambda) = S_n^{(0)} + \lambda S_n^{(1)} + \lambda^2 S_n^{(2)} + \cdots \tag{24}$$

we are able to calculate the updated response order by order. For $n > 0$, the first three orders in λ^k ($k = 0,1,2$), of the approximate density matrix and its response are given by

$$S_n^{(0)} = W_n (S_{n-1}^{(0)})^2 + B_n^{(0)}$$

$$S_n^{(1)} = W_n (S_{n-1}^{(0)} S_{n-1}^{(1)} + S_{n-1}^{(1)} S_{n-1}^{(0)}) + B_n^{(1)}$$

$$S_n^{(2)} = W_n (S_{n-1}^{(0)} S_{n-1}^{(2)} + S_{n-1}^{(1)} S_{n-1}^{(1)} + S_{n-1}^{(2)} S_{n-1}^{(0)}) + B_n^{(2)} \tag{25}$$

and more generally, for all k by

$$S_n^{(k)} = W_n \sum_{j=0}^{k} S_{n-1}^{(j)} S_{n-1}^{(k-j)} + B_n^{(k)} \tag{26}$$

where the bias functions, $B_n^{(k)}$, are given by

$$B_n^{(k)} = (1 - \sigma_n) S_{n-1}^{(k)} \tag{27}$$

We can then expect that $D_\perp^{(k)} = \lim_{n \to \infty} S_n^{(k)}$. At $n = 0$, the initial transformation is given via the expansion

$$S_0(\lambda) = \frac{\varepsilon_N I - H^\perp(\lambda)}{\varepsilon_N - \varepsilon_1}$$

$$= \frac{\varepsilon_N I - H_\perp^{(0)}}{\varepsilon_N - \varepsilon_1} - \lambda \frac{1}{\varepsilon_N - \varepsilon_1} H_\perp^{(1)} - \lambda^2 \frac{1}{\varepsilon_N - \varepsilon_1} H_\perp^{(2)} - \cdots$$

$$= S_0^{(0)} + \lambda S_0^{(1)} + \lambda^2 S_0^{(2)} + \cdots$$

where ε_1 and ε_N are estimates of the smallest and largest eigenvalues of $H_\perp^{(0)}$, respectively. The recursive expansion of the density matrix response in eqs 25 to 27 forms the basis of DMPT in its DNN-SP2 form.

Alternatively, we can formulate this DMPT algorithm purely in terms of the network parameters X, W, and B, which may be more desirable for direct machine learning applications.[24] In this case, we use the relation

$$X_{n+1} = (W_n X_n + B_n)^2 \tag{28}$$

so that, assuming W_n is a scaled identity matrix,

$$X_{n+1}^{(k)} = (W_n)^2 \sum_{j=0}^{k} X_n^{(j)} X_n^{(k-j)} + W_n \sum_{j=0}^{k} \{X_n^{(j)}, B_n^{(k-j)}\}$$

$$+ \sum_{j=0}^{k} B_n^{(j)} B_n^{(k-j)} \tag{29}$$

where we use the anti-commutator notation, $\{A, B\} = AB + BA$. Notice, that for symmetric and commuting matrices, $(AB)^T = B^T A^T = BA = AB$, which is used to speed up the calculations.

In the next section, we adapt the DNN-SP2 method to calculate $D^{(k)}$, for any k, using a blocked matrix representation with the same convolutional deep neural network as the original DNN-SP2 scheme. This block-matrix formulation is valuable because it allows a more straightforward mathematical analysis of DMPT, while at the same time, generates an efficient way of calculating the matrix function derivatives in eq 20, using dense matrix algebra as in eq 26, which is well-suited to the Tensor core design.

3.2. Block Matrix Representation. We now re-express DMPT as a DNN using upper-triangular block matrices through the DNN-SP2 method. We call this technique the DNN-SP2 perturbation theory (DNN-SP2-PT) method. Given a maximum response order m, we map the mth order perturbative expansion for the Hamiltonian, $H = H^{(0)} + \lambda H^{(1)} + \lambda^2 H^{(2)} + \cdots + \lambda^m H^{(m)}$, onto an $(m + 1)N \times (m + 1)N$ block upper-triangular matrix,

$$\mathbf{H} = \begin{pmatrix} H^{(0)} & H^{(1)} & H^{(2)} & \cdots & H^{(m)} \\ & H^{(0)} & H^{(1)} & \cdots & H^{(m-1)} \\ & & H^{(0)} & \cdots & H^{(m-2)} \\ & & & \ddots & \vdots \\ & & & & H^{(0)} \end{pmatrix} \tag{30}$$

The bold notation indicates block matrices and the "⊥" symbols indicating an orthogonalized representation are dropped for convenience. All the lower triangular blocks of $\mathbf{H}$ are zero matrices.

Similarly, the mth order expansion for the density matrix can be mapped onto the matrix

$$\mathbf{D} = \begin{pmatrix} D^{(0)} & D^{(1)} & D^{(2)} & \cdots & D^{(m)} \\ & D^{(0)} & D^{(1)} & \cdots & D^{(m-1)} \\ & & D^{(0)} & \cdots & D^{(m-2)} \\ & & & \ddots & \vdots \\ & & & & D^{(0)} \end{pmatrix} \tag{31}$$

The idempotency conditions for the mth order density matrix expansion are then given by the blocks of $\mathbf{D}^2 = \mathbf{D}$. Similarly, the commutation conditions for DMPT can be captured for each order from the blocks using the block matrix equation $\mathbf{HD} - \mathbf{DH} = 0$.

This block representation can then be used in a straightforward manner to generalize the DNN-SP2 scheme. The input layer is set to be $\mathbf{X}_0 = \mathbf{H}$, in the same way as for DNN-SP2, where the initial weight and bias are also chosen to yield a shifting and rescaling of the spectrum for $\mathbf{H}$ (or, equivalently, $H^{(0)}$):

$$\mathbf{S}_0 = \frac{\varepsilon_N \mathbf{I} - \mathbf{H}}{\varepsilon_N - \varepsilon_1} \tag{32}$$

Here the values ε_1 and ε_N are estimates of the smallest and largest eigenvalues for the non-perturbed system, $H^{(0)}$, i.e., for $H(\lambda)$ in the limit $\lambda = 0$. In the same way as before, the nth layer of the network, $\mathbf{X}_n$, is then defined through

$$\mathbf{S}_n = W_n \mathbf{X}_n + \mathbf{B}_n \tag{33}$$

$$\mathbf{X}_n = f(\mathbf{S}_{n-1}) \tag{34}$$

where f is the matrix square activation function, i.e., $f(\mathbf{X}) = \mathbf{X}^2$, the weights, W_n, are also defined as before through $W_n = \sigma_n \mathbf{I}$ (with $\sigma_n = \pm 1$), and the biases, $\mathbf{B}_n$, are defined by

$$\mathbf{B}_n = (1 - \sigma_n)\mathbf{S}_{n-1} \tag{35}$$

The upper-triangular block matrix

$$\mathbf{S}_n = \begin{pmatrix} S_n^{(0)} & S_n^{(1)} & S_n^{(2)} & \cdots & S_n^{(m)} \\ & S_n^{(0)} & S_n^{(1)} & \cdots & S_n^{(m-1)} \\ & & S_n^{(0)} & \cdots & S_n^{(m-2)} \\ & & & \ddots & \vdots \\ & & & & S_n^{(0)} \end{pmatrix} \tag{36}$$

corresponds to the approximate density matrix expansion at the nth layer, $S_n^{(0)} + \lambda S_n^{(1)} + \lambda^2 S_n^{(2)} + \cdots + \lambda^m S_n^{(m)}$.

Figure 3 presents a visual of the deep neural network for this generalized block matrix formulation of density matrix
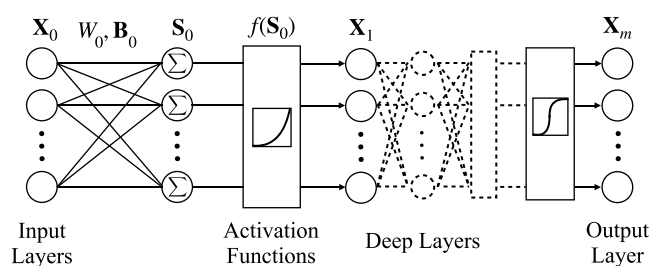

Figure 3. Deep neural network diagram for the generalized block matrix formulation of the DNN-SP2-PT method with $f(\mathbf{X}) = \mathbf{X}^2$, a matrix square activation function. The network is exactly the same as the network for the DNN-SP2 method we previously presented in ref 24 except that here $\mathbf{B}_k, \mathbf{S}_k, \mathbf{X}_k \in \mathbb{R}^{(m+1)N \times (m+1)N}$ are block matrices with the different response orders on the superdiagonals, where m is the highest-order response considered and N is the number of basis orbitals.

perturbation theory. As in the case of regular DNN-SP2, it is possible to append an additional double-precision refinement step to the final output layer, but again, we do not consider this here. No significant benefit in accuracy of the response matrices was observed with this extra refinement step.

3.3. Activation Function. Applying the matrix square activation function, f, to $\mathbf{S}_n$ from the previous section leads to

$$f(\mathbf{S}_n) = \begin{pmatrix} f^{(0)} & f^{(1)} & f^{(2)} & \cdots & f^{(m)} \\ & f^{(0)} & f^{(1)} & \cdots & f^{(m-1)} \\ & & f^{(0)} & \cdots & f^{(m-2)} \\ & & & \ddots & \vdots \\ & & & & f^{(0)} \end{pmatrix} \tag{37}$$

where

$$f^{(k)} = f^{(k)}(S^{(0)}, S^{(1)}, ..., S^{(k)}) \equiv \sum_{j=0}^{k} S^{(j)}S^{(k-j)} \tag{38}$$

It is straightforward to verify that this matrix square activation definition and the upper-triangular block matrix representation

of the DNN-SP2 scheme yield the exact same update rules for each matrix component, $S_n^{(k)}$, as shown in the previous section in eq 26.

It is also worth noting in particular that $f^{(m)}$ depends only on $S^{(0)}, S^{(1)}, ..., S^{(m)}$. Thus, to obtain the mth order response approximation, $S_n^{(m)}$, at layer n, information from only the first m order terms is needed from the previous layer, $n-1$. This observation provides a natural way of parallelizing the matrix multiplications over multiple GPU devices. Given m GPU devices, the lth device, $1 \leq l \leq m$, can calculate the $f^{(l)}$ activation function, requiring only that $S^{(0)}, S^{(1)}, ..., S^{(l-1)}$ be sent from the previous $l-1$ devices unidirectionally. Because the information flow only moves one way, efficiencies can be recovered as latencies are hidden more easily. This multi-GPU approach will be demonstrated in the case of a linear response calculation in Section 4.

Though we now need $m + 1$ matrices to calculate the mth order response (as we see in eq 38), the additional storage requirement will turn out to not be the principal limiting factor for system sizes that can be treated compared to the straightforward ground-state calculations[24] of the density matrix. It is the Tensor core to GPU I/O that remains the bottleneck and limits the system size to around 20,000 orbitals for optimal performance.

3.4. Dual Matrix Representations for the Activation Function. The computational cost for each deep layer is dominated by the matrix activation function, which requires repeated matrix–matrix multiplications. To enhance the accuracy of the low precision floating-point operations from the Tensor cores, we follow the same recipe as our original DNN-SP2 scheme[24] and use a dual matrix representation as in eqs 12 and 13. In the activation functions, $f^{(m)}$, we use this dual matrix representation for each factor $S^{(j)}$, where

$$f^{(m)}(S^{(0)}, ..., S^{(m)}) \equiv \sum_{j=0}^{m} S^{(j)}S^{(m-j)}$$

$$= \sum_{j=0}^{m} (S_{\text{high}}^{(j)} + S_{\text{low}}^{(j)}) \times (S_{\text{high}}^{(m-j)} + S_{\text{low}}^{(m-j)}) \tag{39}$$

In the same way as before, the half-precision representations of the matrices $S^{(j)}$ are given by

$$S_{\text{high}}^{(j)} = \text{FP16}[S^{(j)}] \tag{40}$$

$$S_{\text{low}}^{(j)} = \text{FP16}[S^{(j)} - S_{\text{high}}^{(j)}] \tag{41}$$

where "FP16[S]" denotes the half-precision (16 bit) floating point (FP) representation of S.[24,62,69] We then approximate the mth order activation function, $f^{(m)}$, by

$$f^{(m)} \approx \sum_{j=1}^{m} (S_{\text{high}}^{(j)}S_{\text{high}}^{(m-j)} + S_{\text{low}}^{(j)}S_{\text{high}}^{(m-j)} + S_{\text{high}}^{(j)}S_{\text{low}}^{(m-j)}) \tag{42}$$

and discard the less significant $S_{\text{low}} \times S_{\text{low}}$ terms. Each summand in the activation function $f^{(m)}$ is then approximated by separate half-precision Tensor core matrix–matrix multiplications with their product accumulated in single-precision-(FP32), i.e.

$$S^{(j)}S^{(m-j)} \approx \text{FP32}[A + B + C] \tag{43}$$

$$A = S_{\text{high}}^{(j)} S_{\text{high}}^{(m-j)} (\text{Tensor core mult. })$$

$$B = S_{\text{high}}^{(j)} S_{\text{low}}^{(m-j)} (\text{Tensor core mult. })$$

$$C = S_{\text{low}}^{(j)} S_{\text{high}}^{(m-j)} (\text{Tensor core mult. }) \tag{44}$$

This is, once again, analogous to the original DNN-SP2 scheme. Symmetry of the Hamiltonian matrix H and its perturbations $H^{(l)}$ implies

$$S^{(m-j)} S^{(j)} = (S^{(j)} S^{(m-j)})^T$$

for all $0 \leq j < m$, so that we actually only need $\sim m/2$ matrix multiplications instead of m (see Python code in the Supporting Information).

4. LINEAR RESPONSE CALCULATIONS

To illustrate and analyze our DNN-SP2 DMPT framework, we focus on first-order linear response calculations, for time- and basis set-independent perturbations. Higher orders in the response are given by a straightforward generalization and the extension to basis set-dependent response calculations is also possible.[14] First, we look at calculations using self-consistent density functional-based tight-binding theory and then Hartree−Fock theory. A pseudocode for a first-order response calculation is shown in Algorithms 1 to 3 and a Python translation of it is given in the Supporting Information.

Before analyzing the performance of Algorithm 1, we need to decide how to determine when convergence is reached for the response. This is far from trivial as the numerically noisy low-precision Tensor core arithmetics make it hard to determine convergence and previous stopping criteria for the unperturbed DNN-SP2 scheme may no longer apply.[27]

4.1. Parameter-free Stopping Criterion. The stopping criterion that we develop in this section is based on a generalization of the previous parameter-free convergence measures developed for the ground-state density matrix.[24,69] Parameter-free stopping criteria are based on the idea of identifying an analytical convergence rate that is valid under exact arithmetics. Whenever this convergence rate is broken, which eventually occurs because of the finite precision in the numerical calculations, we expect that we have the best possible answer using the available numerical precision and we terminate the calculation.

In exact arithmetic,

$$\lim_{n \to \infty} \mathbf{S}_n = \lim_{n \to \infty} \begin{pmatrix} S_n^{(0)} & S_n^{(1)} \\ 0 & S_n^{(0)} \end{pmatrix} = \begin{pmatrix} D^{(0)} & D^{(1)} \\ 0 & D^{(0)} \end{pmatrix} = \mathbf{D} \tag{45}$$

where $D^{(1)}$ is the matrix directional derivative (or Gâteaux derivative) in the direction of the linear perturbation, $H^{(1)}$, i.e.

$$D^{(1)} = \lim_{\lambda \to 0} \frac{D(H^{(0)} + \lambda H^{(1)}) - D(H^{(0)})}{\lambda} \tag{46}$$

We may write

$$g\left(\begin{pmatrix} H^{(0)} & H^{(1)} \\ 0 & H^{(0)} \end{pmatrix} \right) = \mathbf{S}_n = \begin{pmatrix} S_n^{(0)} & S_n^{(1)} \\ 0 & S_n^{(0)} \end{pmatrix} \tag{47}$$

where $g = g_n \circ g_{n-1} \circ \cdots \circ g_1$ is the composition of all prior activation functions, weight, and bias applications of the DNN-SP2 method (as in eq 7), or equivalently, polynomials x^2 and

$2x - x^2$, using the original SP2 framework. Consider a σ-flip iteration, that is, a double iteration choosing $\sigma_{n+1} = 1$ and $\sigma_{n+2} = -1$, or vice versa, for the next two iterations in the DNN-SP2-PT method. This is equivalent to applying the function $h(x) = 2x^2 - x^4$ or $h(x) = (2x - x^2)^2$ (the two possible function compositions with x^2 and $2x - x^2$) to $\mathbf{S}_n$ in eq 47 to get

$$\mathbf{S}_{n+2} = h\left(\begin{pmatrix} S_n^{(0)} & S_n^{(1)} \\ 0 & S_n^{(0)} \end{pmatrix} \right) = \begin{cases} 2\mathbf{S}_n^2 - \mathbf{S}_n^4 \\ (2\mathbf{S}_n - \mathbf{S}_n^2)^2 \end{cases} \tag{48}$$

The evaluation of h in eq 48 yields the approximate density matrix and its response at layer $n + 2$, $\mathbf{S}_{n+2}$. Due to the particular upper-triangular block form of $\mathbf{S}_n$, results by Mathias[63,73] say that the evaluation of h at the matrix $\mathbf{S}_n$ in eq 48 can be written as

$$h(\mathbf{S}_n) = \begin{pmatrix} h(S_n^{(0)}) & \left. \dfrac{\partial h(S_n^{(0)} + \lambda S_n^{(1)})}{\partial \lambda} \right|_{\lambda=0} \\ 0 & h(S_n^{(0)}) \end{pmatrix} \tag{49}$$

where the upper right block, $S_{n+2}^{(1)}$, is the directional derivative of h at $S_n^{(0)}$ in the direction of $S_n^{(1)}$. Furthermore, assuming an eigendecomposition $S_n^{(0)} = V \Lambda V^T$, where $\Lambda = \text{diag }(\lambda_i)$ and $\lambda_1 \geq \cdots \geq \lambda_N$, the theorem of Daleckii−Krein[63,74] tells us that this derivative can be computed analytically and is given by

$$\left. \frac{\partial h(S_n^{(0)} + \lambda S_n^{(1)})}{\partial \lambda} \right|_{\lambda=0} = V(G \odot (V^T S_n^{(1)} V)) V^T \tag{50}$$

where in this expression the symbol $\odot$ represents the elementwise, or Hadamard, product and the entries of the matrix G are defined by $G_{ij} = h[\lambda_i, \lambda_j]$, the divided differences quotient for h using eigenvalues λ_i and λ_j. The divided differences quotients are defined as

$$h[\lambda_i, \lambda_j] = \begin{cases} \dfrac{h(\lambda_i) - h(\lambda_j)}{\lambda_i - \lambda_j}, & \lambda_i \neq \lambda_j \\ h'(\lambda_i), & \lambda_i = \lambda_j \end{cases} \tag{51}$$

The expression in eq 48 then becomes

$$\mathbf{S}_{n+2} = \begin{pmatrix} h(S_n^{(0)}) & V(G \odot (V^T S_n^{(1)} V)) V^T \\ 0 & h(S_n^{(0)}) \end{pmatrix} \tag{52}$$

As a first attempt to deriving a stopping criterion, we assume that the zeroth order approximation, $S_n^{(0)}$, has perfectly converged after n iterations to $D^{(0)}$. Then, all of its eigenvalues, λ_i, will be 0 or 1, so that the divided differences quotients, $h[\lambda_i, \lambda_j]$, are also 0 or 1 because

$$h[\lambda_i, \lambda_j] = \begin{cases} 0 = h'(0), & \lambda_i = 0, \lambda_j = 0 \\ 1 = \dfrac{h(0) - h(1)}{0 - 1}, & \lambda_i = 0, \lambda_j = 1 \\ 1 = \dfrac{h(1) - h(0)}{1 - 0}, & \lambda_i = 1, \lambda_j = 0 \\ 0 = h'(1), & \lambda_i = 1, \lambda_j = 1 \end{cases} \tag{53}$$

Therefore, we have

$$\Lambda = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \tag{54}$$

and

$$G = \begin{pmatrix} 0 & \mathbf{1} \\ \mathbf{1} & 0 \end{pmatrix} \tag{55}$$

where $\mathbf{1}$ is a matrix of ones and I is the identity matrix. The form of G then implies that the operation $G \odot$ is idempotent, so that application of h a second time to the expression in eq 48 does not change it. In other words, applying another σ-flip iteration does not improve or worsen the approximation to $D^{(1)}$.

This would suggest that a single σ-flip iteration following the convergence of $S_n^{(0)}$ should be sufficient for the convergence of $S_n^{(1)}$. However, it was sometimes observed numerically that the first-order idempotency error, $\mathrm{Idemp}_n^{(1)} = S_n^{(1)} - S_n^{(0)}S_n^{(1)} - S_n^{(1)}S_n^{(0)}$, continues to decrease if one does additional σ-flip iterations.

To further understand this behavior, let us again assume that $S_n^{(0)}$ has converged exactly. The first-order idempotency error is given by

$$\mathrm{Idemp}_n^{(1)} = S_n^{(1)} - S_n^{(0)}S_n^{(1)} - S_n^{(1)}S_n^{(0)} \tag{56}$$

$$= VV^T S_n^{(1)} VV^T - V\Lambda V^T V^T S_n^{(1)} VV^T \tag{57}$$

$$- VV^T S_n^{(1)} V \Lambda V^T \tag{58}$$

$$= V((I - \Lambda)Y - Y\Lambda)V^T \tag{59}$$

$$= V\begin{pmatrix} -Y_{1,1} & 0 \\ 0 & Y_{2,2} \end{pmatrix} V^T \tag{60}$$

where $Y = V^T S_n^{(1)} V$ and $Y_{i,j}$, $i,j \in \{1,2\}$ are the submatrices of Y conforming to the previous block partitions of Λ and G. This means that the purpose of any additional iterations should be only to zero out the diagonal blocks of Y, ideally leaving the off-diagonal blocks, which do not contribute to the idempotency error, unchanged. Any changes to the off-diagonal blocks are the result of spurious fluctuations at the level of the numerical accuracy. Failure to converge using a single σ-flip iteration is due to the supposedly zero diagonal blocks of G not being exactly zero. Numerical errors in those blocks play a key role in the final convergence of $S_n^{(1)}$.

We therefore model the convergence in this final phase by including a block-diagonal perturbation of G

$$G = \begin{pmatrix} \delta_1 & \mathbf{1} \\ \mathbf{1} & \delta_2 \end{pmatrix} \tag{61}$$

with

$$(\delta_1)_{i,j} = h[\lambda_i, \lambda_j], \quad i,j = 1, ..., N_{occ} \tag{62}$$

and

$$(\delta_2)_{i,j} = h[\lambda_{N_{occ}+i}, \lambda_{N_{occ}+j}], \quad i,j = 1, ..., N - N_{occ} \tag{63}$$

where $\lambda_1 \approx \cdots \approx \lambda_{N_{occ}} \approx 1$ and $\lambda_{N_{occ}+1} \approx \cdots \approx \lambda_N \approx 0$ are the actual eigenvalues of $S_n^{(0)}$. Given this model, we derive a worst case reduction of the first-order idempotency error by a σ-flip

iteration. The first-order idempotency error after a σ-flip iteration is then given by

$$\mathrm{Idemp}_{n+2}^{(1)} = S_{n+2}^{(1)} - S_n^{(0)}S_{n+2}^{(1)} - S_{n+2}^{(1)}S_n^{(0)} \tag{64}$$

$$= V(G \odot Y)V^T - V\Lambda V^T V(G \odot Y)V^T \tag{65}$$

$$- V(G \odot Y_n)V^T V \Lambda V^T \tag{66}$$

$$= V((G - \Lambda G - G\Lambda) \odot Y)V^T \tag{67}$$

$$= V\begin{pmatrix} -\delta_1 \odot Y_{1,1} & 0 \\ 0 & \delta_2 \odot Y_{2,2} \end{pmatrix} V^T \tag{68}$$

Thus, instead of immediate convergence after a single σ-flip iteration, we get an iteration with fast linear convergence and a small prefactor determined by δ_1 and δ_2. Note that we have used $S_n^{(0)}$ rather than $S_{n+2}^{(0)}$ in eq 64 since we have assumed our algorithm stops updating $S_n^{(0)}$ once it has converged.

The absolute values of the entries of δ_1 can be bounded as

$$|(\delta_1)_{i,j}| = |h[\lambda_i, \lambda_j]| \tag{69}$$

$$\leq \max(|h'(\lambda_i)|, |h'(\lambda_j)|) \tag{70}$$

$$\leq \max_k |h'(\lambda_k)| \tag{71}$$

$$\leq C_{pt} \max_k |\lambda_k - \lambda_k^2| \tag{72}$$

$$= C_{pt} \left\| S_n^{(0)} - (S_n^{(0)})^2 \right\|_2 \tag{73}$$

$$\leq C_{pt} \left\| S_n^{(0)} - (S_n^{(0)})^2 \right\|_F \tag{74}$$

The inequality in eq 70 is based on the assumption that all eigenvalues $\lambda_1, ..., \lambda_{N_{occ}}$ are larger than the inflection point of $h(x)$ so that h is concave on their convex hull. The inflection points of $h(x) = (2x - x^2)^2$ and $h(x) = 2x^2 - x^4$ are $1 - 1/\sqrt{3} \approx 0.42$ and $1/\sqrt{3} \approx 0.58$, respectively. The constant introduced in eq 72 can be chosen as

$$C_{pt} \geq 6 + 4\sqrt{0.25 + \left\| S_n^{(0)} - (S_n^{(0)})^2 \right\|_F}$$

which ensures that the inequality is valid on an interval containing all eigenvalues, including the one that gives the maximum. In practice, one may for example set $C_{pt} = 9$, which makes the inequality valid on the $[-0.25, 1.25]$ interval, which should always include all eigenvalues. The corresponding $|(\delta_2)_{i,j}| \leq C_{pt} \| S_n^{(0)} - (S_n^{(0)})^2 \|_F$ bound can be derived similarly.

Using all of this and that the Frobenius norm is unitarily invariant, we get that the reduction of the first-order idempotency error in the Frobenius norm is bounded as follows:

$$\left\| \mathrm{Idemp}_{n+2}^{(1)} \right\|_F = \sqrt{\| \delta_1 \odot Y_{1,1} \|_F^2 + \| \delta_2 \odot Y_{2,2} \|_F^2}$$

$$\leq C_{pt} \left\| S_n^{(0)} - (S_n^{(0)})^2 \right\|_F \times \sqrt{\| Y_{1,1} \|_F^2 + \| Y_{2,2} \|_F^2}$$

$$= C_{pt} \left\| S_n^{(0)} - (S_n^{(0)})^2 \right\|_F \times \left\| \mathrm{Idemp}_n^{(1)} \right\|_F \tag{75}$$

Our proposed parameter-free convergence criteria is then given by this bound no longer holding. We also remark that

there is negligible cost to evaluating the Frobenius norm and idempotency error for determining the convergence of $S_n^{(1)}$ because the matrix square activation function of $S_n^{(0)}$ is no longer computed once $S_n^{(0)}$ has converged. The parameter-free convergence criteria often leads to a few extra (around three or four) iterations beyond what is required to reach convergence for $S_n^{(0)}$, and no additional accuracy is achieved by performing more iterations once the convergence in eq 75 has been reached. The convergence criterion is given in Algorithm 3 as part of the DNN-SP2-PT scheme in Algorithms 1 to 3.

Algorithm 1 Pseudocode for the DNN-SP2-PT formulation of the recursive Fermi-operator expansion algorithm for linear response calculations.

N, Number of orbitals
N_{occ}, Number of occupied orbitals
$H^{(0)}$, Hamiltonian matrix
$H^{(1)}$, First order perturbation of the Hamiltonian
$\varepsilon_1, \varepsilon_N$, Spectral bound estimates of $H^{(0)}$
$X_0^{(0)} = Z^T H^{(0)} Z$, Orthogonlized input
$X_0^{(1)} = Z^T H^{(1)} Z$, Orthogonlized input response
$W_0 = -(\varepsilon_N - \varepsilon_1)^{-1}$, $B_0^{(0)} = \varepsilon_N(\varepsilon_N - \varepsilon_1)^{-1}I$, $B_0^{(1)} = 0$
$S_0^{(0)} = W_0 X_0^{(0)} + B_0^{(0)}$, Initial transform, zeroth order
$N_S = \text{Tr}[S_0^{(0)}]$, Occupation of $S_0^{(0)}$
$S_0^{(1)} = W_0 X_0^{(1)} + B_0^{(1)}$, Initial transform, first order
$n = 1$, Number of layers
$D^{(0)}, D^{(1)}$ converged = **false**
while $D^{(0)}$ and $D^{(1)}$ converged = **false do**
  $X_n^{(1)} = S_{n-1}^{(0)} S_{n-1}^{(1)} + S_{n-1}^{(1)} S_{n-1}^{(0)}$, Activation function $f^{(1)}$
  if $D^{(0)}$ converged = **false then**
    $X_n^{(0)} = (S_{n-1}^{(0)})^2$, Activation function $f^{(0)}$
    $N_X^{(0)} = \text{Tr}[X_n^{(0)}]$
    $\text{IdErr}_n^{(0)} = N_S^{(0)} - N_X^{(0)}$, Idemp. error estimate
    $\sigma_n = \text{sign}\left(|2N_S^{(0)} - N_X^{(0)} - N_{occ}| - |N_X^{(0)} - N_{occ}|\right)$
    Call Layer Update $D^{(0)}$ (Alg. 2)
  else
    $S_n^{(0)} = S_{n-1}^{(0)}$
    $\sigma_n = (-1) \times \sigma_{n-1}$, Do the σ flip
    $W_n = \sigma_n I$
  end if
  Call Layer Update $D^{(1)}$ (Alg. 3)
  $n = n + 1$
end while

Algorithm 2 Parameter-free convergence criteria and deep layer update for $D^{(0)}$. The convergence criteria shown here for $\text{IdErr}_n^{(0)}$ was derived in a previous work [24] by establishing an analytical bound on $\text{IdErr}_n^{(0)}$.

---

if $\text{IdErr}_n^{(0)} <= 0$ **then**
  $D^{(0)}$ converged = **true**
  $\text{FroErr}^{(0)} = ||S_{n-1}^{(0)} - (S_{n-1}^{(0)})^2||_F$, Idemp. error
else if $n > 2$ and $\sigma_{n-1} \neq \sigma_{n-2}$ and $\text{IdErr}_n^{(0)} > 4.5 \times (\text{IdErr}_{n-2}^{(0)})^2$ **then**
  $D^{(0)}$ converged = **true**
  $\text{FroErr}^{(0)} = ||S_{n-1}^{(0)} - (S_{n-1}^{(0)})^2||_F$, Idemp. error
else
  $W_n = \sigma_n I$, $B_n^{(0)} = (I - W_n) S_{n-1}^{(0)}$
  $S_n^{(0)} = W_n X_n^{(0)} + B_n^{(0)}$, Update deep layer, zeroth-order
  $N_S^{(0)} = W_n N_X^{(0)} + (1 - \sigma_n) N_S^{(0)}$, Update occ. of $S_n^{(0)}$
end if

Algorithm 3 Parameter-free convergence criteria and deep layer update for $D^{(1)}$.

---

if $D^{(0)}$ converged = **true then**
  $\text{FroErr}_n^{(1)} = ||S_{n-1}^{(1)} - (S_{n-1}^{(0)} S_{n-1}^{(1)} + S_{n-1}^{(1)} S_{n-1}^{(0)})||_F$, First order idemp. error
  if $\text{FroErr}_n^{(1)} > 9 \times \text{FroErr}^{(0)} \times \text{FroErr}_{n-2}^{(1)}$ **then**
    $D^{(1)}$ converged = **true**
  end if
end if
if $D^{(1)}$ converged = **false then**
  $B_n^{(1)} = (I - W_n) S_{n-1}^{(1)}$
  $S_n^{(1)} = W_n X_n^{(1)} + B_n^{(1)}$, Update deep layer, first order
end if

4.2. Uniform Electric Field Perturbations in DFTB Theory.

To demonstrate and evaluate the DNN-SP2-PT scheme in Algorithms 1 to 3 using Tensor cores, we will first use second-order self-consistent charge density functional tight-binding (SCC-DFTB) theory,[75−79] which is an approximation of Kohn−Sham DFT. SCC-DFTB theory is based on a second-order expansion of the Kohn−Sham DFT energy functional around a reference electron charge distribution of overlapping neutral atomic electron densities, where the electrostatic interaction is approximated by overlapping atom-centered electron charge distributions at short distances and monopole net partial atomic charges at large distances. In all our SCC-DFTB examples, we use periodic boundary conditions but without any k points or Bloch functions (Gamma-point only). Our implementation is based on the SCC-DFTB software package LATTE.[80] As a perturbation, we use an external electric field, $\mathcal{E} = [\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_z]$ that interacts with the system through the dipole moment. Because of the periodic boundary conditions and the real-space representation, the external field interaction term has a simple sawtooth form and our response calculations do not use the geometric (Berry) phase approach that plays a crucial role in the modern theory of polarization.[81,82] This theory is required to better estimate the polarizability measured in experiments for systems with periodic cells. In this way, our calculations only represent an approximate model problem but still fully illustrate the efficiency and accuracy of our approach to quantum response calculations.

In our model, the perturbed DFTB approximation of the Kohn−Sham Hamiltonian matrix is

$$H = H^{(0)} + \sum_\alpha \mathcal{E}_\alpha H_\alpha^{(1)}, \alpha = x, y, z \tag{76}$$

where

$$H_\alpha^{(1)} = \frac{1}{2}(OR_\alpha + R_\alpha O) \tag{77}$$

and $H^{(0)}$ is the DFTB Hamiltonian of the unperturbed system. Here, $O_{kl} = \int \phi_k^*(\mathbf{r})\phi_l(\mathbf{r})d\mathbf{r}$ is the basis set overlap matrix and R_α is the diagonal matrix with the atomic coordinates (in the $\alpha = x, y, z$ directions) for each atom-centered basis set orbital at position $\mathbf{R}$.

Orthogonalization by the square root of the inverse overlap matrix, $Z = O^{-1/2}$, leads to the transformed input Hamiltonian matrices

$$H^{(0)\perp} = Z^T H^{(0)} Z, H^{(1)\perp} = Z^T H^{(1)} Z \tag{78}$$

that can be used to demonstrate and evaluate the DNN-SP2-PT calculations with Tensor cores. In the first calculations below, we will only investigate non-self-consistent perturbation calculations of the linear response in the electron density represented by the first-order perturbation in the density matrix $D^{(0)}$. No coupled perturbed self-consistent field optimizations[5−8,13,15] are performed for the DFTB examples. The computational cost of repeated coupled-perturbed self-consistent-field iterations would depend linearly on the number of iterations required to reach some chosen convergence tolerance, which here can be ignored.

The use of Tensor cores makes a substantial difference in performance. Figure 4 displays the computational performance of the DNN-SP2-PT method when calculating the first order response for water systems of various sizes (periodic boxes
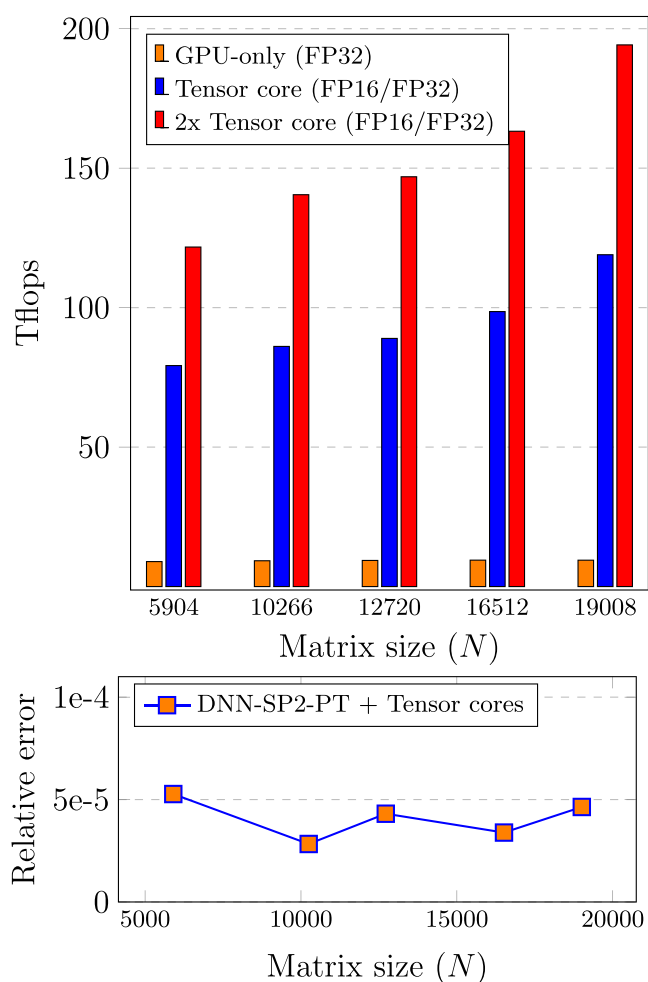
Figure 4. Top figure displays the flop rate for periodic box water systems with 66% occupation, $N_{occ} = (2/3) \times N$, as a function of number of orbitals, N, for the DNN-SP2-PT algorithm using only one Nvidia A100 GPU with Tensor cores (Tensor core) and two separate sets of A100 Tensor cores (2× Tensor core), along with a standard single-precision version using one A100 GPU's single-precision compute cores only (GPU-only). The theoretical flop rate for a single A100 GPU using the Tensor cores is 156 Tflops and for the A100 only using its FP32 compute cores is 9.75 Tflops.[84] Bottom figure displays $\| \cdot \|_2$ norm error in the first-order density matrix response produced by the DNN-SP2-PT method using Tensor cores on the test water boxes. The reference was obtained from a double-precision DNN-SP2-PT response calculation.

containing varying number of water molecules randomly distributed) using Nvidia A100 GPUs. For the water system with a Hamiltonian matrix size of 19,008 × 19,008 (corresponding to 3168 water molecules), the observed peak performance of the DNN-SP2-PT calculations on the Tensor cores of a single A100 GPU (not including memory data transfers or I/O) is approximately 120 Tflops, consistent with previous results for the DNN-SP2 algorithm.[24,27] Using the Tensor cores increases the flop rate by an order of magnitude in comparison to the GPU-only performance. Though, notice that the GPU calculations are single-precision (FP32) and the dual matrix representation that is needed to achieve a comparable level of accuracy approximately doubles the cost for the half-precision (FP16) calculations on the Tensor cores.

To further increase performance, we take advantage of the computational structure inherent in the DNN-DMPT formalism discussed above in Section 3.3. There, it was suggested that a multi-GPU approach could be used to exploit the unidirectional dependency in the response calculation. The two activation functions, $f^{(0)}$ and $f^{(1)}$, can utilize the Tensor cores of two separate A100 GPUs to calculate $D^{(0)}$ and $D^{(1)}$. The algorithm only requires that $S_n^{(0)}$ be passed, once, unidirectionally between the two GPUs at each step. Using the Tensor cores of two GPUs, in parallel, our peak performance is approximately 195 Tflops, an almost 63% gain in performance over a single A100.

The flop rate for both implementations, that is, the single GPU and the multi-GPU configuration, was estimated using the formula

$$\text{Est. flop rate} = \frac{5 \times N^3 \times \text{number of iterations}}{\text{time for main loop[s]}} \quad (79)$$

where timings for eq 79 were measured using CUDA event timers for a single run. The variation in time over consecutive runs is small, less than 5% for the smallest system and less than 1% for the largest one. The factor 5 comes from the fact that five total Tensor core matrix multiplications in half-precision are needed in each iteration using the dual matrix representation, as can be seen from eqs 14 and 44. The update to $S^{(0)}$ requires two Tensor core multiplications, and the update to $S^{(1)}$ requires three Tensor core multiplications.

To test the numerical accuracy of the DNN-SP2-PT method, a reference calculation using double-precision arithmetic instead of mixed precision is produced for each system of water. We then compare the density matrix response generated by the DNN-SP2-PT method using Tensor cores and the dual matrix representation with that of the reference calculation by computing the relative 2-norm error between the two. The result is shown in the bottom panel of Figure 4. These relative errors appear to be independent of system size and hover around 5×10^{-5}, which is about a factor 20 less than the machine epsilon for FP16 arithmetic. With 10 bits in the mantissa, FP16 provides about 3 digits accuracy, because $1/2^{10} \approx 10^{-3}$. Ideally, we would therefore have an accuracy of 6 digits with the dual matrix representation. Even if we do not fully reach this limit, we get an accuracy of $\sim 10^{-5}$. A thorough investigation of Tensor core FP16/FP32 precision can be found in refs 71 and 83.

Because the DNN-SP2-PT method calculates density matrix derivatives in the direction of the perturbation, we could, alternatively, compute these derivatives through finite differencing. How does such a direct approach compare to the

DNN-SP2-PT scheme? To investigate this, we consider two and four-point finite differencing stencils, with an accuracy of second and fourth order in the step size, h. The first-order density matrix $D^{(1)}$ can be then approximated using a finite difference approximation by

$$D^{(1)} = \frac{D(H^{(0)} + hH^{(1)}) - D(H^{(0)} - hH^{(1)})}{2h} + O(h^2)$$
(80)

or by

$$D^{(1)} = \frac{-D(H^{(0)} + 2hH^{(1)}) + 8D(H^{(0)} + hH^{(1)})}{12h}$$
$$+ \frac{-8D(H^{(0)} - hH^{(1)}) + D(H^{(0)} - 2hH^{(1)})}{12h}$$
$$+ O(h^4)$$
(81)

for a suitably chosen step sizes h and where each evaluation of D uses regular DNN-SP2 with Tensor core acceleration. The second order approximation costs about the same as a first-order DNN-SP2-PT calculation (4 instead of 5 matrix multiplications in each layer), whereas the four-point stencil would take about twice as long (8 instead of 5 multiplications). A water box with $N = 5904$ orbitals is used for testing. Figure 5
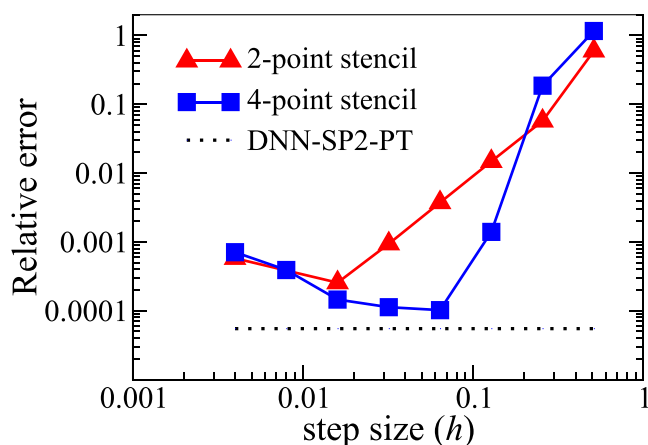

Figure 5. Standard finite differencing 2-point and 4-point stencils for approximating the first-order term in the density matrix response, eqs 80 and 81. The dotted line shows the relative error obtained from using our Tensor core-based DNN-SP2-PT approach.

shows the relative error in approximating $D^{(1)}$ by using both finite differencing stencils for several values of step size. The second order 2-point stencil approximation yields an almost order-of-magnitude decrease in accuracy, for the best possible case, but requires finding the optimal value of step size, h, which depends on machine epsilon for the numerical precision of the algorithm. Using the higher order 4-point stencil yields better accuracy, though for a different h value, but is still a factor 2 less accurate than our DMPT framework even for the most optimal step size displayed. The four-point stencil requires 4 separate density matrix calculations, which represents a considerable overhead compared to the DNN-SP2-PT calculation. From this analysis, it is clear that our DNN-SP2-PT method offers significant advantages. No search for an optimal step size h is needed (which is non-trivial due to the mixed FP16/FP32 precision), the numerical accuracy is higher, and the cost is significantly lower compared to the

more accurate four-point stencil finite difference approximation.

4.3. Hartree-Fock Based Polarizability Calculations. In the demonstration and analysis of the DNN-SP2-PT approach above, we used SCC-DFTB theory with periodic boundary conditions. In this section, we use restricted Hartree−Fock theory, where the response is given from the solution of the coupled-perturbed self-consistent field equations.[5,6,13,15] Molecular clusters without periodic boundary conditions were used. To generate the relevant self-consistent Fockians (i.e., the effective single-particle Hamiltonians) and their first-order perturbations, we have used the electronic structure software package Ergo.[85] Our calculations only include the last step of this coupled-perturbed self-consistent field calculation, with the already converged self-consistent Fockian, and its response obtained from double-precision arithmetic. As a scalable test bed system, we have used water clusters of approximately spherical shape and the Gaussian 6-31G** basis set. The water clusters were extracted from a molecular dynamics simulation of bulk water at standard temperature and pressure by including all water molecules within spheres of varying radius. The water cluster geometries are publicly available at ergoscf. org. The coupled perturbed self-consistent field iterations were converged to within a tolerance $\|Y - Y^T\|_F < 10^{-3}$, $Y = (H^{(0)}D^{(1)} + H^{(1)}D^{(0)})O$, for the first-order commutation conditions.

Table 1 shows the calculated isotropic polarizabilities using the DNN-SP2-PT algorithm with Tensor cores in low-mixed-

Table 1. Isotropic Polarizabilities, $(\alpha_{xx} + \alpha_{yy} + \alpha_{zz})/3$, of Water Clusters Using DNN-SP2-PT with Tensor Cores[a]

N	N_{occ}	DMPT TC FP16	Ergo reference CPU FP64
1128	235	−286.118378567	−286.104481906
1824	380	−463.263240067	−463.26617634
2400	500	−610.91459163	−610.92738903
4560	950	−1169.0446183	−1169.104407943
7224	1505	−1856.85780567	−1856.9148556467

[a]Double-precision Ergo calculations on CPUs are given as reference values, which use the regular SP2 Fermi-operator expansion scheme for the density matrix construction. Here, N is the number of basis functions and N_{occ} is the occupation number. The 6-31G** basis set is used for the Ergo calculations.

precision, FP16/FP32 floating-point arithmetics, in comparison to the Ergo reference CPU calculations in double-precision. The accuracy (4 to 5 digits) is about the same as for the SCC-DFTB calculations with the same system-size independence. This is the expected accuracy for the dual matrix representations using mixed half-precision calculations with single-precision accumulation as in eqs 43 and 44.

5. CONCLUSIONS

We have shown how time-independent quantum response calculations based on density matrix perturbation theory can be generated through a deep neural network. Tensor cores, a new type of accelerated hardware developed for machine learning, are ideally suited for computations using this network structure. Our results add to the nascent, yet growing, body of work that utilizes specialized machine learning hardware for non-AI scientific purposes, e.g., see refs 24−27, and 31−33. With this article, we have further broadened the applicability of Tensor cores to quantum response calculations, which

represents yet another example of a more general science application. This again demonstrates their, mostly untapped, potential for accelerating numerical methods in chemistry and materials science.

Despite the low-precision arithmetics of Tensor cores, we maintain sufficient accuracy for DFTB-based linear response calculations while obtaining significant performance gains, more than an order of magnitude, over what can be achieved with a high-performance GPU-only implementation. The computational structure of density matrix perturbation theory and our corresponding deep neural network formulation are uniquely suited toward leveraging Tensor cores. This structural framework further allows us to design a natural multi-GPU approach to perturbation theory. With two separate intranode A100 devices, we achieved close to 200 Tflops when computing the density matrix and its first order response.

Though we have focused our applications to Nvidia Tensor core computations, our DNN-SP2-PT scheme is quite general and similar gains in performance should be expected from other accelerated hardware architectures such as TPUs and Matrix cores.[22,23]

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.2c00274.

> Appendix section: Kohn–Sham density-functional theory and the density matrix (PDF)

> Complete Python implementation of linear response calculation using DNN-SP2-PT including Python implementation of pseudocode in Algorithms 1–3 (PDF)

AUTHOR INFORMATION

Corresponding Authors

Joshua Finkelstein − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States;* ⓞ orcid.org/0000-0002-1506-6008; Email: jdf@lanl.gov

Emanuel H. Rubensson − *Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala SE-751 05, Sweden;* ⓞ orcid.org/0000-0002-7686-2075; Email: emanuel.rubensson@it.uu.se

Anders M. N. Niklasson − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States;* Email: amn@lanl.gov

Authors

Susan M. Mniszewski − *Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States;* ⓞ orcid.org/0000-0002-0077-0537

Christian F. A. Negre − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States*

Complete contact information is available at:
https://pubs.acs.org/10.1021/acs.jctc.2c00274

Notes

The authors declare no competing financial interest.

REFERENCES

(1) Schrödinger, E. Quantisierung als Eigenwertproblem. *Ann. Phys.* **1926**, *385*, 437−490.

(2) Hylleraas, E. A. Über den Grundterm der Zweielektronenproblbeme von H−, He, Li+, Be++ usw. *Z. Phys.* **1930**, *65*, 209−225.

(3) Sternheimer, R. M. Electronic Polarizabilities of Ions from the Hartree-Fock Wave Functions. *Phys. Rev.* **1954**, *96*, 951−968.

(4) Karplus, M. Weak Interactions in Molecular Quantum Mechanics. *Rev. Mod. Phys.* **1960**, *32*, 455−460.

(5) Stevens, R. M.; Pitzer, R. M.; Lipscomb, W. N. Perturbed Hartree—Fock Calculations. I. Magnetic Susceptibility and Shielding in the LiH Molecule. *J. Chem. Phys.* **1963**, *38*, 550−560.

(6) Pople, J. A.; Krishnan, R.; Schlegel, H. B.; Binkley, J. S. Derivative studies in Hartree-Fock and Møller-Plesset theories. *Int. J. Quant. Chem.* **1979**, *16*, 225−241.

(7) Baroni, S.; Giannozzi, P.; Testa, A. Green's-function approach to linear response in solids. *Phys. Rev. Lett.* **1987**, *58*, 1861−1864.

(8) Baroni, S.; de Gironcoli, S.; Dal Corso, A.; Giannozzi, P. Phonons and related crystal properties from density-functional perturbation theory. *Rev. Mod. Phys.* **2001**, *73*, 515−562.

(9) Helgaker, T.; Jorgensen, P.; Olsen, J. *Molecular Electronic-Structure Theory*, 1st ed.; John Wiley and Sons Ltd: New York, 2002.

(10) Sakurai, J.; Napolitano, J. *Modern Quantum mechanics*, 2nd edition; Cambridge University Press, 2011.

(11) McWeeny, R. Perturbation Theory for the Fock-Dirac Density Matrix. *Phys. Rev.* **1962**, *126*, 1028−1034.

(12) Niklasson, A. M. N.; Challacombe, M. Density Matrix Perturbation Theory. *Phys. Rev. Lett.* **2004**, *92*, No. 193001.

(13) Weber, V.; Niklasson, A. M. N.; Challacombe, M. Ab Initio Linear Scaling Response Theory: Electric Polariz-ability by Perturbed Projection. *Phys. Rev. Lett.* **2004**, *92*, No. 193002.

(14) Niklasson, A. M. N.; Weber, V.; Challacombe, M. Non-Orthogonal Density Matrix Perturbation Theory. *J. Chem. Phys.* **2005**, *123*, 44107.

(15) Weber, V.; Niklasson, A. M. N.; Challacombe, M. Higher Order Response in O(N) by Perturbed Projection. *J. Chem. Phys.* **2005**, *123*, 44106.

(16) Niklasson, A. M. N.; Weber, V. Linear scaling density matrix perturbation theory for basis-set-dependent quantum response calculations: An orthogonal formulation. *J. Chem. Phys.* **2007**, *127*, No. 064105.

(17) Niklasson, A. M. N.; Cawkwell, M. J.; Rubensson, E. H.; Rudberg, E. Canonical Density Matrix Perturbation Theory. *Phys. Rev. E* **2015**, *92*, No. 063301.

(18) Truflandier, L. A.; Dianzinga, R. M.; Bowler, D. R. Notes on density matrix perturbation theory. *J. Chem. Phys.* **2020**, *153*, 164105.

(19) Shang, H.; Duan, X.; Li, F.; Zhang, L.; Xu, Z.; Liu, K.; Luo, H.; Ji, Y.; Zhao, W.; Xue, W.; Chen, L.; Zhang, Y. Many-core acceleration of the first-principles all-electron quantum perturbation calculations. *Comput. Phys. Commun.* **2021**, *267*, No. 108045.

(20) Shang, H.; Li, F.; Zhang, Y.; Zhang, L.; Fu, Y.; Gao, Y.; Wu, Y.; Duan, X.; Lin, R.; Liu, X.; Liu, Y.; Chen, D. Extreme-scale ab initio quantum Raman spectra simulations on the leadership HPC system in China. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 2021; pp 1−13.

(21) NVIDIA corporation, *Nvidia Tensor Cores.* https://www.nvidia.com/en-us/data-center/tensor-cores/, Accessed: 2021-6-8.

(22) *Google, Tensor Processing Units.* https://cloud.google.com/tpu/docs/system-architecture, 2021; .

(23) *Advanced Micro Devices, AMD CDNA^{TM} 2 Architecture.* https://www.amd.com/en/technologies/cdna2, Accessed: 2022-3-10.

(24) Finkelstein, J.; Smith, J. S.; Mniszewski, S. M.; Barros, K.; Negre, C. F. A.; Rubensson, E. H.; Niklasson, A. M. N. Mixed Precision Fermi-Operator Expansion on Tensor Cores from a Machine Learning Perspective. *J. Chem. Theory Comput.* 2021, 17, 2256−2265.

(25) Shang, H.; Xiang, H.; Li, Z.; Yang, J. Linear scaling electronic structure calculations with numerical atomic basis set. *Int. Rev. Phys. Chem.* 2010, 29, 665−691.

(26) Shang, H.; Liang, W.; Zhang, Y.; Yang, J. Efficient parallel linear scaling method to get the response density matrix in all-electron real-space density-functional perturbation theory. *Comput. Phys. Commun.* 2021, 258, No. 107613.

(27) Finkelstein, J.; Smith, J. S.; Mniszewski, S. M.; Barros, K.; Negre, C. F. A.; Rubensson, E. H.; Niklasson, A. M. Quantum-Based Molecular Dynamics Simulations Using Tensor Cores. *J. Chem. Theory Comput.* 2021, 17, 6180−6192.

(28) Abdelfattah, A.; Tomov, S.; Dongarra, J. Fast Batched Matrix Multiplication for Small Sizes Using Half-Precision Arithmetic on GPUs. *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*; IEEE 2019; pp. 111−122.

(29) Haidar, A.; Bayraktar, H.; Tomov, S.; Dongarra, J.; Higham, N. J. Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems. *Proc. R. Soc. A* 2020, 476, 20200110.

(30) Li, B.; Cheng, S.; Lin, J. tcFFT: Accelerating Half-Precision FFT through Tensor Cores. *arXiv preprint arXiv:2104.11471* 2021.

(31) Lewis, A. G. M.; Beall, J.; Ganahl, M.; Hauru, M.; Mallick, S. B.; Vidal, G. Large Scale Distributed Linear Algebra With Tensor Processing Units. *arXiv:2112.09017* 2021.

(32) Hauru, M.; Morningstar, A.; Beall, J.; Ganahl, M.; Lewis, A.; Vidal, G. *Simulation of quantum physics with Tensor Processing Units: brute-force computation of ground states and time evolution. arXiv preprint arXiv:2111.10466.* 2022.

(33) Morningstar, A.; Hauru, M.; Beall, J.; Ganahl, M.; Lewis, A. G. M.; Khemani, V.; Vidal, G. Simulation of quantum many-body dynamics with Tensor Processing Units: Floquet prethermalization. *PRX Quantum* 2022, No. 020331.

(34) Pederson, R.; John Kozlowski, R. S.; Beall, J.; Ganahl, M.; Hauru, M.; Lewis, A. G.; Mallick, S. B.; Blum, V.; Vidal, G. Tensor Processing Units as Quantum Chemistry Supercomputers. *arXiv preprint arXiv:2202.01255* 2021.

(35) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. GPU-accelerated molecular modeling coming of age. *J. Mol. Graphics Modell.* 2010, 29, 116−125.

(36) Germann, T. C.; Kadau, K.; Swaminarayan, S. 369 Tflop/s molecular dynamics simulations on the petaflop hybrid super-computer 'Roadrunner'. *Concurr. Comput.* 2009, 21, 2143−2159.

(37) Ufimtsev, I. S.; Martínez, T. J. Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation. *J. Chem. Theory Comput.* 2008, 4, 222−231.

(38) Ufimtsev, I. S.; Martínez, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *J. Chem. Theory Comput.* 2009, 5, 1004−1015.

(39) Ufimtsev, I. S.; Martínez, T. J. Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics. *J. Chem. Theory Comput.* 2009, 5, 2619−2628.

(40) Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units (GPUs). *J. Chem. Theory Comput.* 2011, 7, 949−954.

(41) Maia, J. D. C.; Carvalho, G. A. U.; Mangueira, C. P.; Santana, S. R.; Cabral, L. A. F.; Rocha, G. B. GPU Linear Algebra Libraries and GPGPU Programming for Accelerating MOPAC Semiempirical Quantum Chemistry Calculations. *J. Chem. Theory Comput.* 2012, 11, 3131−3144.

(42) Hacene, M.; Anciaux-Sedrakian, A.; Rozanska, X.; Klahr, D.; Guignon, T.; Fleurat-Lessard, P. Accelerating VASP electronic structure calculations using graphic processing units. *J. Chem. Theory Comput.* 2012, 33, 2581−2589.

(43) Liu, F.; Luehr, N.; Kulik, H. J.; Martínez, T. J. Quantum chemistry for solvated molecules on graphical processing units using polarizable continuum models. *Comput. Phys. Commun.* 2015, 8, 3072−3081.

(44) Huhn, W. P.; Lange, B.; Yu, V. W.; Yoon, M.; Blum, V. GPU acceleration of all-electron electronic structure theory using localized numeric atom-centered basis functions. *Comput. Phys. Commun.* 2020, 254, No. 107314.

(45) Gordon, M. S.; Windus, T. L. Modern Architectures and Their Impact on Electronic Structure Theory. *Chem. Rev.* 2020, 120, 9015−9020.

(46) Zhou, G.; Nebgen, B.; Lubbers, N.; Malone, W.; Niklasson, A. M. N.; Tretiak, S. Graphics Processing UnitAccelerated Semiempirical Born Oppenheimer Molecular Dynamics Using PyTorch. *J. Chem. Theory Comput.* 2020, 16, 4951−4962.

(47) Genovese, L.; Ospici, M.; Deutsch, T.; Mehaut, J.-F.; Neelov, A.; Goedecker, S. Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures. *J. Chem. Phys.* 2009, 131, No. 034103.

(48) *Still waiting for Exascale: Japan's Fugaku outperforms all competition once again* https://top500.org. Accessed: 2022-3-10.

(49) Fock, V. Naherungsmethode zur Losung des quanten-mechanischen Mehrkorperproblems. *Z. Phys.* 1930, 61, 126−148.

(50) Roothaan, C. New developments in molecular orbital theory. *Rev. Mod. Phys.* 1951, 23, 69.

(51) Hohenberg, P.; Kohn, W. Inhomgenous electron gas. *Rev. Mod. Phys.* 1951, 23, 69.

(52) Parr, R. G.; Yang, W. *Density-functional theory of atoms and molecules*; Oxford University Press: Oxford, 1989.

(53) Dreizler, R.; Gross, K. *Density-functional theory*; Springer Verlag: Berlin Heidelberg, 1990.

(54) McWeeny, R. The density matrix in self-consistent field theory .1. Iterative construction of the density matrix. *Proc. R. Soc. London Ser. A* 1956, 235, 496.

(55) NVIDIA corporation, *cuBLAS.* https://developer.nvidia.com/cuBLAS, 2021; Accessed: 2021-4-15.

(56) Negre, C. F. A.; Mniszewski, S. M.; Cawkwell, M. J.; Bock, N.; Wall, M. E.; Niklasson, A. M. N. Recursive Factorization of the Inverse Overlap Matrix in Linear-Scaling Quantum Molecular Dynamics Simulations. *J. Chem. Theory Comput.* 2016, 12, 3063.

(57) Niklasson, A. M. N. Iterative refinement method for the approximate factorization of a matrix inverse. *Phys. Rev. B* 2004, 70, No. 193102.

(58) Rubensson, E. H.; Bock, N.; Holmtromm, E.; Niklasson, A. M. N. Recursive inverse factorization. *J. Chem. Phys.* 2008, 128, 104105.

(59) Benzi, M.; Meyer, C. D.; Tuma, M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.* 1996, 17, 1135−1149.

(60) Rubensson, E. H.; Artemov, A. G.; Kruchinina, A.; Rudberg, E. Localized inverse factorization. *IMA J. Numer. Anal.* 2021, 41, 729−763.

(61) McWeeny, R. The density matrix in many-electron quantum mechanics I. Generalized product functions. Factorization and physical interpretation of the density matrices. *Proc. R. Soc. London, Ser. A* 1959, 253, 242−259.

(62) Niklasson, A. M. N. In *Linear-Scaling Techniques in Computational Chemistry and Physics: Methods and Applications*; Zalesny, R., Papadopoulos, M. G., Mezey, P. G., Leszczynski, J., Eds.; Springer Netherlands: Dordrecht, 2011; pp. 439−473.

(63) Higham, N. J. *Functions of matrices: theory and computation*; SIAM, 2008.

(64) Niklasson, A. M. N. Expansion algorithm for the density matrix. *Phys. Rev. B* 2002, 66, No. 155115.

(65) Rubensson, E. H. Nonmonotonic recursive polynomial expansion for linear scaling calculation of the density matrix. *J. Chem. Theory Comput.* **2011**, *7*, 1233.

(66) Rubensson, E. H.; Niklasson, A. M. N. Interior eigenvalues from density matrix expansions in quantum mechanical molecular dynamics. *SIAM J. Sci. Comput.* **2014**, *36*, 148.

(67) Schulz, G. Z. Iterative berechung der reziproken matrix. *Z. Angew. Math. Mech.* **1933**, *13*, 57.

(68) Bowler, D. R.; Miyazaki, T. O(N) methods in electronic structure calculations. *Rep. Prog. Phys.* **2012**, *75*, 036503−036546.

(69) Kruchinina, A.; Rudberg, E.; Rubensson, E. H. Parameterless Stopping Criteria for Recursive Density Matrix Expansions. *J. Chem. Theory Comput.* **2016**, *12*, 5788−5802.

(70) Markidis, S.; Chien, S. W. D.; Laure, E.; Peng, I. B.; Vetter, J. S. NVIDIA Tensor Core Programmability, Performance amp; Precision. *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*; IEEE 2018; pp. 522−531.

(71) Ootomo, H.; Yokota, R. Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance. *arXiv preprint arXiv:2203.03341* **2022**.

(72) Ochsenfeld, C.; Head-Gordon, M. A reformulation of the coupled perturbed self-consistent field equations entirely within a local atomic orbital density matrix-based scheme. *Chem. Phys. Lett.* **1997**, *270*, 399−405.

(73) Mathias, R. A Chain Rule for Matrix Functions and Applications. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 610−620.

(74) Daletskii, J. L.; Krein, S. Integration and differentiation of functions of Hermitian operators and applications to the theory of perturbations. *AMS Translations (2)* **1965**, *47*, 10−1090.

(75) Porezag, D.; Frauenheim, T.; Köhler, T.; Seifert, G.; Kaschner, R. Construction of tight-binding-like potentials on the basis of density-functional theory: Application to carbon. *Phys. Rev. B* **1995**, *51*, 12947−12957.

(76) Elstner, M.; Porezag, D.; Jungnickel, G.; Elsner, J.; Haugk, M.; Frauenheim, T.; Suhai, S.; Seifert, G. Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties. *Phys. Rev. B* **1998**, *58*, 7260−7268.

(77) Finnis, M. W.; Paxton, A. T.; Methfessel, M.; van Schilfgarde, M. Crystal structures of zirconia from first principles and self-consistent tight binding. *Phys. Rev. Lett.* **1998**, *81*, 5149.

(78) Frauenheim, T.; Seifert, G.; Aand, Z.; Hajnal, M. E.; Jungnickel, G.; Poresag, D.; Suhai, S.; Scholz, R. A self-consistent charge density-functional based tight-binding method for predictive materials simulations in physics, chemistry and biology. *Phys. Stat. sol.* **2000**, *217*, 41.

(79) Hourahine, B.; et al. DFTB+, a software package for efficient approximate density functional theory based atomistic simulations. *J. Chem. Phys.* **2020**, *152*, 124101.

(80) Bock, N.; Cawkwell, M. J.; Coe, J. D.; Krishnapriyan, A.; Kroonblawd, M. P.; Lang, A.; Liu, C.; Saez, E. M.; Mniszewski, S. M.; Negre, C. F. A.; Niklasson, A. M. N.; Sanville, E.; Wood, M. A.; Yang, P. *LATTE.* 2008; https://github.com/lanl/LATTE.

(81) King-Smith, R. D.; Vanderbilt, D. Theory of polarization of crystalline solids. *Phys. Rev. B* **1993**, *47*, 1651−1654.

(82) Resta, R. Macroscopic Electric Polarization as a Geometric Quantum Phase. *Europhysics Letters (EPL)* **1993**, *22*, 133−138.

(83) Fasi, M.; Higham, N. J.; Mikaitis, M.; Pranesh, S. Numerical behavior of NVIDIA tensor cores. *PeerJ Computer Science* **2021**, *7*, No. e330.

(84) NVIDIA *NVIDIA A100 Tensor Core GPU Architecture.* https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf, Accessed: 2021-4-28.

(85) Rudberg, E.; Rubensson, E. H.; Salek, P.; Kruchinina, A. Ergo: An open-source program for linear-scaling electronic structure calculations. *SoftwareX* **2018**, *7*, 107−111.

■ NOTE ADDED AFTER ASAP PUBLICATION

Due to a production error, this paper was published ASAP on June 7, 2022 with Algorithms 1−3 missing from the paper. The corrected version was reposted on June 7, 2022.

⬡ Recommended by ACS