JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

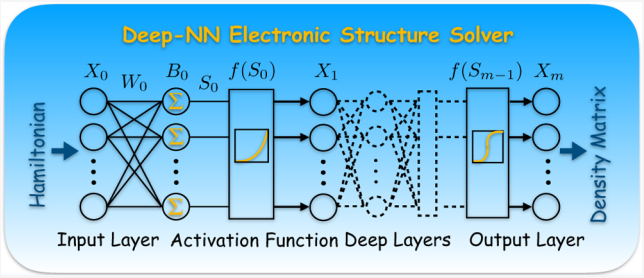
Mixed Precision Fermi-Operator Expansion on Tensor Cores from a Machine Learning Perspective

Joshua Finkelstein,* Justin S. Smith, Susan M. Mniszewski, Kipton Barros, Christian F. A. Negre,* Emanuel H. Rubensson, and Anders M. N. Niklasson*

Cite This: J. Chem. Theory Comput. 2021, 17, 2256−2265 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: We present a second-order recursive Fermi-operator expansion scheme using mixed precision floating point operations to perform electronic structure calculations using tensor core units. A performance of over 100 teraFLOPs is achieved for half-precision floating point operations on Nvidia's A100 tensor core units. The second-order recursive Fermi-operator scheme is formulated in terms of a generalized, differentiable deep neural network structure, which solves the quantum mechanical electronic structure problem. We demonstrate how this network can be accelerated by optimizing the weight and bias values to substantially reduce the number of layers required for convergence. We also show how this machine learning approach can be used to optimize the coefficients of the recursive Fermi-operator expansion to accurately represent the fractional occupation numbers of the electronic states at finite temperatures.



1. INTRODUCTION

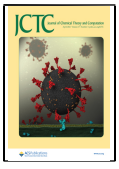
Electronic structure calculations based on Hartree−Fock, density functional theory, or semiempirical methods often require the intermediate construction of the single-particle density matrix.[1−3] This density matrix can be calculated with different techniques, for example, the use of direct diagonalization of the Kohn−Sham Hamiltonian or the Fockian,[4] Green's function methods,[5] variational optimization,[6] and various recursive Fermi-operator expansion schemes.[2,7,8] The method of choice often depends on several criteria such as (1) the electronic structure basis set, for example, if plane waves or localized atomic orbitals are used; (2) the system that is analyzed, for example, if the system is small or large or if it is metallic or nonmetallic; and (3) the computational platform, for example, if the calculation is performed on a single or multiple central processing units (CPUs) or on a hybrid architecture with graphics processing units (GPUs).

In this article, we target density matrix calculations for electronic structure methods on tensor core units with an adapted atomic-orbital-like basis set for intermediate sized nonmetallic systems. For these calculations, we use a second-order recursive Fermi-operator expansion scheme in combination with mixed precision floating point operations, which enables efficient calculations of the density matrix using tensor core accelerators.[9] The density matrix is required in the calculation of energies, expectation values, and, for example, the force evaluations in molecular dynamics simulations. The ability to use tensor cores for the Fermi-operator expansion therefore affects a broad range of possible applications.

Exploring the use of tensor core-based architectures for electronic structure calculations follows the previous transitioning from CPU-only-based electronic structure techniques to the more specialized GPU-based techniques.[10−20] Our Fermi-operator expansion scheme is formulated and presented in terms of a generalized convolutional deep neural network.[21,22] This network formulation provides a powerful machine learning perspective on how we can further optimize and extend the applications of the recursive Fermi-operator expansion. We find that we can optimize the weight and bias values and use a combination of multiple layers to represent Fermi functions at finite electronic temperatures with high numerical accuracy. We also find that an optimized set of weight and bias values can reduce the number of layers required to reach convergence.

This article is outlined as follows. First, we discuss the electronic structure problem and the Fermi-operator representation of the density matrix. Then, we present a second-order recursive Fermi-operator expansion method in terms of a generalized deep neural network using mixed precision floating point operations that are well adapted for tensor core calculations. We then demonstrate and analyze the perform-

Received: January 19, 2021
Published: April 2, 2021



https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

ance on tensor core units for some test examples. Thereafter, we discuss how optimized weight and bias values can be used to accelerate convergence and how they accurately represent the Fermi function for fractional occupation numbers at finite electronic temperatures. The algorithms are presented in pseudo-code throughout the article and implementations in Python are available in the Supporting Information document.

2. DENSITY MATRIX FERMI-OPERATOR EXPANSION

2.1. Density Matrix. The single-particle density matrix, D, is given in terms of the Fermi matrix function, where

$$D = (e^{\beta(H-\mu I)} + I)^{-1} \tag{1}$$

Here, I is the identity matrix, $\beta = 1/(k_B T_e)$ is the inverse electronic temperature, μ is the chemical potential, and $H \in \mathbf{R}^{N\times N}$ is the Hamiltonian (or Fockian) matrix with matrix elements

$$H_{i,j} = \langle \phi_i | \hat{H} | \phi_j \rangle \tag{2}$$

For simplicity, we assume an orthonormal basis set $\{\phi_i\}_{i=1}^{N}$, so that the overlap matrix $S_{i,j} = \langle \phi_i | \phi_j \rangle = \delta_{i,j}$. An orthonormalized basis set representation can always be constructed from a congruence transform based on the inverse factorization of the overlap matrix.[23] The operator $\hat{H}$ is the effective single-particle Hamiltonian operator involved in methods such as Hartree−Fock or Kohn−Sham density functional theory. At zero electronic temperature, $T_e = 0$, the Fermi function becomes a shifted Heaviside step function, $\theta(\cdot)$, and the density matrix reads as

$$D = \theta(\mu I - H) \tag{3}$$

There are several methods that can be used to calculate the density matrix. The traditional method is based on a direct diagonalization of H, that is, finding the orthonormal eigenstates Q such that

$$Q^T H Q = E, \quad E_{ij} = \varepsilon_i \delta_{ij}, \quad Q^T Q = I \tag{4}$$

In the diagonal (eigenvector) representation, E, and the identity matrix, I, are diagonal and the matrix exponential and inversion can therefore be calculated directly,

$$\begin{aligned} D &= QQ^T (e^{\beta(H-\mu I)} + I)^{-1} QQ^T \\ &= Q(e^{\beta(E-\mu I)} + I)^{-1} Q^T \end{aligned} \tag{5}$$

where the chemical potential μ is adjusted to account for the desired orbital occupation, N_{occ}, such that $\mathrm{Tr}[D] = N_{occ}$. Alternatively, in the zero-temperature limit, the density matrix becomes

$$\begin{aligned} D &= QQ^T \theta(\mu I - H) QQ^T \\ &= Q\theta(\mu I - E) Q^T \end{aligned} \tag{6}$$

where the shifted Heaviside step function, $\theta(\mu I - E)$, is evaluated on each of the diagonal elements of E (eigenvalues of H). The chemical potential needs to be shifted to reach a desired occupation also in this case.

2.2. Serial Fermi-Operator Expansions. An alternative to construct the density matrix, D, is the serial Chebyshev Fermi-operator expansion scheme,[24−26] where the density matrix is approximated by a linear combination of Chebyshev matrix polynomials of the Hamiltonian, $T_n(\bar{H})$,

$$D = (e^{\beta(H-\mu I)} + I)^{-1} \approx \sum_{n=1}^{m} c_n T_n(\bar{H}) \tag{7}$$

where $\bar{H}$ is a linearly transformed Hamiltonian matrix H such that its eigenvalue spectrum is within the interval $[-1, 1]$. Alternatively, we may also use a Green's function expansion, which is based on a complex contour integration[27−31] with some complex energy mesh $\{z_n\}$, so that

$$D = (e^{\beta(H-\mu I)} + I)^{-1} \approx \sum_{n=1}^{m} c_n (H - z_n I)^{-1} \tag{8}$$

In both these serial Fermi-operator expansion methods, the coefficients $\{c_n\}_{n=1}^{m}$ need to be adjusted such that the approximate density matrix has the correct occupation and temperature. To reach accurate representations, high-order expansions are required and convergence can be hard to achieve at low temperatures. Higher-order polynomials are needed for the Chebyshev expansion, and the Green's function expansion requires complex energies close to the real axis, which may lead to singularity problems for low-temperature expansions. However, the Chebyshev and Green's function methods can take advantage of sparse matrix algebra for sufficiently large Hamiltonian matrices, which allow computations with linear scaling complexity.[32,33] Chebyshev methods can sometimes also take advantage of a stochastic sampling of expectation values using a smaller randomized trial basis. In these cases, the $O(N^3)$ cubic scaling cost of the diagonalization, a function of the system size N, can be replaced by a linear $O(N)$ scaling complexity.[31−34] This is a particular advantage for very large problems such as those including tens of thousands of atoms or electrons. For smaller problems, the construction of the density matrix with direct diagonalization is typically much faster.

2.3. Recursive Fermi-Operator Expansion. At zero electronic temperature, a Fermi-operator expansion scheme has to approximate the Heaviside step function of H, with the step formed at the chemical potential. In this zero-temperature limit, Chebyshev and Green's function methods can have convergence problems.[24,25] An alternative is given by a recursive Fermi-operator expansion,[7,35−43] where

$$D = \theta(\mu I - H) \approx f_m(f_{m-1}(\ldots f_0(H)\ldots)) \tag{9}$$

The recursion can be performed by successive projections of a matrix X_i that starts with $X_0 = H$ and then $X_{i+1} = f_i(X_i)$ is calculated in each iteration until convergence is reached as $X_i \to D$. The functions $f_i(X_i)$ are chosen to project the eigenvalue spectrum of X_i to a more pure ensemble with eigenvalues closer either to 1 or to 0. The occupation is 1 for the occupied states below the chemical potential μ and 0 for the unoccupied states above. These types of recursive Fermi-operator expansion methods are also referred to as purification or spectral projection schemes.[2,37,44]

If we use sparse matrix algebra, there is an overhead associated with a fill-in of nonzero matrix elements during the recursive expansion. In particular, for metallic systems, the density matrix will become full even if the original Hamiltonian is sparse. However, here we will deal only with dense (nonsparse) matrix algebra. Our main concern is therefore only the number of iterations needed. The advantage of a recursive Fermi-operator expansion is that we can reach a very high polynomial order in the approximation with only a few

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

numbers of iterations. There are many choices of polynomials and techniques to adjust the expansion such that the step is formed at the chemical potential. Possibly, the simplest and most efficient technique is the second-order spectral projection (SP2) method,[37,40,42,45] which is the main focus of this article.

3. SP2 METHOD

3.1. Second-Order Spectral Projection Polynomials. In the SP2 method, the recursive expansion functions in eq 9 are chosen as second-order polynomials acting on the interval [0, 1]. In the original version of SP2

$$X_{i+1} = f_i(X_i) = X_i \pm (X_i - X_i^2) \tag{10}$$

The ± sign is chosen to adjust the trace of X_{i+1} in each projection such that the correct occupation, N_{occ}, is reached at convergence, that is, such that $\mathrm{Tr}[X_i] \to \mathrm{Tr}[D] = N_{occ}$.[37] In this way, the step is formed automatically at the correct chemical potential μ. No prior knowledge of the chemical potential is therefore required and no post-processing adjustment is needed. The polynomial expansion order doubles in each recursion. In only 30 recursion steps, the polynomial expansion order is over 1 billion. The second-order polynomials in eq 10 are continuously increasing and decreasing functions on the expansion interval [0, 1]. The expansion therefore automatically avoids any type of Gibbs oscillations that are sometimes a problem in Chebyshev expansion methods.[25,34] A truncated version of the SP2 scheme, where the recursion is terminated before the convergence to an idempotent solution with integer occupation numbers is reached, can also be used to approximate the Fermi operator at elevated electronic temperatures.[46]

The second-order polynomial projection functions in eq 10 can be modified by a shift-and-scale transformation that accelerates the expansion.[45] To guarantee stability and convergence, this acceleration technique requires prior knowledge of the two eigenvalues right above and right below the chemical potential [the eigenvalues corresponding to the highest occupied molecular orbitals (HOMOs) and the lowest unoccupied molecular orbitals (LUMOs)] or at least some fairly accurate estimate of their values.[42] For repeated applications of the SP2 algorithm, which is necessary, for example, in molecular dynamics simulations, rigorous estimates of the HOMO and LUMO eigenvalues can be calculated from each previous SP2 Fermi-operator expansion.[42] In this case, only the first SP2 expansion cannot use the shift-and-scale acceleration technique because there is no prior knowledge of the HOMO and LUMO energies.

In our generalized deep neural network representation of the SP2 scheme, presented in the next section, we will show how an accelerated convergence can be obtained from an optimization of the weight and bias values of the network. This machine learning perspective can also be used to optimize the coefficients and re-weight the different layers of the SP2 expansion to get a highly accurate representation of the Fermi function at finite electronic temperatures with fractional occupation numbers. This offers a significant improvement over the truncated SP2 scheme that has previously been used to approximate the Fermi function at finite temperatures.[46]

3.2. Deep-NN SP2. There are several ways to implement the SP2 recursive Fermi-operator expansion of eq 9, using the second-order projection polynomials in eq 10. Here, we will present a version that naturally maps onto the algorithmic structure of a generalized convolutional deep neural network (Deep-NN).

The original SP2 expansion has two key properties if we assume all eigenvalues of $X_1 \in [0, 1]$: (1) it converges to a step function with the step formed somewhere in the interval [0, 1] and (2) each projection step either increases or decreases the trace of X_i by projecting the eigenstates either toward a stationary point at 1 or toward a stationary point at 0. An initial linear transform, $X_1 = f_0(H)$, is chosen to scale the eigenstates of H to the interval [0, 1] in reverse order. Following this initial transform, we can choose the projection polynomials that improve the convergence of the trace after each recursion. When the trace corrections no longer improve the occupation or when all the eigenvalues of X_i are as close as possible to 0 or 1, convergence has been reached and the expansion is terminated. At this point, we may use the converged density matrix X_m to calculate various quantum mechanical observables, $\langle A \rangle = \mathrm{Tr}[X_m A]$, where A is the matrix representation of the relevant operator, for example, the Hamiltonian matrix, H, for the energy. It is easy to see how this scheme can be reformulated and mapped onto the structure of a Deep-NN as shown in Figure 1. In the first layer, we use the Hamiltonian $X_0 = H$ as the input descriptor. The weight and bias functions, W_0 and B_0, are then chosen such that S_0 is the rescaled Hamiltonian with eigenvalues in reverse order inside the interval [0, 1]. As an activation function, we chose the matrix function $f(S_0) = S_0^2$, which acts on the eigenvalues of S_0. This is in contrast to regular neural networks where the activation function acts on the individual matrix elements. The matrix square operation of the activation function consists of a single tensor contraction, that is, a matrix−matrix multiplication, which is an advantage since tensor cores are optimized to perform such operations at high speed. At the subsequent layer, where $X_1 = f(S_0)$, we chose the weight and bias values such that $S_1 = W_1 X_1 + B_1$, with $W_1 = \sigma_1 I$ and $B_1 = (\mathrm{I} - W_1)S_0$. The value of $\sigma_1 = \pm 1$ is chosen such that S_1 has the smallest occupation error, $|\mathrm{Tr}[S_1] - N_{occ}|$, of the two sign alternatives. These operations are continued layer by layer and the i-th approximation to the density matrix is computed as follows

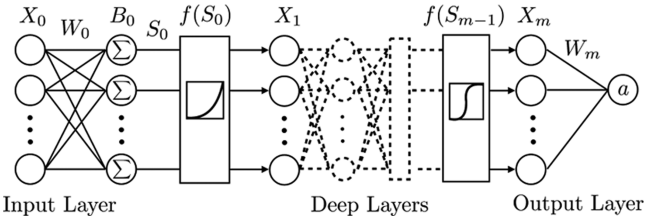


Figure 1. Schematic picture of a deep neural network. The weights W_n and bias values B_n generate a linear transformation of X_n, where $S_n = W_n X_n + B_n$ are given in standard matrix notation. A new layer X_{n+1} is provided after the application of a nonlinear activation function, that is, where $X_{n+1} = f(S_n)$.

$$X_i = f(\ldots f(W_1 f(W_0 X_0 + B_0) + B_1)\ldots) \tag{11}$$

Once the occupation error (estimated by an idempotency measure) has converged to sufficient accuracy, we have reached the last layer, S_{m-1}. The density matrix is then outputted as $D = X_m$.

The Deep-NN formulation of the SP2 algorithm is given in pseudo-code in Algorithm 1 and also includes a parameter-free

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

check for convergence. The convergence is determined from where an expected decrease, under exact arithmetic, of the estimated idempotency error, IdErr_n, is no longer fulfilled in practice.

A motivation for and precise derivation of the convergence criterion is provided in the appendix. Typically, the idempotency error is computed as $\|X - X^2\|$, where $\|\cdot\|$ is either the spectral (2-norm) or the Frobenius norm. Here, we have instead chosen to use $\mathrm{Tr}[X - X^2]$ as the measure of the idempotency error, which is a simpler and more computationally efficient measure. In fact, since $\|X - X^2\|_2 \leq \mathrm{Tr}[X - X^2] \leq N\|X - X^2\|_2$ whenever the eigenvalues of X are in $[0, 1]$, the convergence in $\mathrm{Tr}[X - X^2]$ is equivalent to the convergence in the spectral norm. Only a single $O(N)$ trace operation is needed in each deep layer.

In Algorithm 1, we also include a small constant ϵ which ensures that we get an alternating sign of σ_n if the occupation corrections are very small, inducing faster convergence. Otherwise, the sign, σ_n, is chosen to minimize the occupation error in S_n. The inclusion of the small ϵ term is an ad hoc adjustment that, in general, is not a necessity of the convergence criteria. A Python script implementing the full Deep-NN SP2 algorithm is presented in the Supporting Information.

Algorithm 1 The Deep-NN formulation of the SP2 recursive Fermi-operator expansion algorithm. A Python script is provided in the supplementary information.

N_{occ}, Number of occupied states or orbitals
ϵ, small number close (or equal) to 0
H, Orthonormalized Hamiltonian
ε_1, ε_N, Spectral bound estimates of H
W_n, Observable operator matrix, e.g. $W_n = H$
$X_0 = H$, Input layer
$W_0 = -(\varepsilon_N - \varepsilon_1)^{-1} I$, $B_0 = \varepsilon_N(\varepsilon_N - \varepsilon_1)^{-1} I$
$S_0 = W_0 X_0 + B_0$, $N_S = \mathrm{Tr}[S_0]$, Occupation of S_0
$n = 0$, Number of layers
while Not Converged **do**
  $n = n + 1$
  $X_n = f(S_{n-1})$, See Alg. 2
  $N_X = \mathrm{Tr}[X_n]$
  $\mathrm{IdErr}_n = N_S - N_X$, Idempotency error estimate
  $\sigma_n = \mathrm{Sign}\left(|2N_S - N_X - N_{\mathrm{occ}}| - |N_X - N_{\mathrm{occ}}| - \sigma_{n-1}\epsilon\right)$
  $W_n = \sigma_n I$, $B_n = (I - W_n)S_{n-1}$
  $S_n = W_n X_n + B_n$
  $N_S = W_n N_X + (1 - \sigma_n)N_S$, Updated occupation of S_n
  if $\mathrm{IdErr}_n <= 0$ **then**
    Converged = **true**
  else if $n > 2$ **and** $\sigma_{n-1} \neq \sigma_{n-2}$ **and** $\mathrm{IdErr}_n > 4.5 \times (\mathrm{IdErr}_{n-2})^2$ **then**
    Converged = **true**
  end if
end while
$D = f_{\mathrm{final}}(S_{m-1})$, final X_m layer in Fig. 1
$\alpha = \mathrm{Tr}[DW_m]$, Output observable

3.3. Mixed Precision Operations. The computationally dominant step in the Deep-NN SP2 Fermi-operator expansion is the calculation of the matrix square in the activation function. In dense matrix algebra, such generalized matrix−matrix multiplications can often be performed with very high performance on almost any computational platform. The SP2 scheme therefore stands out as an efficient alternative to diagonalization-based density matrix calculations. Here, we are interested in using tensor core calculations. Tensor core units have been tailored to perform tensor contractions, that is matrix−matrix multiplications, for machine learning applications using convolutional deep neural networks with close to peak performance. Recently, Nvidia's V100 tensor core accelerated GPU broke the 100 teraFLOPs barrier for deep learning applications.[47] Our goal is to use such tensor core accelerators for the calculation of density matrices using the Deep-NN SP2 Fermi-operator expansion.

The tensor core units use low, mixed precision floating point operations. Typically, only half-precision operations with single-precision accumulation are used. The half-precision is in general too low in accuracy for meaningful density matrix calculations, but a single precision accuracy is good enough for many problems. To achieve single precision accuracy, we can represent a single-precision matrix X with a pair of two half-precision matrices,

$$X \approx X^{(0)} + X^{(1)} \tag{12}$$

Using pseudo-code notation, the corresponding dual half-precision representation of a matrix X would be generated by

$$\begin{aligned} X^{(0)} &= FP16[X] \\ X^{(1)} &= FP16[X - X^{(0)}] \end{aligned} \tag{13}$$

where FP16[ ] denotes the half-precision representation. Generalizations to a higher level of accuracy using multiple matrices, $X^{(n)}$, is straightforward and will not be discussed. A matrix−matrix multiplication, $X \times Y$, can then be performed using four separate matrix−matrix multiplications in half-precision with accumulation in single precision (FP32[ ]), that is,

$$\begin{aligned} X \times Y &\approx FP32[(X^{(0)} + X^{(1)})(Y^{(0)} + Y^{(1)})] \\ &= FP32[X^{(0)}Y^{(0)} + (X^{(1)}Y^{(0)} + X^{(0)}Y^{(1)}) \\ &\quad + X^{(1)}Y^{(1)}] \end{aligned} \tag{14}$$

In the Deep-NN SP2 Fermi-operator scheme in Algorithm 1, we only need to calculate matrix squares in the activation function. If we assume that each matrix X is symmetric and neglect the small $X^{(1)}Y^{(1)}$ term, we can reduce the calculation of a matrix square to only two matrix−matrix multiplications in half-precision and single accumulation, that is,

$$X^2 \approx FP32[X^{(0)}X^{(0)} + X^{(0)}X^{(1)} + (X^{(0)}X^{(1)})^T] \tag{15}$$

All the matrix products and sums are assumed to be accumulated in single precision (FP32). This approach would also benefit from multiplications of symmetric matrices where only the upper or lower half of the matrix needs to be calculated.

3.4. Mixed Precision Deep-NN SP2. To adjust the Deep-NN SP2 algorithm in Algorithm 1 to mixed precision floating point operations, we only need to adjust the activation function, $f(X) = X^2$, where the matrix square is performed using tensor contractions on tensor core units in half-precision with single accumulation. This is described by Algorithm 2.

There are two sources of error. The main source of error in the mixed precision Deep-NN SP2 scheme is the eigenvalue distribution at convergence. Because of the finite precision, the eigenstates will not be exactly 1 or 0, corresponding to fully occupied or unoccupied states. This may lead to significant

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

Algorithm 2 Calculation of the activation function, $f(X) = X^2$, in the Deep-NN SP2 scheme in Alg. 1 for tensor core units in half precision and single accumulation.

X Input matrix in single precision
$X^{(0)} = \mathrm{FP16}[X]$
$X^{(1)} = \mathrm{FP16}[X - X^{(0)}]$
$A = \mathrm{FP32}[X^{(0)} \times X^{(0)}]$ Tensor core multiplication
$B = \mathrm{FP32}[X^{(0)} \times X^{(1)}]$ Tensor core multiplication
$f(X) = X^2 \approx \mathrm{FP32}[A + B + B^T]$

errors in energy calculations. However, these errors can be reduced by a post-processing step. The post-processing refinement can be achieved by using a modified activation function in the final step, $f_{\mathrm{final}}(S_n)$, in Algorithm 1. Instead of the matrix square, we use

$$f_{\mathrm{final}}(S_n) = \begin{cases} 2S_n^2 - S_n^4, & \text{if } \sigma_{n-2} = 1 \\ (2S_n - S_n^2)^2, & \text{if } \sigma_{n-2} = -1 \end{cases} \tag{16}$$

which is calculated in an enhanced precision, either standard double precision or in single precision with double accumulation. The refinement in eq 16 is the result of composing two layers in a single step with $\sigma_n = 1$ and $\sigma_{n-1} = -1$ or $\sigma_n = -1$ and $\sigma_{n-1} = 1$. These alternating signs of σ provide for a guaranteed second-order decrease in the error if exact floating point operations are used. The remaining source of error is a small corruption in the eigenvectors due to numerical errors that is manifested in a nonzero commutation, $DH - HD$, between the converged density matrix and the Hamiltonian. In exact arithmetic, $DH - HD = 0$.

Figure 2 shows the convergence in the energy, $\mathrm{Tr}[X_n H]$, with the error compared to the "exact" energy and the idempotency error measured by the spectral norm, $\|X_n^2 - X_n\|_2$. The rapid improvement by multiple orders of magnitude in the last layer is given by the final refinement step performed in an enhanced precision which scales quadratically from $n - 1$ to n. The enhanced precision is often necessary to attain a sufficiently high numerical accuracy.

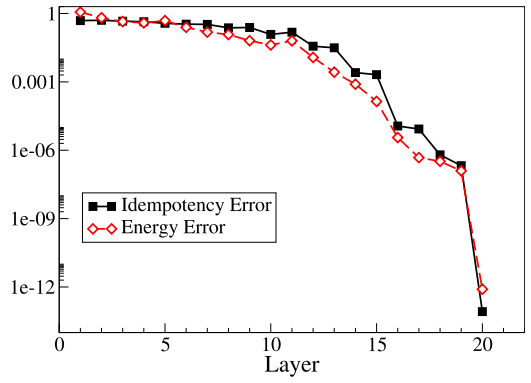


Figure 2. Idempotency convergence of the Deep-NN SP2 Fermi-operator expansion scheme for a small uniformly randomized symmetric Hamiltonian ($H \in \mathbf{R}^{10\times10}$, $N_{\mathrm{occ}} = 5$, $H_{ij} \in [-1, 1]$) using the mixed precision algorithm. The last layer corresponds to the final refinement activation function.

3.5. Convergence Estimate for Low-Precision Floating Point Operations. Using only half-precision floating point operations in the Deep-NN SP2 scheme leads to fairly large errors compared to regular double-precision operations, even if the dual mixed precision, presented above, is used. Thanks to the post-processing refinement step the final error is significantly reduced. However, we first need to determine when convergence is reached. This can be difficult to decide under numerically noisy conditions caused by the low-precision floating point operations. The idea we use to determine convergence is based on the observation that the idempotency estimate we use in Algorithm 1, that is, $\mathrm{IdErr}_n = \mathrm{Tr}[S_{n-1} - X_n] = \mathrm{Tr}[S_{n-1} - S_{n-1}^2]$, decreases quadratically between every second step if we have alternating signs of σ_n. However, limitations in the finite precision will, at some point, prevent the expected decay of the idempotency error. At this point, the expansion can then be terminated because the best possible convergence has been reached. This parameter-free convergence estimate is both efficient and easy to implement.

Our convergence criterion is analogous to the parameterless stopping criteria by Kruchinina, Rudberg, and Rubensson,[48] which here has been adapted to a different idempotency measure. Details of the derivation are given in the Appendix.

4. MIXED PRECISION FERMI-OPERATOR EXPANSION ON TENSOR CORES

To make use of tensor cores with matrix multiplications, the Deep-NN SP2 algorithm was written using CUDA v11.0, the cuBLAS library,[49] and several customized kernels. All matrix multiplications in the SP2 algorithm were carried out using cuBLAS general dense matrix−matrix multiplication (GEMM) calls. The GEMM calls execute tensor core operations automatically and no special commands are required to make use of them. This automatic feature can be disabled with the appropriate cuBLAS application programming interface (API) call. As test matrices, we used a generic set of $N \times N$ Hamiltonian matrices with matrix elements

$$H_{i,j} = \mathrm{e}^{-1/2|i-j|}\sin(i + j), \qquad i, j = 1, 2, \ldots N \tag{17}$$

The test Hamiltonians have a near-constant spectral range, approximately $[-1.867, 1.867]$, and a HOMO−LUMO gap which scales inversely to the system size N. The number of layers required to reach convergence only scales logarithmically with the size of the inverse gap.[37] The system size N was increased until the performance peaked before an abrupt drop-off because of memory limitations. For the V100 unit, the largest matrix size was approximately 14,000 × 14,000, and for the A100 unit, the limit was reached for matrix sizes of about 20,000 × 20,000.

Implementation of the half-precision multiplications needed by the X^2 activation function in Algorithm 2 required several custom kernels. These kernels decompose the matrix X into a sum of two FP16 matrices that are then multiplied and summed as described in eqs 12 through 15 using standard cuBLAS GEMM routines. Custom kernels were also necessary to reduce, as much as possible, the amount of data transfer between the host and the GPU device memory. The Deep-NN SP2 CUDA implementation is available through the PROGRESS[50] library.

The rate of floating point operations (FLOP) for the Deep-NN SP2 algorithm was estimated from simulations using tensor cores on both Nvidia A100 and V100 GPUs and is shown in Figure 3. This estimate does not include the initialization and memory allocation of the routine nor the final layer, the double-precision refinement step. For purposes of comparison, this FLOP rate was also computed on the V100

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

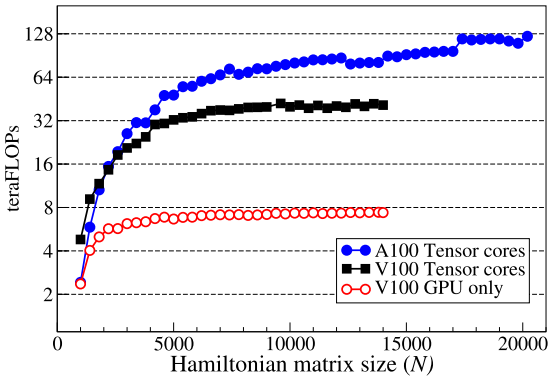


Figure 3. Half-precision tera-floating point operations per second (teraFLOPs) vs Hamiltonian system size for the Deep-NN SP2 Fermi-operator expansion running on Nvidia's Volta V100 tensor core units, on its Volta GPU only, and on the more recent A100 tensor core units. The difference in maximum N values is due to device memory limitations. The test Hamiltonian is provided in eq 17.

unit with the tensor cores disabled; we call this the GPU-only FLOP rate; it is displayed in Figure 3 as well. We observe an approximate 7−8× speed up on the V100 unit when tensor cores are enabled versus when they are disabled and only the GPU is used. Even more impressive, we achieve approximately 120 teraFLOPs on the A100 unit when utilizing tensor cores. At this FLOP rate, the convergence is reached within 4 s of wall-clock time for a 20,000 × 20,000 system.

Although the plots in Figure 3 suggest that the Deep-NN SP2 algorithm may only be beneficial for large N values, a recent publication[51] shows how matrix−matrix multiplications can reach high performance also for smaller N by utilizing batching techniques. The same technique would most likely benefit the SP2 method for small N. Additionally, further speed-up can be achieved by considering a sparse matrix implementation[52] of the Deep-NN SP2 method.

5. CAPITALIZING ON THE MACHINE LEARNING PERSPECTIVE

There are several observations that appear from the machine learning perspective of the SP2 Fermi-operator expansion scheme when it is formulated in terms of a layered network structure: (1) the quantum mechanical problem is solved naturally and with high efficiency through the computational structure of a generalized deep neural network; (2) the bias and weight values could be optimized using machine learning techniques to achieve improved convergence and possibly higher accuracy; (3) other functions besides the matrix Heaviside step function could potentially be approximated through the same generalized deep neural network, including Fermi functions at finite electronic temperatures; (4) recursive Fermi-operator schemes or sign matrix expansions based on higher-order spectral projection polynomials could be mapped onto the same generalized network structure and use the same mixed precision technique; and (5) a recursive calculation of Green's functions via a Dyson series expansion could also be generalized to fit into the algorithmic structure of Deep-NN SP2, for example, $G = G_0 + G_0VG$ (where G, G_0, and V are the Green function, the initial Green function, and a perturbation to the Hamiltonian, respectively) can be rewritten recursively in a similar way to the SP2 scheme, where the corresponding weights and bias values could be optimized for convergence.

Here, we will briefly discuss the ability to accelerate convergence for the Deep-NN SP2 algorithm and how approximate Fermi functions for fractional occupation numbers at elevated electronic temperatures can be generated recursively with high accuracy.

5.1. Accelerated Deep-NN SP2. In machine learning, we try to learn the weight and bias functions by optimizing a regularized penalty function based on, for example, some large set of predetermined data. Here, we may instead use the convergence rate to the idempotent density matrix. Each layer of the Deep-NN SP2 scheme can then be seen as generalized spectral projections with weights W_n and bias values B_n. Instead of choosing the spectral projections from $W_n = \sigma_n I$, where $\sigma_n = \pm 1$, we may optimize over a continuous set of values, $\sigma_n \in \mathbf{R}$, as illustrated in Figure 4. To optimize

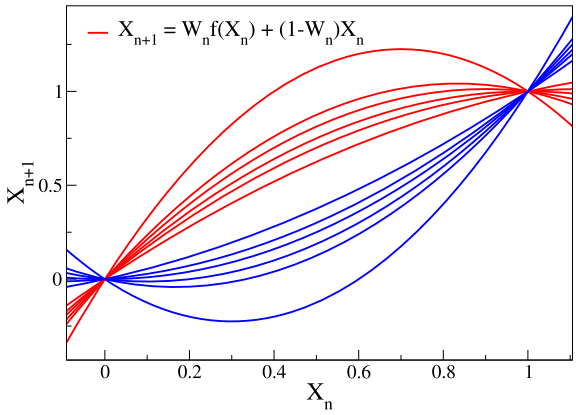


Figure 4. Generalized Deep-NN SP2 projections in each layer, $X_n \rightarrow X_{n+1}$, with various weight values $W_n = \sigma_n I$, with $\sigma_n \in \mathbf{R}$, instead of $\sigma_n = \pm 1$. By locally optimizing σ_n in each layer and using a shift and re-scale transforms to keep eigenvalues within the interval $[0, 1]$, new weight and bias values for the accelerated Deep-NN formulation of the SP2 expansion are generated. A Python script for the accelerated Deep-NN SP2 algorithm is presented in the Supporting Information.

convergence, we chose the values of W_n, which in each separate layer gives the highest slope of the projection around the re-scaled eigenvalues corresponding to the HOMO or LUMO eigenvalues, but without risking switching places between occupied and unoccupied eigenvalues. This local choice of optimization accelerates the separation of the HOMO and LUMO eigenstates, which are the last to reach the fixed points at 1 and 0. This optimization requires prior knowledge of the re-scaled HOMO and LUMO eigenvalues. The optimized spectral projections may push eigenvalues outside of the $[0, 1]$ interval, which could lead to divergence. To avoid this, we need to shift and re-scale the eigenvalue spectrum to $[0, 1]$ after each optimized projection. The combined transform from the choice of σ_n-values, followed by the shift and re-scaling, determines the optimized weight, W_n, and bias values, B_n, in each layer. This local optimization of the weight and bias values of each layer can lead to a significant acceleration. Our accelerated Deep-NN SP2 algorithm is presented as a Python script in the Supporting Information, and the optimized choices of W_n and B_n for the network defined by $X_{n+1} = f(W_nX_n + B_n)$ are given there explicitly. This accelerated Deep-NN SP2 scheme turns out to be an equivalent Deep-NN formulation of the accelerated SP2 Fermi-operator expansion by Rubensson, which uses a shift and re-scale technique.[42,45] However, here we arrive at the same acceleration scheme but based on the

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

Deep-NN perspective and with a different combination of spectral projection polynomials and choice of shift and re-scale transformations. An example of the convergence accelerated Deep-NN SP2 scheme is shown in Figure 5. The convergence is reached after 17 layers instead of 28, which is a significant improvement.

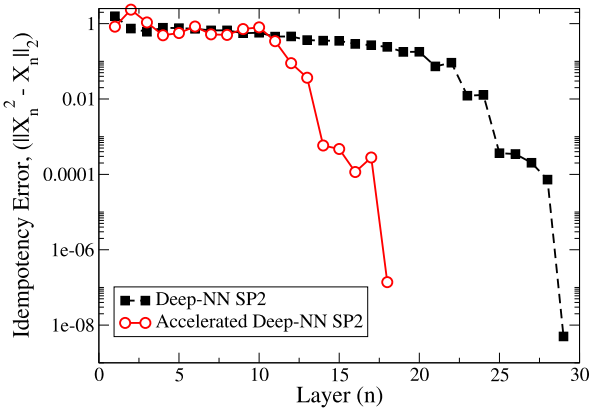


Figure 5. Idempotency error, $\|X_n^2 - X_n\|_2$, for a test Hamiltonian, $H \in \mathbf{R}^{100\times100}$, with $N_{occ} = 10$, with and without acceleration. A Python script of the accelerated Deep-NN SP2 scheme is presented in the Supporting Information.

A disadvantage with the accelerated Deep-NN SP2 scheme is that it requires prior knowledge of the HOMO−LUMO eigenvalues. However, for repeated calculations of the density matrix, for example, in molecular dynamics simulations, the HOMO−LUMO eigenvalues can be estimated from previous time steps with a high level of accuracy.[42] Further acceleration of the Deep-NN SP2 scheme can possibly be achieved by tailoring the optimization of the weight and bias values for Hamiltonian matrices with particular eigenvalue distributions.

5.2. Optimized SP2 Finite Temperature Fermi-Operator Expansion. A recursive SP2 expansion that is stopped before it has reached convergence generates a smooth approximation to the Heaviside step function $\theta(\varepsilon - \mu)$ using the eigenvalues $\varepsilon_i \in [0, 1]$ of the re-scaled Hamiltonian matrix H. This occurs because the second-order spectral projection functions are smooth on the interval $[0, 1]$. These truncated SP2 expansions have previously been used to approximate the Fermi function at elevated electronic temperatures.[46] However, seeing the truncated SP2 scheme in terms of a deep neural network allows for a straightforward optimization. By generalizing the second-order spectral projection functions to a more general second-order polynomial and then optimizing the coefficients in each layer, we may achieve more accurate approximations of the Fermi function than those available to a truncated SP2 scheme alone. In this way, we can also optimize the convergence rate and minimize the error compared to an exact Fermi function.

Instead of using the alternating SP2 projection polynomials x^2 and $2x - x^2$, as is in eq 10, we allow for general second-degree polynomials on $[0, 1]$ to generate an approximation to the Fermi function at any $\varepsilon \in [0, 1]$. We define the initial and n-th layer to be

$$\varepsilon_0 = \varepsilon,$$

$$\varepsilon_n = \theta_{n-1,2}\varepsilon_{n-1}^2 + \theta_{n-1,1}\varepsilon_{n-1} + \theta_{n-1,0} \tag{18}$$

Equation 18 corresponds to the n-th matrix recursion step, where $X_n = \theta_{n-1,2}X_n^2 + \theta_{n-1,1}X_n + \theta_{n-1,0}I$. To increase model flexibility, a linear combination of the intermediate values, $\sum_{i=0}^{n} c_i\varepsilon_i$, is used to enhance the approximation; that is, at the end, we approximate the density matrix as $D = \sum_{i=0}^{n} c_i X_n$. Taking into account the folding of the eigenspectrum by the SP2 scheme, the Fermi function approximation then becomes

$$\tilde{F}(\varepsilon) := 1 - \sum_{i=0}^{n} c_i\varepsilon_i \tag{19}$$

Subsequently, the $\theta_{i,j}$ and c_i are trained to minimize the mean-squared error over a pre-selected grid, $\{\varepsilon_i\}_{i=0}^{N}$, on $[0, 1]$

$$\mathcal{L}(\tilde{F}) = \sum_i w_i[\tilde{F}(\varepsilon_i) - F(\varepsilon_i)]^2 \approx \int_0^1 [\tilde{F}(\varepsilon) - F(\varepsilon)]^2 d\varepsilon \tag{20}$$

We use the Levenberg−Marquardt (LM) optimization method, which is designed specifically for a sum-of-squares loss function. LM dynamically blends the Gauss−Newton method, yielding quadratic convergence where possible, and gradient descent, slower, but having more robust convergence guarantees.

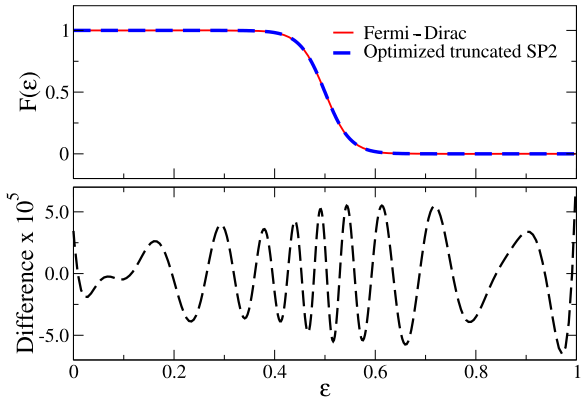


Figure 6. Result from learning the Fermi function with 11 layers and $\beta = 40$. The exact Fermi function (red) is compared with the learned one (dashed blue) in the top panel. The bottom panel shows the error as a function of the scaled energy ε.

Figure 6 shows an example of a globally optimized truncated SP2 recursive Fermi-operator scheme in comparison to the corresponding Fermi−Dirac function

$$F(\varepsilon) = (e^{\beta(\varepsilon-1/2)} + 1)^{-1} \tag{21}$$

with $\beta = 40$. The approximation error is shown in the lower panel. Previously, we have been limited to the use of recursive Fermi-operator expansions that are based on rational Pade' polynomials as their projections to reach this level of accuracy.[38,53] However, these schemes are implicit and require a solution to a system of equations in each iteration. Here, we are able to achieve a similar level of accuracy using the explicit machine-learned generalized SP2 expansion scheme as presented in eq 18.

6. CONCLUSIONS

We have demonstrated how the solution to the quantum mechanical electronic structure problem, for example,

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

appearing in Hartree−Fock and Kohn−Sham density functional theory, can be mapped onto the computational structure of a generalized deep neural network. The solution, in terms of an effective single-particle density matrix, is generated by a recursive Fermi-operator expansion derived from a second-order spectral projection scheme. The main computational bottleneck of the layered network is dominated by the activation function, a matrix square operation, which can be performed with high efficiency on tensor core units using a mixed precision formulation that enhances the intrinsic half-precision floating point operations. A single-precision matrix−matrix multiplication in the activation function is replaced by two half-precision matrix−matrix multiplications, allowing us to make full use of the available tensor core architectures. This leads to an impressive speed-up of about 16× for the calculation of density matrices with respect to the same generation GPUs.

By capitalizing on the machine learning perspective of the deep neural network formulation of the recursive second-order SP2 Fermi-operator expansion, we were able to both accelerate the rate of convergence, by optimizing the weights of the neural net, and apply machine learning techniques to closely approximate Fermi−Dirac functions at finite electronic temperatures.

APPENDIX

Here, we state the result used to justify our parameter-free convergence criterion in Algorithm 1, which is based on a bound of the worst-case error reduction on the general form $\text{Error}_i \leq C(\text{Error}_{i-2})^2$ for some constant C. We use the estimate of the idempotency error, $\text{IdErr}_i = \text{Tr}[S_{i-1} - S_{i-1}^2]$, for the error measure Error_i. We then determine that convergence occurs once the estimated error reduction (in eq 22 below) no longer holds with the available precision of the floating point operations. It is always valid in exact arithmetic. The theory is analogous to the previous parameter-free convergence criterion by Kruchinina et al.,[48] which was based on a different measure of the idempotency error.

Theorem 1. Assume that $\sigma_i \neq \sigma_{i-1}$ so that either $S_i = (2S_{i-2} - S_{i-2}^2)^2$ or $S_i = 2S_{i-2}^2 - S_{i-2}^4$ and $i > 1$. Assume also that S_{i-2} has all eigenvalues in $[0, 1]$. Then

$$\text{IdErr}_{i+1} = \text{Tr}[S_i - S_i^2] \leq C(\text{Tr}[S_{i-2} - S_{i-2}^2])^2 = C(\text{IdErr}_{i-1})^2 \tag{22}$$

with $C = \frac{1}{32}(71 + 17\sqrt{17}) \approx 4.41$.

Proof. Let $\{\lambda_j^{(i)}\}_{j=1}^N$ be the eigenvalues of S_i, where the ordering of eigenvalues is such that $\lambda_j^{(i)} = \lambda_j^{(i-1)} + \sigma_i(\lambda_j^{(i-1)} - (\lambda_j^{(i-1)})^2), \quad j = 1, ..., N$. From ref 48 we have that

$$\max_{\lambda\in(0,1)} \frac{(2\lambda - \lambda^2)^2 - (2\lambda - \lambda^2)^4}{(\lambda - \lambda^2)^2} \tag{23}$$

$$= \max_{\lambda\in(0,1)} \frac{2\lambda^2 - \lambda^4 - (2\lambda^2 - \lambda^4)^2}{(\lambda - \lambda^2)^2} = C \tag{24}$$

which, given that $\sigma_i \neq \sigma_{i-1}$, means for $j = 1, ..., N$

$$\lambda_j^{(i)} - (\lambda_j^{(i)})^2 \leq C(\lambda_j^{(i-2)} - (\lambda_j^{(i-2)})^2)^2 \tag{25}$$

Summing over all eigenvalues

$$\text{IdErr}_{i+1} = \text{Tr}[S_i - S_i^2] \tag{26}$$

$$= \sum_{j=1}^N \lambda_j^{(i)} - (\lambda_j^{(i)})^2 \tag{27}$$

$$\leq \sum_{j=1}^N C(\lambda_j^{(i-2)} - (\lambda_j^{(i-2)})^2)^2 \tag{28}$$

$$= C\left(\left(\sum_{j=1}^N \lambda_j^{(i-2)} - (\lambda_j^{(i-2)})^2\right)^2 - \sum_{j=1}^N \sum_{k\neq j} (\lambda_j^{(i-2)} - (\lambda_j^{(i-2)})^2) \times (\lambda_k^{(i-2)} - (\lambda_k^{(i-2)})^2)\right) \tag{29}$$

$$\leq C(\text{Tr}[S_{i-2} - S_{i-2}^2])^2 \tag{30}$$

$$= C(\text{IdErr}_{i-1})^2 \tag{31}$$

Note that C is not the asymptotic error constant, but since C is finite, the result implies the established quadratic convergence for sequences with alternating polynomials in the limit of idempotent matrices S_i.[37,42]

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.1c00057.

We present two fully implementable Python scripts for the accelerated and non-accelerated, Deep-NN SP2 algorithm. This example code is highly documented and equivalent to the CUDA implementation discussed in the main body of the article (ZIP)

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

Christian F. A. Negre − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States*; Email: cnegre@lanl.gov

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

Authors

Justin S. Smith − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States*

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

Kipton Barros − *Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, United States*

Emanuel H. Rubensson − *Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala SE-751 05, Sweden*

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265

Complete contact information is available at:
https://pubs.acs.org/10.1021/acs.jctc.1c00057

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work is supported by the U.S. Department of Energy Office of Basic Energy Sciences (FWP LANLE8AN), the LANL LDRD-ER program, and by the U.S. Department of Energy through the Los Alamos National Laboratory. We thank the CCS-7 group and Darwin cluster at Los Alamos National Laboratory for computational resources. Darwin is funded by the Computational Systems and Software Environments (CSSE) subprogram of LANL's ASC program (NNSA/DOE). We are thankful to Nicolas Bock for his advice on code development.

■ REFERENCES

(1) McWeeny, R. The Density Matrix in Self-Consistent Field Theory I. Iterative Construction of the Density Matrix. *Proc. R. Soc. London, Ser. A* **1956**, *235*, 496.
(2) McWeeny, R. Some Recent Advances in Density Matrix Theory. *Rev. Mod. Phys.* **1960**, *32*, 335.
(3) Finnis, M. *Interatomic Forces in Condensed Matter*; OUP Oxford, 2003.
(4) Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry*, 1st ed.; Mc Graw−Hill Inc.: New York, 1989.
(5) Economou, E. N. *Green's Functions in Quantum Physics*; Springer Science & Business Media, 2006.
(6) Payne, P. W. Appropriate Constraints for Variational Optimization of Electronic Density Matrices and Electron Densities. *Proc. Natl. Acad. Sci. U.S.A.* **1982**, *79*, 6391−6395.
(7) Palser, A. H. R.; Manolopoulos, D. E. Canonical Purification of the Density Matrix in Electronic Structure Theory. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1998**, *58*, 12704.
(8) Bowler, D. R.; Gillan, M. J. Density Matrices in O(N) Electronic Structure Calculations: Theory and Applications. *Comput. Phys. Commun.* **1999**, *120*, 95−108.
(9) NVIDIA Tensor Cores. https://www.nvidia.com/en-us/data-center/tensor-cores/ (accessed Dec 22, 2020).
(10) 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.
(11) 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.
(12) 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.
(13) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. GPU-accelerated Molecular Modeling Coming of Age. *J. Mol. Graph. Model.* **2010**, *29*, 116−125.
(14) Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units. *J. Chem. Theory Comput.* **2011**, *7*, 949−954.
(15) 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**, *8*, 3072−3081.
(16) 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.
(17) 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**, *11*, 3131−3144.
(18) Huhn, W. P.; Lange, B.; Yu, V. W.-z.; Yoon, M.; Blum, V. GPU Acceleration of All-electron Electronic Structure Theory Using Localized Numeric Atom-centered Basis Functions. *Comput. Phys. Commun.* **2020**, *254*, 107314.
(19) Gordon, M. S.; Windus, T. L. Modern Architectures and Their Impact on Electronic Structure Theory. *Chem. Rev.* **2020**, *120*, 9015−9020.
(20) Zhou, G.; Nebgen, B.; Lubbers, N.; Malone, W.; Niklasson, A. M. N.; Tretiak, S. Graphics Processing Unit-accelerated Semiempirical Born-Oppenheimer Molecular Dynamics Using PyTorch. *J. Chem. Theory Comput.* **2020**, *16*, 4951−4962.
(21) Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Network.* **2015**, *61*, 85−117.
(22) Higham, C. F.; Higham, D. J. Deep Learning: An Introduction for Applied Mathematicians. *SIAM Rev.* **2019**, *61*, 860−891.
(23) 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.
(24) Silver, R. N.; Röder, H. Densities of States of Mega-dimensional Hamiltonian Matrices. *Int. J. Mod. Phys. C* **1994**, *05*, 735.
(25) Weiße, A.; Wellein, G.; Alvermann, A.; Fehske, H. The Kernel Polynomial Method. *Rev. Mod. Phys.* **2006**, *78*, 275−306.
(26) Goedecker, S.; Colombo, L. Efficient Linear Scaling Algorithm for Tight-Binding Molecular Dynamics. *Phys. Rev. Lett.* **1994**, *73*, 122.
(27) Zeller, R.; Deutz, J.; Dederichs, P. Application of Complex Energy Integration To Self-Consistent Electronic Structure Calculations. *Solid State Commun.* **1982**, *44*, 993.
(28) Bernstein, N. Linear Scaling Nonorthogonal Tight-binding Molecular Dynamics for Nonperiodic Systems. *Europhys. Lett.* **2001**, *55*, 52.
(29) Goedecker, S. Integral Representation of the Fermi Distribution and its Applications in Electronic Structure Calculations. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1993**, *48*, 17573.
(30) Ozaki, T. Continued Fraction Representation of the Fermi-Dirac Function for Large-scale Electronic Structure Calculations. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2007**, *75*, 035123.
(31) Lin, L.; Chen, M.; Yang, C.; He, L. Accelerating Atomic Orbital-based Electronic Structure Calculation via Pole Expansion and Selected Inversion. *J. Phys. Condens. Matter* **2013**, *25*, 295501.
(32) Goedecker, S. Linear Scaling Electronic Structure Methods. *Rev. Mod. Phys.* **1999**, *71*, 1085−1123.
(33) Bowler, D. R.; Miyazaki, T. O(N) Methods in Electronic Structure Calculations. *Rep. Prog. Phys.* **2012**, *75*, 036503−036546.
(34) Silver, R. N.; Roeder, H.; Voter, A. F.; Kress, J. D. Kernel Polynomial Approximations for Densities of States and Spectral Functions. *Int. J. Comput. Phys.* **1996**, *124*, 115.
(35) Németh, K.; Scuseria, G. E. Linear Scaling Density Matrix Search Based on Sign Matrices. *J. Chem. Phys.* **2000**, *113*, 6035−6041.
(36) Holas, A. Transforms for Idempotency Purification of Density Matrices in Linear-scaling Electronic Structure Calculations. *Chem. Phys. Lett.* **2001**, *340*, 552−558.
(37) Niklasson, A. M. N. Expansion Algorithm for the Density Matrix. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2002**, *66*, 155115.
(38) Niklasson, A. M. N. Implicit Purification for Temperature-dependent Density Matrices. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2003**, *68*, 233104.
(39) Jordan, D. K.; Mazziotti, D. A. Comparison of Two Genres for Linear Scaling in Density Functional Theory: Purification and Density Matrix Minimization Methods. *J. Chem. Phys.* **2005**, *122*, 084114.
(40) Rudberg, E.; Rubensson, E. H. Assessment of Density Matrix Methods for Linear Scaling Electronic Structure Calculations. *J. Phys.: Condens. Matter* **2011**, *23*, 075502.
(41) Suryanarayana, P. Optimized Purification for Density Matrix Calculations. *Chem. Phys. Lett.* **2013**, *555*, 291.

(42) Rubensson, E. H.; Niklasson, A. M. N. Interior Eigenvalues from Density Matrix Expansions in Quantum Mechanical Molecular Dynamics. *SIAM J. Sci. Comput.* **2014**, *36*, B147.
(43) Truflandier, L. A.; Dianzinga, R. M.; Bowler, D. R. Communication: Generalized Canonical Purification for Density Matrix Minimization. *J. Chem. Phys.* **2016**, *144*, 091102.
(44) Beylkin, G.; Coult, N.; Mohlenkamp, M. J. Fast Spectral Projection Algorithms for Density Matrix Computations. *J. Comp. Physiol.* **1999**, *152*, 32−54.
(45) Rubensson, E. H. Nonmonotonic Recursive Polynomial Expansion for Linear Scaling Calculation of the Density Matrix. *J. Chem. Theor. Comput.* **2011**, *7*, 1233.
(46) Mniszewski, S. M.; Perriot, R.; Rubensson, E. H.; Negre, C. F. A.; Cawkwell, M. J.; Niklasson, A. M. N. Linear Scaling Pseudo Fermi-Operator Expansion for Fractional Occupation. *J. Chem. Theory Comput.* **2019**, *15*, 190−200.
(47) NVIDIA V100. https://www.nvidia.com/en-us/data-center/v100/ (accessed Dec 23, 2020).
(48) Kruchinina, A.; Rudberg, E.; Rubensson, E. H. Parameterless Stopping Criteria for Recursive Density Matrix Expansions. *J. Chem. Theory Comput.* **2016**, *12*, 5788−5802.
(49) NVIDIA cuBLAS. *cuBLAS*, 2020. https://developer.nvidia.com/cuBLAS.
(50) Niklasson, A. M.; Mniszewski, S. M.; Negre, C. F. A.; Wall, M. E.; Cawkwell, M. J.; Bock, N. *Progress*, version 1.0, 2016. https://github.com/lanl/qmd-progress.
(51) 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)*, 2019; pp 111−122.
(52) Zachariadis, O.; Satpute, N.; Gómez-Luna, J.; Olivares, J. Accelerating Sparse Matrix−Matrix Multiplication with GPU Tensor Cores. *Comput. Electr. Eng.* **2020**, *88*, 106848.
(53) Niklasson, A. M. N.; Cawkwell, M. J.; Rubensson, E. H.; Rudberg, E. Canonical Density Matrix Perturbation Theory. *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.* **2015**, *92*, 063301.

https://doi.org/10.1021/acs.jctc.1c00057
J. Chem. Theory Comput. 2021, 17, 2256−2265