The Journal of Chemical Physics

RESEARCH ARTICLE | DECEMBER 09 2021

Massively parallel linear-scaling Hartree–Fock exchange and hybrid exchange–correlation functionals with plane wave basis set accuracy

Special Collection: Beyond GGA Total Energies for Solids and Surfaces

Jacek Dziedzic ; James C. Womack ; Rozh Ali; Chris-Kriton Skylaris

J. Chem. Phys. 155, 224106 (2021)
https://doi.org/10.1063/5.0067781

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

Massively parallel linear-scaling Hartree–Fock exchange and hybrid exchange–correlation functionals with plane wave basis set accuracy

Cite as: J. Chem. Phys. 155, 224106 (2021); doi: 10.1063/5.0067781
Submitted: 19 August 2021 • Accepted: 22 November 2021 •
Published Online: 9 December 2021

Jacek Dziedzic,[1,2] James C. Womack,[1,3] Rozh Ali,[2,4] and Chris-Kriton Skylaris[1,a)]

AFFILIATIONS

[1] School of Chemistry, Highfield, University of Southampton, Southampton SO17 1BJ, United Kingdom
[2] Faculty of Applied Physics and Mathematics, Gdańsk University of Technology, Gdańsk 80-233, Poland
[3] Research Software Engineering, Advanced Computing Research Centre, University of Bristol, Bristol BS1 5QD, United Kingdom
[4] Applied Physics Department, College of Medical and Applied Science, Charmo University, Chamchamal, 46023 Sulaimania, Iraq

Note: This paper is part of the JCP Special Topic on Beyond GGA Total Energies for Solids and Surfaces.
[a)] **Author to whom correspondence should be addressed:** C.Skylaris@soton.ac.uk

ABSTRACT

We extend our linear-scaling approach for the calculation of Hartree–Fock exchange energy using localized *in situ* optimized orbitals [Dziedzic *et al.*, J. Chem. Phys. **139**, 214103 (2013)] to leverage massive parallelism. Our approach has been implemented in the ONETEP (Order-N Electronic Total Energy Package) density functional theory framework, which employs a basis of non-orthogonal generalized Wannier functions (NGWFs) to achieve linear scaling with system size while retaining controllable near-complete-basis-set accuracy. For the calculation of Hartree–Fock exchange, we use a resolution-of-identity approach, where an auxiliary basis set of truncated spherical waves is used to fit products of NGWFs. The fact that the electrostatic potential of spherical waves (SWs) is known analytically, combined with the use of a distance-based cutoff for exchange interactions, leads to a calculation cost that scales linearly with the system size. Our new implementation, which we describe in detail, combines distributed memory parallelism (using the message passing interface) with shared memory parallelism (OpenMP threads) to efficiently utilize numbers of central processing unit cores comparable to, or exceeding, the number of atoms in the system. We show how the use of multiple time-memory trade-offs substantially increases performance, enabling our approach to achieve superlinear strong parallel scaling in many cases and excellent, although sublinear, parallel scaling otherwise. We demonstrate that in scenarios with low available memory, which preclude or limit the use of time-memory trade-offs, the performance degradation of our algorithm is graceful. We show that, crucially, linear scaling with system size is maintained in all cases. We demonstrate the practicability of our approach by performing a set of fully converged production calculations with a hybrid functional on large imogolite nanotubes up to over 1400 atoms. We finish with a brief study of how the employed approximations (exchange cutoff and the quality of the SW basis) affect the calculation walltime and the accuracy of the obtained results.

I. INTRODUCTION

Owing to its favorable balance of accuracy and relatively low computational cost, Kohn–Sham (KS) density functional theory (DFT) is a widely used technique in many branches of computational chemistry and materials science.[1] The accuracy of DFT crucially depends on the approximations invoked in the exchange–correlation (XC) functional used. Hybrid functionals, which include a fraction of Hartree–Fock exchange (HFx), are among the most accurate functionals in use today, offering an elegant way of reducing the self-interaction error and leading to a more faithful description of geometries and of several properties, such as bond energies and bandgaps, particularly for metal oxides.[2]

Despite the continual increase in available computing power, calculating the HFx energy term remains computationally expensive because of its inherent non-locality. In canonical KS-DFT, the HFx energy is given by

$$E_{\mathrm{HFx}} = -\sum_{i=1}^{N_{\mathrm{MO}}}\sum_{j=1}^{N_{\mathrm{MO}}} z_i z_j \iint \frac{\psi_i^*(\mathbf{r})\psi_j(\mathbf{r})\psi_j^*(\mathbf{r}')\psi_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}', \quad (1)$$

where $\{\psi_i\}$ are the canonical molecular orbitals (MOs), z_i are their occupancies, and N_{MO} is the total number of molecular orbitals present in the calculation. Given that MOs extend throughout the entire system and that the Coulomb operator is long-ranged, the cost of each volume integration in Eq. (1) is proportional to the size of the system, whether measured by the number of atoms N or the number of molecular orbitals N_{MO} (which is $\propto N$). The presence of a *double* integral over volume together with a *double* sum over MOs makes a direct calculation of E_{HFx} scale as $\mathcal{O}(N^4)$.

In practical calculations, the MOs are expanded in terms of a finite basis,

$$\psi_i(\mathbf{r}) = \varphi_\alpha(\mathbf{r})M^\alpha{}_i, \quad (2)$$

where we have assumed a summation over repeated greek indices. The techniques for mitigating the unfortunate quartic scaling depend on the employed basis set $\{\varphi_\alpha\}$.

When localized orbitals are used [e.g., Gaussian functions or numerical atomic orbitals (NAOs)], Hartree–Fock exchange energy can be recast as

$$E_{\mathrm{HFx}} = -K^{\beta\alpha}\left(\varphi_\alpha\varphi_\delta|\varphi_\gamma\varphi_\beta\right)K^{\delta\gamma}, \quad (3)$$

where $\left(\varphi_\alpha\varphi_\delta|\varphi_\gamma\varphi_\beta\right)$ is the two-electron four-center electron repulsion integral (ERI) in chemists' notation,

$$\left(\varphi_\alpha\varphi_\delta|\varphi_\gamma\varphi_\beta\right) = \iint \varphi_\alpha^*(\mathbf{r})\varphi_\delta(\mathbf{r})\frac{1}{|\mathbf{r}-\mathbf{r}'|}\varphi_\gamma^*(\mathbf{r}')\varphi_\beta(\mathbf{r}')\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}' \quad (4)$$

$$= \int \varphi_\alpha^*(\mathbf{r})\varphi_\delta(\mathbf{r})u_{\gamma\beta}(\mathbf{r})\mathrm{d}\mathbf{r}, \quad (5)$$

where

$$u_{\gamma\beta}(\mathbf{r}) = \int \frac{\varphi_\gamma^*(\mathbf{r}')\varphi_\beta(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}' \quad (6)$$

is a Coulombic potential of a product of localized orbitals. The contravariant matrix $\mathbf{K}$ is the representation of the single-particle density matrix in the duals of $\{\varphi_\alpha\}$ and is known as the density kernel. The density kernel is, in general, a spin-dependent quantity. Here and in the text that follows, we will omit spin-dependence for clarity and brevity of notation.

Pre-screening ERIs, that is, avoiding their evaluation if they are deemed to be zero or below a given threshold, forms the basis of a number of methods for reducing the computational cost of HFx, particularly, for Gaussian basis sets. Examples of such approaches include LinK[3] and Order-N exchange (ONX).[4] It has also been recognized that rigorous upper bounds for integrals are of secondary importance, with tighter estimates of integral values permitting better control of the precision of calculating HFx.[5] Another approach is to use a truncated Coulomb operator to evaluate the ERIs, which makes them short-range. The long range contribution can then be recovered in the form of a systematically improvable correction.[6]

In the context of extended basis sets, such as plane waves, the non-locality of the exchange operator makes the calculation of HFx more challenging. Nevertheless, suitable techniques have been proposed for calculations on periodic systems[7–9] with some of them being linear-scaling.[10,11] Some of these approaches employ mixed basis sets (e.g., Gaussians and plane waves[12,13] or Wannier functions and plane waves[10,11]).

In the context of localized atom-centered basis sets, such as Gaussians or NAOs, methods based on the resolution of the identity (RI) are commonly employed to make the computational effort associated with computing HFx more manageable. These techniques, pioneered in the 1970s[14,15] and particularly popular in the field of correlated wavefunction methods,[16–18] expand pair products of atomic orbitals in an auxiliary basis whose functions are similarly atom-centered.[19]

One of the most notable developments in this area is the robust fitting formula of Dunlap,[20,21] which, when used instead of directly replacing the pair products with their RI fits, ensures that the resultant error is bilinear in the error of the fitted products,[19] with the linear error removed. Dunlap *et al.* were also credited with establishing[22] that using the Coulomb metric in the fitting offers more accurate energies compared to other metrics, such as the overlap metric.

In recent years, certain difficulties associated with robust fitting have been recognized and solutions or workarounds were proposed. Merlot *et al.*[19] observed that the two-electron integral matrix is not manifestly positive semidefinite under certain conditions. They proposed a pair-atomic resolution of identity (PARI) approach based on local fitting of either the bra or the ket side of the ERI, combined with the robust correction, to achieve quadratic accuracy.[20] Tew[23] recently proposed a "quasi-robust" local density fitting approach that addresses issues with undesired long-range behavior when the auxiliary basis is incomplete. Sodt and Head-Gordon[24] proposed a local modification to RI that yields energies that are differentiable with respect to nuclear positions.

In this paper, we present a massively parallel approach for the efficient calculation of Hartree–Fock exchange in linear-scaling time. The technique employs the resolution of identity and uses truncated spherical waves as the auxiliary basis, with only the ket side of Eq. (4) being fitted, while the products in the bra are unexpanded. It is based on our previous developments,[25] where the original implementation was serial. In Sec. II, we first briefly outline the basics of ONETEP (Order-N Electronic Total Energy Package)—the linear-scaling reformulation of KS-DFT in which our approach is implemented—followed by a description of the theoretical basis of our method. In Sec. III, we describe the implementation details of the algorithm, devoting particular attention to the parallel data distribution and time-memory trade-offs that enable its efficient parallelization. Section IV describes the setup of the calculations we performed to benchmark our method.

In Sec. V, we show the results of these calculations, demonstrate that our algorithm is, indeed, linear-scaling, compare calculation walltimes against non-hybrid functionals, and investigate how individual components of the calculation scale. We show excellent strong and weak parallel scaling of our approach, with superlinear speed-ups in many cases. We briefly demonstrate how preconverging the calculation with a non-hybrid functional before continuing with a hybrid functional significantly shortens

calculation time with negligible loss of accuracy. We conclude this section with a demonstration of the feasibility of the proposed approach by performing calculations with the B3LYP hybrid functional on imogolite nanotube systems with 1416 atoms. We finish with Sec. VI, which contains conclusions and thoughts about future work. In the supplementary material, we show how the additional approximations (the use of a cutoff finiteness of the auxiliary basis) are controllable and how the associated errors are very small.

II. THEORY

A. ONETEP

ONETEP[26] reformulates Kohn–Sham DFT[27] in terms of the single-particle density matrix, $\rho(\mathbf{r},\mathbf{r}')$. The density matrix is represented as

$$\rho(\mathbf{r},\mathbf{r}') = \varphi_\alpha(\mathbf{r})K^{\alpha\beta}\varphi^*_\beta(\mathbf{r}'), \tag{7}$$

where $\{\varphi_\epsilon\}$ are the non-orthogonal generalized Wannier functions (NGWFs)[28] centered at $\mathbf{r}_\epsilon$, which coincide with nuclear coordinates (ϵ being a generic NGWF index). The NGWFs are strictly localized within spherical regions with radii $\{R_\epsilon\}$.

$\mathbf{K}$ is the density kernel, a sparse contravariant matrix, whose elements $K^{\alpha\beta}$ are nonzero only if $|\mathbf{r}_\alpha - \mathbf{r}_\beta| < r_\mathrm{K}$, where r_K is a real-space cutoff length, known as the density kernel cutoff.

The strict localization of NGWFs means that the NGWF overlap matrix $\mathbf{S} = \{S_{\epsilon\zeta}\}$, defined as

$$S_{\epsilon\zeta} = \int \varphi^*_\epsilon(\mathbf{r})\varphi_\zeta(\mathbf{r})\mathrm{d}\mathbf{r}, \tag{8}$$

is sparse.

The NGWFs are expanded as linear combinations of psinc functions,[29] $D_m(\mathbf{r}) = D(\mathbf{r}-\mathbf{r}_m)$,

$$\varphi_\epsilon(\mathbf{r}) = \sum_m^{m\in LR(\epsilon)} D(\mathbf{r}-\mathbf{r}_m)c_{m\epsilon}, \tag{9}$$

where the index m runs over the points of the real-space Cartesian grid $\mathbf{r}_m$, which are the centers of the psinc functions, inside the localization region of φ_ϵ, $LR(\epsilon)$. The psinc functions form an orthogonal basis and are related to plane waves by a Fourier transform, thus sharing many of their desirable properties, notably the independence from the nuclear coordinates and the ability of the basis set to be systematically improved by increasing a single parameter: the kinetic energy cutoff.

The total energy is minimized self-consistently with respect to the density kernel elements $K^{\alpha\beta}$ and the NGWF expansion coefficients $c_{m\epsilon}$ under the constraints of the idempotency of the density matrix and conservation of the number of electrons, N_e.

In typical ONETEP calculations, this is done in two nested loops. In the inner loop, $\mathbf{K}$ is optimized via a modified Li–Nunes–Vanderbilt (LNV) algorithm[30–32] with the NGWFs fixed. The outer loop optimizes the NGWF expansion coefficients $c_{m\epsilon}$ through gradient-based energy minimization. The fact that the NGWFs remain fixed in the inner loop (and thus during the majority of energy evaluations) will play a crucial role in the code optimizations employed in the calculation of the Hartree–Fock exchange energy.

B. Hartree–Fock exchange in ONETEP

By introducing an auxiliary quantity, the (covariant) exchange matrix $\mathbf{X}$,

$$X_{\alpha\beta} = \left(\varphi_\alpha\varphi_\delta|\varphi_\gamma\varphi_\beta\right)K^{\delta\gamma}, \tag{10}$$

we can express E_HFx [Eq. (3)] simply as

$$E_\mathrm{HFx} = -K^{\beta\alpha}X_{\alpha\beta}. \tag{11}$$

Direct evaluation of ERIs from Eq. (4) is impracticable because of the six-dimensional nature of the integral. Proceeding via Eq. (5) is slightly more advantageous—$u_{\gamma\beta}(\mathbf{r})$ [Eq. (6)] can be obtained in reciprocal space[33] or by solving the Poisson equation. Here, the difficulty lies in the fact that this has to be done for each pair-atomic quantity $\varphi^*_\gamma(\mathbf{r}')\varphi_\beta(\mathbf{r}')$ and that the potential $u_{\gamma\beta}(\mathbf{r})$ is long-ranged (cf. Fig. 1). The latter precludes the use of ONETEP's traditional tool, the FFT box,[34] as the FFT box would have to coincide with the entire simulation cell. This approach is quadratically scaling with a very large prefactor.[34] The use of a finite exchange cutoff, e.g., by assuming $X_{\alpha\beta}$ to vanish when $|\mathbf{r}_\alpha - \mathbf{r}_\beta| < r_\mathrm{X}$, where r_X is the exchange cutoff length, makes this approach linear-scaling, but the prefactor remains prohibitively large.

We now briefly recount the linear-scaling approach actually used in ONETEP. For more details, we refer the reader to Ref. 25 where this approach was first described.

We first introduce the electrostatic metric $\mathbf{V}$ with the following elements:

$$V_{ps} = \iint f^*_p(\mathbf{r})\frac{1}{|\mathbf{r}-\mathbf{r}'|}f_s(\mathbf{r}')\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}' = (f_p|f_s), \tag{12}$$

where $\{f_p(\mathbf{r})\}_{p=1}^{N_\mathrm{f}}$ are a set of (in-general) non-orthogonal functions. Using the elements V^{ps} of the inverse metric matrix $\mathbf{V}^{-1}$, we can

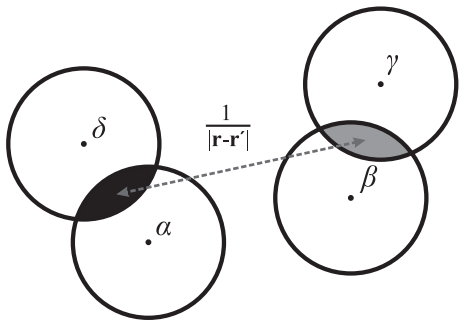

FIG. 1. NGWFs (localized orbitals) featuring in the calculation of E_HFx [Eq. (3)]. The interacting pair-atomic quantities $\varphi^*_\alpha(\mathbf{r})\varphi_\delta(\mathbf{r})$ and $\varphi^*_\gamma(\mathbf{r}')\varphi_\beta(\mathbf{r}')$ are shaded. All terms where the localization sphere of α is disjoint from that of δ vanish, and similarly, all terms where the localization sphere of β is disjoint from that of γ vanish. This is a property of localized orbitals. The non-local nature of Hartree–Fock exchange manifests in the fact that terms where the localization sphere of α is disjoint from the localization sphere of β *do not*, in general, vanish.

define a resolution-of-identity (RI) operator,

$$\hat{I}_{\mathrm{V}} = |f_p)\, V^{ps}(f_s|. \tag{13}$$

Such RI operators are often used in computational quantum chemistry software[14–18,20,22–24,35–39] with the aim of replacing four-center ERIs with more tractable three-center integrals.

By inserting the RI operator into Eq. (3), we obtain[25]

$$E_{\mathrm{HFx}} = -K^{\beta\alpha}(\varphi_\alpha\varphi_\delta|f_p)\, V^{ps}\big(f_s|\varphi_\gamma\varphi_\beta\big)K^{\delta\gamma}, \tag{14}$$

which is exact if the set of auxiliary functions spans the same subspace as the products of NGWFs. In practice, we will be working with finite sets of auxiliary functions, making Eq. (14) an approximate equality; we will denote the resultant approximate Hartree–Fock exchange energy with $\tilde{E}_{\mathrm{HFx}}$ and the approximate exchange matrix whose elements are

$$\tilde{X}_{\alpha\beta} = (\varphi_\alpha\varphi_\delta|f_p)\, V^{ps}\big(f_s|\varphi_\gamma\varphi_\beta\big)K^{\delta\gamma} \tag{15}$$

with $\tilde{\mathbf{X}}$.

The employed auxiliary basis should have two important properties—it should be able to accurately represent the fitted NGWF products and it should enable the computation of Coulomb potentials [Eq. (6)] in $\mathcal{O}(1)$ time per NGWF pair. One such a basis set is formed by truncated spherical waves (SWs), which are solutions of the Schrödinger equation for a particle in a sphere. They are given by

$$f(\mathbf{r}) = \begin{cases} j_l(qr)Z_{lm}(\hat{\mathbf{r}}), & r < a, \\ 0, & r \geq a, \end{cases} \tag{16}$$

where $j_l(qr)$ is a spherical Bessel function, $Z_{lm}(\hat{\mathbf{r}})$ is a real spherical harmonic, m is an integer from the interval $[-l, l]$, and a is the radius of the sphere where the zero boundary condition is imposed. In ONETEP's implementation, we assume $\forall_\epsilon R_\epsilon = a$, that is, all NGWF localization radii are identical and equal to a. The value of q is chosen in such a way that $j_l(qa) = 0$, and the suitable values of q depend on the angular momentum index l. The maximum values of q and l are limited by the kinetic energy cutoff and the corresponding grid spacing. In typical scenarios, it is sufficient to truncate the SW basis at $l_{\max} = 3$ and $q_{\max} = 12$, where $q_{\max}$ is the number of different values of q used for each l. In the text that follows, we will use a single index (p or s) for the SWs for simplicity. This index covers all the possible combinations of l, q, and m and runs from 1 to N_{SW}. We will use $u_p(\mathbf{r})$ to denote the potential of a SW ("SWpot" in the further text),

$$u_p(\mathbf{r}) = \int \frac{f_p(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}\mathrm{d}\mathbf{r}'. \tag{17}$$

This potential can be evaluated analytically in $\mathcal{O}(1)$ time.[33]

In our technique, the products featuring in the ket of Eq. (4), that is, $\varphi^*_\gamma(\mathbf{r}')\varphi_\beta(\mathbf{r}')$, are expanded, while the products in the bra, that is, $\varphi^*_\alpha(\mathbf{r})\varphi_\delta(\mathbf{r})$, do not undergo expansion. The coefficients of expansion $\{c^p_{\gamma\beta}\}_{p=1,\ldots,N_{\mathrm{SW}}}$ are given by

$$c^p_{\gamma\beta} = V^{ps}\big(f_s|\varphi_\gamma\varphi_\beta\big). \tag{18}$$

An important issue is the choice of centers for the SWs used in the expansion of $\varphi^*_\gamma(\mathbf{r}')\varphi_\beta(\mathbf{r}')$. Using SWs centered only on NGWF γ or only on NGWF β breaks symmetry in Eq. (4) because it leads to $c_{\gamma\beta} \approx c_{\beta\gamma}$ rather than $c_{\gamma\beta} = c_{\beta\gamma}$. Using SWs centered on both β and γ alleviates this problem, but there is still the problem of broken $\alpha\delta \leftrightarrow \gamma\beta$ symmetry, i.e., the products in the bra of the ERI Eq. (4) are then exact, while the products in the ket are approximate (fitted). To formally satisfy all the symmetry requirements, one can resort to using SWs centered on all four atoms (α, β, γ, and δ) in the expansion, which keeps the fitting domain identical between all permutations of α, β, γ, and δ in the ERI. Such a fit is then robust in the sense of Ref. 20, that is, the fitted integral does not contain any terms linear in the error in the fitted densities. However, in our case, this is impracticable. Even though it maintains linear-scaling, using four centers in the expansion leads to prefactors that are prohibitively large because the product $\varphi^*_\gamma(\mathbf{r})\varphi_\beta(\mathbf{r})$ then needs to be re-expanded every time α or δ changes in Eq. (14).

We solve the above problem by using a two-center expansion (on β and γ) and recovering the $\alpha\delta \leftrightarrow \gamma\beta$ symmetry by symmetrizing the approximate (SW-expanded) exchange matrix, $\tilde{\mathbf{X}}$,

$$\tilde{X}_{\alpha\beta} \leftarrow \frac{1}{2}\big(\tilde{X}_{\alpha\beta} + \tilde{X}_{\beta\alpha}\big). \tag{19}$$

This change propagates to the expression for the Hartree–Fock energy, which now reads[25]

$$\begin{aligned}\tilde{E}_{\mathrm{HFx}} = -\frac{1}{2}K^{\beta\alpha}\big[&(\varphi_\alpha\varphi_\delta|f_{p,\gamma\beta})\, V^{ps}\big(f_{s,\gamma\beta}|\varphi_\gamma\varphi_\beta\big) \\ &+ (\varphi_\alpha\varphi_\delta|f_{p,\delta\alpha})\, V^{ps}\big(f_{s,\delta\alpha}|\varphi_\gamma\varphi_\beta\big)\big]K^{\delta\gamma}.\end{aligned} \tag{20}$$

Here, we used additional indices to the SWs $\{f\}$ to indicate where they are centered. While the above expression formally includes SWs centered on α and δ, we again stress that the symmetrization procedure [Eq. (19)] makes it sufficient to only expand using SWs on β and γ. We have shown[25] that, in practice, any differences between the elegant but prohibitively expensive four-center fit and the two-center fit we employ are negligible, although our approach is no longer robust in the sense of Ref. 20.

An important advantage of the two-center expansion is that it only includes SWs centered on atoms whose NGWFs overlap, making the electrostatic matrix in Eq. (12) effectively sparse (even though the potential of a SW is not localized). This is necessary for the approach to be linear-scaling.

We subsequently build linear combinations of SWpots (corresponding to expanded potentials of NGWF pair products),

$$e_{\gamma\beta}(\mathbf{r}) = \sum_{p=1}^{N_{\mathrm{SW}}} u_p(\mathbf{r})c^p_{\gamma\beta} = \hat{I}_{\gamma\beta}\big|\,\varphi_\gamma\varphi_\beta\big) \tag{21}$$

[where $\hat{I}_{\gamma\beta}$ is defined in Eq. (24)], and we contract them with the density kernel over the index γ and use the resultant potential

$$\tilde{u}^\delta_\beta(\mathbf{r}) = K^{\delta\gamma}e_{\gamma\beta}(\mathbf{r}) \tag{22}$$

TABLE I. Summary of truncation parameters employed in the ONETEP approach and in our HFx algorithm.

Parameter	Symbol	Typical value	Used in	Description
Density kernel cutoff	r_K	$50–100a_0$	ONETEP	Distance between two atomic centers beyond which the density matrix is assumed to vanish.
NGWF localization radius	R_ϵ	$8–10a_0$	ONETEP	Radius of an atom-centered sphere beyond which the NGWF is assumed to be strictly zero.
Exchange cutoff	r_X	$20–25a_0$	HFx method	Distance between two atomic centers beyond which the exchange matrix is assumed to vanish.

on the product $\varphi^*_\alpha(\mathbf{r})\varphi_\delta(\mathbf{r})$ that appears in the bra of the ERI of Eq. (3). Having repeated this step for all requisite NGWFs δ, we obtain an element of the exchange matrix, $\tilde{X}_{\alpha\beta}$ [Eq. (15)]. In the final step, we contract the exchange matrix with the negative of the density kernel $\mathbf{K}$ over α and β according to (11) to obtain the Hartree–Fock exchange energy.

For every exchange matrix element [a pair of indices (α,β)], the cost of the above calculation is asymptotically constant (independent of system size). This is because all the NGWFs are strictly localized and their overlap matrix is sparse. For our approach to be linear-scaling, the number of pairs (α,β) must increase linearly with system size, i.e., the exchange interaction must be truncated. This is a standard practice in linear-scaling methods.[40] Our method employs a simple distance-based cutoff, where the exchange matrix $\mathbf{X}$ is made sparse by neglecting contributions from pairs (α,β) when $|\mathbf{r}_\alpha - \mathbf{r}_\beta| > r_X$, where r_X is an assumed real-space cutoff for exchange. We have shown[25] that even in more demanding applications, $r_X \approx 20a_0$, or even less, is sufficient to keep the truncation error below 0.01%.

The fact that in ONETEP the NGWFs are optimized *in situ* necessitates calculating the gradient of the energy with respect to the NGWF expansion coefficients (which, mathematically, is the functional derivative of the energy with respect to the complex conjugate of some NGWF α). The derivation of the relevant expression for Hartree–Fock exchange was presented in Ref. 25; here, for the sake of brevity, we only recount the following final form:

$$G^\alpha(\mathbf{r}) = G^\alpha_1(\mathbf{r}) + G^\alpha_2(\mathbf{r}) = 2\varphi_\delta(\mathbf{r})K^{\beta\alpha}K^{\delta\gamma}\hat{I}_{\gamma\beta}\big|\varphi_\gamma\varphi_\beta\big) + 2\varphi_\delta(\mathbf{r})\hat{I}_{\delta\alpha}\Big(K^{\beta\alpha}K^{\delta\gamma}\big|\varphi_\gamma\varphi_\beta\big)\Big), \tag{23}$$

where the factor of 2 is due to the fact that the NGWFs are assumed to be real-valued, and the operator $I_{\kappa\lambda}$ is the resolution of the identity operator in terms of SWs centered on NGWFs κ and λ,

$$\hat{I}_{\kappa\lambda} = \big|f_{p,\kappa\lambda}\big)\, V^{ps}\, \big(f_{s,\kappa\lambda}\big|. \tag{24}$$

The first term in Eq. (23) involves summing large numbers of expansions of pair NGWF products $\varphi_\gamma\varphi_\beta$ in terms of SWs centered on these NGWFs. In the second, more involved term linear combinations of NGWF products are expanded in SWs centered not on these NGWFs, but rather on the NGWF with respect to which we differentiate (α) and the NGWFs that overlap with it (δ).

The expression in Eq. (23) is the *contravariant* gradient, which cannot be directly used to update NGWFs $\{\varphi_\alpha\}$, which are *covariant*. To achieve tensorial correctness, it must be converted to a covariant gradient: $G_\epsilon(\mathbf{r}) = G^\alpha(\mathbf{r})S_{\alpha\epsilon}$.

C. Summary of truncation parameters

As an aid to the reader, in Table I, we recount and briefly describe the truncation parameters used in ONETEP and in this work.

III. IMPLEMENTATION

In this section, we describe the parallel implementation of our algorithm, highlighting choices of parallel decompositions and time-memory trade-offs that make it efficient and massively parallelizable.

A. Preliminary comments and notation

We begin with a minor implementation detail that will gain more importance later in the text. As mentioned earlier [cf. Eq. (9)], in ONETEP, the NGWFs are stored as psinc expansion coefficients on a Cartesian grid. In practice, we use a so-called parallelepiped representation, illustrated in Fig. 2.

The simulation cell is tiled into suitably sized parallelepipeds (PPDs). Each NGWF localization region is encompassed by a number of PPDs (shaded areas in the diagram). The psinc

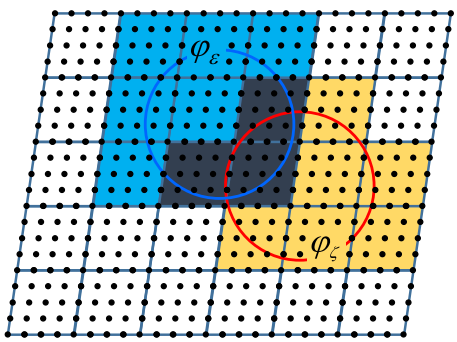

FIG. 2. Diagram of the parallelepiped (PPD) representation used in ONETEP (simplified to 2D). Two example NGWFs are shown (circles). Grid data for NGWFs and their overlaps are stored as contents of parallelepipeds (PPDs) tiling the grid (shaded areas).

coefficients are stored for all points in these PPDs, arranged into a continuous 1D array. Such a packed representation, while introducing some overhead (points within PPDs but outside of the NGWF sphere), makes it easy to pass NGWF data between message passing interface (MPI) ranks and enables efficient processing. In the current implementation of HFx, this representation is also used for NGWF overlaps (dark-shaded area in the diagram).

In the text that follows, we will use a convention where capital letters (A, B, C, D, I, J, and M) denote atoms. Lowercase letters (a, b, c, and d) will be used to index NGWFs on a corresponding atom; for example, a will count NGWFs on atom A. For indexing NGWFs globally, we will use greek letters (α, β, γ, δ, ϵ, and ζ) like we have already done in the Introduction. We will occasionally switch between these two ways of indexing NGWFs ($Aa \equiv \alpha$, etc.) as some concepts are easier explained using one notation or the other. In addition, implicit summation will be assumed only over repeated greek indices.

Matrix sparsity plays an important role in the algorithm we present. The sparsity of $\mathbf{S}$ (due to strict localization of NGWFs), the sparsity of $\mathbf{K}$ (due to the assumed kernel cutoff), and the sparsity of $\tilde{\mathbf{X}}$ (due to the assumed exchange cutoff) are all crucial for achieving linear scaling. In the text that follows, it will often be useful to think of matrix sparsity patterns in terms of pairs of atoms that are either *within* a sparsity pattern of a matrix (meaning that the corresponding matrix element is non-zero) or *outside* it (meaning that the corresponding element is zero and is not stored). We will use the following terminology to describe this: two atoms whose NGWFs overlap (and so the corresponding $\mathbf{S}$ element is non-zero) will be termed S-neighbors. Two atoms whose centers are within the exchange cutoff (and so the corresponding $\tilde{\mathbf{X}}$ element is non-zero) will be termed X-neighbors. The relations of being an S-neighbor or X-neighbor are commutative. Finally, by saying an atom I is an X–S-neighbor of atom J, we will mean that there exists an atom M such that J and M are X-neighbors and M and I are S-neighbors. This is best understood referring to Fig. 1, where atom D (NGWF δ) is an X–S-neighbor of atom B (NGWF β). Note that this relation is not, in general, commutative.

B. Hybrid MPI-openMP parallelism

ONETEP employs hybrid parallelism for all its major algorithms.[41] Distributed-memory process-based (MPI[42]) parallelism is used in two ways: for the geometric decomposition of the simulation cell (where each MPI rank deals with a number of slabs comprising the cell) and to divide atoms across MPI ranks (each MPI rank "owns" or is responsible for a subset of atoms). Such a scheme is naturally limited in the number of MPI ranks that can be used—performance begins to deteriorate once the number of MPI processes exceeds the number of slabs in the cell (because some ranks no longer have any work to do) or when the number of MPI ranks exceeds the number of atoms in the system (at which point ONETEP cannot be run). This problem can be alleviated by reducing the number of MPI ranks and, instead, having each rank spawn a number of threads to saturate the available central processing unit (CPU) cores.

Shared-memory thread-based parallelism (OpenMP[43]) is then used on a finer scale to subdivide larger work grains into threads. There are two main advantages using such a hybrid model: One is being able to utilize a large number of CPU cores without significant loss of performance and another stems from the fact that threads share an address space, so the memory load associated with quantities that would normally be replicated across MPI ranks is lowered since fewer MPI ranks are used. This second advantage does not play a large role in ONETEP, as the most memory-intensive quantities are distributed, not replicated, across MPI ranks.

The two main bottlenecks in ONETEP's approach to Hartree–Fock exchange are as follows: (a) the evaluation of SWpots [Eq. (17)] from analytical expressions involving trigonometric and radial terms (see Ref. 33 for details) and (b) the evaluation of SW-expanded potentials of NGWF pair products [Eq. (21)]. Both of these are performed repeatedly, often, for the same parameters. For example, Eq. (21) will be evaluated with the same γ and β for multiple combinations of α and δ, of which it is independent.

Naturally, this opens up opportunities for time-memory tradeoffs, where the already calculated results are cached in memory and subsequently reused, reducing the computational cost to mere lookups. However, this has to be done carefully. Both quantities considered here (SWpots and expanded potentials) require very large amounts of memory to store. Further in the text (Sec. III E), we outline how the requirements scale with the system size; here, we will only provide an estimate—in typical scenarios, this memory requirement begins to exceed 1 TB at about 400 atoms, just at the onset of linear scaling in the evaluation of HFx. This is a requirement on total memory, and so it could, in principle, be divided across multiple compute nodes. In today's high performance computing environments, using this quantity of distributed memory is practical since compute nodes are routinely equipped with 128–512 GB of memory.

Not to be neglected, however, is the cost to accessing such *distributed* cached data—accessing data that are not local to an MPI rank entails interprocess communication. The calculation of SWpots in ONETEP has been extensively optimized—calculating ≈200 SWpots that are required in typical scenarios in one PPD that contains 125 grid points takes only about 5 μs, and so any attempt to access a cached copy from a remote MPI rank would be much slower. For this reason, our algorithm relies on caching all relevant data *locally* (within the same MPI rank), even if this means replicating some data across ranks.

To minimize this replication and to increase the available RAM for the cache, it stands to reason to use as many OpenMP threads as possible and as few MPI ranks as threads can then share a large cache without any need for message passing. Suitable thread synchronization mechanisms must be used when populating the cache to avoid data races, but once the cache is populated and becomes read-only, there is no need for synchronization.

For the above approach to be efficient, our algorithm had to be optimized for large numbers of threads. This is accomplished by ordering operations in such a way as to move OpenMP parallelism as high in the loop structure as possible, making computational grains larger and by avoiding synchronization mechanisms (critical sections) whenever possible. As we will demonstrate in Sec. V, the result is an algorithm that scales well to at least tens

of OpenMP threads per process and at least thousands of CPU cores.

Another component crucial for massive parallelism is load balancing. As described above, ONETEP distributes atoms across MPI ranks. Looking at relevant expressions [e.g., Eqs. (18), (21), and (23)], it becomes apparent that the key quantity being processed is the kets $\left|\varphi_\gamma\varphi_\beta\right)$. For this reason, our algorithm distributes atom *pairs* (B, C) rather than atoms as in ONETEP's original parallel distribution.[44] In the text that follows, we will term ONETEP's original distribution "scheme 1," while the scheme used by the HFx algorithm will be termed "scheme 2." The quantities that are inputs to the algorithm, such as NGWFs, density kernel elements, and the metric matrix $\mathbf{V}$, will need to re-distributed from scheme 1 to scheme 2 before actual processing begins, and the quantities produced by the algorithm [the exchange matrix $\tilde{\mathbf{X}}$ and the NGWF gradients $G^\alpha(\mathbf{r})$] will have to be re-distributed from scheme 2 back to scheme 1 before they can be used by the rest of ONETEP machinery. The walltime cost of these operations is very modest below 0.5%. The associated memory cost (due to having to store additional NGWFs and density kernel elements) is also acceptable at a level below 1 MB/atom per MPI rank.

C. Algorithm overview

Examining expressions in Eqs. (18) and (21) makes it clear that they are independent of the density kernel and involve only the NGWFs. This means that the calculation of $\tilde{E}_{\mathrm{HFx}}$ can be broken up into two distinct stages—one that is independent of the density kernel and one that is density-kernel-dependent. The first stage involves calculating expansion coefficients of NGWF pair products [Eq. (18)] and the potential of these expansions [Eq. (21)] does not need to be repeated in the inner (kernel optimization, LNV) loop. The second stage {Eq. (22), subsequent calculation of the elements of $\tilde{\mathbf{X}}$ and of $\tilde{E}_{\mathrm{HFx}}$ itself [Eq. (11)]} must be repeated every time $\mathbf{K}$ changes, that is, multiple times in the inner loop.

The density-kernel-independent stage will be described in Sec. III D, and the density-kernel-dependent stage, which entails the actual calculation of $\tilde{E}_{\mathrm{HFx}}$—will be described in Sec. III E.

The last component is the calculation of the gradient of $\tilde{E}_{\mathrm{HFx}}$ with respect to the NGWFs, which is required for *in situ* optimization of NGWFs. It will be described in Sec. III F.

D. Density-kernel-independent stage

The density-kernel-independent stage is described in Algorithm 1. In step 1, the workload associated with all kets in Eq. (14), that is, the set of all (B, C) atom pairs in the calculation, is distributed across MPI ranks. During load-balancing, we assume the computational effort associated with a (B, C) atom pair to be proportional to the product of the number of NGWFs on atoms B and C. This is justified by the fact that the algorithm deals with products of pairs of NGWFs on these atoms. In the final distribution, each MPI rank has a subset of (B, C) atom pairs and these subsets are mutually disjoint, i.e., a given (B, C) pair is only found on one MPI rank.

In step 2, each MPI rank constructs a set of all B atom indices and a set of all C atom indices present in its set of (B, C) pairs. We will refer these as the rank's B atoms and rank's C atoms. Unlike (B, C) pairs, more than one MPI rank can have the same B or C atom in its subset (i.e., the subsets are not disjoint).

In step 3, each MPI rank determines the set of its A atoms, which we define as all atoms that are X-neighbors to any B atom the MPI rank owns. Similarly, in step 4, each MPI rank determines the set of its D atoms, which we define as all atoms that are X–S-neighbors to any B atom the MPI rank owns.

In step 5, the electrostatic metric matrix $\mathbf{V}$ [Eq. (12)] is redistributed from scheme 1 (which was used during its calculation) to scheme 2 (which is used in all subsequent calculations). This also involves a change in the underlying datastructure—from a distributed sparse matrix to rank-local hash tables.

In step 6, the NGWFs themselves are redistributed from scheme 1 to scheme 2. Each MPI rank requests scheme-1-owners of all NGWFs it would need to send them (in PPD format), and simultaneously, each MPI rank listens for requests addressed to it and satisfies them by sending the NGWFs its scheme-1-owns. Following this step, each MPI rank has all the NGWFs it would need in the calculation of HFx locally. No further communication of NGWFs is going to be necessary. This is an important trade-off—we sacrifice some memory (because most NGWFs are now replicated on more than one rank), but in return, we avoid interspersing calculation with communication in all subsequent stages of the calculation. Our algorithm will thus be free of any idle waits of a rank for communication with another rank and of any potential convoy effects. The memory cost of this overhead is acceptable—below 1 MB/atom on each MPI

ALGORITHM 1. Density-kernel-independent stage.

1: Distribute (B, C) pairs across MPI ranks
2: Determine B and C atoms for each MPI rank
3: Determine A atoms for each MPI rank
4: Determine D atoms for each MPI rank
5: Redistribute the matrix $\mathbf{V}$ from scheme 1 to scheme 2
6: Redistribute NGWFs from scheme 1 to scheme 2
7: Determine the set of all PPDs where SW expansions of the potential will be needed for each MPI rank
8: Populate the SWpot cache on each MPI rank (→Algorithm 2)
9: Populate the NGWF product cache (→Algorithm 3)
10: Expand NGWF pair products (→Algorithm 4)

ALGORITHM 2. Populate the SWpot cache.

1: Perform a dry-run of the calculation, where only the numbers of accesses to each SWpot are counted (for every MPI rank separately)
2: Determine how many SWpots can be cached before the user-specified memory limit is reached, $N_{\mathrm{SWpot}}^{\mathrm{max}}$
3: Precalculate the $N_{\mathrm{SWpot}}^{\mathrm{max}}$ most accessed SWpots and store them in a hash table

rank, increasing linearly with system size (owing to the sparsity of **S** and **X**). We use the same approach for communicating elements of the density kernel (see step 1 of Algorithm 5).

In step 7, each MPI rank establishes the set of PPDs where SW-expanded potentials of ket NGWF pairs will be required. This is a union of the sets of PPDs on all the rank's atoms A.

In step 8, each MPI rank populates its SWpot cache, following Algorithm 2.

In step 9, each MPI rank populates its NGWF product cache, following Algorithm 3.

Finally, in step 10, NGWF pair products are expanded in terms of SWs, following Algorithm 4.

This completes the description of the density-kernel-independent stage. We now briefly describe the algorithms used in the last three steps.

The SWpot cache is populated according to Algorithm 2.

The NGWF product cache is populated following Algorithm 3. First, each rank counts and enumerates the atom pairs (A, D) relevant to it, building a list of pairs. This is done to switch from atom indices to a fused (pair) index, which lends itself better to OpenMP (OMP) parallelization. Subsequently, for all the atom pairs, the products of their NGWFs are calculated and stored, with the pair loop parallelized over OpenMP threads. If at any time the user-specified maximum size of the product cache is reached, the loop terminates early. That means that we only store as many NGWF products as possible, with the memory requirement strictly bounded. In subsequent steps, products that are not found in the cache will simply be recalculated on the fly. This precludes runaway memory use and ensures graceful performance degradation in low-memory scenarios.

NGWF pair products are expanded in terms of SWs following Algorithm 4. For all NGWF pairs on all (B, C) atom pairs owned by an MPI rank, we first calculate the products of the NGWFs and then calculate their overlaps with all SWs s (centered on B and C). The overlaps are then used as the constant term in a system of linear equations to determine the expansion coefficients (step 7). OpenMP parallelism is employed over PPDs in the product.

ALGORITHM 3. Populate the NGWF product cache.

1: $n_{\mathrm{pairs}} = 0$
2: **for all** my A atoms **do** (▷) count (A, D) pairs
3: **for all** D atoms that are S-neighbors of this A **do**
4: $n_{\mathrm{pairs}} = n_{\mathrm{pairs}} + 1$
5: **end for** D
6: **end for** A
7: **OMP for** $i_{\mathrm{pair}} = 1$ **to** n_{pairs} **do** (▷) process (A, D) pairs
8: **for all** NGWFs d on atom D in pair **do**
9: **for all** NGWFs a on atom A in pair **do**
10: **if** room left in NGWF product cache **then**
11: Compute and store $\varphi_{Aa}\varphi_{Dd}$ in PPDs
12: **else**
13: exit **OMP for**
14: **end if**
15: **end for** a
16: **end for** d
17: **end OMP for** i_{pair}

E. Density-kernel-dependent stage

In this stage, the Hartree–Fock exchange energy $\tilde{E}_{\mathrm{HFx}}$ is calculated using the SW expansion of NGWF products obtained from Algorithm 1. Unlike the previous stage, this one depends on the values of the density kernel, and as such, it needs to be performed multiple times within the kernel optimization loop. The procedure is outlined in Algorithm 5 and will now be described.

In step 1, the density kernel matrix **K** is redistributed from scheme 1 (which is used in the rest of ONETEP) to scheme 2 (which is used in all subsequent calculations). This also involves a change in the underlying datastructure—from a distributed sparse matrix to rank-local hash tables.

Next, each MPI rank processes the atom pairs (B, C) it was assigned. The result of this processing is the expanded potentials of NGWF products $e_{\gamma\beta}$ for all NGWFs on atoms B and C in each pair. The expanded potentials are calculated at all points where they will later be required, that is, in the PPDs of all atoms A that are X-neighbors to atom B. This set of PPDs (termed "PPDs relevant to atom B") is established in step 3.

Subsequently, we iterate over all the NGWF pairs on atoms B and C. The previously calculated SW expansion coefficients $c^p_{\gamma\beta}$ for

ALGORITHM 4. Expand NGWF pair products.

1: **for all** (B, C) pairs owned by this MPI rank **do**
2: **for all** NGWFs b on atom B in pair **do**
3: **for all** NGWFs c on atom C in pair **do**
4: Compute $\varphi_{Bb}\varphi_{Cc}$ in PPDs
5: **OMP for all** PPDs in product $\varphi_{Bb}\varphi_{Cc}$ **do**
6: Calculate $b_{s,\gamma\beta} = \left(f_s|\varphi_\gamma\varphi_\beta\right)$ for all SWs s
7: For all SWs p, obtain expansion coefficients $c^p_{\gamma\beta} = V^{ps}b_{s,\gamma\beta}$ by solving a linear equation system $V_{sp}c^p_{\gamma\beta} = b_{s,\gamma\beta}$
8: **end OMP for** PPD
9: **end for** c
10: **end for** b
11: **end for** (B, C)

ALGORITHM 5. Calculation of $\tilde{E}_{\mathrm{HFx}}$ from previously (Algorithm 1) expanded NGWF products.

1: Redistribute $\mathbf{K}$ from scheme 1 to scheme 2
2: **for all** (B, C) pairs owned by this MPI rank **do**
3: Find the PPDs relevant to atom B from the pair.
This is a union of the sets of PPDs of all atoms A that are X-neighbors with this atom B
4: **for all** NGWFs b on atom B in pair **do**
5: **for all** NGWFs c on atom C in pair **do**
6: Retrieve expansion coefficients for $\varphi_{Bb}\varphi_{Cc}$
7: **OMP for all** PPDs relevant to atom B **do**
8: Calculate the expanded potential
$e_{\gamma\beta} = \sum_{p=1}^{N_{\mathrm{SW}}} c_{\gamma\beta}^{p} u_p(\mathbf{r})$ in PPD i
9: **if** room left in expansion cache **then**
Store $e_{\gamma\beta}$ in expansion cache
10: **else**
11: exit (B, C) loop
12: **end if**
13: **end OMP for** PPD
14: **end for** c
15: **end for** b
16: **end for** (B, C)
17: **for all** my B atoms **do**
18: Find the PPDs relevant to atom B like in step 3
19: Calculate $\tilde{u}_{\beta}^{\delta} = \sum_{\gamma} K^{\delta\gamma} e_{\gamma\beta}$ in all relevant PPDs, thereby eliminating index γ (→Algorithm 6)
20: Calculate contribution to $\tilde{\mathbf{X}}$ from atom B, thereby eliminating index β (→Algorithm 7)
21: **if** NGWF gradient needed **then**
22: Accumulate contribution to contravariant gradient G_1 from this atom B (→Algorithm 8)
23: **end if**
24: **end for** B
25: Redistribute $\tilde{\mathbf{X}}$ from scheme 2 back to scheme 1
26: **if** NGWF gradient needed **then**
27: Finalize NGWF gradient term G_1 (→Algorithm 9)
28: Calculate the NGWF gradient term G_2 (→Algorithm 10)
29: **end if**
30: Symmetrize $\tilde{\mathbf{X}}$ [Eq. (19)]
31: Calculate $\tilde{E}_{\mathrm{HFx}}$ [Eq. (20)]

the NGWF products are retrieved (step 6), and the expanded potential is calculated (step 8) in all relevant PPDs. OpenMP parallelism is leveraged for the loop over PPDs. The expanded potential is stored in a hash table termed the "expansion cache."

Storing *all* the expanded potentials would require enormous amounts of memory—for all but the smallest systems, they need to be calculated in a sphere with a radius $r_{\mathrm{X}} + a$ (where r_{X} is the exchange cutoff and a is the NGWF localization radius). At typical settings, this translates to about 6.5 MB of storage per single expansion. The number of expansions on an atom B is $n_B n_C N_B^{\mathrm{S}}$, where n_B and n_C are the numbers of NGWFs on atoms B and C, respectively, and N_B^{S} is the number of C atoms that are S-neighbors of atom B. At typical settings, this number is about 800. For typical n_B and n_C, we arrive at a value of about 5 GB per atom, which is clearly excessive even if we assume this cost to be distributed across compute nodes. For this reason, we set a user-adjustable upper bound on the size of the expansion cache. Once the cache is full, the loop exits (step 11). Expanded potentials that did not fit in the cache are later (Algorithm 6) calculated on the fly. This precludes runaway memory use and ensures graceful performance degradation in low-memory scenarios.

Once all the expanded potentials are calculated or the expansion cache cannot accommodate more elements, each MPI rank iterates over all its B atoms (step 17). First, PPDs relevant to current atom B are identified, the same as was done in step 3. Subsequently, the expansion coefficients are contracted with the density kernel over index γ, eliminating this index from further computation (step 19). This is done by Algorithm 6, described further in Sec. III E.

Next, the contribution from current atom B to all elements of the exchange matrix $\tilde{\mathbf{X}}$ is calculated (step 20). This is done by Algorithm 7, described further in this section. Similarly, contributions to the NGWF gradient from atom B are accumulated. This is done by Algorithm 8, described in Sec. III F.

In step 25, the exchange matrix is redistributed from scheme 2 back to scheme 1 to make it possible to use standard sparse algebra routines on it in the rest of ONETEP.

ALGORITHM 6. Eliminate index γ for a single atom B.

1: **for all** atoms C in my (B, C) pair list for current B **do**
2: **for all** NGWFs b on atom B **do**
3: **for all** NGWFs c on atom C **do**
4: **OMP for all** PPDs relevant to atom B **do**
5: Retrieve from cache or calculate $e_{\gamma\beta}$
6: **end OMP for** PPD
7: **end for** c
8: **end for** b
9: **OMP for all** PPDs relevant to atom B **do**
10: **for all** atoms D participating in this PPD **do**
11: **for all** NGWFs d on atom D **do**
12: **for all** NGWFs b on atom B **do**
13: **for all** NGWFs c on atom C **do**
14: $\tilde{u}^{\delta}_{\beta} = \tilde{u}^{\delta}_{\beta} + K^{\delta Cc} e_{Cc\beta}$
15: **end for** c
16: **end for** b
17: **end for** d
18: **end for** D
19: **end OMP for** PPD
20: **end for** C

The main part of the NGWF gradient calculation takes place in step 27; this is carried out by Algorithm 10, described in Sec. III F.

Finally, the exchange matrix $\tilde{\mathbf{X}}$ is symmetrized [cf. Eq. (19)], and the Hartree–Fock exchange energy is computed according to Eq. (20).

This concludes the general description of the density-kernel-dependent stage. We will now briefly describe Algorithms 6 and 7.

ALGORITHM 7. Eliminate index β.

1: **OMP for all** PPDs relevant to atom B **do**
2: **for all** atoms D participating in this PPD **do**
3: **for all** my atoms A participating in this PPD **do**
4: **for all** NGWFs d on atom D **do**
5: **for all** NGWFs a on atom A **do**
6: Retrieve from cache or calculate $p_{AaDd} = \varphi_{Aa}\varphi_{Dd}$ in current PPD
7: **end for** a
8: **for all** NGWFs b on atom B **do**
9: **for all** NGWFs a on atom A **do**
10: $\tilde{X}_{AaBb} = \tilde{X}_{AaBb} + \sum_{\mathbf{r}}^{\mathrm{PPD}} p_{AaDd}(\mathbf{r})\tilde{u}^{Dd}_{Bb}$
11: **if** NGWF gradient needed **then**
12: $k_{Bb} = k_{Bb} + \varphi_{Dd}\tilde{u}^{Dd}_{Bb}$
13: **end if**
14: **end for** a
15: **end for** b
16: **end for** d
17: **end for** A
18: **end for** D
19: **end OMP for** PPD

Algorithm 6 processes a single atom B owned by a given MPI rank. It iterates over all the C atoms that are S-neighbors of B and are assigned to this MPI rank. Thus, the loop body is executed for a (B, C) pair. First (steps 2–8), the expanded potentials for all combinations of NGWFs on atoms B and C are retrieved from the expansion cache into a local array for efficient access later on. This is done for all the PPDs relevant to atom B, and OpenMP parallelism is leveraged for the loop over PPDs. Subsequently, for all the PPD (again leveraging OpenMP parallelism) and all the atoms D whose localization sphere features this PPD, the quantity $\tilde{u}^{\delta}_{\beta}$, which is a contraction of expanded potentials with the density kernel over the index γ, is accumulated (step 14). As the density kernel depends on another index (δ), $\tilde{u}^{\delta}_{\beta}$ is a two-center quantity. As it is stored only for a single atom B before being processed (via Algorithm 7) and discarded, the associated memory requirement remains modest. Algorithm 6 finishes after calculating and storing all requisite $\tilde{u}^{\delta}_{\beta}$ for a given atom B.

The next stage (Algorithm 7) eliminates index β from the calculation. Like Algorithm 6, it is also performed for a single atom B. The algorithm processes all PPDs relevant to this atom in an OpenMP-parallelized loop. For each PPD, the algorithm iterates over all atoms D whose localization sphere spans this PPD and all atoms A owned by the MPI rank whose localization sphere spans this PPD. The unusual loop ordering with the loop over PPDs being outermost is cumbersome programmatically but was found to lead to best performance, allowing for OpenMP parallelism to be leveraged for largest grain sizes and for any parallel contention to be avoided. No thread synchronization is required in this loop.

First (step 6), the algorithm retrieves all NGWF products for the atom pair (A, D) from the NGWF product cache. Cache misses lead to the recalculation of the products on the fly. Subsequently (steps 8–11), elements of the exchange matrix are accumulated in a loop over NGWFs on atoms B and A and over all points in the PPD. The accumulated quantity is the product $\varphi_{\alpha}\varphi_{\delta}$ multiplied by $\tilde{u}^{\delta}_{\beta} = \sum_{p}\sum_{s} u_p(\mathbf{r}) V^{ps}\left(f_s|\varphi_{\gamma}\varphi_{\beta}\right)K^{\delta\gamma}$, which completes the integration required for obtaining the exchange matrix element $\tilde{X}_{\alpha\beta}$ [cf. Eq. (15)]. Once the algorithm completes, a column stripe of the exchange matrix corresponding to atom B has been calculated.

F. Gradient with respect to the NGWFs

The last component of our algorithm is the calculation of the functional derivative of $\tilde{E}_{\mathrm{HFx}}$ with respect to an NGWF—a rather involved procedure required for *in situ* optimization of NGWFs. The calculation is split into two terms, G^{α}_1 and G^{α}_2, where α is the NGWF with respect to which we differentiate, as defined in Eq. (23).

Calculating the G_1 term is straightforward, beginning already in step 12 of Algorithm 7, where the auxiliary quantity k_{Bb} is calculated for every atom B occurring in the atom pairs (B, C) assigned to an MPI rank. Subsequently, in step 22 of Algorithm 5, Algorithm 8 is invoked for every atom B owned by the MPI rank. This is a simple algorithm, contracting the auxiliary quantity k_{Bb} with the density kernel over the index Bb. Following its completion, the contravariant version of term G_1 has been calculated.

ALGORITHM 8. Accumulate contravariant gradient G_1^{Aa} for one atom B owned by this MPI rank.

1: **for all** PPDs relevant to atom B **do**
2: **for all** my atoms A participating in this PPD **do**
3: **if** atom A is an X-neighbor of atom B **then**
4: **for all** NGWFs b on atom B **do**
5: **for all** NGWFs a on atom A **do**
6: $G_1^{Aa} = G_1^{Aa} + k_{Bb}K^{BbAa}$
7: **end for** a
8: **end for** b
9: **end if**
10: **end for** A
11: **end for** PPD

The calculation is finalized in step 27 of Algorithm 5, where Algorithm 9 is invoked. Here, the calculated NGWF gradient is prepared for subsequent use outside of the HFx code. First, the calculated contravariant gradient is redistributed back to scheme 1, that is, to the atom-based distribution. Each MPI rank obtains the gradient for the NGWFs of the atoms local to it. Next, the points within the PPDs but outside of the NGWF localization radius are zeroed ("shaved"), as they cannot contribute to the gradient, having only been calculated for numerical convenience. Finally, a prefactor is applied [the 2 that is in Eq. (23), together with a grid weight], the contravariant gradient is converted to the covariant form, and reciprocal-space preconditioning[29] is performed on the gradient. At this point, the calculation of the first term, in its final form, is completed.

Calculation of the second term G_2 is more involved. It is performed entirely by Algorithm 10, invoked in step 28 of Algorithm 5. The algorithm is best understood by referring to the second term of Eq. (23). This term, crucially, involves a version of the resolution-of-identity operator $\hat{I}_{\delta\alpha}$ that employs SWs centered on atoms D and A that feature in the *bra* of the ERIs in Eqs. (3) and (4). This is unfortunate because, as explained earlier, our algorithm divides the workload by distributing *kets* [atom pairs (B, C)] across MPI ranks. Furthermore, in the evaluation of HFx energy and the first term of the gradient (Algorithm 4, step 6 and Algorithm 5, step 8), our algorithm always operates on SWs centered on *ket* atoms (B and C), and it is those SWs that are cached. Fortunately, we can simply rename dummy indices in the sums to arrive at a less cumbersome expression for the second term in the gradient. Once we rename $Aa\alpha \leftrightarrow Bb\beta$ and $Dd\delta \leftrightarrow Ccy$, we

ALGORITHM 10. Calculate the NGWF gradient term G_2 [Eq. (23)].

1: **for all** my C atoms **do**
2: Determine atoms B relevant to this atom C
3: Determine atoms A relevant to this atom C
4: **OMP for all** relevant A atoms **do**
5: Calculate auxiliary quantity $P_\alpha^\gamma = \varphi_\alpha \sum_\delta K^{\gamma\delta}\varphi_\delta$, thereby eliminating index δ (→Algorithm 11)
6: **end OMP for** A
7: **for all** NGWFs c on atom C **do**
8: **OMP for all** atoms B relevant to atom C **do**
9: **for all** atoms A relevant to atom C **do**
10: **if** A is not an S-neighbor of B **then**
11: **next** A
12: **end if**
13: **for all** NGWFs b on atom B **do**
14: $Q^{CcBb} = Q^{CcBb} + \sum_a P_{Aa}^{Cc}K^{AaBb}$
15: **end for** b
16: **end for** A
17: **end OMP for** B
18: **for all** atoms B relevant to atom C **do**
19: **for all** NGWFs b on atom B **do**
20: Expand Q^{CcBb} in SWs on B and C
21: **OMP for all** PPDs in $B \cap C$ **do**
22: Calculate the expanded potential E^{CcBb} of Q^{CcBb} in PPD
23: **end OMP for** PPD
24: **for** PPDs in $B \cap C$ **do**
25: Accumulate $G_2^{Bb} = G_2^{Bb} + \varphi_{Cc}E^{CcBb}$
26: **end for** PPD
27: **end for** b
28: **end for** B
29: **end for** c
30: **end for** C
31: Finalize NGWF gradient term G_2 (→Algorithm 9)

ALGORITHM 9. Finalize the NGWF gradient term $G_i (i = 1, 2)$.

1: Redistribute the contravariant gradient G_i from scheme 2 back to scheme 1.
2: **for all** atoms A scheme-1-local to this MPI rank **do**
3: **for all** NGWFs a on atom A **do**
4: **for all** PPDs spanned by NGWF Aa **do**
5: Zero points outside of NGWF sphere
6: Apply prefactor
7: **end for** PPD
8: Convert G_i^{Aa} to covariant form
9: Perform reciprocal-space preconditioning
10: **end for** a
11: **end for** A

ALGORITHM 11. Calculate $P_\alpha^\gamma = \varphi_\alpha \sum_\delta K^{\gamma\delta}\varphi_\delta$ for a given atom pair (A, C).

1: **for all** atoms D that are S-neighbors of atom A **do**
2: **for all** NGWFs d on atom D **do**
3: **for all** NGWFs a on atom A **do**
4: Compute $\varphi_{Dd}\varphi_{Aa}$
5: **for all** NGWFs c on atom C **do**
6: $P_{Aa}^{Cc} = P_{Aa}^{Cc} + K^{CcDd}\varphi_{Dd}\varphi_{Aa}$
7: **end for** c
8: **end for** a
9: **end for** d
10: **end for** D

arrive at (where we explicitly indicate summations for clarity) the following:

$$G_2^\beta(\mathbf{r}) = 2\sum_\gamma \varphi_\gamma(\mathbf{r})\hat{I}_{\gamma\beta}\left(\sum_\alpha K^{\alpha\beta}\sum_\delta K^{\gamma\delta}|\varphi_\delta\varphi_\alpha)\right). \tag{25}$$

This expression is more amenable to our approach since its RI operator involves SWs centered on *ket* atoms, as in the previous steps of the calculation. The fact that we now obtain a gradient term for NGWF β and not NGWF α does not matter, as we will need to redistribute the calculated gradients to scheme 1 in any case. The fact that we will operate on φ_α and φ_δ is also not a problem since our algorithm already requires them (cf. Algorithm 7, step 10).

We will now describe Algorithm 10, which calculates the second term in the NGWF gradient. The main loop iterates over atoms C owned by an MPI rank, corresponding to the sum over γ in Eq. (25). For each atom C in the sum, the algorithm first determines the atoms B with respect to whose NGWFs the gradient needs to be calculated. These are S-neighbors of atom C, which are in the list of (B, C) pairs assigned to this MPI rank and will be termed "B atoms relevant to atom C." Similarly, atoms A featured in the second sum (over α) in Eq. (25) are determined. These atoms, which we term "A atoms relevant to atom C," are all atoms that are X-neighbors of relevant B atoms.

For all such atoms A, the algorithm then (steps 4–6) calculates an auxiliary quantity

$$P_\alpha^\gamma(\mathbf{r}) = \varphi_\alpha(\mathbf{r})\sum_\delta K^{\gamma\delta}\varphi_\delta(\mathbf{r}) \tag{26}$$

in PPDs spanned by atom A [cf. the last sum in Eq. (25)]. This is done by a very simple algorithm, Algorithm 11, which we present below. The algorithm is a simple accumulation of the linear combination of NGWF products. The loop over A leverages OpenMP parallelism to make this operation efficient, requiring Algorithm 11 itself to be thread-safe. Given that every instance of Algorithm 11 operates on a different Aa, this can be accomplished without resorting to synchronization mechanisms. Once this is complete, index δ has been eliminated from further calculations.

We note in passing that the same P_α^γ can be calculated on more than one MPI rank since there is some overlap between C atoms in (B, C) atom pairs assigned to MPI ranks. Even so, as will be shown in Sec. V C, the cost of this stage of the algorithm is almost negligible, except in conduction calculations, where it accounts for ≈20% of the total cost—a result of large numbers of NGWFs per atom and lower overlap matrix sparsity.

Algorithm 10 then proceeds to move left through Eq. (25), taking care of the sum over α (steps 13–15). In this way, another auxiliary quantity is calculated,

$$Q^{\gamma\beta}(\mathbf{r}) = \sum_\alpha \varphi_\alpha(\mathbf{r})K^{\alpha\beta}\sum_\delta K^{\gamma\delta}\varphi_\delta(\mathbf{r}), \tag{27}$$

which should be compared with the quantity in parentheses in Eq. (25). This eliminates the index α from further calculations.

The next stage is the application of the RI operator $\hat{I}_{\gamma\beta}$, that is, the calculation of the SW-expanded potential of $Q^{\gamma\beta}$ in all PPDs spanned by the intersection of localization spheres of atoms B and C. $Q^{\gamma\beta}$ is first expanded in SWs (step 20), similarly to what was done earlier to NGWF pair products (steps 5–8 of Algorithm 4). The expanded potential itself (denoted as E^{CcBb}) is calculated in steps 21–23 in the same PPDs, leveraging OpenMP parallelism. Finally, the contributions are multiplied by $\varphi_\gamma(\mathbf{r})$ and summed over γ in steps 24–26, corresponding to the leftmost operation in Eq. (25). The gradient is finalized ("shaved," converted to the covariant form, and preconditioned) through the application of Algorithm 9, just as was done to its first term.

G. Conduction calculations

Apart from the usual mode of operation where only occupied states are considered, ONETEP has the capability to perform conduction calculations, which finds use, e.g., in calculating optical absorption spectra. Conduction calculations optimize an energy expression involving a separate (conduction) density kernel and a projected conduction Hamiltonian.[45] They typically use larger numbers of NGWFs (i.e., not a minimal basis) and larger localization radii of the NGWFs (by a factor of ≈ 1.5), which means that the overlap matrices are not as sparse as in conduction calculations. This, coupled with the fact that in conduction calculations with hybrid functionals the inner (LNV) loop does not involve HFx, leads to a shift of the computational bottlenecks to different parts of the algorithm. Because of the larger NGWF basis, the memory requirement of conduction calculations is also increased relative to valence calculations. For these reasons, we will benchmark conduction calculations separately, and only for the polymer systems, where they are more relevant.

H. Preconverging with a non-hybrid functional

A simple commonly used optimization is to preconverge the calculation with a non-hybrid [typically a GGA (generalized gradient approximation) functional] and, once it is converged, to continue it with a hybrid. The expectation here is for the pre-converged calculation to reach convergence faster than when starting from the initial guess, thus reducing the number of expensive (hybrid functional) self-consistency iterations. The cost associated with the initial steps performed with a GGA is typically much lower.

We briefly examined two such optimizations—in the first one (referred in the text as B3LYP-1), we pre-converged the

calculation with the Perdew–Burke–Ernzerhof (PBE)[46] functional, and once convergence has been obtained, we continued it with B3LYP, optimizing both the density kernel and the NGWFs. In the second optimization (referred in the text as B3LYP-2), we similarly pre-converged with PBE, but then, we kept the NGWF basis fixed, only optimizing the density kernel. This corresponds to running a fixed-NGWF calculation in a GGA-optimized NGWF basis. In this scenario, ONETEP operates similarly to other fixed-atomic-orbital-basis electronic structure codes, but with the advantage of using a pre-optimized minimal set of atomic orbitals (NGWFs).

We demonstrate the excellent accuracy of both optimizations and significant performance gains in Sec. V H.

IV. CALCULATION DETAILS

We demonstrate the behavior and performance of our implementation on two classes of systems. The first class comprises roughly spherical "scoops" of a protein (cGMP-specific phosphodiesterase type 5, PDE5) of increasing size, suitably truncated and protonated. These range from 44 atoms (≈12 Å across) to 4048 atoms (≈54 Å across) and constitute a good model of a typical three-dimensional system of interest in biochemical applications. The second class comprises stacked polymer chains of increasing length (four chains of the PBTZT-stat-BDTT-8 polymer analog, as studied in Ref. 47). These range from 228 atoms (≈21 Å in length) to 2868 atoms (≈265 Å in length) and are a good model of typical systems of interest in materials science. The two classes differ in topology (true 3D systems vs 1D chains) and composition (H-dominated vs C-dominated).

Unless stated otherwise, all calculations used a kinetic energy cutoff of 827 eV, an NGWF localization radius of $8a_0$ (≈4.2 Å), and a minimal NGWF basis (one NGWF on H and four NGWFs on p-block elements). Eight inner loop (LNV) iterations were used throughout. The SW basis quality was $l_{\max} = 3$ and $q_{\max} = 12$, and an exchange interaction cutoff of $20a_0$ (≈10.6 Å) was used. Calculations on polymer chains employed density kernel truncation with a cutoff of $50a_0$ (≈26.5 Å), while the calculations on proteins did not employ kernel truncation. Because the size of the polymer systems could easily be increased in uniform steps, we will use these to demonstrate weak parallel scaling (where the system size is proportional to the number of CPU cores). Strong parallel scaling (where the system size remains constant and the number CPU cores is increased) will be demonstrated for both classes of systems.

Additionally, for the polymer systems, we benchmarked calculations of *conduction states*, where, as explained in Sec. III, the computational bottlenecks are expected to be different. Here, to account for the expected orbital delocalization, we increased the NGWF radius to $12a_0$ (≈6.35 Å), used an extended basis (five NGWFs on H and 13 NGWFs on p-block elements), and increased the number of inner loop iterations to 12, reflecting typical settings used for conduction calculations in ONETEP.

V. RESULTS

All walltimes shown in this section correspond to fully converged calculations with default convergence thresholds and so are good representations of real-life scenarios. However, we feel obliged to note two important points. First, in the interest of reducing the load on the high-performance computing (HPC) facility, we only ran calculations for one outer loop (NGWF) iteration and carefully extrapolated the timings to a fully converged calculation from these measurements, separately taking into account the measured constant overhead and the measured per-iteration time. The number of iterations was taken from GGA calculations that could be run at a fraction of the cost. Cross-checks with full hybrid functional calculations that were actually run to convergence (for a subset of the systems) validated our estimates to be within 0.5% of the true walltimes.

Second, in the interest of clarity, we show results normalized to a constant number of outer loop iterations (e.g., Ref. 12 for the protein systems), whereas in reality, the actual number of iterations differed slightly ($\sigma = 3.0$) between individual systems (for calculations with a GGA and hybrid functional alike). This slight difference in the number of iterations was not systematic, but rather, it reflected statistical noise from how the fragments of the total protein were carved out and truncated, obscuring the linear-scaling behavior that we set out to demonstrate. For the polymer systems, the effect was less pronounced ($\tilde{N}_{\text{iter}} = 10$ and $\sigma = 2.2$) but also present. To disentangle this statistical noise from the actual performance of our algorithm, we chose to show values normalized to the same number of outer loop iterations in all cases.

A. Linear scaling with system size

We begin by demonstrating that our approach is, indeed, linear-scaling, i.e., beyond a certain system size ("onset of linear scaling"), the walltime of the calculation is linearly proportional to the number of atoms in the system. First, however, we stress an important point regarding *memory scaling*. The performance of our approach heavily depends on the amount of available memory, with maximum performance attained with a greedy algorithm that allocates as much memory as it needs to cache all requisite quantities. The RAM requirement of such a greedy approach increases super-linearly with the size of the system, at least for system sizes considered here, and quickly exceeds typical RAM sizes available on today's computing nodes—this happens already at about 150 atoms. Thus, to ensure a fair and realistic benchmark, in this work, we employ the conditions of linear scaling in memory, i.e., we settle on a constant memory requirement per atom and limit the RAM use to this value. We choose this value in such a way as to be able to perform the largest calculations demonstrated here without exceeding the available RAM on the Iridis5 system at the University of Southampton. We will term this the *standard memory requirement*. Additionally, we will show how our implementation scales at a quarter of this value (mimicking a low-memory environment) and with four times as much RAM (mimicking typical high-memory nodes often available on HPC systems), giving an idea of how the employed time-memory trade-offs perform. The detailed values are given in Table II.

For demonstrating the effect of available RAM on calculation walltime, we chose a setup, where all calculations are run on 16 compute nodes (40 CPU cores each). In order to maximize the size of the caches shared by threads, we use the maximum number of OMP threads per non-uniform memory access (NUMA) region, that is,

TABLE II. The assumed memory requirement of the HFx engine (in MB/atom per MPI rank) for valence calculations.

Memory requirement	SWpot cache	Expansion cache	NGWF product cache	Total
Low (×0.25)	2.5	1.25	0.25	4
Standard	10	5	1	16
High (×0.4)	40	20	4	64

we run with 32 MPI ranks, each spawning 20 OMP threads. In this way, each node holds only two MPI ranks and each of the node's two NUMA regions is fully populated by OMP threads.

Figure 3 shows the scaling of the total time for the protein systems. Linear scaling is achieved in all cases, with the onset at about 400 atoms. In the low-memory scenario (green squares), all calculations were feasible, although the largest ones barely completed one outer loop iteration within the maximum job walltime on Iridis5 (60 h). Calculations that cannot complete even one iteration within this limit (and thus cannot be checkpointed and continued later) we deem infeasible. Given that the total calculation runs for 12 iterations, we marked a "walltime limit" on the plot at 720 h. We can thus estimate the maximum system size feasible on 16 Iridis5 nodes to be about 4200 atoms.

At standard memory requirements (blue crosses), all calculations ran 17%–29% faster, but an out-of-memory condition prevented jobs larger than 3328 atoms from running, corresponding to a memory requirement of ≈100 GB per node. This means that with standard memory memory requirement, the maximum limit on job size is dictated not by the available job time but by available memory per node.

At high memory requirements, a further speedup of 15%–50% was achieved, but the maximum job size was only 909 atoms before available RAM was exhausted.

Figure 4 demonstrates linear scaling for the polymer systems. Here, it was feasible to run all systems studied (up to 2868 atoms) with low and standard memory requirements, with all calculations completing in well under 200 h. Compared to the low-memory scenario, the gain from having standard memory requirements was almost constant for all system sizes and averaged at a sizable 26%. In a high-memory scenario, a further speedup of about 20% was achieved, but the largest system that did not run out of memory was 1108 atoms.

We now turn our attention to the performance of conduction state calculations. Because of the different way the energy minimization is structured compared to valence calculations (cf. Sec. III), we note that there is no benefit to caching NGWF expansions in this case, as NGWFs always change between invocations of the HFx engine. In addition, because of the vastly increased number of NGWFs per atom and the increased NGWF localization radius, caching NGWF products becomes counterproductive, as the cache hit ratio achieved with default memory allowance is very low (in the order of 1%–2%). For the above reasons, when benchmarking conduction calculations, we decided to devote the entire memory allowance to the SWpot cache while keeping the total amount of RAM per atom the same as in valence calculations. This is summarized in Table III.

Figure 5 shows the scaling of the total walltime for a conduction calculation on the polymer systems. The largest system that was feasible on 16 nodes had 1768 atoms, and beyond that, calculations with the low memory requirements could not complete a single iteration within the 60 h job time window. With standard memory requirements, calculations ran about 22% faster, but beyond 1548 atoms, they ran out of memory. With high memory requirements, further gains were very modest and the maximum system size was limited to only 448 atoms. Linear scaling was achieved in all cases.

In summary, we find that 16 compute nodes, with about 100 GB RAM on each node devoted to HFx engine caches, were sufficient to

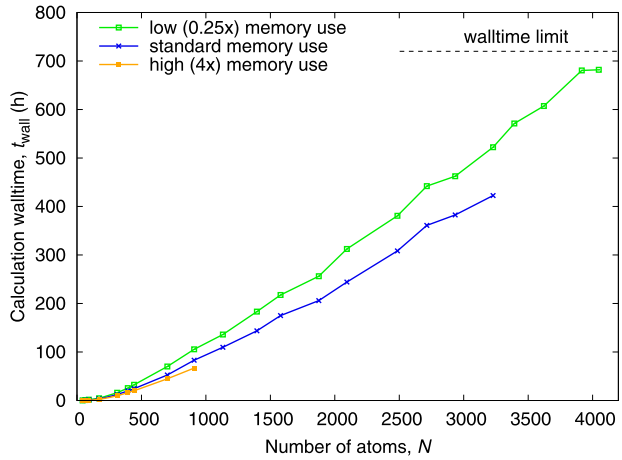

FIG. 3. Total calculation walltime (on 16 compute nodes) for protein systems of different sizes, assuming 12 outer loop iterations, for different memory requirements (see the text). For high and standard memory requirements, the maximum system size is bounded by available RAM. For low memory requirements, all systems (up to 4048 atoms) were feasible on 16 compute nodes.

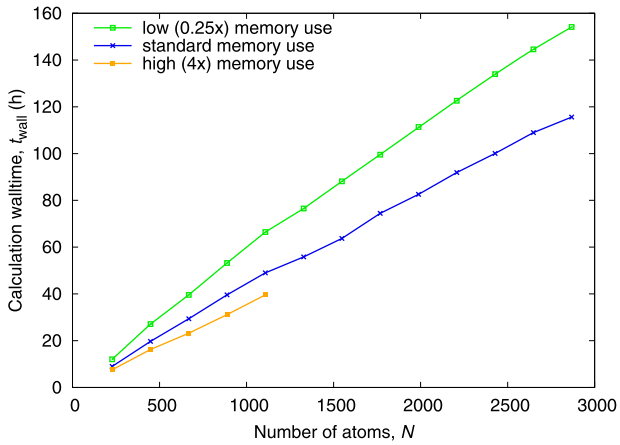

FIG. 4. Total calculation walltime (on 16 compute nodes) for polymer chains of different lengths, assuming ten outer loop iterations, for different memory requirements (see the text). For high memory requirements, the maximum system size is bounded by available RAM. At low and standard memory requirements, all systems (up to 2868 atoms) were feasible on 16 compute nodes.

TABLE III. The assumed memory requirement of the HFx engine (in MB/atom per MPI rank) for conduction calculations.

Memory requirement	SWpot cache	Expansion cache	NGWF product cache	Total
Low (×0.25)	4	0	0	4
Standard	16	0	0	16
High (×0.4)	64	0	0	64

perform valence calculations on systems up to about 4000 atoms and conduction calculations up to about 1800 atoms. These calculations were performed with linear-scaling CPU effort and linear-scaling memory.

We highlight that smaller calculations (up to several hundred atoms) could be made faster, in practice, by not limiting their memory as much. We demonstrate this briefly in Fig. 6, where we show the calculation walltime for a 443-atom protein system as a function of memory devoted to HFx caches. This particular calculation could be made just over twice as fast by using a generous memory allowance.

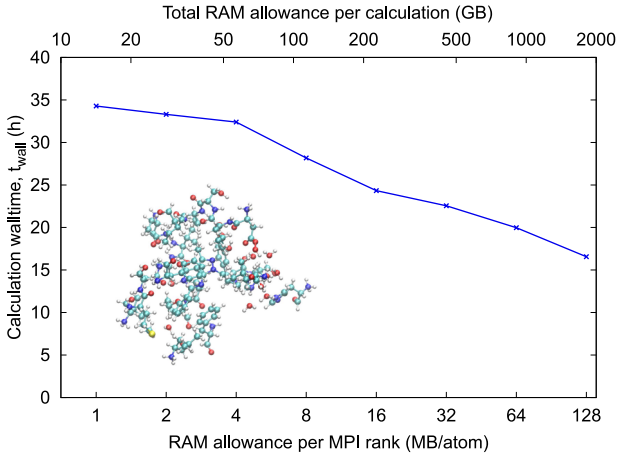

FIG. 6. Total calculation walltime (on 16 compute nodes) for a 443-atom protein system as a function of RAM devoted to HFx caches (log scale).

B. Calculation walltime compared to a GGA calculation

To give a better idea about the cost of HFx calculations, in Figs. 7 and 8, we show calculation walltimes relative to a calculation with a GGA functional (PBE) for the protein systems and polymer systems.

For the protein systems (Fig. 7), the computational effort plateaus at ≈ 100 (for standard memory requirements), meaning that for larger systems, hybrid calculations should be expected to be two orders of magnitude slower compared to calculations with a GGA functional. Using less or more memory has the effect discussed already in Sec. V A. For small systems (below 100 atoms), hybrid calculations are only about an order of magnitude slower than calculations with a GGA functional. The reason for that is that exchange matrix sparsity is reached later than NGWF overlap sparsity, which puts larger HFx calculations at a bigger disadvantage.

For the polymer systems (Fig. 8), HFx performs somewhat better, being consistently slower than a GGA by a factor of ≈ 60 for all system sizes, for standard memory requirements. Using less or more memory has the effect discussed already in Sec. V A. The reason for better performance here is the one-dimensional nature of the polymer systems, which enables exchange matrix sparsity to be reached already for the smallest system, and a more homogeneous structure, which makes load balancing easier.

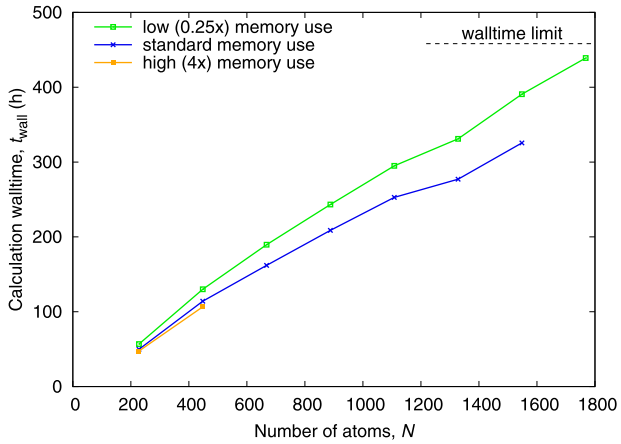

FIG. 5. Total calculation walltime (on 16 compute nodes) for a conduction calculation on polymer chains of different lengths, assuming 11 outer loop iterations, for different memory requirements (see the text). For high and standard memory requirements, the maximum system size is bounded by available RAM. At low memory requirements, the walltime limit in conduction calculations is reached at about 1800 atoms.

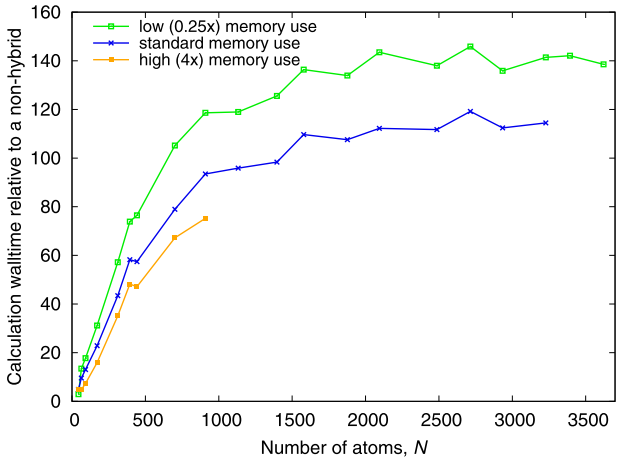

FIG. 7. Total calculation walltime (on 16 compute nodes), relative to a calculation with a GGA functional, for protein systems of different sizes, assuming 12 outer loop iterations, for different memory requirements (cf. Fig. 3). Lower is better.

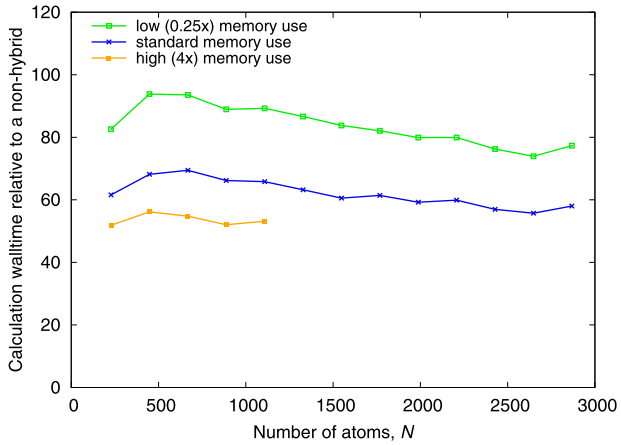

FIG. 8. Total calculation walltime (on 16 compute nodes), relative to a calculation with a GGA functional, for polymer chains of different lengths, assuming ten outer loop iterations, for different memory requirements (cf. Fig. 4). Lower is better.

Finally, we turn to the conduction state calculations on the polymer systems. A comparison against a GGA calculation is shown in Fig. 9. We observe that overall HFx does not perform as well as it did for valence calculations on the same systems, but relative performance clearly improves systematically as the systems become bigger. This indicates that the employed algorithm scales better with system size than standard (GGA) ONETEP, at least at this range of system sizes. For the largest systems, the cost of a hybrid calculation is ≈50–60 times larger than for a GGA.

We note that all calculations shown in this subsection were performed with the maximum number of OMP threads (20), which is optimal for HFx but suboptimal for standard GGA calculations with ONETEP. This is because the scaling with the number of threads

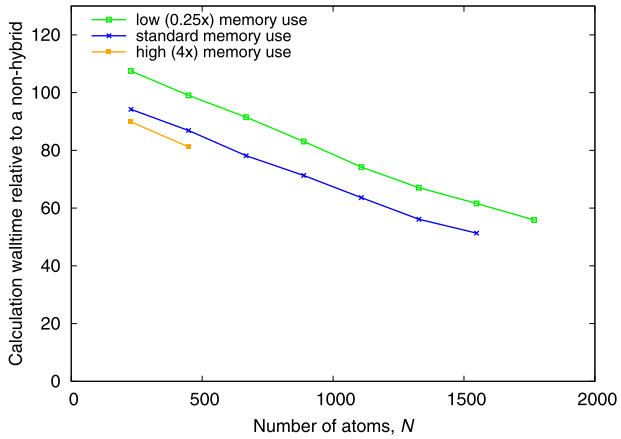

FIG. 9. Total calculation walltime (on 16 compute nodes), relative to a calculation with a GGA functional, for conduction state calculations on polymer chains of different lengths, assuming 11 outer loop iterations, for different memory requirements (cf. Fig. 5). Lower is better.

is much better for the HFx engine than for the rest of ONETEP, with the latter typically achieving optimum performance at 4–5 OMP threads. Indeed, once the GGA calculations are switched from an $N_{\mathrm{MPI}} = 32$ and $N_{\mathrm{OMP}} = 20$ setup to a more favorable $N_{\mathrm{MPI}} = 160$ and $N_{\mathrm{OMP}} = 4$ setup, their performance increases almost exactly by a factor of 2 for all systems. Thus, in practical scenarios, the reported slowdown of HFx relative to GGA would be twice as big.

C. Individual components of computational effort

We will now provide a detailed breakdown of which components of the calculation are responsible for most of the computational effort for the classes of systems under study, demonstrating that each component also separately scales linearly with system size.

The majority of computational effort in HFx calculations with ONETEP is associated with the following stages of the calculation:

1. **Evaluating SWpots**—calculating the values of the spherical wave potentials[33] originating on atomic centers at Cartesian grid points of NGWFs whose centers are within the exchange cutoff. Although this stage has been extensively optimized, it remains a bottleneck in all studied systems. This effort is mitigated by the SWpot cache.
2. **Expansions**—evaluating the linear combinations (cf. Algorithm 5, step 8) of SWpots originating on atomic centers at Cartesian grid points within the exchange cutoff. This effort is mitigated by the expansion cache.
3. **Sum over *Cc***—calculation of the sums over Cc (cf. Algorithm 5, step 19 and Algorithm 6).
4. **Gradient *P***—calculation of the auxiliary term P in the exchange NGWF gradient, Eq. (26).
5. **Load imbalance**—idle time spent waiting for other MPI ranks to finish their share of calculations. This component would be zero if load could be balanced perfectly.
6. **MPI comms**—time spent on message passing (communication between MPI ranks).
7. **HFx other**—all the remaining stages of evaluating the HFx energy and gradient, excluding initialization (calculation of the metric matrix).
8. **Rest of *ONETEP***—all the non-HFx related calculations performed within ONETEP, excluding initialization.
9. **Initialization**—constant overhead, independent of the number of energy evaluations, HFx-related and HFx-unrelated alike, e.g., calculating the metric matrix and structure factor, setting up the parallel distribution, calculating the Ewald energy term, and initialization of radial Bessel functions.

Figure 10 shows a breakdown of the total calculation walltime for the protein systems, on 16 compute nodes, under standard memory requirements. The same data are presented in a cumulative percentage form in Fig. 11. It is clear that all individual components are linear-scaling with the size of the system and that the evaluation of SWpots (red) constitutes the most computationally intensive part of the calculation, accounting for about half of the walltime. For these systems, at standard memory requirements, only

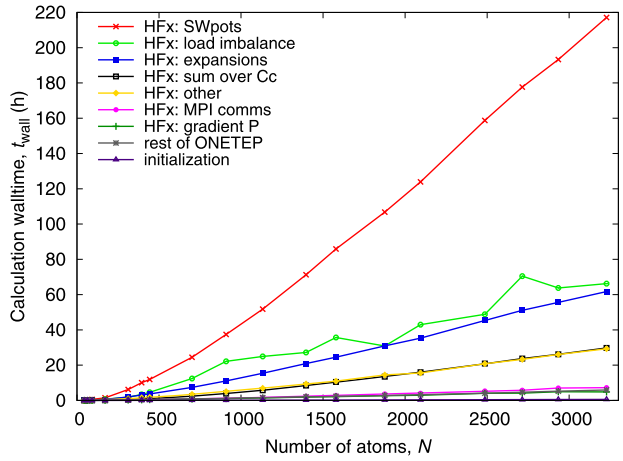

FIG. 10. Individual components of the total calculation walltime (on 16 compute nodes) for protein systems of different sizes, assuming 12 outer loop iterations, for standard memory requirement.

(2.6 ± 1.0)% SWpots can be stored in the cache. Since the SWpots are heavily reused and are stored in the order of reusability, actual SWpot cache hit ratios are much higher here, at (42.6 ± 4.2)%, but this still means that in over half of the cases, SWpots have to be re-evaluated. This cost can be mitigated by increasing the memory allowance for the SWpot cache, but after a certain point, this strategy yields diminishing returns—SWpots evaluated at the outer edges of the system, where there is little NGWF overlap, are necessarily less often reused, and caching them offers less benefit compared to SWpots evaluated at grid points where many NGWFs overlap.

Load imbalance (light green) accounted for (17.4 ± 4.8)% of the calculation walltime. This relatively large value reflects the fact that the protein constitutes a rather heterogeneous system, where it is difficult to simultaneously achieve a good load balance for the energy calculation and for the NGWF gradient calculation. Here, it is the imbalance in the energy calculation that accounts for ≈80% of the total load imbalance. This is mostly caused by the differences in SWpot cache hit ratios between MPI ranks—the current balancing algorithm strives to balance the total number of *Bb*–*Cc* NGWF products, without accounting for the fact that SWpots on the outer edges of the system are unlikely to be cached.

Calculating expansions of NGWF products in terms of SWs (cf. Algorithm 5, step 8) (blue) accounted for (14.5 ± 1.0)% of the calculation walltime. This cost can be mitigated by increasing the allowance for the expansions cache, but the associated memory requirement grows superlinearly, meaning that with linear-scaling memory allowance, cache hit ratios will decrease as the systems are made bigger. Here, in the smallest (44-atom) system, the cache hit ratio was 46%, while in the largest (3228-atom) system, it was only 2.8%.

Summing over the index *Cc* (cf. Algorithm 5, step 19 and Algorithm 6), all the remaining HFx operations were minor contributions, accounting for less than 7% of the total walltime each. The remaining terms (MPI communication, calculation of the NGWF gradient term *P*, and non-HFx operations and initialization) were almost negligible below 2% each.

We now turn our attention to the polymer systems, where corresponding plots are shown in Figs. 12 and 13. Qualitatively, the breakdown of timings resembles that for the protein systems, and all components retain their linear-scaling behavior. Evaluating SWpots remains the most costly component, accounting for (53.7 ± 1.7)% of the walltime. The main difference is the reduced load imbalance (light green) and an increase in the cost of generating the SW expansions (blue). The polymer systems are composed of identical repeat units and increase only along one dimension, which explains the much easier load balancing—associated overheads were below 9%, except for the smallest systems. Higher effort associated with SW expansions, compared to the protein systems, is a consequence of a higher average number of NGWFs per atom (≈3.0 vs ≈2.5). Other contributions to the walltime, similarly as for the pro-

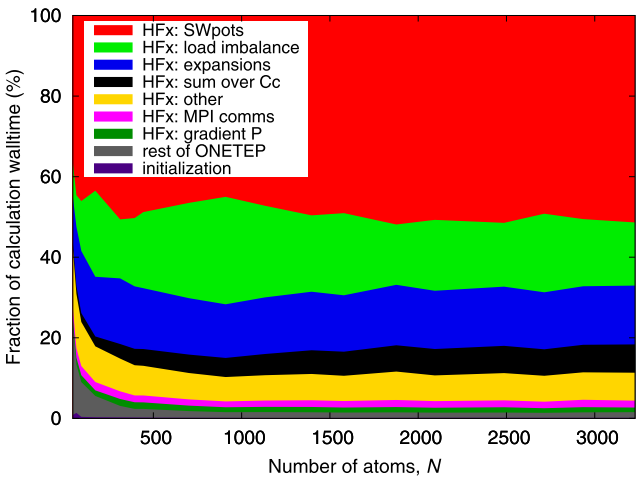

FIG. 11. As in Fig. 10, shown as cumulative percentages of the total.

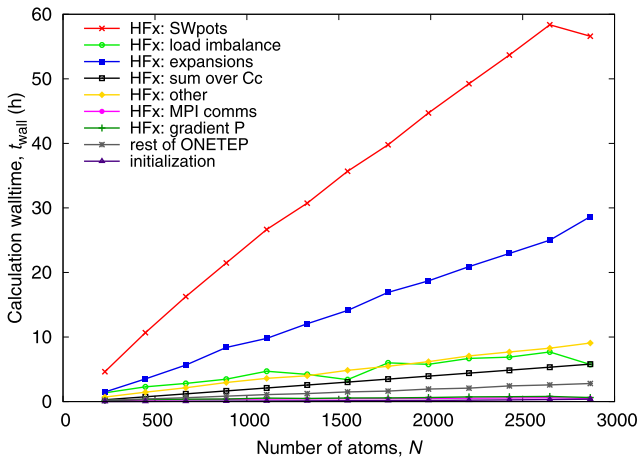

FIG. 12. Individual components of the total calculation walltime (on 16 compute nodes) for polymer chains of different lengths, assuming ten outer loop iterations, for standard memory requirement.

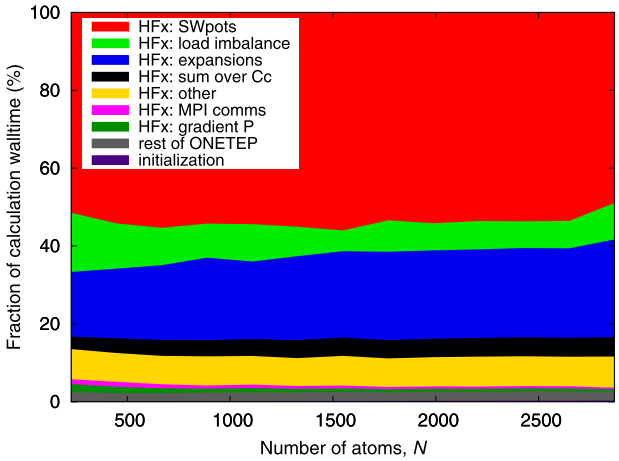

FIG. 13. As in Fig. 12, shown as cumulative percentages of the total.

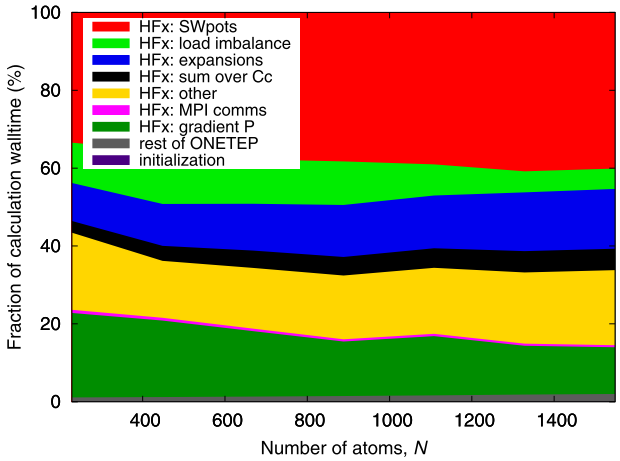

FIG. 15. As in Fig. 14, shown as cumulative percentages of the total.

tein systems, are much smaller. The slight anomaly for the final data point, where the cost of evaluating SWpots decreases despite the increase in the size of the system, is a result of fortuitous load balancing, increasing the average SWpot cache hit ratio from 50.0% to 54.5%.

For conduction state calculations, the distribution of the computational effort is somewhat different. The main reasons are as follows: the use of much bigger NGWF localization regions (which lead to a decrease in metric matrix and overlap matrix sparsities), the use of a large number of NGWFs per atom (vastly increasing the number of *Bb*–*Cc* NGWF products), and the fact that inner loop (LNV) iterations do not require the evaluation of HFx energy in conduction NGWF optimization.

The plots for conduction state calculations on the polymer chains are shown in Figs. 14 and 15. Linear-scaling behavior

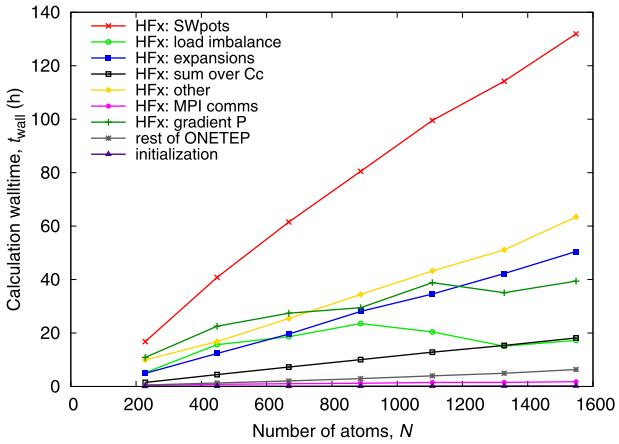

FIG. 14. Individual components of the total calculation walltime (on 16 compute nodes) in a *conduction state* calculation for polymer chains of different lengths, assuming 11 outer loop iterations, for standard memory requirement.

is retained for all the components of the algorithm. Evaluating SWpots remains the bottleneck, but its cost is slightly reduced to (37.9 ± 2.4)%. One reason is the relative increase in the SWpot cache memory allowance (cf. Table III), and another one is the increase in the effort associated with other components, particularly those associated with the NGWF gradient term.

The main difference from the valence calculation is the increased cost of "HFx: other" (orange diamonds) from (7.5 ± 0.2)% to (17.3 ± 1.8)%. This increase is driven by the increased NGWF overlap, leading to larger efforts associated with computing SWpot-NGWF-product overlaps (cf. Algorithm 4, step 6) and the auxiliary term Q in the NGWF gradient [Eq. (27)]. The same reason drives the significant increase in the effort associated with calculating the term P, which increased to (16.0 ± 3.4)% from (1.0 ± 0.4)% for the valence calculation, becoming more costly on average than the calculation of SW expansions.

The cost of evaluating SW expansions is significantly decreased [(21.4 ± 2.3)% in the valence calculation to (12.8 ± 1.9)% here], which is a direct consequence of the fact that inner loop iterations in a conduction calculation do not involve HFx. The cost of imperfect load balancing is similar [(9.3 ± 3.0)%], but here, it is mostly the NGWF gradient calculation that is less than optimally balanced. Finally, the non-HFx components of the calculation ("rest of ONETEP") are no longer negligible, accounting for (1.4 ± 0.3)% of the total calculation walltime.

In summary, we showed that all components of the calculation scale linearly with the system size, that evaluating SWpots is the bottleneck in all types of calculations, that the relative effort associated with other components depends on the type of calculation (valence vs conduction) and type of the system, that communication overheads are negligible, and that load balancing could be improved for more difficult systems.

Here, we briefly demonstrate how the total walltime of HFx calculations depends on the number of compute nodes and what system sizes are feasible. In Fig. 16, we plot the walltimes of calculations for protein systems, with standard memory requirements (cf. Fig. 3), run on 4, 8, 16, and 32 Iridis5 compute nodes (160, 320, 640, and 1280 CPU cores, respectively). The computational

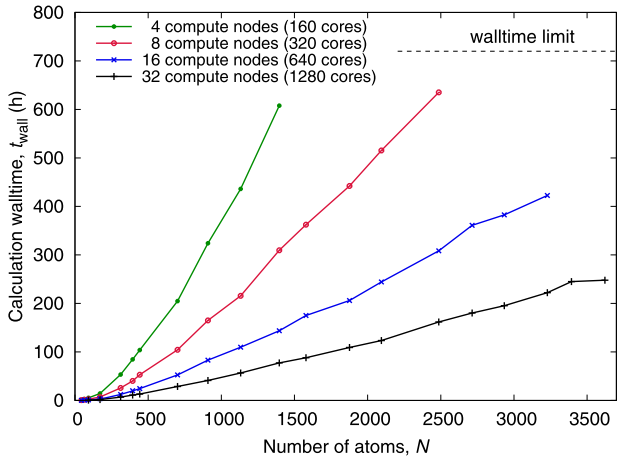

FIG. 16. Total calculation walltime (on 4, 8, 16, and 32 compute nodes) for protein systems of different sizes, assuming 12 outer loop iterations.

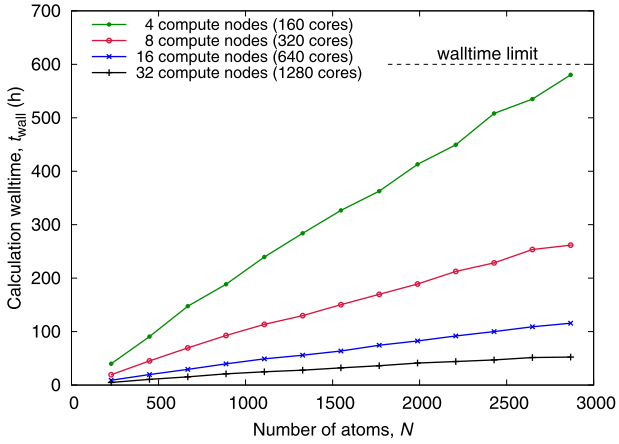

FIG. 17. Total calculation walltime (on 4, 8, 16, and 32 compute nodes) for polymer chains of different lengths, assuming ten outer loop iterations.

effort is linear-scaling (with an onset at ≈400 atoms) in all cases. Calculations on four and eight nodes reach the walltime limit (60 h per outer loop iteration) beyond 1396 and 2486 atoms, respectively. Larger calculations would only be possible on these configurations only if the memory allowance could be increased significantly or if the walltime window per job could be increased beyond 60 h.

D. Calculation walltime and feasibility depending on number of compute nodes

On 16 and 32 nodes, the maximum system size is much larger (3228 and 3622 atoms, respectively) and, in this case, is bounded by available RAM. Larger calculations would be possible on these configurations if memory allowance was decreased. Indeed, we showed in Fig. 3 that *all* the studied systems (up to 4048 atoms) could be already run on 16 compute nodes under low memory requirements.

For the polymer chain systems, we show an equivalent plot in Fig. 17. Again, the calculation is linear-scaling (with the onset already at the smallest system). Here, owing to the fact that the systems are effectively one-dimensional and thus exchange matrix sparsity is reached very early, all the studied systems fit within the walltime limit (60 h per outer loop iteration) even on four compute nodes, and available RAM is not exhausted under standard memory conditions.

E. Strong parallel scaling

We now turn our attention to arguably the most important metric for describing how the computational effort of a calculation changes with the allocated resources. Strong parallel scaling characterizes the speedup obtained when the same calculation is run on an increasingly larger number of CPU cores. Parallel speedup is defined as

$$S(N_{\text{cores}}) = \frac{t(1)}{t(N_{\text{cores}})}, \tag{28}$$

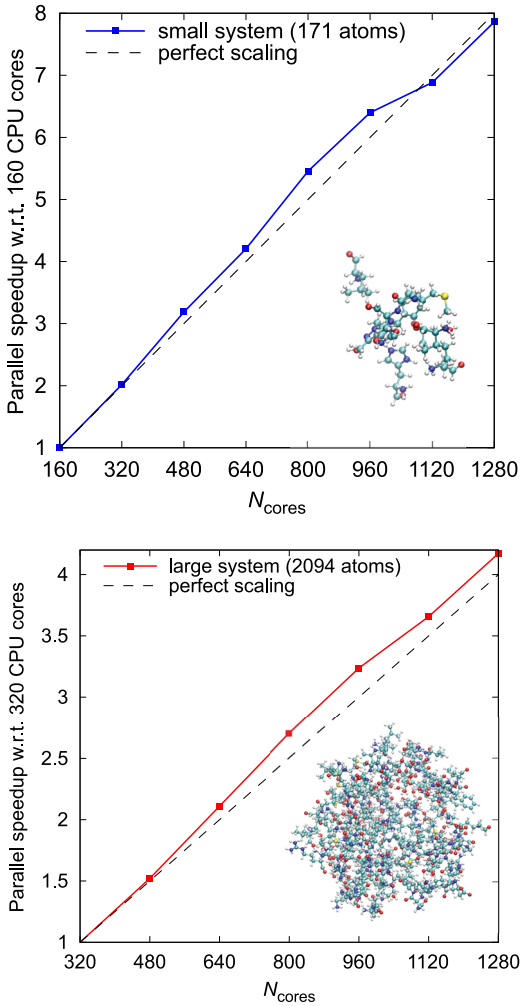

FIG. 18. Strong parallel scaling for two representative protein systems: small (top) and large (bottom). Higher is better.

where $t(N_{\text{cores}})$ is the walltime of the calculation on N_{cores} CPU cores. Since in many scenarios it is not feasible to run the calculation on one CPU core, speedup relative to a fixed number of cores N^0_{cores} is often used instead,

$$S_{N^0_{\text{cores}}}(N_{\text{cores}}) = \frac{t(N^0_{\text{cores}})}{t(N_{\text{cores}})}. \tag{29}$$

Well-parallelized algorithms achieve near-linear speedup, which usually becomes sublinear and then plateaus for larger N_{cores}, due to small but non-zero fractions of the algorithm that could not or have not been parallelized, an effect captured by Amdahl's law.[48] Non-negligible communication overheads and imperfect load balancing also play a role. A linear speedup is typically termed *perfect* or *optimal* scaling because it corresponds to a scenario where all the overheads vanish and the fraction of the algorithm that has not been parallelized is zero.

In the case of our implementation, this simplified analysis does not strictly hold because our algorithm makes good use of the extra memory that becomes available as additional compute nodes are allocated to the calculation. We showed in Sec. V A (see, e.g., Fig. 6) that the performance of our implementation strongly depends on the amount of memory that can be devoted to the calculation. This particular feature allows our approach to *exceed* what is typically deemed to be perfect parallel scaling by improving the degree of caching as more CPU cores are added. We demonstrate this for representative protein systems (one small and one large in Fig. 18) and for representative polymer systems (one small and one large in Fig. 19). We show parallel speedups relative to 160 CPU cores (four nodes) or, where the system is too large to run on four nodes, relative to 320 CPU cores (eight nodes).

For the small protein system (Fig. 18, top), we modestly exceed perfect scaling, except for the last two data points, where the speedup becomes marginally worse than linear. This happens due to imperfect load balancing, which is difficult to achieve when the number of CPU cores exceeds the number of atoms by a factor of about 7. For the large protein system (Fig. 18, bottom), we exceed perfect scaling in all cases, even for the largest number of CPU cores, where at 1280 cores we achieve a 4.17-fold speedup over 320 cores.

The polymer chains, being more homogeneous and practically one-dimensional, constitute an easier system for our approach, which achieves better-than-perfect scaling at least until 1280 CPU cores for a smaller and a larger system alike (Fig. 19). For the larger system, in particular the gains are substantial—an eightfold increase in the number of CPU cores (from 160 to 1280) yields a 10.2-fold speedup.

For conduction state calculations, our algorithm does not scale superoptimally (see Fig. 20), but the scaling remains respectable. For a small system (228 atoms), an eightfold increase in the number of CPU cores offers a 4.22-fold speedup, corresponding to a parallel efficiency of 0.53. For a larger system (888 atoms), which was benchmarked against 320 cores, a fourfold increase in the number of CPU cores yielded a 2.72-fold speedup (efficiency of 0.68). The main culprit responsible for the worse scaling of conduction calculations is parallel load imbalance in the NGWF gradient part. A more careful load balancing scheme, perhaps one dedicated to the particular requirement of conduction calculations, might be able to mitigate the problem.

F. Weak parallel scaling

We will now look at how the computational effort of our approach changes as both the size of the system and the available resources (CPU cores and RAM) are uniformly increased. This is known as weak parallel scaling, and it is easiest to illustrate using parallel speedup relative to the number of CPU cores on which this speedup has been obtained, also known as *parallel efficiency*,

$$e(N_{\text{cores}}) = \frac{s(N_{\text{cores}})}{N_{\text{cores}}}. \tag{30}$$

Since in many scenarios it is not feasible to run the calculation on one CPU core, efficiency relative to a fixed number of cores N^0_{cores} is often used instead,

$$e_{N^0_{\text{cores}}}(N_{\text{cores}}) = \frac{s_{N^0_{\text{cores}}}(N_{\text{cores}})}{N_{\text{cores}}/N^0_{\text{cores}}}. \tag{31}$$

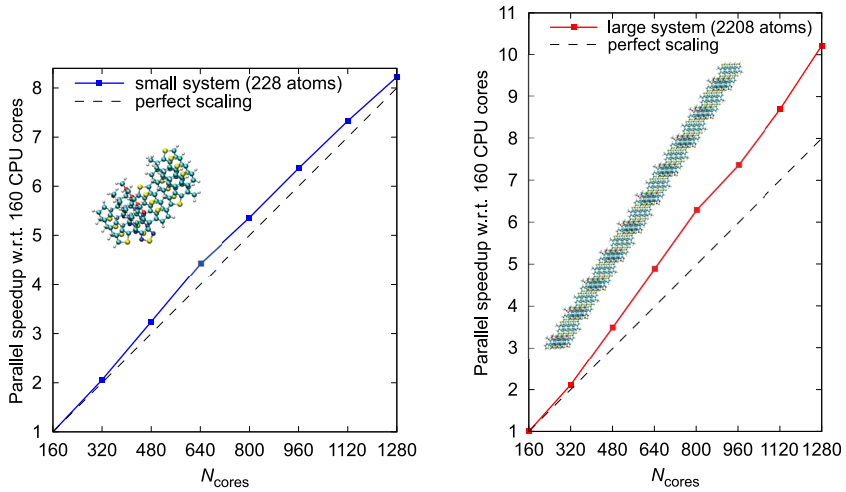

FIG. 19. Strong parallel scaling for two representative polymer chains: small (left) and large (right). Higher is better.

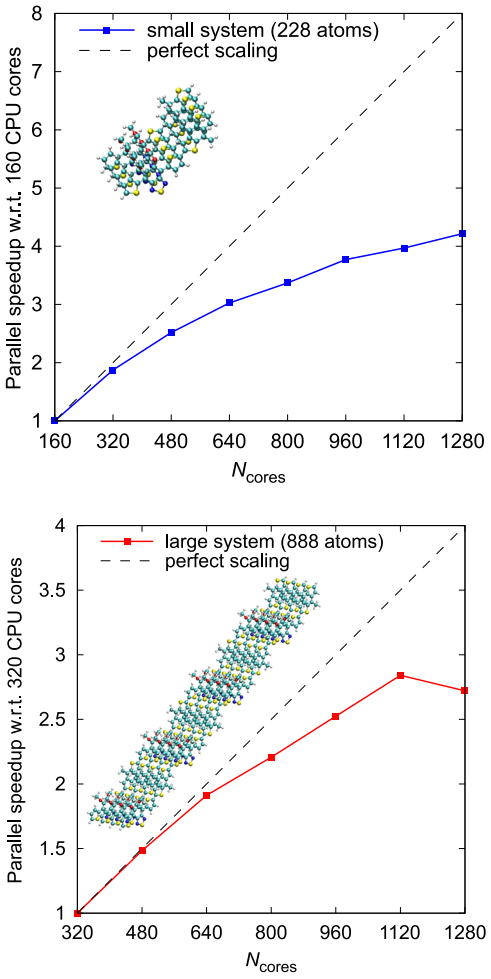

FIG. 20. Strong parallel scaling for two representative polymer chains: small (top) and large (bottom) in a *conduction state* calculation. Higher is better.

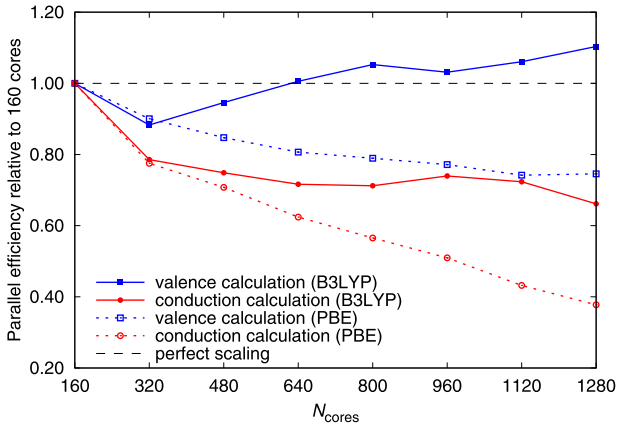

FIG. 21. Weak parallel scaling for the polymer systems (higher is better). The solid lines correspond to a calculation with a hybrid, and the dashed lines show a GGA calculation for comparison. System sizes range from 228 to 1768 atoms for ≈1.4 atoms per CPU core.

Here, we will perform a rather stringent test of performance, showing parallel efficiency relative to 160 CPU cores, with a fairly small (and thus difficult) number of atoms per core (≈1.4).

From the definition of efficiency and the discussion in Sec. V E, it follows that perfect scaling will correspond to $e = 1$. In typical scenarios, Amdahl's law together with load imbalance and communication overheads will cause efficiency to drop below 1 for larger core counts. As our algorithm can make good use not only of the additional CPU cores but also of the additional RAM, we expect it to achieve "superoptimal" efficiency at least in some scenarios.

When investigating weak parallel scaling, one must be able to increase the system size in a uniform fashion so that the number of atoms per CPU core is constant. For this reason, we will only show weak scaling for the polymer systems, where this can be achieved by adding identical units to the system.

The results of our measurements are presented in Fig. 21, which shows plots of efficiency (relative to 160 CPU cores) for both the valence and conduction state calculations. For valence calculations, scaling is near-optimal at lower system sizes and core counts, slightly exceeding perfect scaling for larger systems, which is an excellent result. Standard GGA calculations do not scale as well, although their performance is still good, with efficiencies of about 0.8 for all setups.

For conduction calculations, scaling is suboptimal, but respectable, across the board, with $e \approx 0.7$–0.8, which is still much better than GGA conduction calculations, where efficiency drops to below 0.4 for the largest core counts. Several factors are responsible for the worse scaling performance of conduction calculations compared to valence calculations. First, because of how conduction calculations are structured (cf. Sec. III), there is no benefit to caching NGWF expansions in this case, as NGWFs always change between invocations of the HFx engine. This means that the SW expansion stage does not benefit from the additional RAM at higher core counts. Second, the calculation of the P term in the NGWF gradient does not scale as well as the other components, presumably because NGWF products are not cached either in this case. The additional RAM is not wasted; it is devoted to caching SWpots (see Table III), and this stage does achieve $e > 1$. Finally and most importantly, the impact of parallel load imbalances increases as the calculations move to larger core counts. In contrast to valence calculations, here, it is the NGWF gradient calculation stage that becomes increasingly poorly balanced. This again (cf. the end of Sec. V E) suggests that taking the NGWF gradient stage into account in the load balancing scheme would be worth pursuing.

G. Calculation walltime vs MPI/OMP balance

We will now briefly consider how the calculation walltime depends on the division of work across MPI ranks and OMP threads. ONETEP supports the so-called hybrid parallelism, that is, it runs on multiple MPI processes (termed ranks), each of which spawns OMP threads. Processes reside in separate address spaces (and often on distinct physical machines, termed nodes), while threads

spawned from a single rank share memory. All large data structures are thus distributed across ranks, but shared across threads. It is up to the user to divide the pool of available CPU cores N_{cores} across N_{MPI} ranks and N_{OMP} threads such that $N_{cores} = N_{MPI}N_{OMP}$. The maximum number of ranks is limited by system size. Roughly speaking, the number of atoms must be larger than N_{MPI}, and in practice, load balancing issues cause performance to deteriorate when the number of atoms per rank becomes small. The maximum number of threads is limited by the number of CPU cores on a node.

Standard (non-hybrid) ONETEP calculations typically attain best performance at four or five OMP threads, where load balancing overheads are balanced between MPI ranks and OMP threads.[41] The HFx engine, however, benefits from additional memory that becomes available when the MPI/OMP balance is shifted toward using more OMP threads and fewer MPI ranks. For instance, on a typical 128 GB node with 32 CPU cores, one could allocate 16 GB of memory per MPI rank when using $N_{MPI} = 8$ and $N_{OMP} = 4$ but as much as 64 GB per MPI rank when using $N_{MPI} = 2$ and $N_{OMP} = 16$ (neglecting the memory used by the OS and supporting software architecture). The additional memory devoted to HFx engine caches will substantially increase performance (cf. Fig. 6). However, for this to happen, the algorithm needs to scale well with the number of OMP threads.

In Fig. 22, we show how the calculation walltime changes with the number of OMP threads. Crucially, the total memory use of the HFx engine has been kept constant across all data points in each of the two curves. Clearly, for both systems, the calculation is fastest when 20 OMP threads are used, becoming somewhat slower as balance is shifted toward lower numbers of threads. The performance decrease at 40 OMP threads, despite allocating all of the node's memory to a single MPI rank, is due to the fact that Iridis5 nodes encompass two NUMA regions, and splitting a process across two regions results in a severe performance hit due to nonlocal memory accesses. This is a typical situation on many HPC machines. We did not run the small system with one or two OMP threads because the system was too small for the resulting large N_{MPI}. The large system could not be run with four or fewer OMP threads because the resulting large number of MPI ranks per node led to an out-of-memory condition due to the cost of non-HFx components of ONETEP.

In summary, we confirmed our expectation that our algorithm performs best at the highest possible number of OMP threads, provided that processes do not cross NUMA boundaries. This also confirms excellent OMP scaling of the algorithm, without which the above result would not have been possible.

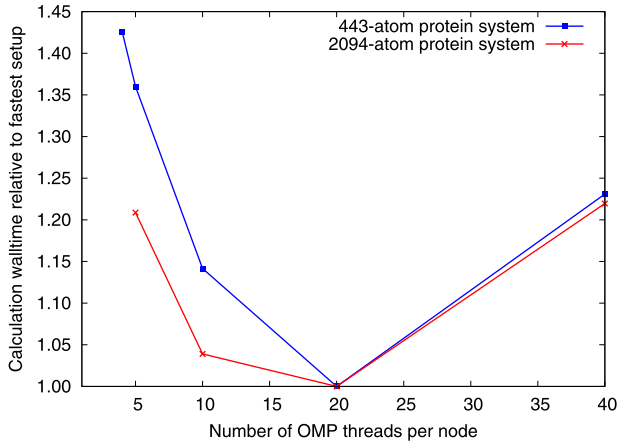

FIG. 22. Effect of the MPI-OMP balance on calculation walltime for a small protein system (blue squares) and a large protein system (red crosses) run on 16 compute nodes (40 CPU cores each). The x axis shows N_{OMP}, and $N_{MPI} = (16 \times 40)/N_{OMP}$. Lower is better.

H. Preconverging with a non-hybrid functional

We now briefly show how the performance of hybrid functional calculations can be improved by employing the optimization described in Sec. III H, and we assess the effect of this optimization on energies.

We begin by demonstrating that the error in the energies associated with restarting from a pre-converged calculation is practically negligible (and certainly much smaller than the error introduced by switching to PBE) in both scenarios—when the calculation is continued with a hybrid functional, optimizing both the density kernel and the NGWFs ("B3LYP-1"), and when only the density kernel is optimized ("B3LYP-2")—with the latter approach avoiding the NGWF gradient computation stage entirely.

As the investigated energies we use (a) the bond-stretch energy curve of ethene, (b) the interaction energy curve of water with a chloride ion, (c) the interaction energy of a sodium cation with its first solvation shell for a number of snapshots obtained from classical molecular dynamics, and (d) the HOMO–LUMO gap of an 888-atom polymer system investigated earlier in the text and shown in Fig. 20. In this way, we probe not only bonded but also non-bonded interactions, and we investigate small systems (a)–(c) and large systems (d).

The smallest systems were run using correspondingly modest computational resources—two MPI processes with four OpenMP threads each for ethene and $H_2O{:}Cl^-$ and eight MPI processes with five OpenMP threads each for $Na^+{:}6H_2O$. Memory load did not exceed 4 GB per MPI processes, and as such, it was not capped. The 888-atom polymer system was run on 16 Iridis5 nodes (two MPI processes per node, with 20 OpenMP threads each), with memory capped to 50 GB per MPI processes.

The results are shown in Figs. 23–25 and in Tables IV and V. For ethene (Fig. 23), the bond-stretch curves are practically indistinguishable between the full B3LYP calculation and both approaches based on restarts, with mean errors in the order of 0.1 kcal/mol or less, while PBE consistently overbinds by as much as 17 kcal/mol (Table IV). For the $H_2O{:}Cl^-$ system (Fig. 24), the interaction energy curves are also very similar between the full B3LYP calculation and both approaches based on restarts, with errors never exceeding 0.1 kcal/mol, while PBE overbinds by 1–2 kcal/mol (Table IV). In the $Na^+{:}6H_2O$ system, PBE underbinds by 1–2.5 kcal/mol, and again, the results of the two restart-based approaches are almost always within 0.1 kcal/mol from the full B3LYP calculation. For these very small systems ($N_{atoms} < 20$), the efficiency gains from using a restart-based approach are either very modest or non-existent (Table V). This is due to the fact that for

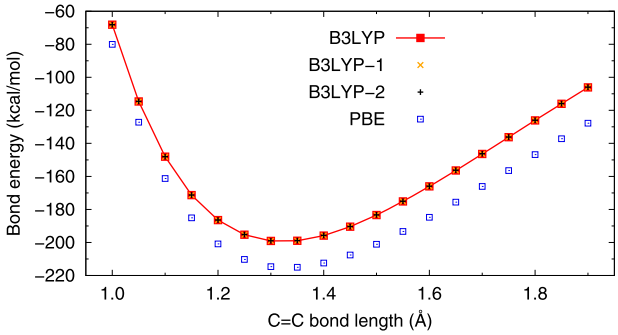

FIG. 23. Bond stretch curve for ethene, computed with B3LYP (red squares), the two approaches where B3LYP is used after pre-converging with PBE (orange crosses and black plusses—see the text), and PBE (blue squares). The line is meant as a guide for the eye.

small systems, all requisite SWs and expansions can be cached in RAM, making the employed time-memory trade-offs very efficient and the calculation of HFx near-optimal. As shown earlier in Figs. 3 and 4, the memory load of caching everything quickly becomes prohibitive, and indeed, the efficiency gains from using a restart-based approach for a large polymer system become quite significant—over twofold for B3LYP-1 and over 19-fold for B3LYP-2 (Table V). As we did not look at the energy of binding for the polymer system, we instead calculate its HOMO–LUMO gap to assess the accuracy of the restart-based approach. Table VI shows that the associated error was in the order of 0.02 eV (less than 2%), again much smaller than the one associated with switching to PBE for the entire calculation.

We thus conclude that it is practicable, and sufficiently accurate, to significantly reduce the cost of hybrid functional calculations by first preconverging with a GGA for all systems except the smallest ones.

I. Demonstration of practicability—example calculations on large imogolite nanotube systems

We finish with a demonstration of the practicability of the presented approach for calculating Hartree–Fock exchange by showing

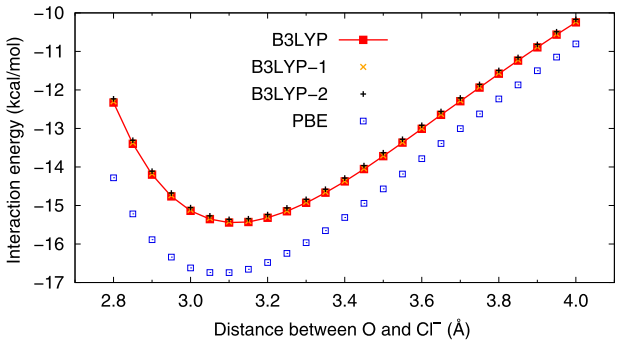

FIG. 24. Interaction energy curve for $H_2O{:}Cl^-$, computed with B3LYP (red squares), the two approaches where B3LYP is used after pre-converging with PBE (orange crosses and black plusses—see the text), and PBE (blue squares). The line is meant as a guide for the eye.

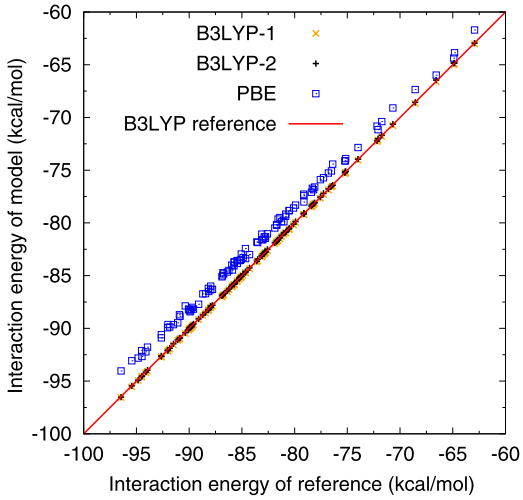

FIG. 25. Interaction energies between Na^+ and its first solvation shell ($6H_2O$) for 100 snapshots extracted at regular intervals from a molecular dynamics run. The results computed with B3LYP (red squares), the two approaches where B3LYP is used after pre-converging with PBE (orange crosses and black plusses—see the text), and PBE (blue squares) are compared against the reference (B3LYP) results.

fully converged results obtained with B3LYP for large nanotube systems (up to 1416 atoms). The potential of these aluminosilicate nanotubes and their derivatives for selective photo-catalytic applications has been recently explored computationally[49–53] and started to be verified experimentally.[54,55] In this work, we calculated the electronic density of states (Fig. 26), the HOMO–LUMO bandgap (Fig. 27), and the free energy of solvation in implicit solvent water (Fig. 28) of pristine (undefected), hydrated, and aluminosilicate imogolite nanotubes of three different sizes—three, five, and seven units (see Ref. 49 for more details).

The structure of the largest considered nanotube, surrounded with implicit solvent, is shown in Fig. 29. Our calculations ran on 32 compute nodes, each with 40 CPU cores and 192 GB of RAM

TABLE IV. Errors in the interaction energy—mean signed, root mean square, and maximum—relative to the B3LYP reference, averaged over all data points.

System	Model	Mean signed error (kcal/mol)	RMS error (kcal/mol)	Maximum error (kcal/mol)
	PBE	−17.017	17.268	21.651
C_2H_4	B3LYP-1	−0.016	0.035	0.091
	B3LYP-2	0.110	0.121	0.159
	PBE	−1.046	1.121	1.953
$H_2O{:}Cl^-$	B3LYP-1	0.021	0.022	0.030
	B3LYP-2	0.085	0.085	0.096
	PBE	1.650	1.689	2.490
$Na^+{:}6H_2O$	B3LYP-1	−0.087	0.091	0.148
	B3LYP-2	−0.012	0.042	0.111

TABLE V. Speed-up (reduction in walltime) of the two restart-based approaches relative to a full B3LYP calculation: $s = t_{B3LYP}/t_{B3LYP\text{-}[12]}$. Higher is better.

System	Model	Speed-up relative to a full B3LYP calculation
C_2H_4	B3LYP-1	0.75
	B3LYP-2	1.02
$H_2O{:}Cl^-$	B3LYP-1	0.71
	B3LYP-2	1.01
$Na^+{:}6H_2O$	B3LYP-1	1.10
	B3LYP-2	1.72
888-atom polymer	B3LYP-1	2.21
	B3LYP-2	19.44

TABLE VI. HOMO–LUMO gaps for the 888-atom polymer system—calculated with PBE, B3LYP, and the two restart-based approaches, and their errors relative to a full B3LYP calculation.

Model	HOMO–LUMO gap (eV)	Error relative to B3LYP (eV)	Error relative to B3LYP (%)
PBE	0.482	−0.810	−62.69
B3LYP	1.293	0.000	0.00
B3LYP-1	1.267	−0.026	−2.01
B3LYP-2	1.281	−0.012	−0.89

and, for the largest system, took about 14 h per NGWF optimization iteration, thus requiring several restarts to complete (calculations typically take 10–20 iterations to fully converge).

As expected, we observe (Figs. 27 and 28) a widening of the HOMO–LUMO gap by about 1.5 eV once the hybrid functional is switched from PBE to a hybrid (B3LYP). The subsequent addition of the water environment (modeled using our minimal-parameter solvent model[56,57]) further increases the gap by about 0.4 eV. The magnitude of the gap appears reasonably well-converged with system size at seven nanotube units.

The calculated free energy of solvation is in the order of −200 kcal/mol per one nanotube unit of length and is more favorable by about 13 kcal/mol when calculated with B3LYP, compared to PBE. It changes appreciably between system sizes, presumably due to the effect of the ends of the nanotube and different structural relaxation.[49] The calculated free energy of solvation still seems slightly underconverged with system size even at seven nanotube units. The presented method development in ONETEP paves the way for follow-up research into this aspect of imogolite nanotubes, as well as hybrid (linear-scaling) DFT simulations of the mechanisms of the assembly of solvated proto-imogolite fragments into the final nanotubes.[58,59]

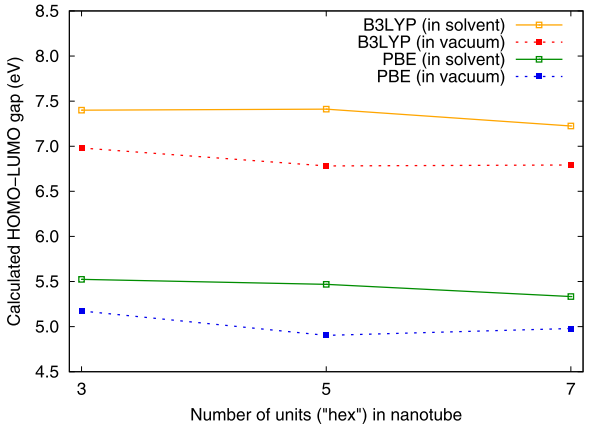

FIG. 27. HOMO–LUMO gap for the three studied nanotube lengths, as predicted by B3LYP (in implicit solvent: solid orange line, empty squares; in vacuum: dashed red line, full squares), and PBE (in implicit solvent: solid green line, empty squares; in vacuum: dashed blue line, full squares).

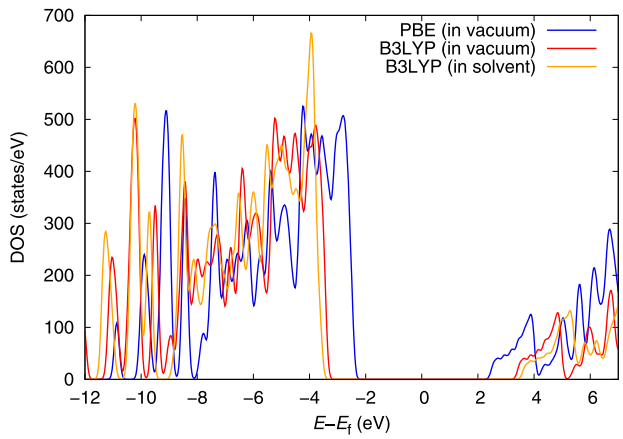

FIG. 26. Electronic density of states for the studied imogolite nanotube. The results obtained in vacuum with a GGA (PBE, blue) and with a hybrid functional (B3LYP, red) are compared. In addition, the effect of implicit solvent is shown for the B3LYP calculation (orange).

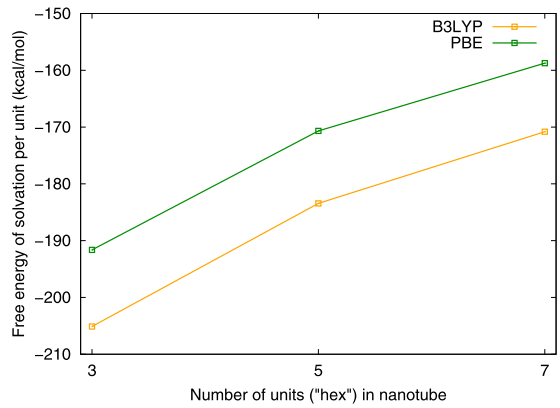

FIG. 28. Free energy of solvation per nanotube unit for the three studied nanotube lengths, as predicted by B3LYP (orange, empty squares) and PBE (green, empty squares).

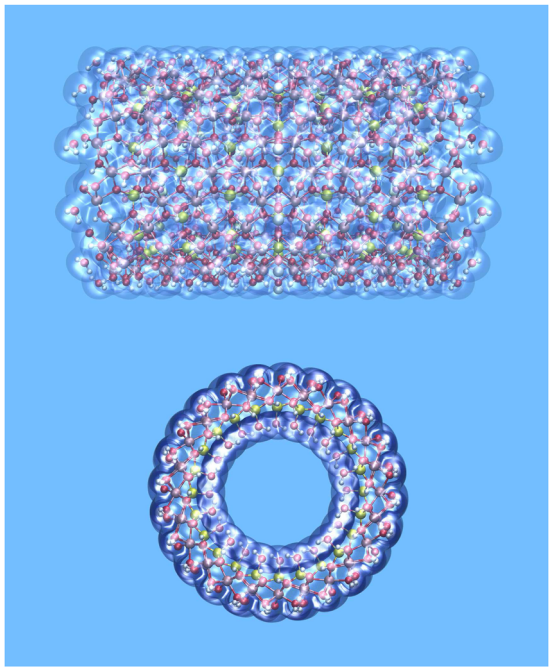

FIG. 29. Structure of the largest studied imogolite nanotube (seven units), indicating the positions of the nuclei (spheres) and the implicit solvent dielectric cavity (surrounding bubbles).

VI. CONCLUSIONS

We presented a massively parallel linear-scaling algorithm for the calculation of Hartree–Fock exchange and hybrid functionals, subsequently discussing and benchmarking its implementation on a number of systems relevant to industrial applications.

Our approach is based on expressing products of localized orbitals (NGWFs) in terms of truncated spherical waves, the expressions for the electrostatic potential of which are known analytically. The carefully thought out parallel distribution of both data and algorithms, together with the aggressive use of time-memory trade-offs, allows our approach to achieve very high parallel efficiency in scenarios where the number of CPU cores is comparable to the number of atoms and beyond.

We showed how on today's machines our approach is able to treat systems of up to about 1500 atoms routinely—requiring several hundred CPU cores to achieve a walltime of under one week. The largest system that we demonstrated to be practicable on 32 compute nodes contained 4048 atoms. The excellent scaling properties of our approach mean that systems even larger than that will be treatable, although they would require substantial computational resources ($N_{\text{cores}} \approx N_{\text{atoms}}$). Conduction state calculations have larger requirements due to reduced matrix sparsity and larger numbers of NGWFs, and the largest system we showed to be practicable on 16 nodes was 1768 atoms.

The computational effort of our approach scales linearly with system size, and the same is true for all of its components individually. The strong parallel scaling is excellent, occasionally becoming superlinear owing to the extensive use of time-memory trade-offs, and although there is still room for optimization in conduction state calculations, even these scale better than corresponding calculations with a GGA. Our implementation scales very well to high thread counts and to large numbers of MPI processes, retaining very good efficiency even in the regime where $N_{\text{cores}} \gg N_{\text{atoms}}$.

In light of the fact that, through the use of a finite auxiliary basis, our approach is an approximation, in the supplementary material, we assessed the magnitude of the introduced error and showed how it converges with the tunable parameters of the SW basis set. In all cases, we found the magnitude of the error to be extremely small and controllable.

The methods presented in this paper not only significantly narrow the performance gap between hybrid functionals and GGAs and *meta*-GGAs in linear-scaling DFT but also pave the way for future developments in ONETEP, which would employ four-center electron repulsion integrals—such as Random Phase Approximation (RPA) or Møller–Plesset perturbation theories or calculations employing screened hybrids.

SUPPLEMENTARY MATERIAL

See the supplementary material for how the additional approximations employed in our approach (the use of an exchange cutoff and the finiteness of the auxiliary basis) are controllable and how the associated errors are very small.

ACKNOWLEDGMENTS

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC), UK, as part of a flagship project of the CCP9 consortium (EPSRC Grant No. EP/P02209X/1). We acknowledge the support of the high-performance computing centers where we ran the calculations: Iridis5 at the University of Southampton (UK) and tryton at the TASK Academic Computer Centre (Gdańsk, Poland). We also acknowledge the UKCP for access to ARCHER and ARCHER2 (EPSRC Grant No. EP/P022030/1) and the MMM hub for access to Young (EPSRC Grant No. EP/T022213/1). J.D. would like to thank Gilberto Teobaldi and Emiliano Poli for fruitful discussions regarding the imogolite nanotube systems and for making their structures available.

AUTHOR DECLARATIONS

Conflict of Interest

The authors declare that there is no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] K. Burke, J. Chem. Phys. **136**, 150901 (2012).
[2] V. N. Staroverov, G. E. Scuseria, J. Tao, and J. P. Perdew, J. Chem. Phys. **119**, 12129 (2003).
[3] C. Ochsenfeld, C. A. White, and M. Head-Gordon, J. Chem. Phys. **109**, 1663 (1998).

[4]E. Schwegler, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **106**, 9708 (1997).
[5]S. A. Maurer, D. S. Lambrecht, D. Flaig, and C. Ochsenfeld, J. Chem. Phys. **136**, 144107 (2012).
[6]M. Guidon, J. Hutter, and J. VandeVondele, J. Chem. Theory Comput. **5**, 3010 (2009).
[7]M. C. Gibson, S. Brand, and S. J. Clark, Phys. Rev. B **73**, 125120 (2006).
[8]M. Marsman, J. Paier, A. Stroppa, and G. Kresse, J. Phys.: Condens. Matter **20**, 064201 (2008).
[9]S. J. Clark and J. Robertson, Phys. Rev. B **82**, 085208 (2010).
[10]H.-Y. Ko, J. Jia, B. Santra, X. Wu, R. Car, and R. A. DiStasio, Jr., J. Chem. Theory Comput. **16**, 3757 (2020).
[11]X. Wu, A. Selloni, and R. Car, Phys. Rev. B **79**, 085102 (2009).
[12]Q. Sun, T. C. Berkelbach, J. D. McClain, and G. K.-L. Chan, J. Chem. Phys. **147**, 164119 (2017).
[13]M. Del Ben, J. Hutter, and J. VandeVondele, J. Chem. Theory Comput. **8**, 4177 (2012).
[14]E. J. Baerends, D. E. Ellis, and P. Ros, Chem. Phys. **2**, 41 (1973).
[15]J. L. Whitten, J. Chem. Phys. **58**, 4496 (1973).
[16]M. Feyereisen, G. Fitzgerald, and A. Komornicki, Chem. Phys. Lett. **208**, 359 (1993).
[17]K. Eichkorn, O. Treutler, H. Öhm, M. Häser, and R. Ahlrichs, Chem. Phys. Lett. **240**, 283 (1995).
[18]F. Weigend, M. Häser, H. Patzelt, and R. Ahlrichs, Chem. Phys. Lett. **294**, 143 (1998).
[19]P. Merlot, T. Kjaergaard, T. Helgaker, R. Lindh, F. Aquilante, S. Reine, and T. B. Pedersen, J. Comput. Chem. **34**, 1486 (2013).
[20]B. I. Dunlap, Phys. Chem. Chem. Phys. **2**, 2113 (2000).
[21]B. I. Dunlap, J. Mol. Struct.: THEOCHEM **529**, 37 (2000).
[22]B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, J. Chem. Phys. **71**, 3396 (1979).
[23]D. P. Tew, J. Chem. Phys. **148**, 011102 (2018).
[24]A. Sodt and M. Head-Gordon, J. Chem. Phys. **128**, 104106 (2008).
[25]J. Dziedzic, Q. Hill, and C.-K. Skylaris, J. Chem. Phys. **139**, 214103 (2013).
[26]J. C. A. Prentice, J. Aarons, J. C. Womack, A. E. A. Allen, L. Andrinopoulos, L. Anton, R. A. Bell, A. Bhandari, G. A. Bramley, R. J. Charlton, R. J. Clements, D. J. Cole, G. Constantinescu, F. Corsetti, S. M.-M. Dubois, K. K. B. Duff, J. M. Escartín, A. Greco, Q. Hill, L. P. Lee, E. Linscott, D. D. O'Regan, M. J. S. Phipps, L. E. Ratcliff, Á. Ruiz Serrano, E. W. Tait, G. Teobaldi, V. Vitale, N. Yeung, T. J. Zuehlsdorff, J. Dziedzic, P. D. Haynes, N. D. M. Hine, A. A. Mostofi, M. C. Payne, and C.-K. Skylaris, J. Chem. Phys. **152**, 174111 (2020).
[27]W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).
[28]C.-K. Skylaris, A. A. Mostofi, P. D. Haynes, O. Diéguez, and M. C. Payne, Phys. Rev. B **66**, 035119 (2002).
[29]A. A. Mostofi, P. D. Haynes, C.-K. Skylaris, and M. C. Payne, J. Chem. Phys. **119**, 8842 (2003).
[30]R. W. Nunes and D. Vanderbilt, Phys. Rev. B **50**, 17611 (1994).
[31]J. M. Millam and G. E. Scuseria, J. Chem. Phys. **106**, 5569 (1997).
[32]P. D. Haynes, C.-K. Skylaris, A. A. Mostofi, and M. C. Payne, J. Phys.: Condens. Matter **20**, 294207 (2008).
[33]Q. Hill, "Development of more accurate computational methods within linear-scaling density functional theory," Ph.D. thesis, University of Southampton, Southampton, 2010.
[34]C.-K. Skylaris, A. A. Mostofi, P. D. Haynes, C. J. Pickard, and M. C. Payne, Comput. Phys. Commun. **140**, 315 (2001).
[35]B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, J. Chem. Phys. **71**, 4993 (1979).
[36]O. Vahtras, J. Almlöf, and M. W. Feyereisen, Chem. Phys. Lett. **213**, 514 (1993).
[37]C.-K. Skylaris, L. Gagliardi, N. C. Handy, A. G. Ioannou, S. Spencer, and A. Willetts, J. Mol. Struct.: THEOCHEM **501–502**, 229 (2002).
[38]B. I. Dunlap, N. Rösch, and S. B. Trickey, Mol. Phys. **108**, 3167 (2010).
[39]R. Polly, H.-J. Werner, F. R. Manby, and P. J. Knowles, Mol. Phys. **102**, 2311 (2004).
[40]S. Goedecker and G. E. Scuseria, Comput. Sci. Eng. **5**, 14 (2003).
[41]K. A. Wilkinson, N. D. M. Hine, and C.-K. Skylaris, J. Chem. Theory Comput. **10**, 4782 (2014).
[42]Message Passing Interface Forum, MPI: A message-passing interface standard, version 2.2, specification, 2009, http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf.
[43]OpenMP architecture review board, OpenMP application program interface version 3.0, 2008, http://www.openmp.org/mp-documents/spec30.pdf.
[44]C.-K. Skylaris, P. D. Haynes, A. A. Mostofi, and M. C. Payne, Phys. Status Solidi B **243**, 973 (2006).
[45]L. E. Ratcliff, N. D. M. Hine, and P. D. Haynes, Phys. Rev. B **84**, 165131 (2011).
[46]J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).
[47]G. Boschetto, H.-T. Xue, J. Dziedzic, M. Krompiec, and C.-K. Skylaris, J. Phys. Chem. C **121**, 2529 (2017).
[48]G. M. Amdahl, in *Proceedings of the April 18–20, 1967, Spring Joint Computer Conference* (Association for Computing Machinery, New York, 1967), pp. 483–485.
[49]E. Poli, J. D. Elliott, Z. Chai, and G. Teobaldi, Crystals **10**, 1051 (2020).
[50]E. Poli, J. D. Elliott, S. K. Chulkov, M. B. Watkins, and G. Teobaldi, Front. Chem. **7**, 210 (2019).
[51]E. Poli, J. D. Elliott, N. D. M. Hine, A. A. Mostofi, and G. Teobaldi, Mater. Res. Innovations **19**, S272 (2015).
[52]E. Poli, J. D. Elliott, L. E. Ratcliff, L. Andrinopoulos, J. Dziedzic, N. D. M. Hine, A. A. Mostofi, C.-K. Skylaris, P. D. Haynes, and G. Teobaldi, J. Phys.: Condens. Matter **28**, 074003 (2016).
[53]J. D. Elliott, E. Poli, I. Scivetti, L. E. Ratcliff, L. Andrinopoulos, J. Dziedzic, N. D. M. Hine, A. A. Mostofi, C.-K. Skylaris, P. D. Haynes, and G. Teobaldi, Adv. Sci. **4**, 1600153 (2017).
[54]M.-C. Pignié, V. Shcherbakov, T. Charpentier, M. Moskura, C. Carteret, S. Denisov, M. Mostafavi, A. Thill, and S. Le Caër, Nanoscale **13**, 3092 (2021).
[55]S. Patra, D. Schaming, P. Picot, M.-C. Pignié, J.-B. Brubach, L. Sicard, S. Le Caër, and A. Thill, Environ. Sci.: Nano **8**, 2523 (2021).
[56]J. Dziedzic, H. H. Helal, C.-K. Skylaris, A. A. Mostofi, and M. C. Payne, Europhys. Lett. **95**, 43001 (2011).
[57]J. C. Howard, J. C. Womack, J. Dziedzic, C.-K. Skylaris, B. P. Pritchard, and T. D. Crawford, J. Chem. Theory Comput. **13**, 5572 (2017).
[58]G. I. Yucelen, D.-Y. Kang, I. Schmidt-Krey, H. W. Beckham, and S. Nair, Chem. Eng. Sci. **90**, 200 (2013).
[59]P. Du, P. Yuan, A. Thill, F. Annabi-Bergaya, D. Liu, and S. Wang, Appl. Clay Sci. **150**, 115 (2017).