Computer Physics Communications 236 (2019) 188–204

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

PySCF-NAO: An efficient and flexible implementation of linear response time-dependent density functional theory with numerical atomic orbitals☆

Peter Koval *, Marc Barbry, Daniel Sánchez-Portal

Centro de Física de Materiales, CSIC-UPV/EHU, Paseo Manuel de Lardizabal 5, 20018 Donostia-San Sebastián, Spain
Donostia International Physics Center, Paseo Manuel de Lardizabal 4, 20018 Donostia-San Sebastián, Spain

ARTICLE INFO

Article history:
Received 22 February 2018
Received in revised form 13 June 2018
Accepted 10 August 2018
Available online 27 August 2018

Keywords:
Numerical atomic orbitals
Dominant product basis
Time-dependent density functional theory
Iterative algorithm

ABSTRACT

We present an algorithm and its implementation to calculate the properties of electronic excitations in molecules and clusters from first principles, using time-dependent density functional theory (TDDFT). The algorithm assumes the use of some localized functions as a basis set to represent the spatial degrees of freedom. It relies on an iterative computation of the induced density according to the Dyson-like equation for the linear response function. The current implementation is built upon so-called numerical atomic orbitals. It is suitable for a wide variety of density functional theory (DFT) software. In this work, we demonstrate TDDFT calculations starting from preceding DFT runs with `SIESTA`, `GPAW` and `PySCF` packages, while a coupling with the other DFT packages such as `Fireball` and `OpenMX` is planned. The mentioned packages are capable of performing *ab initio* molecular dynamics simulations, and the speed of our TDDFT implementation makes feasible to perform a configuration average of the optical absorption spectra. Our code is written mostly in Python language allowing for a quick and compact implementation of most numerical methods and data-managing tasks with the help of NumPy/SciPy libraries and Python intrinsic constructs. Part of the code is written in C and Fortran to achieve a competitive speed in particular sections of the algorithm. Many parts of the current algorithm and implementation are useful in other *ab initio* methods for electronic excited states, such as Hedin's *GW*, Bethe–Salpeter equation and DFT with hybrid functionals. Corresponding proof-of-principles implementations are already part of the code.

Program summary
Program Title: `PySCF-NAO`
Program Files doi: http://dx.doi.org/10.17632/9wgp6255hn.1
Licensing provisions: Apache License, Version 2.0
Programming language: Python (2 or 3), Fortran90 and C
Supplementary material: We provide the source code and input files to organize example and benchmark calculations discussed in the paper.
Nature of the problem: The study of the interaction of photons and charged particles with matter depends upon understanding of electronic excitations in matter. A description of the electronic excitations within time-dependent density functional theory (TDDFT) is popular due to the combination of its reasonable accuracy and its relatively low computational cost. Despite the relative simplicity of TDDFT, its application becomes difficult for quantum systems containing several hundreds of atoms. Such microscopically small systems are relevant in organic electronics, plasmonics and surface science. Therefore, much work has been devoted to the development of adequate electronic structure methods. Moreover, thermal motion of atoms and the presence of solvents affects the properties of electronic excited states in a decisive manner. Modeling of the electronic excited states including the system's dynamics within the Born–Oppenheimer approximation leads to even stronger efficiency requirements from the corresponding electronic structure methods.
Solution method: We discretize the Kohn–Sham Hamiltonian using a basis of numerical atomic orbitals. The induced electronic density is determined in response to a dipolar external perturbation, according to linear response TDDFT, using an iterative algorithm. The method takes advantage of the sparsity generated by the finite support of the numerical atomic orbitals. This allows computing the dynamical polarizability of molecules and clusters containing up to several thousands of atoms.

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author at: Donostia International Physics Center, Paseo Manuel de Lardizabal 4, 20018 Donostia-San Sebastián, Spain.
E-mail address: koval.peter@gmail.com (P. Koval).

https://doi.org/10.1016/j.cpc.2018.08.004

Additional comments including Restrictions and Unusual features: The current algorithm is formulated within the formalism of density response functions. Therefore, one needs to know explicitly an exchange–correlation kernel (second-derivative of energy with respect to variation of the density). The exchange–correlation kernel is analytically known for (semi-)local density functionals, but not for the hybrid density functionals. To date, we implemented only the local density approximation (LDA) for the exchange–correlation kernel to be used with the iterative TDDFT in PySCF-NAO. Spin-restricted formalism for finite systems is covered in the current implementation. Moreover, although our code is prepared to compute the electronic response of all-electron systems, our current implementation of an auxiliary product basis (density-fitting basis) is working best in combination with the use of pseudopotentials.

1. Introduction

Interaction of photons and charged particles with matter is at the root of several widely studied, technologically important phenomena. The behavior of electrons in matter is decisive to the unfolding and outcomes of these phenomena, leading to the notion of electronic excitations. A theoretical description of electronic excitations becomes desirable, important and difficult as the devices shrink to the nanometer scale. There is a vast number of theories and methods to cope with the electronic excitations and an even larger number of their realizations. Among such theories, time-dependent density functional theory (TDDFT) is probably the most popular because of its accuracy and relatively low computational cost [1–3]. Being an *ab initio* theory, TDDFT is in principle as accurate as the employed density functional approximation allows. The computational cost of the TDDFT-based methods depends on the chosen density functional and on the quality of the real-space discretization among other factors. TDDFT has been realized in many software packages. For the sake of presentation, we mention here a few open-source packages such as Abinit (Yambo), Quantum Espresso, Octopus and GPAW.

The packages Abinit (Yamboo) [4,5] and Quantum Espresso [6,7] profit from the plane-wave (PW) basis sets for the discretization of the real-space degrees of freedom. They can be used to model both extended and finite systems with a linear response TDDFT and other more sophisticated methods for excited states, allowing to study complex organic chemicals at interfaces [8,9], molecular crystals [10,11], and semi-conductor clusters containing dozens of atoms [12–15].

The comprehensive suite for TDDFT Octopus profits from the real-space grids (RSG) for discretization of the Kohn–Sham (KS) Hamiltonian [16]. In Octopus, TDDFT has been realized both within linear response approximation (Casida and Sternheimer formulations) as well as with wave-packet time-propagation techniques. The code can be used to model a wide range of finite systems such as large organic compounds [17], metallic [18–20] and semiconductor clusters [21] containing several hundreds of atoms.

The Python-based suite GPAW [22,23] is a comprehensive and easy-to-use software which is capable to use PW, RSG as well as numerical atomic orbitals (NAO) to discretize the real-space variables. A propagation of NAO-discretized wavepackets is particularly suitable for plasmonic applications. For instance, silver and sodium clusters containing several hundreds of atoms were studied with GPAW [24,25].

Besides the named packages, there are a few open-source suites realizing density functional theory (DFT) for which the option of linear response TDDFT is limited or missing. Such packages as Fireball [26], SIESTA [27,28], OpenMX [29], Plato [30], DFTB+ [31] profit from relatively short-ranged NAOs generating sparse and parsimonious representations of the KS Hamiltonian. These packages are designed for efficient *ab initio* molecular dynamics (AIMD) simulations and there is a considerable interest in coupling AIMD with TDDFT [32–37] and other methods [38–41] for electronic excited states. In this work, we present a method and an implementation of a fast linear response TDDFT solver for finite systems. The method is suitable for KS DFT Hamiltonians with semi-local density functionals and using basis sets of localized functions. The particular implementation is applicable to Hamiltonians discretized with NAOs and with the core electrons substituted by norm-conserving pseudo-potentials (PP). Although the scope of the present work is limited to finite systems, we aim to realize the iterative TDDFT also for systems with periodic boundary conditions. In our experience, NAOs can be more advantageous than PWs for systems with large unit cells, such as organic crystals or interfaces [42].

The rest of the paper is organized as follows. In Section 2, we briefly review the equations of DFT and linear response TDDFT for the sake of a self-contained presentation. The basis set of atomic orbitals will be introduced in Section 3. In Section 4, we discuss the general scheme of the iterative TDDFT solver. More details on the construction of the product basis set are given in Section 5. In Section 6, we discuss the implementation of the iterative TDDFT solver, covering installation, unit testing and running in Sections 6.1–6.3, correspondingly. Finally, we present two benchmark calculations in Section 7 and conclude in Section 8.

2. Generalities of linear response TDDFT within KS framework

Within KS DFT, the electron density is given by [43,3]

$$n(\boldsymbol{r}) = \sum_{i\in \mathrm{occ}} f_i \Psi_i^*(\boldsymbol{r})\Psi_i(\boldsymbol{r}), \tag{1}$$

where the summation goes over the occupied KS orbitals $\Psi_i(\boldsymbol{r})$ and f_i are the occupation numbers. In this work, we focus on spin-saturated electronic structure calculations for the sake of computational efficiency. The KS orbitals satisfy the eigenvalue equation

$$\hat{H}_{\mathrm{KS}}\Psi_i(\boldsymbol{r}) = E_i\Psi_i(\boldsymbol{r}), \tag{2}$$

where $\hat{H}_{\mathrm{KS}}$ and E_i are KS Hamiltonian and eigenenergies. The lowest eigenenergy orbitals are occupied according to a Fermi–Dirac statistics so that the density (1) integrates to the necessary number of electrons. The KS Hamiltonian $\hat{H}_{\mathrm{KS}}$ is a single-particle operator. The KS Hamiltonian is an effective Hamiltonian that depends on the electron density (1), and possibly on its derivatives or even on the KS orbitals themselves, subject of the particular choice of the density functional approximation. Linear response TDDFT is a perturbation theory built on top of DFT and it depends chiefly on the particular choice of the density functional approximation. Our iterative TDDFT is suitable for the widely used local density approximation [44,43] (LDA) and generalized gradient approximation [45,43,46] (GGA) of an exchange–correlation (xc) energy density functional $E_{\mathrm{xc}} = E_{\mathrm{xc}}[n]$ while a large part of the method can be applied to more complex density functionals within a suitable framework. In LDA or GGA, the KS Hamiltonian reads

$$\hat{H}_{\mathrm{KS}}(\boldsymbol{r}) = \hat{T} + V_{\mathrm{ext}}(\boldsymbol{r}) + V_{\mathrm{Hxc}}[n](\boldsymbol{r}), \tag{3}$$

where $\hat{T}$ is the kinetic energy operator, the external potential $V_{\text{ext}}(\boldsymbol{r})$ is created by bare nuclei in full-potential (FP) DFT or PPs representing the core electrons in valence-electron DFT. The density-dependent, single-particle potential $V_{\text{Hxc}}[n](\boldsymbol{r}) \equiv \delta E_{\text{Hxc}}[n]/\delta n$ is split into Hartree, exchange and correlation parts [43].

In linear response TDDFT, we want to find an induced density change $\delta n(\boldsymbol{r}, \omega)$ upon a small, generally time-dependent change in the external potential $\delta V_{\text{ext}}(\boldsymbol{r}, \omega)$

$$\delta n(\boldsymbol{r}, \omega) = \int \chi(\boldsymbol{r}, \boldsymbol{r}', \omega)\delta V_{\text{ext}}(\boldsymbol{r}', \omega)d\boldsymbol{r}', \tag{4}$$

where $\chi(\boldsymbol{r}, \boldsymbol{r}', \omega)$ is the so-called (linear) interacting density response function. The utility of the density response function $\chi(\boldsymbol{r}, \boldsymbol{r}', \omega)$ is many-sided. In this work, we use the density response function $\chi(\boldsymbol{r}, \boldsymbol{r}', \omega)$ to define the optical polarizability tensor in the dipole approximation

$$P_{ij}(\omega) = \iint \boldsymbol{r}_i \chi(\boldsymbol{r}, \boldsymbol{r}', \omega)\boldsymbol{r}'_j d\boldsymbol{r} d\boldsymbol{r}', \tag{5}$$

where i and j are indices enumerating Cartesian directions $i = (x, y, z)$. The optical polarizability $P_{ij}(\omega)$ is connected to the optical absorption cross-section and is one of the target properties of our method. Unfortunately, the interacting response function $\chi \equiv \frac{\delta n}{\delta V_{\text{ext}}}$ cannot be expressed explicitly via KS orbitals $\Psi_i(\boldsymbol{r})$, but satisfies a Dyson-like equation [1]

$$\chi(\boldsymbol{r}, \boldsymbol{r}', \omega) = \chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) + \iint \chi_0(\boldsymbol{r}, \boldsymbol{r}'', \omega)K(\boldsymbol{r}'', \boldsymbol{r}''')\chi(\boldsymbol{r}''', \boldsymbol{r}', \omega)d\boldsymbol{r}'' d\boldsymbol{r}''', \tag{6}$$

where $\chi_0 \equiv \frac{\delta n}{\delta V_{\text{eff}}}$ is the non-interacting density response function which, in contrast to the interacting response function χ, can be expressed explicitly via KS orbitals. The effective potential is defined by $V_{\text{eff}}(\boldsymbol{r}) \equiv V_{\text{ext}}(\boldsymbol{r}) + V_{\text{Hxc}}[n](\boldsymbol{r})$, thence the interaction kernel becomes $K \equiv \frac{\delta V_{\text{Hxc}}}{\delta n}$. It is straightforward to find the interaction kernel K in LDA or GGA. Therefore, the linear response TDDFT becomes numerically tractable in the frequency domain. Unfortunately, the formalism of density response functions is not suitable for popular hybrid density functionals relying on the Fock-like operators [47,48]. Likewise, the formalism of density response functions is applicable neither to simple time-dependent Hartree–Fock nor to Bethe–Salpeter equation solvers. In all the mentioned cases, the formalism of density response functions is not applicable because the interaction kernel K appearing in the Dyson equation (6) is unknown and existing approximations are cumbersome [49,50]. However, using the formalism of two-particle Greens function or Casida equation one can get the induced two-particle density for Hamiltonians that contain Fock-like operators. Hybrid functionals are being used increasingly often because of their good balance between accuracy and computational cost. In spite of this relevance, because of the difficulties to formulate the problem using only changes of the density, the hybrid functionals lie outside the scope of the iterative method described in this paper. However, our construction of a basis set to express the products of atomic orbitals is relevant for the treatment of the Fock operator and indeed this approach has been used in Hartree–Fock, Hedin's *GW* approximation and Bethe–Salpeter calculations [38,51,39,52].

The non-interacting response function $\chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega)$ in Eq. (6) possesses a well-known Lehmann representation [53]

$$\chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) = \sum_{nm}(f_n - f_m)\frac{\Psi^*_n(\boldsymbol{r})\Psi_m(\boldsymbol{r})\Psi^*_m(\boldsymbol{r}')\Psi_n(\boldsymbol{r}')}{\omega - (E_m - E_n) + \mathrm{i}\varepsilon}. \tag{7}$$

Here the *products* of KS eigenstates $\Psi_n(\boldsymbol{r})$ appear and ε is a regularization parameter which phenomenologically accounts for the lifetime of the electronic excited states. A discretization of the real-space degrees of freedom in the KS Hamiltonian (3), in the Dyson equation (6) and in the non-interacting response function (7) will be discussed in the following.

3. Numerical atomic orbitals as a basis set

Several AIMD packages profit from the parsimonious expansion of the KS orbitals $\Psi_i(\boldsymbol{r})$ in terms of atomic orbitals (AO)

$$\Psi_i(\boldsymbol{r}) = \sum_a X^i_a f^a(\boldsymbol{r} - \boldsymbol{R}_a). \tag{8}$$

Here the expansion coefficients X^i_a are determined while self-consistently solving Eqs. (1) and (2) and the atomic orbitals $f^a(\boldsymbol{r})$ are normally centered on atomic nuclei at positions $\boldsymbol{R}_a$. Moreover, the atomic orbitals $f^a(\boldsymbol{r})$ possess a radial-angular decomposition

$$f^a(\boldsymbol{r}) = f^a(r)\mathrm{Y}_{l_a,m_a}(\hat{\boldsymbol{r}}), \tag{9}$$

where $\mathrm{Y}_{l,m}(\hat{\boldsymbol{r}})$ are spherical harmonics which depend on the spatial direction $\hat{\boldsymbol{r}}$ and $f^a(r)$ are corresponding radial functions. The spherical harmonics $\mathrm{Y}_{l,m}(\hat{\boldsymbol{r}})$ are chosen as real spherical harmonics [54] when we store the matrix elements of Hamiltonian, overlap and other tensor quantities. However, we often use complex spherical harmonics to facilitate the derivation and computation of the real-spherical-harmonics matrix elements, using the algebra of angular momentum [55] effectively.

There could be several radial orbitals $f^a(r)$ per angular momentum l, but the radial orbitals are usually independent on the magnetic quantum numbers m. In order to assert in the notation for the radial orbitals $f^a(r)$ their *independence* on the magnetic quantum number m_a, we use also a multiplet index ζ

$$f^{a_{\zeta,m}}(\boldsymbol{r}) = f^{\zeta,m}(\boldsymbol{r}) = f^{\zeta}(r)\mathrm{Y}_{l_\zeta,m}(\hat{\boldsymbol{r}}). \tag{10}$$

In this notation, the multiplet index ζ and magnetic quantum number m determine the orbital index $a = a_{\zeta,m}$.

Representation of radial orbitals. In the case of NAO, the radial functions $f^\zeta(r)$ are given on a radial grid $\{r_n\}$, $n \in [0 \dots N_r - 1]$ and possess a well-defined radial cutoff r^ζ_{cut}. The radial grid can be chosen equidistant for PP calculations, but we use a logarithmic grid, profiting from the existing matrix-element machinery. The choice of logarithmic-grid parameters is covered in Section 5.5. The reliance of the logarithmic radial grid leaves open the possibility for FP calculations reusing most of the matrix-element machinery. For the matrix elements of the overlap, Coulomb interaction and Laplacian we use the methods involving fast spherical Bessel transforms put forward by J. Talman [56–60]. The computation of Coulomb matrix elements between non-overlapping functions is done via a multipole method by D. Foerster [61]. For the matrix elements of the xc potential/kernel, we use the numerical methods of integration in real space that are common in quantum chemistry [62–65].

Local product basis: difficulties and advantage. Having chosen a compact expansion of the KS orbitals via NAOs, we face the difficulty of working with the products of KS orbitals that appear in the response function (7). The basis set of NAOs is normally too restricted to represent all the products of NAOs and we need another (auxiliary) basis set to expand these products. One can introduce an auxiliary product basis (PB) $\{F^\mu(\boldsymbol{r})\}$ which is adequate to expand the products of NAOs

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = V^{ab}_\mu F^\mu(\boldsymbol{r}), \tag{11}$$

where V^{ab}_μ are the product vertex coefficients. Here and later in this paper, we understand summation over indices which appear twice on the right-hand side of an equation only. The product vertex coefficients V^{ab}_μ and PB functions $F^\mu(\boldsymbol{r})$ can be constructed according to several methods [66–70]. All these methods preserve

locality of the PB functions $F^{\mu}(\boldsymbol{r})$. The locality of the PB functions $F^{\mu}(\boldsymbol{r})$ and NAOs $f^{a}(\boldsymbol{r})$ results in a *double sparsity* of the product vertex coefficients V_{μ}^{ab} in the limit of large molecules. The double sparsity of the product vertex V_{μ}^{ab} means that there are only $O(N)$ non-zero elements of the product vertex V_{μ}^{ab} albeit each of its indices a, b, μ runs in the $O(N)$ range, where N is the number of atoms in the molecule.

The localized basis sets have an important potential advantage over the non-local PW basis sets. In order to demonstrate this advantage, we contemplate the product vertex V_{μ}^{ab} for the PW basis. The exponential function satisfies the fundamental multiplicative identity $e^{x+y} = e^{x}e^{y}$, hence the product vertex $V_{G''}^{GG'}$ for the plane waves e^{iGr} is given by a Kronecker delta $V_{G''}^{GG'} = \delta_{G+G',G''}$. Despite the remarkable simplicity of the product vertex $V_{G''}^{GG'}$, it contains $O(N^2)$ non-zero elements which is worse than $O(N)$ for the localized NAOs.

Similarly to NAO, any local basis set will have this advantage over the non-local PW basis set. However, the gain will depend on the particular type of the local basis set: on the localization (or spatial extent) of the basis functions and on the construction of the product vertex V_{μ}^{ab}. For instance, RSG would have a strikingly simple product vertex with $O(N)$ non-zero elements. However, RSGs generate a raw, overgenerous description of virtual states. In contrast to RSG, NAOs are rather economical and permit to expand the occupied and lowest-energy unoccupied states with sufficient accuracy. The characterization of NAO's accuracy is beyond the scope of the paper but we refer the reader to several representative convergence studies using the NAOs [38,71–78].

4. Iterative computation of the optical polarizability

After inserting Eqs. (8) and (11) into the non-interacting response function (7), we obtain

$$\chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) = F^{\mu}(\boldsymbol{r})\chi^0_{\mu\nu}(\omega)F^{\nu}(\boldsymbol{r}'), \tag{12}$$

where the non-interacting response matrix is given by

$$\chi^0_{\mu\nu}(\omega) = \sum_{nm}(f_n - f_m)\frac{(X_a^n V_{\mu}^{ab} X_b^m)(X_c^m V_{\nu}^{cd} X_d^n)}{\omega - (E_m - E_n) + \mathrm{i}\varepsilon}. \tag{13}$$

Furthermore, we assume for the interacting response function $\chi(\boldsymbol{r}, \boldsymbol{r}', \omega)$ a similar expansion as in Eq. (12)

$$\chi(\boldsymbol{r}, \boldsymbol{r}', \omega) = F^{\mu}(\boldsymbol{r})\chi_{\mu\nu}(\omega)F^{\nu}(\boldsymbol{r}'). \tag{14}$$

Using the expansions (12) and (14), we turn the Dyson equation (6) into the corresponding matrix expression [32,78]

$$\chi_{\mu\nu}(\omega) = \chi^0_{\mu\nu}(\omega) + \chi^0_{\mu\mu'}(\omega)K^{\mu'\nu'}\chi_{\nu'\nu}(\omega). \tag{15}$$

Here, $K^{\mu'\nu'}$ are the matrix elements of the interaction kernel $K(\boldsymbol{r}, \boldsymbol{r}')$ computation of which we discuss in Section 4.2.

Inserting the expansion (14) into the optical polarizability (5), taking into account Eq. (15) and the symmetry of response functions $\chi^T = \chi$, we get

$$P_{ij}(\omega) = d_i^{\mu}\chi^0_{\mu\mu'}(\omega)[\delta_{\nu}^{\mu'} - K^{\mu'\nu'}\chi^0_{\nu'\nu}(\omega)]^{-1}d_j^{\nu}, \tag{16}$$

where $\delta_{\nu}^{\mu'}$ is Kronecker delta, $d_i^{\mu} = \int F^{\mu}(\boldsymbol{r})\boldsymbol{r}_i\, d\boldsymbol{r}$ are the dipole moments of the PB functions $F^{\mu}(\boldsymbol{r})$. The dipole moments d_i^{μ} appear because of the dipolar approximation to the electron–photon coupling $\delta V_{\mathrm{ext}}(\boldsymbol{r}, \omega) = \hat{\boldsymbol{E}}_0(\omega)\cdot\boldsymbol{r}$, where $\hat{\boldsymbol{E}}_0(\omega)$ is the unit-vector in the direction of the external electric field. Since we assume the linear response regime, a possible frequency dependence of the strength of the external electric field $\boldsymbol{E}_0(\omega)$ can be easily taken into account after the unit-strength induced density $\delta n(\boldsymbol{r}, \omega)$ is determined.

In order to derive Eq. (16), we used the transpose of the interacting polarizability χ^T to facilitate a straightforward interpretation of intermediate quantities. Namely, we split the calculation of the polarizability (16) into a calculation of the effective KS potential $\delta V^{\mu}_{\mathrm{eff},j}(\omega)$

$$[\delta - K\chi^0(\omega)]\delta V_{\mathrm{eff},j}(\omega) = d_j, \tag{17}$$

an application of the non-interacting response to get the induced density $\delta n_j(\omega) = \chi^0(\omega)\delta V_{\mathrm{eff},j}(\omega)$, and a final scalar product with the dipole moments $P_{ij}(\omega) = d_i\delta n_j(\omega)$. In the last equations, we discarded the product indices μ, ν for the sake of clarity. The linear equation (17) is solved with a generalized minimal residual method (GMRES) [79,80] that belongs to Krylov subspace methods. Krylov subspace methods require only the action of a matrix A onto given vectors. In our case, the matrix reads $A = \delta - K\chi^0(\omega)$. The product of this matrix with a vector z can be computed in terms of subsequent matrix–vector products of the non-interacting response function $\chi^0(\omega)$ with the vector z and of the interaction kernel K with yet another vector $y = \chi^0(\omega)z$. The interaction kernel K is a dense matrix and the latter product Ky can be understood as a simple matrix–vector product. The former product with the response function $\chi_0(\omega)$ is more involved, but owing to the explicit expression (13), it can be split further into a sequence of matrix–vector and matrix–matrix operations. The sequence of operations for the matrix–vector product $\chi_0(\omega)z$ must be chosen carefully, as we discuss below.

4.1. Application of the non-interacting response function

There are several possible sequences of operations for the non-interacting response function $\chi_0(\omega)$ to be applied to a vector z. The most advantageous sequence known to us is schematically depicted in Fig. 1. In Fig. 1, we show with boxes the sequence of operations—the inner boxes indicate operations that precede the operations indicated by the outer boxes. In our previous publication [78], we proposed this sequence as an alternative diminishing the memory footprint. The practice has revealed that this sequence is also the fastest alternative for any appreciable system's size (for example, for more than 55 atoms for the compact silver clusters as those described in Section 7.1). The procedure starts with a trace over the product-function index ν to obtain a sparse overlap-like matrix $\alpha^{cd} = V_{\nu}^{cd}z^{\nu}$. Notice that the sparsity of the matrix α^{cd} originates in the spatial localization of atomic orbitals c and d. This operation takes asymptotically $O(N)$ operations and, as a matter of fact, it claims only a minor portion of run time. In the second step, we multiply the sparse matrix α^{cd} with the occupied-state eigenvectors $\beta^{cn} = \alpha^{cd}X_d^n$. The product takes $O(N^2)$ operations because the eigenvectors X_d^n form a dense matrix. In the third step, we multiply the (dense) matrix β^{cn} with the virtual-state eigenvectors $\gamma^{mn} = X_c^m\beta^{cn}$. This operation takes $O(N^3)$ operations, but it becomes the largest time consumer only for big systems (for instance, for the icosahedral cluster Ag_{923} and larger). In the next step, the matrix γ^{mn} is updated with the frequency denominator $\tilde{\gamma}^{mn} = \gamma^{mn}((f_n - f_m)/(\omega - (E_m - E_n) + \mathrm{i}\varepsilon))$. The update is not shown in Fig. 1 and takes a negligible run time. The remaining steps include a matrix–matrix multiplication $\beta_b^n = X_b^m\tilde{\gamma}^{mn}$, which takes $O(N^3)$ operations and a matrix–matrix multiplication $\alpha_{ab} = \beta_b^n X_a^n$ in which only those elements corresponding to overlapping orbitals a and b need to be computed, hence requiring only $O(N^2)$ operations. Finally, a trace over orbital indices a and b delivers the result $\delta n_{\mu} = V_{\mu}^{ab}\alpha_{ab}$.

The sequence described above has not been realized before in sparse matrix algebra and its superiority for practical computations was not recognized in our previous publications. Namely, in our previous works [32,81–83,78], we were using an alternative sequence in which an auxiliary table $A_{\mu}^{an} = V_{\mu}^{ab}X_b^n$ was precomputed

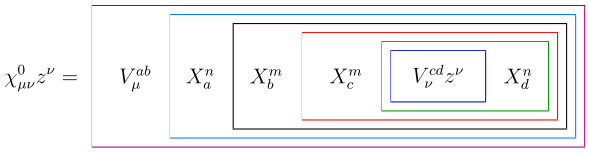

Fig. 1. The operation sequence for the action of the non-interacting response function $\chi_0(\omega)$ on to a vector z. This sequence is adopted in this work as the best alternative in terms of memory footprint and speed.

to diminish the number of mathematical operations. This table takes $O(N^2)$ storage elements of random-access memory (RAM) that is of the same order as required by the KS eigenvectors X^n_a. However, the absolute amount of RAM needed for the table A^{an}_μ is much larger than the RAM occupied by the KS eigenvectors X^n_a and by the vertex coefficients V^{ab}_μ. For instance, for the icosahedral silver cluster Ag_{1415} exemplified later in this work, with a double-zeta polarized basis set of NAOs and single-precision numbers, the RAM taken by the auxiliary table A^{an}_μ will be ∼111 Gigabytes (GB), while it is only ∼1.7 GB for the KS eigenvectors X^n_a and ∼0.21 GB for the dominant product vertex V^{cd}_ν. Comparing these figures, we identify the auxiliary table A^{an}_μ as a hampering bottleneck. Apart from the potential RAM shortage, the size of the table A^{an}_μ causes the machine cache to be easily obstructed and the computation gets slower than with the current sequence of operations.

In current implementation, the product vertex V^{ab}_μ is stored in compressed sparse row (CSR) format, with the product index μ treated as row index and the orbital indices a and b treated as a composite, column index. The effectively matrix–vector operation $\alpha^{cd} = V^{cd}_\nu z^\nu$ is realized with the aid of SciPy library—the standard Python library for scientific computations [84]. Moreover, for the NAO basis sets we are working with, it is not worth to implement the next matrix–matrix product $\beta^{cn} = \alpha^{cd}X^n_d$ in sparse-matrix algebra because of a noticeable degradation of the computational performance with respect to dense-matrix algebra. This solution could be easily improved once we meet a calculation in which this step becomes a bottleneck.

4.2. Application of the interaction kernel in large calculations

The interaction kernel K appearing in Eqs. (6), (16), (17) is a matrix $K^{\mu\nu} \equiv K^{\mu\nu}_{\rm H} + K^{\mu\nu}_{\rm xc}$, where the Hartree $K^{\mu\nu}_{\rm H}$ and xc $K^{\mu\nu}_{\rm xc}$ kernels read

$$K^{\mu\nu}_{\rm H} = \int \frac{F^\mu(\boldsymbol{r})F^\nu(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d\boldsymbol{r}\, d\boldsymbol{r}'; \; K^{\mu\nu}_{\rm xc} = \int F^\mu(\boldsymbol{r})K_{\rm xc}(\boldsymbol{r})F^\nu(\boldsymbol{r})d\boldsymbol{r}. \quad (18)$$

In this work, we use LDA kernel $K_{\rm xc}(\boldsymbol{r})$ that is local in spatial variables [85–87,78]. Due to this locality and to the finite support of the PB functions $F^\mu(\boldsymbol{r})$, the matrix $K^{\mu\nu}_{\rm xc}$ is sparse. However, the Hartree kernel matrix $K^{\mu\nu}_{\rm H}$ is dense for localized functions $F^\mu(\boldsymbol{r})$. We suggested above that the whole application of TDDFT kernel can be sought as a simple matrix–vector operation. However, for large systems, the storage of the $O(N^2)$ matrix elements of TDDFT kernel $K^{\mu\nu}$ is prohibitive. For instance, in case of silver clusters, using the atom-centered PB set, we have to spend about 60 product functions $F^\mu(\boldsymbol{r})$ per atom. Therefore, for such large clusters as Ag_{5083} [88], the storage only of the (upper part of) interaction kernel $K^{\mu\nu}$ in single precision would take 173 GB which is prohibitive for many machines.

In order to alleviate the problem, we suggest here a more sophisticated kernel operator. Namely, the Hartree kernel $K^{\mu\nu}_{\rm H}$ is split into overlapping and non-overlapping parts. The overlapping part of the Hartree kernel $K^{\mu\nu}_{\rm H}$ consists of matrix elements between spatially overlapping PB functions $F^\mu(\boldsymbol{r})$ and $F^\nu(\boldsymbol{r})$, while the rest of the Hartree kernel forms the non-overlapping part. The non-overlapping part of Hartree kernel $K^{\mu\nu}_{\rm H}$ can be computed much faster than the overlapping part, using the multipole moments of PB functions [61,32]. Therefore, in order to save RAM, we store only the overlapping part, spending $O(N)$ memory elements, while the remaining non-overlapping elements are computed on the fly just before a matrix–vector operation Kz that is organized block-wise. In fact, this sophistication slows down the whole iterative loop only by a minor amount. This more sophisticated matrix–vector procedure was tested in our closely related Fortran90 code MBPT-LCAO [34]. To date, this feature is not yet a part of the Python implementation. Therefore, in result Section 7.1, we show the polarizabilities of silver clusters of up to 1415 atoms which is still affordable using the dense-matrix kernel $K^{\mu\nu}$.

5. Construction of the product basis set

The expansion of the products of atomic orbitals (11) in terms of local auxiliary functions is rather *ad hoc*, not yet fully specified. So far, we used only the principal double sparsity of the product vertex coefficients V^{ab}_μ, originating from the locality of NAOs $f^a(\boldsymbol{r})$ and the PB functions $F^\mu(\boldsymbol{r})$. However, the run-time performance of a particular realization of the expansion (11) will depend on the construction details of the PB functions $F^\mu(\boldsymbol{r})$ and the complementary product vertex V^{ab}_μ. In this section, we describe our construction of the PB functions $F^\mu(\boldsymbol{r})$ and the PB vertex V^{ab}_μ.

In our approach, the PB functions are constructed as linear combinations of the original NAOs' products $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$. The linear combinations are chosen in a way to achieve their mutual orthogonality. The orthogonality of the PB set implies its minimal possible size. Unfortunately, the orthogonalization of all the NAOs' products within the system destroys the locality of the original products. Therefore, in order to preserve locality, we limit sets of the original NAOs' products $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$ before their orthogonalization. Namely, we perform the orthogonalization procedure individually, for each atom pair within the molecule. The orthogonalization procedure within the atom-pair constraint leads to what we call a *dominant product* basis [69,89,32,38]. Naturally, there are intra-atomic orbital pairs (when atomic orbital indices a and b belong to the same atom) and inter-atomic orbital pairs (when atomic orbital indices a and b belong to different atoms). The intra-atomic and inter-atomic pairs are treated differently while constructing the dominant product basis, generating the dominant functions centered on atoms $A^\mu(\boldsymbol{r})$ and between atoms $F^\mu(\boldsymbol{r})$ for intra-atomic and inter-atomic pairs, correspondingly. By construction, the inter-atomic dominant functions $F^\mu(\boldsymbol{r})$ are optimal for a given atom pair. However, the inter-atomic dominant functions $F^\mu(\boldsymbol{r})$ of different atom pairs may overlap strongly, causing the unwanted collinearity in the PB set. Fortunately, we have found that the inter-atomic functions $F^\mu(\boldsymbol{r})$ can be reexpressed $F^{\tilde{\mu}}(\boldsymbol{r}) = c^{\tilde{\mu}}_\nu A^\nu(\boldsymbol{r})$ in terms of subsets of the intra-atomic functions $A^\nu(\boldsymbol{r})$, owing to the rather large spatial extent of NAOs. The intra-atomic functions $A^\nu(\boldsymbol{r})$ are not so strongly overlapping because their centers are always separated by at least one inter-atomic distance. Thus, we use the atom-centered functions $A^\nu(\boldsymbol{r})$ for discretizing the interaction kernel (18) and the induced density (4). However, the application of the non-interacting response function to a vector (see Section 4.1) works fastest using the full basis of dominant products, i.e. containing the inter-atomic dominant functions $F^{\tilde{\mu}}(\boldsymbol{r})$ because the corresponding dominant product vertex $V^{ab}_{\tilde{\mu}}$ contains a smallest number of non-zero entries among different choices of PB sets.

To summarize, in our approach we use two types of expansions for the product of orbitals $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$. The first expansion is the dominant product basis defined via a local pair-by-pair orthogonalization of the orbitals' products. We only keep those linear combinations with a norm larger than certain threshold value. The functions in the dominant product basis can be intra-atomic

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = V^{ab}_\mu A^\mu(\boldsymbol{r}),$$

where the orbital indices a and b belong to the same atom, or inter-atomic (bilocal)

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = V^{ab}_{\mu}F^{\mu}(\boldsymbol{r}),$$

where the orbital indices a and b belong to different atoms. This dominant product expansion is the most adequate for the application of the non-interacting response in the iterative procedure. The second expansion of NAOs' products uses only intra-atomic dominant functions $A^{\mu}(\boldsymbol{r})$ as explained in detail in Ref. [78]. We find that this atom-centered expansion is sufficiently accurate and extremely convenient to expand the interaction kernel (18) and the induced density (4).

According to the motivation above, we detail the initial expansion of the NAO products (11), using the atom-centered, intra-atomic dominant functions $A^{\nu}(\boldsymbol{r})$

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = V^{ab}_{\tilde{\mu}}c^{\tilde{\mu}}_{\nu}A^{\nu}(\boldsymbol{r}), \tag{19}$$

where $V^{ab}_{\tilde{\mu}}$ is the dominant product vertex, $c^{\tilde{\mu}}_{\nu}$ is a matrix of conversion between the inter-atomic dominant functions $F^{\tilde{\mu}}(\boldsymbol{r})$ and the intra-atomic dominant functions $A^{\nu}(\boldsymbol{r})$. Atom-centered auxiliary basis to express the density and products of orbitals is commonly used in quantum chemistry [66,67,90,68,70,78]. However, our combination is optimal for the representation of the PB vertex V^{ab}_{μ}. The optimal representation is important for the fast application of the non-interacting response function discussed above in Section 4.1.

In the rest of this section, we will provide further details to realize the expansion (19). In Section 5.1, we describe a general procedure to identify the linearly independent product functions $F^{\mu}(\boldsymbol{r})$. The treatment of the intra-atomic and inter-atomic atom pairs is detailed in Sections 5.2 and 5.3, correspondingly. In Section 5.4, we detail the usage of only intra-atomic, atom-centered functions $A^{\nu}(\boldsymbol{r})$ to represent all the atomic orbital products $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$. An optimal choice of numerical grids is discussed in Section 5.5. Finally, in Section 5.6, we discuss the accuracy and performance issues of presented numerical procedures when they are applied to FP calculations.

5.1. General procedure to identify the dominant functions

The dominant functions $F^{\mu}(\boldsymbol{r})$ are in principle linear combinations of original products of atomic orbitals [69]

$$F^{\mu}(\boldsymbol{r}) = \sum_{ab} C^{\mu}_{ab} f^a(\boldsymbol{r})f^b(\boldsymbol{r}), \tag{20}$$

which are chosen orthogonal to each other. The orthogonality of the functions $F^{\mu}(\boldsymbol{r})$ is required with respect to a Coulomb metric $|\boldsymbol{r}-\boldsymbol{r}'|^{-1}$, which is a common wisdom [67,70,91,92] in quantum chemistry

$$g^{\mu\nu} = \iint \frac{F^{\mu}(\boldsymbol{r})F^{\nu}(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d\boldsymbol{r}d\boldsymbol{r}' = \delta^{\mu\nu}\Lambda^{\mu}. \tag{21}$$

Here, we keep the eigenvalue Λ^{μ} because we do not demand the normality of the basis functions $F^{\mu}(\boldsymbol{r})$. The orthogonality condition (21) can be satisfied if the coefficients of expansion C^{μ}_{ab} in the linear combination (20) are eigenvectors of the Coulomb metric $g^{ab,cd}$ between the products of atomic orbitals

$$g^{ab,cd}C^{\nu}_{cd} = \Lambda^{\nu}C^{\nu}_{ab}, \tag{22}$$

where the matrix elements $g^{ab,cd}$ are defined by

$$g^{ab,cd} = \iint \frac{f^a(\boldsymbol{r})f^b(\boldsymbol{r})f^c(\boldsymbol{r}')f^d(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d\boldsymbol{r}d\boldsymbol{r}'. \tag{23}$$

Moreover, the eigenvalue Λ^{ν} is a Coulomb-weighted measure of the importance of a given linear combination $F^{\mu}(\boldsymbol{r})$ by virtue of Eq. (21). In practice, we use this eigenvalue to discard unimportant functions $F^{\nu}(\boldsymbol{r})$ for which $\Lambda^{\nu} < \Lambda_{\text{tol}}$ according to a tolerance threshold Λ_{tol}. A transformation of the linear combination (20) into the product vertex coefficients in the ansatz (11) is straightforward because the Coulomb metric $g^{ab,cd}$ is symmetric: $C^{\mu}_{ab} = V^{ab}_{\mu}$.

The Coulomb metric (23) should not be computed for the whole molecule at once, because the problem will get too resource-demanding for any appreciable molecular size. The set of atomic orbitals $\{f^a(\boldsymbol{r})\}$ participating in the linear combination (20) must be limited both because the resulting PB $\{F^{\mu}(\boldsymbol{r})\}$ must remain localized and because the burden of diagonalization (22) is too high for any appreciable molecule/basis size. The construction of orthogonal linear combinations based on diagonalization procedures has been proposed numerous times [93,94,90,95,92]. However, the diagonalization had been used either in the context of intra-atomic products or in the context of the whole molecule. Using the diagonalization for individual inter-atomic pairs was proposed and first realized by D. Foerster [69] to the best of our knowledge. We call the functions constructed in this way "dominant functions". The dominant functions are built individually for each atom pair. Moreover, we distinguish between local (intra-atomic) and bilocal (inter-atomic) pairs, using the angular momentum symmetry to further facilitate the diagonalization procedure (22).

5.2. Intra-atomic dominant functions

For the calculation of the Coulomb metric (23) when all orbitals $f^a(\boldsymbol{r})$, $f^b(\boldsymbol{r})$, $f^c(\boldsymbol{r})$ and $f^d(\boldsymbol{r})$ belong to the same center, we will use well-known expressions for the product of two spherical harmonics [55]

$$\mathrm{Y}_{l_1,m_1}(\hat{\boldsymbol{r}})\mathrm{Y}_{l_2,m_2}(\hat{\boldsymbol{r}}) = \sum_{L,M} G^{L,M}_{l_1,m_1,l_2,m_2}\mathrm{Y}^*_{L,M}(\hat{\boldsymbol{r}}) \tag{24}$$

via Gaunt coefficients $G^{L,M}_{l_1,m_1,l_2,m_2}$, the Fourier transform of the Coulomb interaction

$$\frac{1}{|\boldsymbol{r}|} = \frac{4\pi}{(2\pi)^3}\int \frac{\exp(\mathrm{i}\boldsymbol{r}\cdot\boldsymbol{p})}{p^2}d\boldsymbol{p}, \tag{25}$$

and the expansion of plane-wave $\exp(\mathrm{i}\boldsymbol{r}\cdot\boldsymbol{p})$ into spherical harmonics

$$\exp(\mathrm{i}\boldsymbol{r}\cdot\boldsymbol{p}) = 4\pi\sum_{lm}\mathrm{i}^l j_l(rp)\,\mathrm{Y}^*_{lm}(\hat{\boldsymbol{r}})\mathrm{Y}_{lm}(\hat{\boldsymbol{p}}). \tag{26}$$

Using also the orthogonality of the spherical harmonics, we obtain

$$g^{ab,cd} = 8\sum_{LM} G^{L,M}_{l_a,m_a,l_b,m_b}G^{L,M}_{l_c,m_c,l_d,m_d}g^{l_a,l_b,l_c,l_d}_{L}. \tag{27}$$

Here a symmetry-adapted Coulomb metric g^{l_a,l_b,l_c,l_d}_{L} reads

$$g^{l_a,l_b,l_c,l_d}_{L} = \int f^{ab}_{L}(p)f^{cd}_{L}(p)dp, \tag{28}$$

where $f^{ab}_{L}(p)$ is a spherical Bessel transform of the radial–orbital products

$$\int f^a(r)f^b(r)j_L(rp)r^2dr = f^{ab}_{L}(p). \tag{29}$$

Due to the independence of the symmetry-adapted Coulomb metric g^{l_a,l_b,l_c,l_d}_{L} on the magnetic quantum number M, the orthogonal linear combinations (20) can be identified separately for each possible angular momentum $L \in [0\ldots 2l_{\max}]$, where $l_{\max}$ is the maximal angular momentum present in the basis of NAO for a given atomic specie. Correspondingly, constructing the product vertex (11) for the atom-centered situation, we use the product of spherical harmonics (24) and NAOs in the form (10) to obtain

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = \sum_{L,M,\zeta} G^{L,M}_{l_a,m_a,l_b,m_b}(-1)^M\mathcal{F}^{L,\zeta}(r)\mathrm{Y}_{L,M}(\hat{\boldsymbol{r}}), \tag{30}$$

where the radial functions $\mathcal{F}^{L,\zeta}(r) = f^{\kappa}(r)f^{\eta}(r)$ are simply products of original radial functions. The index ζ enumerates the set of multiplet indices κ and η. The set of multiplet indices κ and η is limited according to the triangular inequalities $|l_\kappa - l_\eta| \leq L \leq |l_\kappa + l_\eta|$ for Gaunt coefficients [55] hence determining the size of original products sets $\{\zeta\}$ for each angular momentum L individually $\{\zeta\} = \{\zeta\}(L)$. The symmetry-adapted Coulomb metric (28) will have the dimension of the set $\{\zeta\}$. Therefore, inserting the Kronecker delta $\delta_{\zeta,\zeta'} = X_\zeta^{L,\mu} X_{\zeta'}^{L,\mu}$ expressed via the eigenvectors $X_\zeta^{L,\mu}$ of the symmetry-adapted metric (28) into Eq. (30), we obtain a symmetry-adapted decomposition of the orbital product

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = \sum_{L,M,\mu} V_\mu^{L,ab} F^{L,\mu}(r) \mathrm{Y}_{L,M}(\hat{\boldsymbol{r}}), \tag{31}$$

where

$$V_\mu^{L,ab} = G^{L,M}_{l_a,m_a,l_b,m_b}(-1)^M X_\zeta^{L,\mu} \tag{32}$$

and

$$F^{L,\mu}(r) = \sum_\zeta X_\zeta^{L,\mu} \mathcal{F}^{L,\zeta}(r). \tag{33}$$

After the radial functions $F^{L,\mu}(r)$ and vertices $V_\mu^{L,ab}$ are identified, we reenumerate the dominant function indices $(L,\mu) \to \mu$ to exclude their explicit dependence on the angular momentum L as long as it is convenient in the iterative TDDFT. Moreover, in practice we use real spherical harmonics [54] and the vertex coefficients V_μ^{ab} are transformed accordingly.

5.3. Inter-atomic dominant functions

In the case of bilocal atomic pairs, we expand the products of original atomic orbitals into spherical harmonics, centered at a point $\boldsymbol{R}_c$ on the line segment joining the two centers. A general form of this expansion was derived by J. Talman [60]. We will restate his result here to discuss the use of the remaining *cylindrical* symmetry

$$\begin{aligned} f^a(\boldsymbol{r}-\boldsymbol{R}_a)f^b(\boldsymbol{r}-\boldsymbol{R}_b) &= F^{ab}(\boldsymbol{r}) \\ &= \sum_{jLL'M} \begin{pmatrix} l_a & l_b & j \\ m_a & m_b & m \end{pmatrix} \begin{pmatrix} j & L & L' \\ m & M & M' \end{pmatrix} \\ &\quad \times X_{jLL'}(|\boldsymbol{r}-\boldsymbol{R}_c|, R, \alpha) \mathrm{Y}_{LM}(\widehat{\boldsymbol{r}-\boldsymbol{R}_c}) \mathrm{Y}_{L'M'}(\hat{\boldsymbol{R}}), \end{aligned} \tag{34}$$

where $\boldsymbol{R}_c = \alpha\boldsymbol{R}_a + (1-\alpha)\boldsymbol{R}_b$, $0 < \alpha < 1$, $\boldsymbol{R} = \boldsymbol{R}_b - \boldsymbol{R}_a$ and the Wigner $3j$ coefficients $\begin{pmatrix} l_a & l_b & l_c \\ m_a & m_b & m_c \end{pmatrix}$ are used. The functions $X_{jLL'}(r, R, \alpha)$ are expressed in a rather involved form [60]. To simplify the angular-momentum algebra, we diagonalize the corresponding metric $g^{ab,cd}$ in a custom coordinate system with the origin at $\boldsymbol{R}_c$ and the z axis rotated to the joining line $\boldsymbol{R} = R\boldsymbol{z}$. In this case, the spherical harmonic $\mathrm{Y}_{L'M'}(\hat{\boldsymbol{z}}) = \sqrt{\frac{(2L'+1)}{4\pi}}\delta_{M',0}$ is nonzero only for the projection of angular momentum $M' = 0$, and the sum over the magnetic number M reduces to a single term $M = m_a + m_b$. Correspondingly, the projection of angular momentum M is used to generate the symmetry-adapted orthogonal linear combinations of the original orbital products. Analogously to the case of local pairs above, we generally write

$$f^a(\boldsymbol{r}-\boldsymbol{R}_a)f^b(\boldsymbol{r}-\boldsymbol{R}_b) = \sum_p \mathcal{V}_p^{M,ab} \mathcal{F}^{Mp}(\boldsymbol{r}), \tag{35}$$

where $\mathcal{V}_p^{M,ab} = \delta_{m_a+m_b,M}$, $\mathcal{F}^{Mp}(\boldsymbol{r}) = \sum_L \mathcal{F}_L^{Mp}(r)\mathrm{Y}_{LM}(\hat{\boldsymbol{r}})$ and the set of products $\{p\}$ depends on the magnetic number $M = m_a + m_b$ due to the properties of Wigner $3j$ coefficients. The radial functions $\mathcal{F}_L^{Mp}(r)$ are computed via Talman's general formulation (34). The symmetry-adapted metric

$$g_M^{p,p'} = \iint \frac{\mathcal{F}^{Mp}(\boldsymbol{r})\mathcal{F}^{Mp'}(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d\boldsymbol{r}\, d\boldsymbol{r}' \tag{36}$$

has the dimension of the set $\{p\}$ and is computed in momentum space via (fast) Bessel transforms [56,59,96]. After the symmetry-adapted metric $g_M^{p,p'}$ is diagonalized, we use its eigenvectors to form the identity $\delta_{p,p'} = X_p^{M\mu} X_{p'}^{M\mu}$. Inserting this identity into the original expansion (35), we obtain

$$f^a(\boldsymbol{r}-\boldsymbol{R}_a)f^b(\boldsymbol{r}-\boldsymbol{R}_b) = \sum_\mu V_\mu^{M,ab} F^{M\mu}(\boldsymbol{r}), \tag{37}$$

where $V_\mu^{M,ab} = \sum_p \mathcal{V}_p^{M,ab} X_p^{M\mu}$ and $F^{M\mu}(\boldsymbol{r}) = \sum_p X_p^{M\mu}\mathcal{F}^{Mp}(\boldsymbol{r})$. After the vertices $V_\mu^{M,ab}$ are identified, we transform these to the use with real spherical harmonics and rotate them to the common system of coordinates using the Wigner rotation matrices for real-valued harmonics [54].

5.4. Atom-centered product basis

The dominant functions described above have been used in TDDFT, Hedin's *GW* approximation and for solving a Bethe–Salpeter equation [32,38,52]. However, the construction of dominant functions $F^\mu(\boldsymbol{r})$, although mathematically rigorous and optimal within a given pair, has the disadvantage of generating functions with large overlaps within the whole molecule. This disadvantage stems from the localization constraints in the construction procedure, which is repeated independently for each atom pair. Namely, the dominant functions $F^\mu(\boldsymbol{r})$ constructed for different pairs can strongly overlap because the procedure is "aware" of only one atom pair at a time. This fact results in a redundant description of the orbital products when looking from the perspective of the whole system. Such redundancy leads to a large count of the product functions $F^\mu(\boldsymbol{r})$ and a large number of iterations in the calculation of the effective perturbation (17).

Adapting the idea of auxiliary basis sets from quantum chemistry, we reexpress the inter-atomic, bilocal dominant functions

$$F^\mu(\boldsymbol{r}) = c_\nu^\mu A^\nu(\boldsymbol{r}), \tag{38}$$

in terms of the atom-centered, intra-atomic functions $A^\nu(\boldsymbol{r})$ in a sphere of contributing centers. For the local functions $A^\nu(\boldsymbol{r})$, we use the atom-centered, intra-atomic functions discussed in Section 5.2. Experience tells that even two contributing centers can describe bilocal NAO products fairly well [92]. The projection coefficients can be readily calculated $c_\nu^\mu = M^{\mu\nu'}(v^{\nu'\nu})^{-1}$ in terms of matrix elements of the Coulomb interaction

$$M^{\mu\nu} = \int \frac{F^\mu(\boldsymbol{r})A^\nu(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d\boldsymbol{r}\, d\boldsymbol{r}', \; v^{\mu\nu} = \int \frac{A^\mu(\boldsymbol{r})A^\nu(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d\boldsymbol{r}\, d\boldsymbol{r}'. \tag{39}$$

The inversion of the metric $v^{\mu\nu}$ is still a local operation because the set of the atom-centered indices μ, ν is limited to a sphere of contributing centers. The sphere of contributing centers is centered at the midpoint between atoms and it contains not more than a given number of atoms (by default 8) located closest to the center of the sphere.

The conversion ansatz (38) is useful because the conversion of a vector in the atom-centered basis to the dominant product basis and back turns out to be a fast operation. For instance, it is faster to apply the non-interacting response $\chi^0_{\mu\nu}(\omega)$ in the basis of dominant functions $F^\mu(\boldsymbol{r})$ and, on the other hand, it is faster to compute and easier to store the matrix elements of the interaction kernel $K_{\mathrm{Hxc}}^{\mu\nu}$ in the basis of atom-centered functions $A^\mu(\boldsymbol{r})$.

Using the product vertex ansatz (11) and the expansion of bilocal products (38), we get the factorized form of the product

vertex ansatz (19). The form (19) is used in the iterative procedure described in Section 4. The advantage of the form is a small number of non-zero elements in the table $V^{ab}_{\tilde{\mu}}$ and a fast matrix–vector multiplication with the (sparse) matrix $c^{\tilde{\mu}}_{\nu}$.

5.5. Numerical grid parameters

As we mentioned in Section 3, the distinguishing feature of NAOs is the use of numerical grids to discretize the radial functions $f^{\zeta}(r)$ of atomic orbitals (10). The choice of numerical grid is of great importance because it determines the calculation methods of matrix elements. For instance, in `SIESTA` and `GPAW` packages, the numerical grid is chosen to be equidistant, with different grid spacing h^{ζ} for each radial orbital $f^{\zeta}(r)$.

In contrast to `SIESTA`, we use a logarithmic grid to represent the radial functions in `PySCF-NAO`. This choice is motivated by the construction of PB set (covered above in this section), where we employ essentially quantum-chemistry matrix elements of Coulomb interaction. The logarithmic grid discretization of radial orbitals has been used in FP Hartree–Fock calculations by J. Talman [57,58,60] and we adopt his methods. The methods rely on Fourier (Bessel) transform of radial functions [56,59,96]. Therefore the logarithmic grid is defined for both real and momentum space

$$r_n = \exp(\log(r_{\min}) + \Delta\rho n),\ k_n = r_n \cdot (k_{\max}/r_{\max}),\quad n \in [0 \ldots N_r - 1]\,. \tag{40}$$

where $\Delta\rho = (\log(r_{\max}) - \log(r_{\min}))/(N_r - 1)$ and N_r is the total number of points in the grid. There are four independent parameters $r_{\min}$, $r_{\max}$, $k_{\max}$ and N_r defining the logarithmic grids (40). In order to avoid interpolations, we use only one set of logarithmic-grid parameters $r_{\min}$, $r_{\max}$, $k_{\max}$ and N_r for all radial functions $\{f^{\zeta}(r)\}$ present in calculation. These parameters must be chosen carefully for an accurate representation of all equidistant-grid NAOs we import from `SIESTA`. Fortunately, this choice can be done automatically.

For an accurate representation of the `SIESTA`, equidistant-grid NAOs on the sole logarithmic grid, we choose the minimal distance to the origin in real-space grid $r_{\min} = h^{\zeta}|_{\min}$ coinciding with the minimal grid spacing among all NAOs present in the calculation. Furthermore, the extent of the grid in the momentum space $k_{\max}$ is proportional to the inverse of $r_{\min}$. Hence, we fixed $k_{\max} = 1/(\pi r_{\min})$ after experimenting with the accuracy of the recomputed overlap matrix. Using the same method of accuracy estimation, we determined the optimal value for the maximal extent of the logarithmic grid in real space $r_{\max}$. The maximal extent $r_{\max}$ must be a multiple of maximal extent of NAOs r^{ζ}_{cut} because otherwise the small density of the logarithmic grid at the end of the grid prohibits an accurate description of equidistant-grid NAOs. We found that $r_{\max} = 2.3 \cdot r^{\zeta}_{\text{cut}}|_{\max}$ allows for an accurate representation of `SIESTA`'s orbitals on the logarithmic grid (40). Finally, the (default) number of points $N_r = 1024 = 2^{10}$ in the logarithmic grid was chosen such to speedup fast Fourier transforms and to ensure a similar grid spacing near the origin between the logarithmic and equidistant grids. The latter condition is satisfied by this choice because `SIESTA`'s NAOs are given by default in 500 points and, in order to ensure a similar grid spacing near the origin in the logarithmic and equidistant grids, we have to use more points in the logarithmic grid.

5.6. Limitation of present dominant products

The described above numerical procedures are well suitable for pseudo-potential calculations. However, our efforts to apply the dominant product basis to the FP calculations were rather numerically expensive enterprises [51,52]. The implemented construction of the PB set is based on the decomposition of the inter-atom orbital products by J. Talman [60]. This decomposition was originally devised to be applied to FP Hartree–Fock calculations. However, in order to achieve an acceptable accuracy, a careful choice of the expansion center is necessary that in general is different for each pair of numerical orbitals. An implementation of the inter-atomic dominant functions using the radial-function pairs rather than the atomic pairs is easy to realize [51,39,52]. However, the result will be an unwieldy set of NAO-like functions centered in many points between atoms. The angular momentum of these NAO-like functions is not bound from above either in PP nor in FP calculations. However, the angular momentum cutoff must be set higher in FP calculations. Owing to these accuracy-related issues, the computational performance of the multi-center construction degrades dramatically (∼50 times for any molecule containing more than half a dozen of atoms) comparing to that of the one-center-per-atom-pair dominant functions. The dominant functions are accurate and fast for PP orbitals, generating the sparse dominant product vertex $V^{ab}_{\tilde{\mu}}$ in expansion (19). However, the degradation of computational performance for the FP orbitals and a poor scaling of performance with the cardinal number (or angular momentum content) of NAO basis set suggest considering alternative solutions. Perhaps real-space techniques of dealing with the inter-atomic orbital products could be used [70,91,97] for competitive FP calculations.

6. Implementation of the iterative TDDFT

The algorithm we sketched above was implemented in Fortran language several years ago [33,34,32]. The current implementation is done in the Python language while keeping the computationally intensive parts in a Fortran library. Moreover, we implemented the iterative TDDFT as a part of the Python package `PySCF` [98,99], profiting from the open-source distributed development cycle. The `PySCF` package is a large, well-documented suite of programs for quantum chemistry focused on FP, Gaussian-based calculations. The basic methods we use to calculate the matrix elements (between NAOs) are compatible with the FP framework. Therefore we can couple our algorithm to the `PySCF`'s mean-field objects and profit from other functionality already implemented in `PySCF`. Several of our pull requests have been accepted in a `dev` branch of `PySCF` main repository [99], while most recent developments are available in the `nao` branch of our fork [100].

The design of the classes/functions/file distribution is largely determined by the `PySCF` package. Namely, we added a subdirectory `nao` to the root directory of the `PySCF` package to place there our Python code, and a subdirectory `lib/nao` to place there our Fortran library. A suite of unit tests is placed in the subdirectory `nao/test`. A standard Python module `unittest` is used to organize the unit tests (see Section 6.2). Users can view these unit tests also as examples for their own calculations. Several examples are discussed below in Section 6.3.

6.1. Download and installation

We are using open-source, distributed development environment provided by the `GitHub` platform [101]. `GitHub` provides a convenient interface for the distributed development, involving many contributors. One can profit from this distributed development environment employing the version control system `git`, which is widely popular and recommended for managing the presented source code. For instance, on a Linux machine, one can get the recent version of the NAO-enabled `PySCF` package by using the sequence of commands shown in Fig. 2.

```
git clone https://github.com/cfm-mpc/pyscf.git
cd pyscf/
git fetch
git checkout nao
```

Fig. 2. The sequence of commands to get the source code from our fork of the PySCF package.

This sequence downloads (clones) the main repository into a directory `pyscf` and switches (checkout) to the branch `nao`. There is also a way of contributing to open source projects in a fork/push/pull-request cycle. We gather some recommendations tailored to our package on a Wiki page [102].

The design of the installation procedure follows the PySCF package. Namely, we are using `cmake` utility to compile the low-level libraries, then we adjust a `PYTHONPATH` shell variable to allow Python finding the PySCF modules. After the PySCF repository is cloned, and the `nao` branch is checked out, user is supposed to step in into the directory `lib` and perform a compilation procedure. For instance, on a Linux machine, one can compile the NAO-enabled PySCF package by using a sequence of commands shown in Fig. 3. The sequence of commands in Fig. 3 uses the standard `gcc/gfortran` compilers and creates a set of low-level libraries. These libraries will be using system's `blas/lapack/fftw` libraries, which must be installed prior to the compilation of the PySCF package. The architecture file `lib/cmake.arch.inc` contains machine and installation-specific options. The file is not part of the distribution, because it is very much machine dependent. Instead, we provide a collection of these architecture files in the directory `lib/cmake_arch_config/`. The user may find ready-to-go architecture files for popular Python distributions such as Intel Python and Anaconda in this directory. Further installation instructions for the NAO-enabled PySCF package are gathered in the file `lib/nao/README.md`. We advise the users to review it to get the peak-performance and troubleshooting tips.

6.2. Unit tests

We provide a few dozens of unit tests for the essential functionality of the NAO package. We advise to verify these tests before using the NAO-enabled PySCF package. A standard module `unittest` drives these small calculations. The tests are implemented as Python scripts and gathered in the directory `nao/test`. One can start them one-by-one from the command line

```
python nao/test/test_0001_system_vars.py
python nao/test/test_0002_prod_basis.py
...
```

or in a single run

```
python -m unittest discover nao/test
```

If the installation has been done correctly, there should be no error reported. However, some known issues are discussed in the readme file `lib/nao/README.md`.

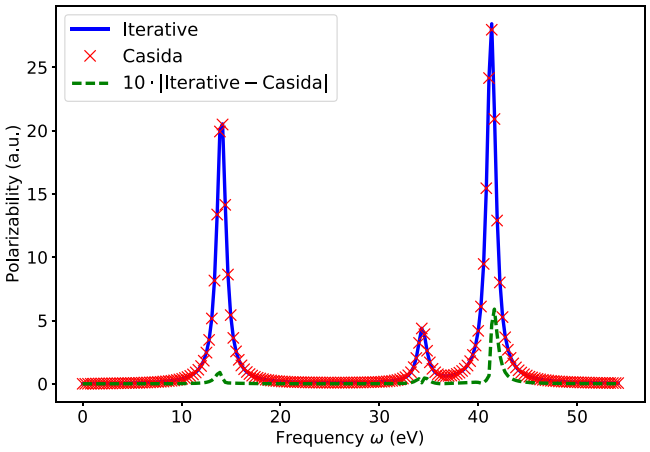

Fig. 4. Polarizability of H_2 molecule computed via Casida method with GTOs in the PySCF package and via the iterative TDDFT with NAOs. The absolute difference between two curves is small in a wide frequency range $\omega = 0 \ldots 2$ a.u.

6.3. Running

After the installation and testing procedures are successfully finished, the user can do DFT/TDDFT calculations using only small scripts, profiting from the flexibility of Python language. Below we discuss five representative applications of NAO-enabled PySCF package illustrating its useful features. The source of the examples and benchmarks we present below is available in the supporting information [103].

6.3.1. Using Gaussian-based DFT to perform iterative TDDFT

The first example is a stand-alone TDDFT calculation for which the DFT calculation is done with Gaussian type of orbitals (GTO), within the PySCF package. To start the iterative TDDFT, we built an interface with a mean-field class of the PySCF package. The product basis we implement in the current version is not generally suitable for a FP calculation since it is thought mostly for PP-based calculations (see Section 5.6). However, it is accurate enough for hydrogen dimer and we have chosen this molecule. There is an original implementation of the Casida approach to TDDFT [104,3] in PySCF package. We use the PySCF's class `TDDFT` to setup and diagonalize the Casida matrix. An average dynamical polarizability $\sum_i P_{ii}(\omega)/3$ is then composed from the eigenvectors of the Casida matrix and the dipole matrix elements. Alternatively, we compute the polarizability using our implementation of the iterative TDDFT described above. The two calculations give very similar results as one can see in Fig. 4.

The Python script to organize these two calculations is shown in Fig. 5. A similar script is part of the testing suite `nao/test/test_0046*`. In the first line of the script, we import the PySCF's objects `gto`, `tddft` and `scf`. The hydrogen molecule with given coordinates of atoms and a correlation-consistent atomic orbital basis set by Dunning [105] is defined in the second line. The KS

```
cd pyscf/pyscf/lib
cp cmake_arch_config/cmake.arch.inc-nao-gnu cmake.arch.inc
mkdir build
cd build
cmake ..
make
```

Fig. 3. The sequence of commands to compile the low-level libraries for the NAO-enabled PySCF package.

```
1  from pyscf import gto, scf, tddft
2  mol = gto.M(atom='''H 0 0 0; H 0.2 0 0.6''', basis='cc-pvdz',)
3  gto_mf = scf.RKS(mol)
4  gto_mf.kernel()
5  gto_td = tddft.TDDFT(gto_mf)
6  gto_td.nstates = 9
7  gto_td.kernel()
8  from pyscf.nao import tddft_iter, polariz_inter_ave
9  nao_td  = tddft_iter(mf=gto_mf, gto=mol)
10 omegas = [1.0+1j*0.02, 1.5+1j*0.02]
11 p_iter = -nao_td.polariz_inter_ave(omegas).imag
12 p_ave = -polariz_inter_ave(gto_mf, mol, gto_td, omegas).imag
```

Fig. 5. Python script to compute the average dynamical polarizability via the Casida formulation and via the iterative TDDFT for a hydrogen dimer molecule.

```
SystemLabel          water
%block ChemicalSpeciesLabel
 1  8  O
 2  1  H
%endblock ChemicalSpeciesLabel
NumberOfAtoms              3
NumberOfSpecies            2
AtomicCoordinatesFormat  Ang
%block AtomicCoordinatesAndAtomicSpecies
    0.00  0.00  0.00   1       1  O
    0.78  0.59  0.00   2       2  H
   -0.78  0.59  0.00   2       3  H
%endblock AtomicCoordinatesAndAtomicSpecies
ElectronicTemperature      300 K
MD.TypeOfRun               CG
MD.NumCGsteps              0

coop.write      .true.
xml.write       .true.
```

Fig. 6. SIESTA input file to organize the self-consistent LDA calculation of a water molecule and storage of the KS Hamiltonian and orbitals.

calculation is defined in the 3rd line and performed in the 4th line, while the Casida calculation is done by line 7.

The class `tddft_iter` is imported and initialized with the eigenvectors from the preceding mean-field calculation in the lines 8 and 9. During construction of the class `tddft_iter`, the NAOs are initialized, the PB is constructed and the interaction kernel (18) is computed. The iterative procedure is done in the line 11, delivering the dynamical polarizability for a given list of (complex-valued) frequencies `omegas`, which must be given in the Hartree atomic units. In the last line 12, we compute the reference dynamical polarizability, taking advantage of the PySCF's Casida calculation (in lines 5–7).

6.3.2. Reading data from SIESTA and doing TDDFT

Currently, we are able to use the PP starting points provided by SIESTA [27,106,107] and GPAW [23] packages, while we are confident that an interfacing with OpenMX [29,108] and Fireball [26,109] packages is easy to achieve.

In this example, we use the data from a preceding SIESTA calculation, providing the shape of NAOs, the geometry of the molecule, the KS orbitals etc. The interface with SIESTA is done via files. Most of the files would be generated upon a normal self-consistent field loop, while the writing of KS Hamiltonian and orbitals is requested by corresponding flags in the SIESTA input file. To illustrate the interfacing with SIESTA package, we have chosen a water molecule. SIESTA input file is shown in

```
from pyscf.nao import tddft_iter
from numpy import linspace, savetxt, array

td  = tddft_iter(label='water', xc_code='RPA')
ww = linspace(0.0,1.0,150)+1j*0.02
p = -td.comp_polariz_inter_ave(ww).imag
savetxt('inter.ave.txt', array([ww.real*27.2114, p]).T)
```

Fig. 7. Python script for computing the average dipole polarizability in a random-phase approximation starting from a data exported during a preceding SIESTA run.

Fig. 6. The two last lines are crucial for the post-SIESTA, iterative TDDFT calculation. The option `coop.write` causes the export of Hamiltonian and the KS eigenstates to the files `water.HSX` and `water.fullBZ.WFSX`, correspondingly. The intended use of this data in the SIESTA suite is for a post analysis of a crystal orbital overlap population (COOP) [110] which explains the name of the option. The option `xml.write` causes an XML version of the output to be written, from which we extract such information as geometry of the molecule, Fermi energy, KS eigenvalues, unit cell vectors, k-points etc.

After the SIESTA calculation is completed, we can load the stored data and compute the optical polarizability, using the algorithm described in Section 4. The corresponding Python script is shown in Fig. 7. As one can see from the script, the class `tddft_iter` can take the *system label* as an argument `label` in order to load the data from the preceding SIESTA calculation. During the class construction, the PB set (19) is generated and the interaction kernel (18) is computed. The type of the interaction kernel could be set with the option `xc_code`. The option `xc_code` is to a large extent inherited from the PySCF package. However, we augmented the option by recognizing also a random-phase approximation (RPA) via this option. In effect of the option `xc_code='RPA'`, we drop the xc part of the interaction kernel (18) in the iterative TDDFT.

In the last three lines, a frequency grid is defined, the polarizability is computed at these frequencies and the imaginary part of the polarizability is stored in a text file for posterior analysis.

6.3.3. Reading data from ASE/GPAW and doing TDDFT

ASE and GPAW are a package bundle for manipulating molecular geometries and electronic structure simulations [111,23] written in Python. ASE is the abbreviation of *atomic simulation environment*. It serves to organize AIMD simulations among other tasks. GPAW is an *ab initio* calculator of the electronic structure and properties. By default GPAW uses a *projected augmented waves* framework for representing the core electron's degrees of freedom and RSGs to represent the spatial variables. However, the GPAW package can also use NAOs and the norm-conserving Vanderbilt PPs [112] in DFT and TDDFT. Interfacing our code with an ASE/GPAW calculation

```
from ase import Atoms
from gpaw import GPAW, PoissonSolver

H2O_gp = Atoms("H2O", [[0.0, -0.757, 0.587],
                       [0.0, +0.757, 0.587],
                       [0.0, 0.0, 0.0]])
H2O_gp.center(vacuum=3.5)
gpaw_calc = GPAW(xc="PBE", h=0.3, nbands=6,
    convergence={"density": 1e-7},
    poissonsolver=PoissonSolver(eps=1e-14, remove_moment=1+3),
    mode="lcao", setups="sg15", txt=None)

H2O_gp.set_calculator(gpaw_calc)
efree_gpaw = H2O_gp.get_potential_energy()
gpaw_calc.write('gpaw.bin', mode='all') # write DFT output
```

Fig. 8. Python script for computing the KS ground state with the norm-conserving Vanderbilt pseudo-potentials.

```
...
gpaw_calc = GPAW('gpaw.bin')
td  = tddft_iter(gpaw=gpaw_calc, tddft_iter_tol=1e-4)
omegas = linspace(0.0,1.0,150)+1j*0.02
p = -td.comp_polariz_inter_ave(omegas).imag
savetxt('inter.ave.txt', array([omegas.real*27.2114, p]).T)
p = -td.comp_polariz_nonin_ave(omegas).imag
savetxt('nonin.ave.txt', array([omegas.real*27.2114, p]).T)
print(td.rf0_ncalls)
```

Fig. 9. Python script for computing the average dipole polarizability from GPAW output. The interacting polarizability (5) using Perdew–Zunger LDA and the non-interacting polarizability (41) are calculated for comparison.

can be readily done in our TDDFT solver. The motivation for the interfacing with GPAW package is to access advanced DFT functionals, such as the Gritsenko–van Leeuwen–van Lenthe–Baerends [113] functional, which is absent in many of the other software packages.

In order to enable the Vanderbilt PPs, one has to download/enable a special set of GPAW "setups" named sg15. This can be done by running a command in terminal

```
gpaw install-data --sg15 /path/to/gpaw/setups/root
```

After installation of the sg15 setups, GPAW will be ready to use the norm-conserving PPs. Furthermore, to use the NAO basis sets, we have to specify an "lcao" mode as shown in the script in Fig. 8. In this script, we define a water molecule, run DFT calculation and save a GPAW object H2O_gp containing the calculation to the file gpaw.bin.

The next script in Fig. 9 would load the data, initialize the object tddft_iter and compute the polarizability (5) for a set of frequencies omegas, storing the result to files inter.ave.txt and nonin.ave.txt for the averaged interacting polarizability and non-interacting polarizability, correspondingly.

The non-interacting polarizability $P^0_{ij}(\omega)$ is defined analogously to the interacting polarizability (5), but using the non-interacting response function

$$P^0_{ij}(\omega) = \iint \boldsymbol{r}_i \chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) \boldsymbol{r}'_j d\boldsymbol{r} d\boldsymbol{r}'. \tag{41}$$

The non-interacting polarizability $P^0_{ij}(\omega)$ can serve as a quick-to-test quantity to control the quality of the PB set, and as a rough approximation to the true interacting polarizability $P_{ij}(\omega)$.

We skipped the import commands in Fig. 9. In this example, we demand a higher accuracy of the iterative procedure set by the tolerance tddft_iter_tol as an optional argument of the class constructor tddft_iter(...).

In the following two examples, we discuss simpler calculations which might be useful both for pure SIESTA users as well as for users doing TDDFT calculations. In Section 6.3.4, we discuss the calculation of density of states and projected density of states, using SIESTA output. Although SIESTA suite already contains the corresponding utilities, the user might find Python scripts more convenient. In Section 6.3.5, we show how to calculate the KS molecular orbital on a Cartesian grid and export this data into the commonly used Gaussian Cube format. The plotting of 3-dimensional isosurfaces of the molecular orbitals is a common practice during the analysis of the excited states calculations [74,9,21,76].

6.3.4. Computing DOS/PDOS with SIESTA output

In this example, we will compute the density of states (DoS) defined via the KS eigenvalues E_n and for a Lorentzian representation of the Dirac δ-function

$$\mathrm{DoS}(\omega) = -\frac{1}{\pi} \Im \sum_n \frac{1}{\omega - E_n + \mathrm{i}\varepsilon}. \tag{42}$$

A projected density of states (PDoS) is defined via the KS eigenvalues E_n, the KS eigenstate coefficients X^n_a and the atomic-orbital overlap S^{ab}

$$\mathrm{PDoS}(l, \omega) = -\frac{1}{\pi} \Im \sum_n \frac{X^n_a \delta_{l,l_a} S^{ab} X^n_b}{\omega - E_n + \mathrm{i}\varepsilon}. \tag{43}$$

In this example, the KS data will be imported from a preceding SIESTA calculation. Assuming one has done the SIESTA calculation with the FDF-file as in Fig. 6, we load this data and perform calculation of DoS and PDoS with a script shown in Fig. 10.

After the script is finished, the files dos.txt and pdos.txt are written. The file dos.txt will contain two columns: the real parts of the set of frequencies omegas and the corresponding DoS (42). The file pdos.txt will contain four columns: the real parts of the

```
from pyscf.nao import scf
from numpy import arange, savetxt, array, zeros

dft  = scf(label='water')
omegas = arange(-1.0,1.0,0.05)+1j*0.01
dos = dft.dos(omegas)
savetxt('dos.txt', array([omegas.real*27.2114, dos]).T)
pdos = dft.pdos(omegas)
data = zeros((pdos.shape[0]+1, pdos.shape[1]))
data[0,:] = omegas.real*27.2114
data[1:,:] = pdos
savetxt('pdos.txt', data.T)
```

Fig. 10. Python script computing the DoS and PDoS with the KS orbitals and eigenvalues imported from a preceding SIESTA calculation.

set of frequencies omegas, and corresponding PDoS (43) for each of the possible atomic-orbital angular momenta $l = 0, 1, 2$. Other types of molecular orbital overlap (Hamiltonian) population analysis is easy to achieve thanks to the flexibility of Python language.

6.3.5. SIESTA's molecular orbitals in Gaussian cube

Plotting of the KS orbitals to a Gaussian Cube format can be also easily accomplished with the imported data. Assuming one has done the SIESTA calculation with the FDF-file as in Fig. 6, we load this data and perform calculation of the highest-occupied molecular orbital (HOMO) with the script in Fig. 11. Additionally to the class scf, we used the class Cube from the pyscf.tools.cubegen module. When constructed, the class Cube can use any Python object with the methods atom_coords(), atom_charge() and the attribute natm defined. Optionally, the number of grid points along the Cartesian axes can be also specified during the construction of the class Cube. After initialization, the class Cube can provide the Cartesian coordinates $\{\boldsymbol{r}\}$ of the 3-dimensional grid via the method get_coords(). In turn, the class scf defines a method comp_aos_den() to compute the atomic orbitals $\{f^a(\boldsymbol{r})\}$ for a set of Cartesian coordinates $\{\boldsymbol{r}\}$ and returning these values in a dense matrix. The values of a molecular orbital $\psi_n(\boldsymbol{r}) = X_a^n f^a(\boldsymbol{r})$ are given by a matrix–vector product, for which we use the NumPy's procedure dot(...). Finally, the Gaussian Cube formatted file is written with the method write() of the class Cube. The Gaussian Cube files can be viewed with a variety of software including the XCrysDen package [114].

7. Benchmark

In the examples above, we illustrated the use of the PySCF-NAO package for doing a variety of calculations for small molecules. However, the main advantage of the iterative TDDFT is in its ability to cope with much larger molecules and clusters, including metallic clusters with many nearly-degenerated states in the KS spectrum. In this section, we describe two sets of larger TDDFT calculations: for silver clusters and for a set of geometries generated in an AIMD simulation.

7.1. Optical absorption by silver clusters

Silver is a metal widely used in plasmonic applications [115–117]. There were already several calculations for silver clusters within TDDFT [118,24,20,78]. In particular, it has been shown that simple DFT functionals describe reasonably well the localized surface plasmonic resonance characteristic to the optical response of silver clusters [118,18,24,20,78]. In this subsection, we perform TDDFT calculations for a series of icosahedral clusters containing between 3 and 8 layers of atoms, which corresponds to number of atoms from 55 to 1415. The geometries of these clusters have been generated with the ASE package, using a Python script shown in Fig. 12. The geometry relaxation at DFT level preserves the icosahedral symmetry and optical absorption spectra to a large extent [78] although claiming a substantial computational effort.

Additionally to the electron-containing atomic layers in the icosahedral clusters, we placed a layer of ghost atoms around each of the clusters [73,78]. The ghost atoms provide basis functions to better describe the decay of the electron cloud at the surface.

The generated geometries are stored to fdf-files in a Z-matrix format. The procedure write_zmat() generates these files. It is omitted in Fig. 12, but is defined in the file ico_geoms.py provided in the supporting information [103].

To produce the DFT starting point, we used SIESTA calculations defined by the FDF-file shown in Fig. 13. The basis of pseudo-atomic orbitals (PAO) shall be defined via a PAO.Basis block, because we intent to include ghost atoms. We use the double-zeta polarized basis set with soft-confinement by Junquera et al. [119]. Because the confinement radii are omitted in the FDF-file (set to zero in the PAO.Basis block), they are determined by the energy shift parameter PAO.EnergyShift. The initial geometry is defined in the included file "geometry.siesta.fdf". A conjugated-gradients (CG) algorithm is used for geometry relaxation. The GGA functional by Wu and Cohen [46] is used. We have chosen this functional because it has been shown to provide the most accurate structural parameters among the available semi-local functionals [120,121]. The rest of the parameters are chosen to achieve a stable convergence of the KS self-consistent procedure. Their meaning is described in the SIESTA manual [107].

After the DFT calculation is finished, the iterative TDDFT calculation is started with the generated data very much similar to the

```
from numpy import dot
from pyscf.nao import scf
from pyscf.tools.cubegen import Cube

dft  = scf(label='water')
cc = Cube(dft, nx=40, ny=40, nz=40)
co2val = dft.comp_aos_den(cc.get_coords())

mol_orb = int(dft.nelectron / 2)-1
print('mol_orb ', mol_orb, dft.nelectron/2)

mo = dft.mo_coeff[0,0,mol_orb,:,0]
c2orb =  dot(co2val, mo).reshape((cc.nx, cc.ny, cc.nz))
cc.write(c2orb, "water.homo.cube", comment="mol_orb=%i"%(mol_orb))
```

Fig. 11. Python script to compute the real-space representation of the HOMO orbital with the data imported from a preceding SIESTA calculation and generating the Gaussian Cube file.

```
from ase.cluster.icosahedron import Icosahedron
for nsh in range(3,9):
  cl = Icosahedron("Ag", nsh, latticeconstant=4.0)
  cl.positions = cl.positions - cl.get_center_of_mass()
  syms = cl.get_chemical_symbols()
  for i,t in enumerate(cl.get_tags()):
    if t>nsh-1: syms[i] = "X"
  cl.set_chemical_symbols(syms)
  write_zmat(cl, "Ag", fname='geometry.siesta_nsh{0}'.format(nsh-1))
```

Fig. 12. Python script to determine a series of geometries for icosahedral silver clusters.

```
%block PAO.Basis
  Ag    2
  n=5  0  2  P  E 50.0 7.0
    0.000  0.000
  n=4  2  2     E 50.0 4.0
    0.000  0.000
  X  1
  n=5  0  1  P  E 50.0 7.0
    0.000
%endblock PAO.Basis
PAO.EnergyShift      100 meV

%include "geometry.siesta.fdf"

MD.TypeOfRun         CG
MD.NumCGsteps        400
XC.functional        GGA
XC.authors           WC

MeshCutOff           250 Ry

DM.MixingWeight      0.01
DM.Tolerance         1e-4
DM.NumberKick        40
DM.NumberPulay       4
DM.KickMixingWeight  0.15
MaxSCFIterations     1500

SolutionMethod       Diagon
COOP.Write           .True.
XML.Write            .True.
```

Fig. 13. SIESTA input file for silver clusters.

example in Section 6.3.2. In contrast to Section 6.3.2, we compute only the *xx*-component of the polarizability (5). Moreover, we use the lowest-possible `level` of accuracy for the Lebedev–Laikov grid to generate the xc kernel (18) and also less strict tolerances in the generation of the intra-atomic (local) `tol_loc` and inter-atomic (bilocal) `tol_biloc` dominant functions $F^{\mu}(\boldsymbol{r})$.

```
td = tddft_iter(label="siesta", level=0,
tol_loc=1e-4, tol_biloc=1e-6)
...
pxx = td.comp_polariz_inter_xx(omegas)
...
```

The relaxed tolerances lead to converged results nevertheless because the PP NAOs are rather smooth functions. On the other hand, the less strict tolerances speedup computations.

The polarizability $P_{xx}(\omega)$ is equal to the direction-averaged polarizability $\sum_i P_{ii}(\omega)/3$ in the case of icosahedral cluster geometries. Therefore, doing the calculation only for *xx*-component saves about 2/3 of the run time in the iterative procedure. The iterative procedure is done for a set of 100 frequencies between 1.0 and 6.0 eV, with an imaginary part equal to 0.15 eV. The interacting polarizabilities are collected in Fig. 14, together with the frequency of the surface-plasmon maximum ω_{sp} as a function of the number of atoms N in the clusters. Inspecting the polarizability in Fig. 14, panel a, we see pronounced plasmonic peaks. The frequency of this resonance peak is slowly changing with the cluster size, giving rise to a $1/D$ scaling of the plasmonic frequency with the cluster diameter D. In Fig. 14, panel b, we plot the energy of the resonance ω_{sp} as a function of $N^{-1/3}$, because this is better defined than the diameter D for these non-spherical clusters. The plasmonic frequencies compare rather accurately to the other calculations. For instance, for the smallest cluster Ag_{55}, we get $\omega_{sp} = 3.7$ eV, while Idrobo and Pantelides got 3.5 eV using LDA functional [122]. For the next-size cluster Ag_{147}, we get $\omega_{sp} = 3.5$ eV, while Lozano et al. [123] found 3.2 eV with the Perdew–Burke–Ernzerhof (PBE) functional [45]. More detailed characterization of the method accuracy for the plasmonic response of silver clusters is available elsewhere [78].

In Fig. 15, we show the run time needed to compute the polarizability on 12 cores of Intel® Xeon® Processor E5-2680 v3. As it can be clearly appreciated, the actual run-time scaling is closer to $T \sim N^2$ for the cluster sizes we consider. The number of iterations per frequency grows very slowly with the size of the cluster, varying from approximately 12 iterations per frequency for Ag_{55} to approximately 20 iterations per frequency for Ag_{1415}.

7.2. AIMD configuration average of the dynamical polarizability

In the final benchmark, we estimate the effect of the temperature fluctuations on the optical polarizability of a photo-chromic diarylethene-derivative depicted in Fig. 16. The compound corresponds to the molecular derivative III in Ref. [124] (Figure 2a) in the "closed" configuration. We assume a complete decoupling of electronic and nuclear degrees of freedom, treating the latter within the framework of Newtonian mechanics with the forces computed from ground state DFT with PBE functional [45]. We record the molecular geometries visited along the AIMD simulation. These geometries are used in subsequent calculations of the optical polarizability (5) to find the so-called configuration average of the averaged polarizability $\sum_i P_{ii}(\omega)/3$.

AIMD is done with `SIESTA` package, using a Nosé thermostat [107,125]. The equilibrium geometry, from which the AIMD propagation is started, is determined with a conjugated gradients method with a tolerance to the remaining force of 0.02 eV/Å. After the geometry relaxation, we computed the phonon modes with the `VIBRA` utility of the `SIESTA` package. The phonon calculation gave us minimal ($18.13\ \mathrm{cm}^{-1}$) and maximal ($3138.12\ \mathrm{cm}^{-1}$) frequencies of vibrations. These frequencies translate to periods 1840 and 10.63 fs, correspondingly. These periods are used to choose the time step and total time for AIMD runs. We performed two AIMD simulations for the target temperatures $T_0 = 100$ and 300 K. In

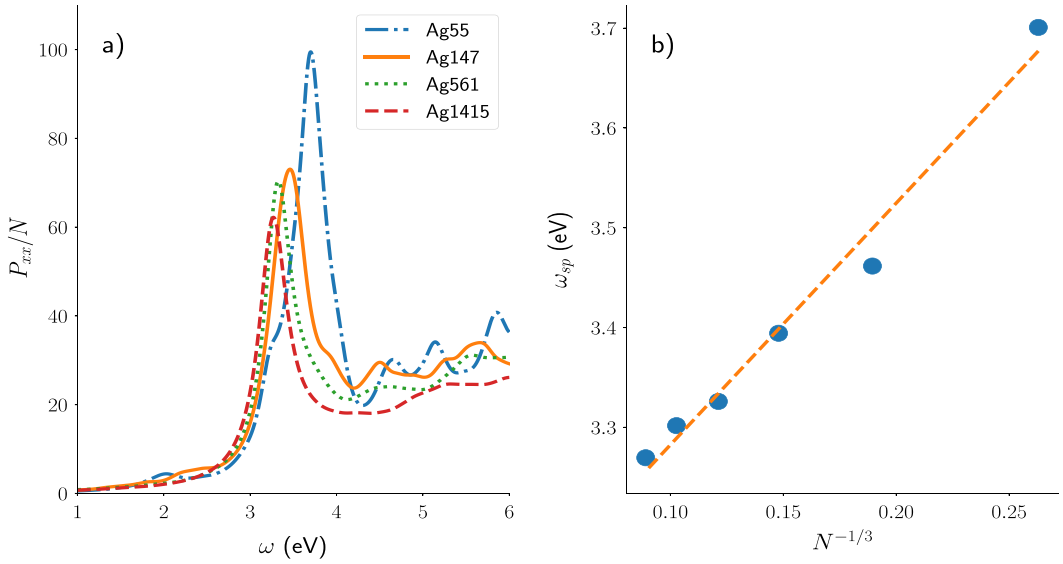

Fig. 14. Interacting polarizability for a set of silver clusters of icosahedral shape (panel a) and the dependence of the frequency of the first maximum depending on the number of atoms in the cluster (panel b).

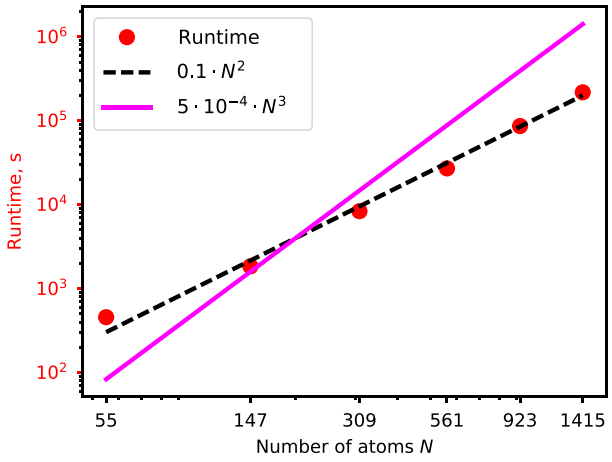

Fig. 15. Total run time to compute the dynamical polarizability of the compact silver clusters (see Fig. 14).

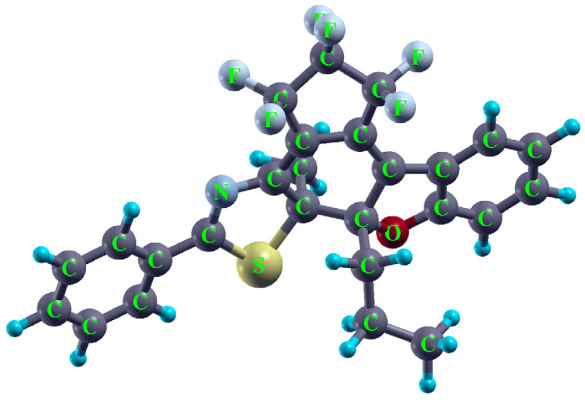

Fig. 16. Ball-stick model of the diarylethene derivative studied in Section 7.2.

both AIMD runs, we used a time step of 1 fs and propagated for 20 000 steps. The excessive AIMD steps were used for a thermalization period of 8000 and 12 000 fs for target temperatures 100 and 300 K, correspondingly. The average of the instantaneous temperature $\langle T \rangle$ was close to the thermostat temperature T_0 for both target temperatures. Moreover, fluctuations of the instantaneous temperature $\langle (\Delta T)^2 \rangle$ after thermalization periods were approximately T_0^2/C_v, $C_v = 3/2\,N$, for both thermostat temperatures.

After the AIMD simulation is finished, `SIESTA` produces an ANI file `system_label.ANI`, containing the generated molecular geometries for all the AIMD steps. This file consists of XYZ files stacked one after another. We split this file with a shell command `split`

```
split -l 56 -d -a 6 siesta.ANI
```

generating a series of XYZ files named `xNNNNNN`. In the next step, we use a subset of these files to organize another set of `SIESTA` calculations followed by the (iterative) TDDFT calculations. In the configurational average of the optical polarizability we have to choose a subset of the 20 000 configurations generated in AIMD because TDDFT is much more computationally expensive than DFT. Therefore, we have chosen a larger time step of 5 fs (which is still twice smaller than the period of shortest vibrations) and averaged over 800 configurations (which gives rise to a total average time $800 \cdot 5 = 4000$ fs that is more than twice larger than the longest vibration period). In the iterative TDDFT, we use a small broadening constant $\varepsilon = 2 \cdot 10^{-3}$ eV, resulting essentially in a stick spectrum of the polarizability's $P(\omega)$ imaginary part. Finally, we compute the configuration average of the polarizability $P_{\text{ca}}(\omega)$ with yet another Lorentzian broadening $\varepsilon_{\text{ca}} = 0.0125$ eV

$$P_{\text{ca}}(\omega) = \frac{\varepsilon_{\text{ca}}}{\pi N_{\text{c}}} \sum_{\text{c}} \frac{\text{Im}P_{\text{c}}(\omega')}{(\omega - \omega')^2 + \varepsilon_{\text{ca}}^2}. \tag{44}$$

The broadening ε_{ca} is chosen so as to simulate a finite spectral resolution of the detector, which is sensitive enough not to smear the features of the configuration average.

In Fig. 17, we show the configuration average of the dynamical polarizability (44) for the thermostat temperatures $T_0 = 100$ K and $T_0 = 300$ K together with the spectrum obtained at the equilibrium geometry (labeled $T_0 = 0$ K). The polarizability at equilibrium geometry was computed with the Lorentzian broadening $\varepsilon_{\text{ca}} = 0.0125$ eV (i.e. the same as for the configurational averaged spectra) and yet we scaled down the polarizability at equilibrium geometry by a factor of four to approximately match its magnitude to the thermally averaged spectra.

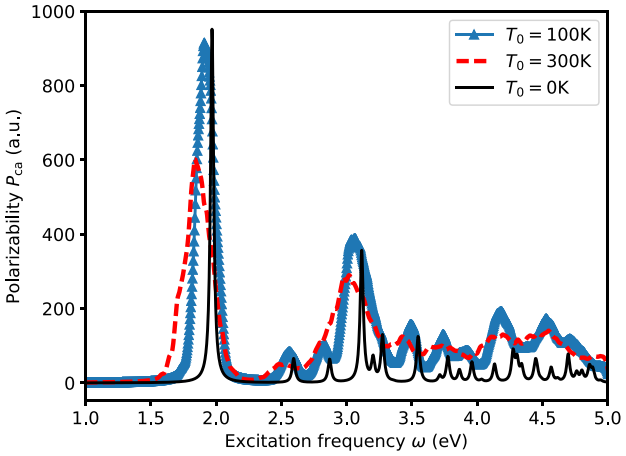

Fig. 17. Configuration-average polarizability $P_{ca}(\omega)$ of the diarylethene-derivative depicted in Fig. 16 at different thermostat temperatures. The average is done with a Nosé thermostat AIMD as implemented in the SIESTA package. The optical spectrum at the equilibrium geometry is shown for comparison. Further details are in the text.

The figure shows that the motion of nuclei dramatically affects the dynamical polarizability of the molecule. The width of the peaks increases with the ionic temperature T_0. For instance, for the first resonance given by HOMO–LUMO transition, the full-width at half maximum is 0.16 and 0.28 eV for 100 and 300 K, correspondingly. Moreover, the position of the first maximum is noticeably red-shifted by the temperature fluctuations (1.97, 1.91 and 1.84 eV for 0, 100 and 300 K, correspondingly). A cross-check calculation using all-electron GTO basis (631G) as implemented in PySCF gives the first excitation energy at 2.03 eV (for equilibrium geometry obtained by SIESTA and using PBE GGA for the xc density functional) and 2.33 eV (using a B3LYP functional [126]).

The iterative TDDFT used on average 15 iterations per frequency and per Cartesian component. The calculation took on average 1.61 s per frequency on an Intel machine Xeon(R) E5-2680 v3 2.50 GHz, using 6 computing cores and less than 2 GB of RAM.

8. Conclusion and outlook

We presented an iterative method for linear response TDDFT and a Python implementation of the method. The method is suitable for DFT starting points and LDA/GGA functionals. The implementation is done as a part of the PySCF package utilizing some of its features. The implementation is interfaced with the SIESTA package, while it can be readily used with GPAW and for light molecules with full-potential calculations performed by PySCF. The dipole polarizability tensor is computed by our code using an efficient iterative algorithm.

We are currently working on the extension of our code to estimate the electronic energy loss spectra [127]. The iterative TDDFT can be readily used to compute non-resonant Raman spectra by finite-differences of the dipole polarizability [128]. Moreover, many parts of the implementation can be used in more complex electronic-structure methods such as time-dependent Hartree–Fock, TDDFT with hybrid functionals and *GW*/BSE solvers. The corresponding proof-of-principles implementations are already included in the NAO-enabled PySCF package and will be a subject of future work.

Acknowledgments

We thank referees for providing detailed constructive comments, leading to significant improvements of the manuscript. PK and DSP acknowledge support from Spanish MINECO Grants MAT2016-78293-C6-4-R and RTC-2016-5681-7 (SIESTA-PRO). PK acknowledges financial support from the Fellows Gipuzkoa program of the Gipuzkoako Foru Aldundia through the FEDER funding scheme of the European Union. MB acknowledges support from the Departemento de Educación of the Basque Government through a PhD grant, as well as from Euskampus and the DIPC at the initial stages of this work.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cpc.2018.08.004.

References

[1] M. Petersilka, U. Gossmann, E. Gross, Phys. Rev. Lett. 76 (8) (1996) 1212–1215, http://dx.doi.org/10.1103/PhysRevLett.76.1212. URL http://link.aps.org/doi/10.1103/PhysRevLett.76.1212.

[2] W. Kohn, Rev. Modern Phys. 71 (1999) 1253–1266, http://dx.doi.org/10.1103/RevModPhys.71.1253. URL https://link.aps.org/doi/10.1103/RevModPhys.71.1253.

[3] M.E. Casida, J. Mol. Struct. THEOCHEM 914 (1) (2009) 3–18, http://dx.doi.org/10.1016/j.theochem.2009.08.018. Time-dependent density-functional theory for molecules and molecular solids. URL http://www.sciencedirect.com/science/article/pii/S0166128009005363.

[4] X. Gonze, B. Amadon, P.-M. Anglade, J.-M. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, P. Ghosez, M. Giantomassi, S. Goedecker, D. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G.-M. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M. Verstraete, G. Zerah, J. Zwanziger, Comput. Phys. Comm. 180 (12) (2009) 2582–2615, http://dx.doi.org/10.1016/j.cpc.2009.07.007. URL http://linkinghub.elsevier.com/retrieve/pii/S0010465509002276.

[5] A. Marini, C. Hogan, M. Grüning, D. Varsano, Comput. Phys. Comm. 180 (8) (2009) 1392–1403, http://dx.doi.org/10.1016/j.cpc.2009.02.003.

[6] I. Timrov, N. Vast, R. Gebauer, S. Baroni, Comput. Phys. Comm. 196 (Supplement C) (2015) 460–469, http://dx.doi.org/10.1016/j.cpc.2015.05.021. URL http://www.sciencedirect.com/science/article/pii/S0010465515002015.

[7] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M.B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A.D. Corso, S. de Gironcoli, P. Delugas, R.A.D. Jr, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N.L. Nguyen, H.V. Nguyen, A. Otero-de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A.P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, S. Baroni, J. Phys.: Condens. Matter 29 (46) (2017) 465901. URL http://stacks.iop.org/0953-8984/29/i=46/a=465901.

[8] G. Bussetti, M. Campione, L. Ferraro, L. Raimondo, B. Bonanni, C. Goletti, M. Palummo, C. Hogan, L. Duò, M. Finazzi, A. Sassella, J. Phys. Chem. C 118 (29) (2014) 15649–15655, http://dx.doi.org/10.1021/jp501594d.

[9] P. Umari, L. Giacomazzi, F. De Angelis, M. Pastore, S. Baroni, J. Chem. Phys. 139 (1) (2013) 014709, http://dx.doi.org/10.1063/1.4809994. URL http://aip.scitation.org/doi/10.1063/1.4809994.

[10] P. Cudazzo, M. Gatti, A. Rubio, Phys. Rev. B 86 (2012) 195307, http://dx.doi.org/10.1103/PhysRevB.86.195307. URL https://link.aps.org/doi/10.1103/PhysRevB.86.195307.

[11] D. Lu, Y. Li, D. Rocca, G. Galli, Phys. Rev. Lett. 102 (2009) 206411, http://dx.doi.org/10.1103/PhysRevLett.102.206411. URL https://link.aps.org/doi/10.1103/PhysRevLett.102.206411.

[12] J. Frenzel, S. Gemming, G. Seifert, Phys. Rev. B 70 (2004) 235404, http://dx.doi.org/10.1103/PhysRevB.70.235404. URL https://link.aps.org/doi/10.1103/PhysRevB.70.235404.

[13] S. Xiong, K. Yang, Y.A. Kosevich, Y. Chalopin, R. D'Agosta, P. Cortona, S. Volz, Phys. Rev. Lett. 112 (2014) 114301, http://dx.doi.org/10.1103/PhysRevLett.112.114301. URL https://link.aps.org/doi/10.1103/PhysRevLett.112.114301.

[14] A.R. Botello-Méndez, F. López-Urías, M. Terrones, H. Terrones, Nano Res. 1 (5) (2008) 420–426, http://dx.doi.org/10.1007/s12274-008-8042-3.

[15] C. Mauney, M.B. Nardelli, D. Lazzati, Astrophys. J. 800 (1) (2015) 30. URL http://stacks.iop.org/0004-637X/800/i=1/a=30.

[16] X. Andrade, D. Strubbe, U. De Giovannini, A.H. Larsen, M.J.T. Oliveira, J. Alberdi-Rodriguez, A. Varas, I. Theophilou, N. Helbig, M.J. Verstraete, L. Stella, F. Nogueira, A. Aspuru-Guzik, A. Castro, M.A.L. Marques, A. Rubio, Phys. Chem. Chem. Phys. 17 (2015) 31371–31396, http://dx.doi.org/10.1039/C5CP00351B.

[17] A. Castro, M.A.L. Marques, D. Varsano, F. Sottile, A. Rubio, C.R. Phys. 10 (6) (2009) 469–490, http://dx.doi.org/10.1016/j.crhy.2008.09.001. URL http://www.sciencedirect.com/science/article/pii/S1631070508001266.

[18] X. Lopez-Lozano, H. Barron, C. Mottet, H.-C. Weissker, Phys. Chem. Chem. Phys. 16 (2014) 1820–1823, http://dx.doi.org/10.1039/C3CP53702A.

[19] A. Varas, P. García-González, F.J. García-Vidal, A. Rubio, J. Phys. Chem. Lett. 6 (10) (2015) 1891–1898, http://dx.doi.org/10.1021/acs.jpclett.5b00573. pMID: 26263265.

[20] O. Baseggio, M. De Vetta, G. Fronzoni, M. Stener, L. Sementa, A. Fortunelli, A. Calzolari, J. Phys. Chem. C 120 (23) (2016) 12773–12782, http://dx.doi.org/10.1021/acs.jpcc.6b04709.

[21] N.C. Forero-Martinez, H.-L. Thi Le, N. Ning, H. Vach, H.C. Weissker, Nanoscale 7 (2015) 4942–4948, http://dx.doi.org/10.1039/C4NR04905E.

[22] A.H. Larsen, M. Vanin, J.J. Mortensen, K.S. Thygesen, K.W. Jacobsen, Phys. Rev. B 80 (19) (2009) 195112, http://dx.doi.org/10.1103/PhysRevB.80.195112.

[23] M. Walter, H. Häkkinen, L. Lehtovaara, M. Puska, J. Enkovaara, C. Rostgaard, J.J. Mortensen, J. Chem. Phys. 128 (24) (2008) 244101, http://dx.doi.org/10.1063/1.2943138.

[24] M. Kuisma, A. Sakko, T.P. Rossi, A.H. Larsen, J. Enkovaara, L. Lehtovaara, T.T. Rantala, Phys. Rev. B 91 (11) (2015) 115431, http://dx.doi.org/10.1103/PhysRevB.91.115431. URL http://link.aps.org/doi/10.1103/PhysRevB.91.115431.

[25] T.P. Rossi, A. Zugarramurdi, M.J. Puska, R.M. Nieminen, Phys. Rev. Lett. 115 (2015) 236804, http://dx.doi.org/10.1103/PhysRevLett.115.236804. URL https://link.aps.org/doi/10.1103/PhysRevLett.115.236804.

[26] J.I. Mendieta-Moreno, R.C. Walker, J.P. Lewis, P. Gómez-Puertas, J. Mendieta, J. Ortega, J. Chem. Theory Comput. 10 (5) (2014) 2185–2193, http://dx.doi.org/10.1021/ct500033w. pMID: 26580543.

[27] J.M. Soler, E. Artacho, J.D. Gale, A. García, J. Junquera, P. Ordejón, D. Sánchez-Portal, J. Phys.: Condens. Matter 14 (11) (2002) 2745 URhttp://stacks.iop.org/0953-8984/14/i=11/a=302.

[28] R. Sánchez-de Armas, J. Oviedo López, M.A. San-Miguel, J.F. Sanz, P. Ordejón, M. Pruneda, J. Chem. Theory Comput. 6 (9) (2010) 2856–2865, http://dx.doi.org/10.1021/ct100289t. pMID: 26616086.

[29] T. Ozaki, H. Kino, Phys. Rev. B 72 (2005) 045121, http://dx.doi.org/10.1103/PhysRevB.72.045121. URL https://link.aps.org/doi/10.1103/PhysRevB.72.045121.

[30] S. Kenny, A. Horsfield, Comput. Phys. Comm. 180 (12) (2009) 2616–2621, http://dx.doi.org/10.1016/j.cpc.2009.08.006. 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures. URL http://www.sciencedirect.com/science/article/pii/S0010465509002562.

[31] B. Aradi, B. Hourahine, T. Frauenheim, J. Phys. Chem. A 111 (26) (2007) 5678–5684, http://dx.doi.org/10.1021/jp070186p.

[32] P. Koval, D. Foerster, O. Coulaud, J. Chem. Theory Comput. 6 (9) (2010) 2654–2668, http://dx.doi.org/10.1021/ct100280x.

[33] O. Coulaud, P. Bordat, P. Fayon, V. Le Bris, I. Baraille, R. Brown, Extensions of the Siesta DFT Code for Simulation of Molecules, Research Report RR-8221, INRIA, 2013, p. 25. https://hal.inria.fr/hal-00787088.

[34] M. Barbry, F. Marchesin, M. Per Ljungberg, P. Koval, D. Foerster, D. Sánchez-Portal, MBPT-LCAO code, http://mbpt-domiprod.wikidot.com.

[35] M. Ohfuchi, Y. Miyamoto, Carbon 114 (2017) 418–423, http://dx.doi.org/10.1016/j.carbon.2016.12.052. URL http://www.sciencedirect.com/science/article/pii/S0008622316311319.

[36] M. Boleininger, A.P. Horsfield, J. Chem. Phys. 147 (4) (2017) 044111, http://dx.doi.org/10.1063/1.4995611.

[37] A. Domnguez, B. Aradi, T. Frauenheim, V. Lutsker, T.A. Niehaus, J. Chem. Theory Comput. 9 (11) (2013) 4901–4914, http://dx.doi.org/10.1021/ct400123t. pMID: 26583409.

[38] D. Foerster, P. Koval, D. Sánchez-Portal, J. Chem. Phys. 135 (7) (2011) 074105, http://dx.doi.org/10.1063/1.3624731.

[39] A.A.M.H.M. Darghouth, M.E. Casida, W. Taouali, K. Alimi, M.P. Ljungberg, P. Koval, D. Sánchez-Portal, D. Foerster, Computation 3 (4) (2015) 616–656, http://dx.doi.org/10.3390/computation3040616. URL http://www.mdpi.com/2079-3197/3/4/616.

[40] W.-C. Lu, C.Z. Wang, L.-Z. Zhao, W. Qin, K.M. Ho, Phys. Rev. B 92 (2015) 035206, http://dx.doi.org/10.1103/PhysRevB.92.035206. URL https://link.aps.org/doi/10.1103/PhysRevB.92.035206.

[41] D. Jacob, J. Phys.: Condens. Matter 27 (24) (2015) 245606. URL http://stacks.iop.org/0953-8984/27/i=24/a=245606.

[42] P. Koval, M.P. Ljungberg, D. Foerster, D. Sánchez-Portal, Nucl. Instrum. Methods Phys. Res. B 354 (2015) 216–219, http://dx.doi.org/10.1016/j.nimb.2014.11.080. 26th International Conference on Atomic Collisions in Solids. URL http://www.sciencedirect.com/science/article/pii/S0168583X14009811.

[43] F. Nogueira, M.A.L. Marques, C. Fiolhais, A Primer in Density Functional Theory, in: Lecture Notes in Physics, Springer, Berlin, 2003. URL https://cds.cern.ch/record/1391332.

[44] J.P. Perdew, A. Zunger, Phys. Rev. B 23 (1981) 5048–5079, http://dx.doi.org/10.1103/PhysRevB.23.5048. URL https://link.aps.org/doi/10.1103/PhysRevB.23.5048.

[45] J.P. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (1996) 3865–3868, http://dx.doi.org/10.1103/PhysRevLett.77.3865. URL https://link.aps.org/doi/10.1103/PhysRevLett.77.3865.

[46] Z. Wu, R.E. Cohen, Phys. Rev. B 73 (2006) 235116, http://dx.doi.org/10.1103/PhysRevB.73.235116. URL http://link.aps.org/doi/10.1103/PhysRevB.73.235116.

[47] A.D. Becke, J. Chem. Phys. 98 (2) (1993) 1372–1377, http://dx.doi.org/10.1063/1.464304.

[48] L. Kronik, T. Stein, S. Refaely-Abramson, R. Baer, J. Chem. Theory Comput. 8 (5) (2012) 1515–1531, http://dx.doi.org/10.1021/ct2009363.

[49] S. Rigamonti, S. Botti, V. Veniard, C. Draxl, L. Reining, F. Sottile, Phys. Rev. Lett. 114 (2015) 146402, http://dx.doi.org/10.1103/PhysRevLett.114.146402. URL https://link.aps.org/doi/10.1103/PhysRevLett.114.146402.

[50] V. Turkowski, N.U. Din, T.S. Rahman, Computation 5 (3) (2017).

[51] P. Koval, D. Foerster, D. Sánchez-Portal, Phys. Rev. B 89 (2014) 155417, http://dx.doi.org/10.1103/PhysRevB.89.155417. URL http://link.aps.org/doi/10.1103/PhysRevB.89.155417.

[52] M.P. Ljungberg, P. Koval, F. Ferrari, D. Foerster, D. Sánchez-Portal, Phys. Rev. B 92 (2015) 075422, http://dx.doi.org/10.1103/PhysRevB.92.075422. URL https://link.aps.org/doi/10.1103/PhysRevB.92.075422.

[53] M.A.L. Marques, E.K.U. Gross, in: C. Fiolhais, F. Nogueira, M.A.L. Marques (Eds.), A Primer in Density Functional Theory, Springer, Berlin, Heidelberg, 2003, pp. 144–184.

[54] M.A. Blanco, M. Flórez, M. Bermejo, J. Mol. Struct. THEOCHEM 419 (1) (1997) 19–27, http://dx.doi.org/10.1016/S0166-1280(97)00185-1. URL http://www.sciencedirect.com/science/article/pii/S0166128097001851.

[55] D.A. Varshalovich, A.N. Moskalev, V.K. Khersonskii, Quantum Theory of Angular Momentum, World Scientific, Singapore, 1988.

[56] J.D. Talman, Comput. Phys. Comm. 30 (1) (1983) 93–99, http://dx.doi.org/10.1016/0010-4655(83)90126-1. URL http://www.sciencedirect.com/science/article/pii/0010465583901261.

[57] J.D. Talman, J. Chem. Phys. 80 (5) (1984) 2000–2008, http://dx.doi.org/10.1063/1.446963.

[58] J.D. Talman, Phys. Rev. Lett. 84 (2000) 855–858, http://dx.doi.org/10.1103/PhysRevLett.84.855. URL https://link.aps.org/doi/10.1103/PhysRevLett.84.855.

[59] J.D. Talman, Comput. Phys. Comm. 180 (2) (2009) 332–338, http://dx.doi.org/10.1016/j.cpc.2008.10.003. URL http://www.sciencedirect.com/science/article/pii/S0010465508003329.

[60] J.D. Talman, Int. J. Quantum Chem. 107 (2007) 1578–1584, http://dx.doi.org/10.1002/qua.

[61] D. Foerster, J. Chem. Phys. (2006) 1–4. URL http://arxiv.org/abs/physics/0612187.

[62] O. Treutler, R. Ahlrichs, J. Chem. Phys. 102 (1) (1995) 346–354, http://dx.doi.org/10.1063/1.469408.

[63] R. Stratmann, G.E. Scuseria, M.J. Frisch, Chem. Phys. Lett. 257 (3) (1996) 213–223, http://dx.doi.org/10.1016/0009-2614(96)00600-8. URL http://www.sciencedirect.com/science/article/pii/0009261496006008.

[64] M.E. Mura, P.J. Knowles, J. Chem. Phys. 104 (24) (1996) 9848–9858, http://dx.doi.org/10.1063/1.471749.

[65] V.I. Lebedev, D.N. Laikov, Dokl. Math. 59 (3) (1999) 477–481.

[66] B.I. Dunlap, J.W.D. Connolly, J.R. Sabin, J. Chem. Phys. 71 (12) (1979) 4993–4999, http://dx.doi.org/10.1063/1.438313. URL http://aip.scitation.org/doi/abs/10.1063/1.438313.

[67] C. Fonseca Guerra, J.G. Snijders, G. te Velde, E.J. Baerends, Theor. Chem. Acc. 99 (6) (1998) 391–403, http://dx.doi.org/10.1007/s002140050353.

[68] X. Blase, C. Attaccalite, V. Olevano, Phys. Rev. B 83 (2011) 115103, http://dx.doi.org/10.1103/PhysRevB.83.115103. URL https://link.aps.org/doi/10.1103/PhysRevB.83.115103.

[69] D. Foerster, J. Chem. Phys. 128 (3) (2008) 034108, http://dx.doi.org/10.1063/1.2821021.

[70] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, M. Scheffler, Comput. Phys. Comm. 180 (11) (2009) 2175–2196, http://dx.doi.org/10.1016/j.cpc.2009.06.022. URL http://www.sciencedirect.com/science/article/pii/S0010465509002033.

[71] J. Junquera, O. Paz, D. Sánchez-Portal, E. Artacho, Phys. Rev. B 64 (2001) 235111, http://dx.doi.org/10.1103/PhysRevB.64.235111. URL http://link.aps.org/doi/10.1103/PhysRevB.64.235111.

[72] E. Anglada, J.M. Soler, J. Junquera, E. Artacho, Phys. Rev. B 66 (2002) 205101, http://dx.doi.org/10.1103/PhysRevB.66.205101. URL http://link.aps.org/doi/10.1103/PhysRevB.66.205101.

[73] S. García-Gil, A. García, N. Lorente, P. Ordejón, Phys. Rev. B 79 (2009) 075441, http://dx.doi.org/10.1103/PhysRevB.79.075441. URL https://link.aps.org/doi/10.1103/PhysRevB.79.075441.

[74] C. Faber, I. Duchemin, T. Deutsch, C. Attaccalite, V. Olevano, X. Blase, J. Mater. Sci. 47 (21) (2012) 7472–7481, http://dx.doi.org/10.1007/s10853-012-6401-7.

[75] M.J. Louwerse, G. Rothenberg, Phys. Rev. B 85 (2012) 035108, http://dx.doi.org/10.1103/PhysRevB.85.035108. URL https://link.aps.org/doi/10.1103/PhysRevB.85.035108.

[76] A. Calborean, L. Buimaga-Iarinca, F. Graur, Phys. Scr. 90 (5) (2015) 55803 URL http://stacks.iop.org/1402-4896/90/i=5/a=055803.

[77] T.P. Rossi, S. Lehtola, A. Sakko, M.J. Puska, R.M. Nieminen, J. Chem. Phys. 142 (9) (2015) 94114, http://dx.doi.org/10.1063/1.4913739.

[78] P. Koval, F. Marchesin, D. Foerster, D. Sánchez-Portal, J. Phys.: Condens. Matter 28 (21) (2016) 214001, http://dx.doi.org/10.1088/0953-8984/28/21/214001. URL http://stacks.iop.org/0953-8984/28/i=21/a=214001.

[79] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[80] V. Frayssé, L. Giraud, S. Gratton, J. Langou, ACM Trans. Math. Software 31 (2) (2005) 228–238, http://dx.doi.org/10.1145/1067967.1067970. URL http://doi.acm.org/10.1145/1067967.1067970.

[81] A. Manjavacas, F. Marchesin, S. Thongrattanasiri, P. Koval, P. Nordlander, D. Sánchez-Portal, F.J. García De Abajo, ACS Nano 7 (4) (2013) 3635–3643, http://dx.doi.org/10.1021/nn4006297.

[82] M. Barbry, P. Koval, F. Marchesin, R. Esteban, A.G. Borisov, J. Aizpurua, D. Sánchez-Portal, Nano Lett. 15 (5) (2015) 3410–3419, http://dx.doi.org/10.1021/acs.nanolett.5b00759. URL http://pubs.acs.org/doi/abs/10.1021/acs.nanolett.5b00759.

[83] F. Marchesin, P. Koval, M. Barbry, J. Aizpurua, D. Sánchez-Portal, ACS Photonics 3 (2) (2016) 269–277, http://dx.doi.org/10.1021/acsphotonics.5b00609. URL http://pubs.acs.org/doi/abs/10.1021/acsphotonics.5b00609.

[84] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001, [Online; accessed 21.05.2018]. URL http://www.scipy.org/.

[85] C. Van Caillie, R.D. Amos, Chem. Phys. Lett. 317 (1–2) (2000) 159–164, http://dx.doi.org/10.1016/S0009-2614(99)01346-9. URL http://www.sciencedirect.com/science/article/pii/S0009261499013469.

[86] T.J. Giese, D.M. York, J. Chem. Phys. 133 (24) (2010), http://dx.doi.org/10.1063/1.3515479.

[87] V.U. Nazarov, G. Vignale, Phys. Rev. Lett. 107 (21) (2011), http://dx.doi.org/10.1103/PhysRevLett.107.216402.

[88] M. Barbry, N.E. Koval, J. Aizpurua, D. Sánchez-Portal, P. Koval, Size dispersion of the plasmon frequency in metal clusters: *ab initio* atomistic description, in preparation doi- URL-.

[89] D. Foerster, P. Koval, J. Chem. Phys. 131 (4) (2009) 044103, http://dx.doi.org/10.1063/1.3179755. URL http://scitation.aip.org/content/aip/journal/jcp/131/4/10.1063/1.3179755.

[90] A. Stan, N.E. Dahlen, R. van Leeuwen, J. Chem. Phys. 130 (11) (2009) 114105, http://dx.doi.org/10.1063/1.3089567.

[91] F. Neese, F. Wennmohs, A. Hansen, U. Becker, Chem. Phys. 356 (1–3) (2009) 98–109, http://dx.doi.org/10.1016/j.chemphys.2008.10.036. URL http://linkinghub.elsevier.com/retrieve/pii/S0301010408005089.

[92] X. Ren, P. Rinke, V. Blum, J. Wieferink, A. Tkatchenko, A. Sanfilippo, K. Reuter, M. Scheffler, New J. Phys. 14 (5) (2012) 053020. URL http://stacks.iop.org/1367-2630/14/i=5/a=053020.

[93] F. Aryasetiawan, O. Gunnarsson, Phys. Rev. B 49 (1994) 16214–16222, http://dx.doi.org/10.1103/PhysRevB.49.16214. URL https://link.aps.org/doi/10.1103/PhysRevB.49.16214.

[94] X. Blase, P. Ordejón, Phys. Rev. B 69 (2004) 085111, http://dx.doi.org/10.1103/PhysRevB.69.085111. URL https://link.aps.org/doi/10.1103/PhysRevB.69.085111.

[95] P. Umari, G. Stenuit, S. Baroni, Phys. Rev. B 79 (2009) 201104, http://dx.doi.org/10.1103/PhysRevB.79.201104. URL https://link.aps.org/doi/10.1103/PhysRevB.79.201104.

[96] P. Koval, J. Talman, Comput. Phys. Comm. 181 (12) (2010) 2212–2213, http://dx.doi.org/10.1016/j.cpc.2010.08.024. URL http://www.sciencedirect.com/science/article/pii/S0010465510003188.

[97] A.D. Bochevarov, H. Edward, T.F. Hughes, J.R. Greenwood, D.A. Braden, M.P. Dean, D. Rinaldo, M.D. Halls, J. Zhang, R.A. Friesner, Int. J. Quantum Chem. 113 (18) (2013) 2110–2142, http://dx.doi.org/10.1002/qua.24481. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.24481.

[98] Q. Sun, T.C. Berkelbach, N.S. Blunt, G.H. Booth, S. Guo, Z. Li, J. Liu, J.D. McClain, E.R. Sayfutyarova, S. Sharma, S. Wouters, G.K.-L. Chan, Wiley Interdiscip. Rev.: Comput. Mol. Sci. (2017) e1340e1340, http://dx.doi.org/10.1002/wcms.1340.

[99] Q. Sun, contributors, Main repository of the `pyscf` project, https://github.com/sunqm/pyscf.

[100] M. Barbry, P. Koval, contributors, Fork of the PYSCF project in which we add the features to extend NAO functionality, https://github.com/cfm-mpc/pyscf, 2017.

[101] GitHub-community, Webpage of the GitHub project, https://github.com/, 2017.

[102] M. Barbry, P. Koval, Recommended workflow for getting of and contributing to the NAO-enabled PySCF package, https://github.com/cfm-mpc/pyscf/wiki/Git-workflow (2017).

[103] P. Koval, M. Barbry, D. Sánchez-Portal, Comput. Phys. Comm. (2017).

[104] R. Bauernschmitt, R. Ahlrichs, Chem. Phys. Lett. 256 (4) (1996) 454–464, http://dx.doi.org/10.1016/0009-2614(96)00440-X. URL http://www.sciencedirect.com/science/article/pii/000926149600440X.

[105] T.H. Dunning, Jr., J. Chem. Phys. 90 (2) (1989) 1007–1023, http://dx.doi.org/10.1063/1.456153.

[106] E. Artacho, E. Anglada, O. Diéguez, J.D. Gale, A. García, J. Junquera, R.M. Martin, P. Ordejón, J.M. Pruneda, D. Sánchez-Portal, J.M. Soler, J. Phys.: Condens. Matter 20 (6) (2008) 064208. URL http://stacks.iop.org/0953-8984/20/i=6/a=064208.

[107] A. Garcia, N. Papior, contributors, Main repository of the `SIESTA` package, https://launchpad.net/siesta, 2017.

[108] T. Ozaki, contributors, Website of the `OpenMX` package, http://www.openmx-square.org/, 2017 .

[109] Open-source repository of the `Fireball` package, https://github.com/fireball-QMD , 2017.

[110] R. Dronskowski, P.E. Blöchl, J. Phys. Chem. 97 (33) (1993) 8617–8624.

[111] A.H. Larsen, J.J. Mortensen, J. Blomqvist, I.E. Castelli, R. Christensen, M. Dułak, J. Friis, M.N. Groves, B. Hammer, C. Hargus, E.D. Hermes, P.C. Jennings, P.B. Jensen, J. Kermode, J.R. Kitchin, E.L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J.B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K.S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K.W. Jacobsen, J. Phys.: Condens. Matter 29 (27) (2017) 273002. URL http://stacks.iop.org/0953-8984/29/i=27/a=273002.

[112] D.R. Hamann, Phys. Rev. B 88 (2013) 085117, http://dx.doi.org/10.1103/PhysRevB.88.085117. URL https://link.aps.org/doi/10.1103/PhysRevB.88.085117.

[113] O. Gritsenko, R. van Leeuwen, E. van Lenthe, E.J. Baerends, Phys. Rev. a 51 (1995) 1944–1954, http://dx.doi.org/10.1103/PhysRevA.51.1944. URL https://link.aps.org/doi/10.1103/PhysRevA.51.1944.

[114] A. Kokalj, Comput. Mater. Sci. 28 (2) (2003) 155–168, http://dx.doi.org/10.1016/S0927-0256(03)00104-6. proceedings of the symposium on software development for process and materials design. URL http://www.sciencedirect.com/science/article/pii/S0927025603001046.

[115] C. Sönnichsen, T. Franzl, T. Wilk, G. Von Plessen, J. Feldmann, New J. Phys. 4 (2002), http://dx.doi.org/10.1088/1367-2630/4/1/393.

[116] A. Desireddy, B.E. Conn, J. Guo, B. Yoon, R.N. Barnett, B.M. Monahan, K. Kirschbaum, W.P. Griffith, R.L. Whetten, U. Landman, T.P. Bigioni, Nature 501 (7467) (2013) 399–402, http://dx.doi.org/10.1038/nature12523. URL http://www.ncbi.nlm.nih.gov/pubmed/24005327.

[117] P. Aubertin, M.A.B. Aissa, N. Raouafi, S. Joiret, A. Courty, E. Maisonhaute, Nano Res. 8 (5) (2015) 1615–1626, http://dx.doi.org/10.1007/s12274-014-0650-5.

[118] K. Yabana, G.F. Bertsch, Phys. Rev. A 60 (5) (1999) 3809–3814, http://dx.doi.org/10.1103/PhysRevA.60.3809. URL http://link.aps.org/doi/10.1103/PhysRevA.60.3809.

[119] J. Junquera, O. Paz, D. Sánchez-Portal, E. Artacho, Phys. Rev. B 64 (2001) 235111, http://dx.doi.org/10.1103/PhysRevB.64.235111. URL https://link.aps.org/doi/10.1103/PhysRevB.64.235111.

[120] P. Haas, F. Tran, P. Blaha, Phys. Rev. B 79 (2009) 085104, http://dx.doi.org/10.1103/PhysRevB.79.085104. URL http://link.aps.org/doi/10.1103/PhysRevB.79.085104.

[121] P. Haas, F. Tran, P. Blaha, Phys. Rev. B 79 (2009) 209902, http://dx.doi.org/10.1103/PhysRevB.79.209902. URL http://link.aps.org/doi/10.1103/PhysRevB.79.209902.

[122] J.C. Idrobo, S.T. Pantelides, Phys. Rev. B 82 (2010) 085420, http://dx.doi.org/10.1103/PhysRevB.82.085420. URL http://link.aps.org/doi/10.1103/PhysRevB.82.085420.

[123] X.L. Lozano, C. Mottet, H.-C. Weissker, J. Phys. Chem. C 117 (6) (2013) 3062–3068, http://dx.doi.org/10.1021/jp309957y.

[124] M. Irie, T. Fukaminato, K. Matsuda, S. Kobatake, Chem. Rev. 114 (24) (2014) 12174–12277, http://dx.doi.org/10.1021/cr500249p. pMID: 25514509.

[125] S. Nosé, J. Chem. Phys. 81 (1) (1984) 511–519, http://dx.doi.org/10.1063/1.447334.

[126] P.J. Stephens, F.J. Devlin, C.F. Chabalowski, M.J. Frisch, J. Phys. Chem. 98 (45) (1994) 11623–11627, http://dx.doi.org/10.1021/j100096a001.

[127] M. Barbry, P. Koval, D. Sánchez-Portal, Atomistic *ab initio* theory of the electron energy loss spectroscopy, in preparation, 2018.

[128] M. Barbry, Plasmons in Nanoparticles: Atomistic *Ab Initio* Theory for Large Systems (Ph.D. thesis), University of Basque Country, Donostia-San Sebastián, Spain, 2018 . submitted for publication. http://cfm.ehu.es/education/thesis/.