Received: 1 November 2021 | Revised: 4 March 2022 | Accepted: 6 March 2022

DOI: 10.1002/wcms.1617

SOFTWARE FOCUS

WIREs COMPUTATIONAL MOLECULAR SCIENCE

WILEY

n2v: A density-to-potential inversion suite. A sandbox for creating, testing, and benchmarking density functional theory inversion methods

Yuming Shi[1] | **Victor H. Chávez**[2] | **Adam Wasserman**[1,2]

[1]Department of Physics and Astronomy, Purdue University, West Lafayette, Indiana, USA

[2]Department of Chemistry, Purdue University, West Lafayette, Indiana, USA

Correspondence
Adam Wasserman, Department of Chemistry, Purdue University, 560 Oval Drive, West Lafayette, IN, USA.
Email: awasser@purdue.edu

Funding information
This work was supported by the National Science Foundation under Grant No. CHE-1900301. Victor H. Chavez was supported by a fellowship from The Molecular Sciences Software Institute under NSF grant OAC-1547580.

Edited by: Peter R. Schreiner, Editor-in-Chief

Abstract

From the most fundamental to the most practical side of density functional theory (DFT), Kohn–Sham inversions (iKS) can contribute to the development of functional approximations and shed light on their performance and limitations. On the one hand, iKS allows for the direct exploration of the Hohenberg–Kohn and Runge–Gross density-to-potential mappings that provide the foundations for DFT and time-dependent DFT. On the other hand, iKS can guide the analysis and development of approximate exchange–correlation and noninteracting kinetic energy functionals, and diagnose their errors. iKS can also play a similar role in the development of nonadditive functionals for modern density-based embedding methods. Various strategies to perform iKS calculations have been explored since the inception of DFT. We introduce *n2v*, a density-to-potential inversion Python module that is capable of performing the most useful and state-of-the-art inversion calculations. Currently based on *NumPy*, *n2v* was developed to be easy to learn by newcomers to the field. Its structure allows for other inversion methods to be easily added. The code offers a general interface that gives the freedom to use different software packages in the computational molecular sciences (CMS) community, and the current release supports the *Psi4* and *PySCF* packages. Six inversion methods have been implemented into *n2v* and are reviewed here along with detailed numerical illustrations on molecules with numbers of electrons ranging from ∼10 to ∼100.

This article is categorized under:
Software > Quantum Chemistry
Electronic Structure Theory > Density Functional Theory

KEYWORDS
density functional theory, Kohn–Sham inversion, open-source software

Yuming Shi and Victor H. Chávez contributed equally to this study.

1 | INTRODUCTION

Density functional theory (DFT) is the most widely used method to calculate the electronic properties of molecules and materials. The art of developing approximations for the exchange–correlation (XC) functional of Kohn–Sham (KS) DFT,[1] has been practiced for over 55 years.[2,3] Most KS-DFT calculations are performed as forward problems (fKS): Given an external potential and total number of electrons, choose an XC functional approximation and solve the (forward) KS equations to find the ground-state density and energy of the system. However, there are occasions in which solutions of the *inverse* problem (iKS) are needed and one wants to find some or all of the KS quantities (potentials, KS orbitals, and eigenvalues) that correspond to a given density. Examples include studies of exact properties,[4–7] analysis and improvement of approximations,[8–17] and potential-reconstruction steps for embedding methods.[18–24] Numerical approaches to address both fKS and iKS problems are needed to fully explore the one-to-one maps between densities and potentials that establish the foundations for ground-state DFT[25] and time-dependent DFT.[26]

Many different methods have been put forth to solve numerically the iKS problem. These can be classified as either pure density-to-potential methods, where only the densities are taken as input,[27–40] or as wavefunction-to-potential methods,[41–44] where a many-electron wavefunction or density matrix is employed. On the other hand, iKS problems can also be classified based on how the convergence is reached. While some methods are based on self-consistent calculations, others depend on constrained optimizations that are usually much more efficient, especially when implemented on finite potential basis sets. The price to pay is less robustness and more sensitivity to various factors, as discussed recently in reference 45.

Unlike the fKS problem, whose accuracy is limited by the XC functional approximations, iKS methods are often "exact" in the sense that, at the analytical convergence limit on a complete basis set, the potential corresponding to the input density should be reproduced exactly by any of the methods mentioned above, and be identical for all. However, in practice, this is not the case. Indeed, iKS are much more sensitive than fKS and ill-posed.[46,47] Additional flaws include numerical instabilities that occur due to the use of incomplete basis sets leading to unphysical oscillations in the potential.[48–51] Chayes et al.[52] showed that such numerical deficiencies are not present when grids are used instead of basis sets, but grids are not practical for larger molecules. There seems to be an implicit efficiency-accuracy tradeoff,[45,46] but different methods perform differently on this seesaw, as will be stressed in this paper. Consequently, each iKS method has zones in which it works better than others, just as different XC functional approximations are often used for different computational purposes/systems in fKS.

A human/social factor that has slowed the development of iKS methods is that, although the algorithms are accessible, the codes in which they were implemented are not. Some methods are hidden behind a paywall that (1) makes the codes inaccessible to research groups that cannot afford them, and (2) even if the licensing can be afforded, these codes usually just ship as an executable, so users do not have access to the source codes. iKS methods are not trivial, and their implementation and testing requires plenty of development time. This requirement creates an unnecessary additional barrier for researchers to understand, reproduce, and modify an algorithm for their own purposes.

As a response, we present the library *n2v* (*n2v* stands for "density-to-potential"). *n2v* is a free, open-source library that offers many of the widely used and recent inversion methods. Six selected methods are implemented and studied: the direct KS-inversion method,[41] the modified Ryabinkin–Kohut–Staroverov method (mRKS),[42,43] the Wu–Yang method (WY),[40] the error function partial-differentiation-equation constrained-optimization method (PDE-CO),[46,53] the Zhao–Morrison–Parr method (ZMP)[31] and the Ou–Carter methods (OC).[37] mRKS is accurate but expensive to run.[54] WY is the lightest and most efficient method. ZMP and WY are most widely used. PDE-CO was implemented in various approaches and shows promising features and capabilities. OC is a pure iKS method that balances well the efficiency and accuracy. Helpful tips and insights regarding these methods are discussed in references 8, 45, 54–56. A comparison of methods in the context of embedding potentials is discussed in references 57, 58. *n2v* includes a repository with Jupyter Notebooks that allows users to test all the methods and reproduce all of the procedures shown in this paper.

There are two other recent packages that feature density-to-potential inversions. *Serenity*[59] is a C++ code for calculations on systems composed of several subsystems, and it includes inversion subroutines that are useful in the context of embedding theories; *KS-pies*,[60] is another high-level code that allows users to perform iKS; *n2v* is introduced as an alternative that provides users with access to more inversion methods as well as the possibility to interface with more than one computational chemistry package.

n2v offers users sufficient options when they are faced with an iKS problem. Users can test and compare the results from the WY method (most efficient) to the mRKS (very accurate) and identify the methods that will serve them best. Also, iKS method-developers can compare their own methods to those implemented into *n2v*. We provide in this paper numerical comparisons of various methods and benchmarks on systems from 2 to about 100 electrons (largest system reported for iKS). We begin with a brief overview of the theory underlying the methods implemented so far into *n2v*. Atomic units are used throughout.

2 | METHODS CURRENTLY IMPLEMENTED INTO *n2v*

All of the methods relate to the Kohn–Sham equations:

$$\left[-\frac{1}{2}\nabla^2 + v_{\text{ext}}(\mathbf{r}) + v_{\text{H}}(\mathbf{r}) + v_{\text{XC}}(\mathbf{r})\right]\psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}), \tag{1}$$

where $\{\psi_i(\mathbf{r})\}$ and $\{\epsilon_i\}$ are the KS orbitals and corresponding eigenvalues, $v_{\text{ext}}(\mathbf{r})$ is the external potential due to the nuclei, $v_{\text{H}}(\mathbf{r}) = v_{\text{H}}[n](\mathbf{r}) = \int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$ is the Hartree potential, and $v_{\text{XC}}(\mathbf{r}) = v_{\text{XC}}[n](\mathbf{r})$ is the XC potential. For a given approximation to the XC energy functional $E_{\text{XC}}[n]$, the *forward* KS problem consists on solving Equation (1) self-consistently with $v_{\text{XC}}[n](\mathbf{r}) = \delta E_{\text{XC}}[n]/\delta n(\mathbf{r})$, where the electron density is defined as:

$$n(\mathbf{r}) = 2\sum_{i=1}^{occ} |\psi_i(\mathbf{r})|^2. \tag{2}$$

In the following, any function that carries the subscript "in" (as in $n_{\text{in}}(\mathbf{r})$ below) represents a quantity that enters iKS as *input*.

2.1 | Direct Kohn–Sham inversion

Let us start with the most direct inversion (the "Kohn–Sham Inversion Formula"), applicable when the set of occupied orbitals is available. Reorganize Equation (1) by pre-multiplying by $\psi^*_{i,\text{in}}(\mathbf{r})$, summing i over the N occupied orbitals, and then dividing by $n_{\text{in}}(\mathbf{r})$ to find $v_{\text{XC}}(\mathbf{r})$. The result is a weighted average of orbital-specific potentials[41]:

$$v_{\text{XC}}(\mathbf{r}) = \frac{1}{n_{\text{in}}(\mathbf{r})}\sum_{i=1}^{N}\left[\frac{1}{2}\psi^*_{i,\text{in}}(\mathbf{r})\nabla^2\psi_{i,\text{in}}(\mathbf{r}) + \epsilon_{i,\text{in}}\left|\psi_{i,\text{in}}(\mathbf{r})\right|^2\right] - v_{\text{ext}}(\mathbf{r}) - v_{\text{H}}(\mathbf{r}). \tag{3}$$

This direct method is rarely useful in practice because it requires KS orbitals that are already the result of a KS calculation. Nevertheless, Equation (3) has been used for determining exchange–correlation potentials,[61,62] where the direct $v_{\text{XC}}(\mathbf{r})$ is less obvious to obtain directly on a grid for a high level XC functional. It has also been used for determining the functional derivative of a kinetic energy functional,[20] and for constructing orbital-averaged exchange–correlation potentials for orbital-dependent functionals.[41]

Unfortunately, the procedure indicated by Equation (3) suffers from numerical difficulties when the $\psi_i(\mathbf{r})$'s are expanded on finite basis-sets.[50] These numerical issues are almost exclusively dependent on the basis-sets employed. Gaiduk et al.[48] showed how one could build an oscillation profile from a simple method such as the Slater exchange functional. The profile can then be used to subtract the errors from any other more sophisticated density-functional approximations. This method along with its basis-set correction is the first method available in *n2v*.

2.2 | The Zhao–Morrison–Parr method

One of the earlier iKS methods is the ZMP, named after their developers Zhao, Morrison, and Parr[31] in the early 1990's, and recently generalized by Kumar and Harbola.[56] The method uses a self-consistent calculation to determine the one-electron effective potential from the target density $n(\mathbf{r})$. To do so, a constraint enforces a density to be equal to $n_{\text{in}}(\mathbf{r})$ at a certain limit:

$$C[n(\mathbf{r}), n_{\text{in}}(\mathbf{r})] = \frac{1}{2}\int\int \frac{\{n(\mathbf{r}) - n_{\text{in}}(\mathbf{r})\}\{n(\mathbf{r}') - n_{\text{in}}(\mathbf{r}')\}}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}d\mathbf{r}' = 0. \tag{4}$$

By attaching a Lagrange multiplier λ, one finds the corresponding potential

$$v_c^{\lambda}(\mathbf{r}) = \lambda \int \frac{n(\mathbf{r}') - n_{\text{in}}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'. \tag{5}$$

So that we can solve the differential equations

$$\left[-\frac{1}{2}\nabla^2 + v_{\text{ext}}(\mathbf{r}) + \left(1 - \frac{1}{N}\right) v_{\text{H}}(\mathbf{r}) + v_c^{\lambda}(\mathbf{r})\right]\psi_i^{\lambda}(\mathbf{r}) = \epsilon_i^{\lambda}\psi_i^{\lambda}(\mathbf{r}), \tag{6}$$

where ψ_i^{λ} and ε_i^{λ} are the eigenfunctions and eigenvalues found with v_c^{λ} with a particular λ, and the term $\left(1 - \frac{1}{N}\right)v_{\text{H}}(\mathbf{r})$ is the Fermi–Amaldi approximation. This term, besides being exact for 1 electron,[63] ensures the correct—$1/r$ asymptotic behavior of the resulting potential.

In the limit $\lambda \to \infty$ Equation (5) becomes Equation (1) and we can retrieve the Kohn–Sham orbitals and orbital energies up to a constant. For finite λ, the solution will only be an approximation to the exact orbitals.

With a choice of λ, at convergence, the resulting $v_{\text{XC}}(\mathbf{r})$ is given by:

$$v_{\text{XC}}^{\lambda}(\mathbf{r}) = v_c^{\lambda}(\mathbf{r}) - \frac{1}{N} v_{\text{H}}(\mathbf{r}). \tag{7}$$

Finite values of λ lead to numerical difficulties.[31] This issue is immediately clear by examining the results for large values of λ. Thus, the most important and time-consuming aspect of this method relies on carefully choosing one or several values of λ. The original work[31] uses an extrapolation technique in which the potentials from the values of $\lambda = \{100, 140, 200\}$ were expanded using a three-term power series in $1/\lambda$. To avoid fitting and make the method more generally applicable, the default for *n2v* is the iterative method used by Morrison and Zhao.[64] In this approach, one starts with a small value of $\lambda = \lambda_i$, then increases its value to $\lambda = \lambda_{i+1}$ and incorporates the previous self-consistent potential as a starting potential for the calculation of λ_{i+1}. This process is iterated until the desired convergence criteria are reached.

2.3 | The Wu–Yang method

The Wu–Yang method[40] implemented with finite potential basis sets is one of the most efficient methods for iKS problems.[45] With explicitly derived Hessian and gradient, a regular optimizer (e.g., trust-region) will usually converge within 10 iterations. Because of the absence of mesh points, the memory required is also usually acceptable.

The Wu–Yang method optimizes the KS noninteracting kinetic energy T_s under the constraint that the output density $n(\mathbf{r})$ is the same as the target density $n_{\text{in}}(\mathbf{r})$. By setting the Lagrangian multiplier as the KS potential $v_{\text{KS}}(\mathbf{r})$, which is the condition for the stationary point, the Lagrangian can be built as:

$$W[v_{\text{KS}}] = T_s[\Psi_{\text{det}}[v_{\text{KS}}]] + \int d\mathbf{r} v_{\text{KS}}(\mathbf{r})[n(\mathbf{r}) - n_{\text{in}}(\mathbf{r})], \tag{8}$$

where $\Psi_{\text{det}}[v_{\text{KS}}]$ is the Slater determinant corresponding to $v_{\text{KS}}(\mathbf{r})$. $W[v_{\text{KS}}]$ is proven to be concave.[40] Thus, in order to reach the stationary point, the Lagrangian can be optimized with its gradient:

$$\frac{\delta W[v_{\text{KS}}]}{\delta v_{\text{KS}}(\mathbf{r})} = n(\mathbf{r}) - n_{\text{in}}(\mathbf{r}) \tag{9}$$

and hessian

$$\begin{aligned}&\frac{\delta^2 W[\Psi_{\text{det}}[v_{\text{KS}}], v_{\text{KS}}]}{\delta v_{\text{KS}}(\mathbf{r})\delta v_{\text{KS}}(\mathbf{r}')}\\ &= \frac{\delta n(\mathbf{r})}{\delta v_{\text{KS}}(\mathbf{r}')}\\ &= 2\sum_i^{occ.}\sum_a^{unocc.}\frac{\psi_i^*(\mathbf{r})\psi_a(\mathbf{r})\psi_i(\mathbf{r}')\psi_a^*(\mathbf{r}')}{\epsilon_i-\epsilon_a}.\end{aligned} \tag{10}$$

$v_{\text{KS}}(\mathbf{r})$ can be decomposed similarly as in ZMP:

$$v_{\text{KS}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_0(\mathbf{r}) + v_{\text{PBS}}(\mathbf{r}), \tag{11}$$

where $v_0(\mathbf{r})$ is a guide potential designed to capture known features of the correct KS potential; the Fermi–Amaldi potential is the usual choice. The rest is expanded on a finite potential basis set (PBS) $\{\phi_t\}$

$$v_{\text{PBS}}(\mathbf{r}) = \sum_t b_t\phi_t(\mathbf{r}). \tag{12}$$

In *n2v*, finite difference checks are implemented for both the gradient and hessian in the PBS.[45] This can be a practically useful tool to check the error and convergence. Since the Wu–Yang optimization can be very sensitive for multiple reasons, regularization is important for practice.[45]

2.4 | PDE-constrained optimization

Even though the iKS problem is generally ill-posed,[46] it is still supposed that within a reasonably close region around the exact XC potential, the closer the KS density is to the exact density, the more accurate the potential will be.[46,53] Following this intuition, a density error is defined and optimized under several constraints required by the KS model.[35,46,53] The density error is defined as:

$$N_{\text{error}} = \int d\mathbf{r}|n(\mathbf{r}) - n_{\text{in}}(\mathbf{r})|^2. \tag{13}$$

The Lagrangian is defined as:

$$\begin{aligned}&L[v_{\text{KS}}, \{\psi_i\}, \{\epsilon_i\}, \{p_i\}, \{\mu_i\}]\\ &= \int (n(\mathbf{r}) - n_{\text{in}}(\mathbf{r}))^2 d\mathbf{r}\\ &+ \sum_{i=1}^{N/2}\int p_i(\mathbf{r})\left(-\frac{1}{2}\nabla^2 + v_{\text{KS}}(\mathbf{r}) - \epsilon_i\right)\psi_i(\mathbf{r})d\mathbf{r}\\ &+ \sum_{i=1}^{N/2}\mu_i\left(\int|\psi_i(\mathbf{r})|^2 d\mathbf{r} - 1\right),\end{aligned} \tag{14}$$

where $\{p_i\}$ and $\{\mu_i\}$ are Lagrange multipliers for the constraints that $\{\psi_i\}$ are the KS orbitals of v_{KS} with corresponding eigenvalues $\{\epsilon_i\}$ (Equation 1) and that $\{\psi_i\}$ are normalized. The normal equations with respect to p_i, μ_i, ψ_i, ϵ_i, and v_{KS} are:

$$\left(-\frac{1}{2}\nabla^2 + v_{\text{KS}}(\mathbf{r})\right)\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}), \tag{15a}$$

$$\int |\psi_i(\mathbf{r})|^2 d\mathbf{r} = 1, \tag{15b}$$

$$\left(-\frac{1}{2}\nabla^2 + v_{\mathrm{KS}}(\mathbf{r}) - \epsilon_i\right) p_i(\mathbf{r}) = 8(n_{\mathrm{in}}(\mathbf{r}) - n(\mathbf{r}))\psi_i(\mathbf{r}) - 2\mu_i\psi_i(\mathbf{r}), \tag{15c}$$

$$\int p_i(\mathbf{r})\psi_i(\mathbf{r}) d\mathbf{r} = 0, \tag{15d}$$

$$\frac{\delta L}{\delta v_{\mathrm{KS}}(\mathbf{r})} = \sum_{i=1}^{N/2} p_i(\mathbf{r})\psi_i(\mathbf{r}). \tag{15e}$$

Equations (15a) with (15b) are solved for the $\{\psi_i\}$; Equations (15c) with (15d) are solved for the $\{p_i\}$. Then $\{\psi_i\}$ and $\{p_i\}$ can be plugged into (15e) to solve for the gradient.

The PDE-constrained optimization has been implemented into *n2v* in finite basis sets as described in reference 45. Since the Lagrangian (Equation (14)) is not concave as in WY (Equation (8)),[40,45] there is no guarantee that the optimization will perform well. Usually a good guide potential (such as Fermi–Amaldi) in Equation (11) with a zero initial guess in Equation (12) are sufficient to achieve convergence. Moreover, special care should be given to the singularity introduced by subtracting the Hamiltonian operator by its eigenvalues as in Equation (15c) and the orthogonality as defined in Equation (15d).[53]

2.5 | The modified Ryabinkin–Kohut–Staroverov method

The three methods above are *pure* KS inversion methods, in which only the densities are taken as input. The modified Ryabinkin–Kohut–Staroverov (mRKS) method[42,43] is an accurate method that makes use of one-electron reduced density matrices $\gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}')$ and two-electron reduced density matrices $\Gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}_2;\mathbf{r}',\mathbf{r}_2')$ obtained from higher-level wavefunction calculations (WF). In mRKS, the $v_{\mathrm{XC}}(\mathbf{r})$ is solved self-consistently on Kohn–Sham systems (KS) in equation:

$$v_{\mathrm{XC}}(\mathbf{r}) = v_{\mathrm{XC}}^{\mathrm{hole}}(\mathbf{r}) + \overline{\epsilon}^{\mathrm{KS}}(\mathbf{r}) - \overline{\epsilon}^{\mathrm{WF}}(\mathbf{r}) + \frac{\tau_P^{\mathrm{WF}}(\mathbf{r})}{n^{\mathrm{WF}}(\mathbf{r})} - \frac{\tau_P^{\mathrm{KS}}(\mathbf{r})}{n^{\mathrm{KS}}(\mathbf{r})}. \tag{16}$$

The potential of the exchange–correlation hole ($v_{\mathrm{XC}}^{\mathrm{hole}}$), the average local electron energy ($\overline{\epsilon}$), and the Pauli kinetic energy density (τ_P) are defined as:

$$v_{\mathrm{XC}}^{\mathrm{hole}}(\mathbf{r}) = \int d\mathbf{r}_2 \frac{n_{\mathrm{XC}}(\mathbf{r},\mathbf{r}_2)}{|\mathbf{r}-\mathbf{r}_2|}, \tag{17a}$$

$$\overline{\epsilon}^{\mathrm{KS}}(\mathbf{r}) = \frac{2}{n^{\mathrm{KS}}(\mathbf{r})}\sum_{i=1}^{N/2} \epsilon_i |\psi_i(\mathbf{r})|^2, \tag{17b}$$

$$\overline{\epsilon}^{\mathrm{WF}}(\mathbf{r}) = \frac{2}{n^{\mathrm{WF}}(\mathbf{r})}\sum_{k=1}^{M} \lambda_k |f_k(\mathbf{r})|^2, \tag{17c}$$

$$\tau_P^{\mathrm{WF}}(\mathbf{r}) = \frac{2}{n^{\mathrm{WF}}(\mathbf{r})}\sum_{k<l}^{M} n_k n_l |\chi_k(\mathbf{r})\nabla\chi_l(\mathbf{r}) - \chi_l(\mathbf{r})\nabla\chi_k(\mathbf{r})|^2, \tag{17d}$$

$$\tau_P^{\mathrm{KS}}(\mathbf{r}) = \frac{2}{n^{\mathrm{KS}}(\mathbf{r})}\sum_{i<j}^{M} n_i n_j |\psi_i(\mathbf{r})\nabla\psi_j(\mathbf{r}) - \psi_j(\mathbf{r})\nabla\psi_i(\mathbf{r})|^2, \tag{17e}$$

where the exchange–correlation hole $n_{\mathrm{XC}}(\mathbf{r},\mathbf{r}_2)$ is

$$n_{\mathrm{XC}}(\mathbf{r},\mathbf{r}_2)=\frac{\Gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}_2;\mathbf{r},\mathbf{r}_2)}{n_{\mathrm{in}}(\mathbf{r})}-n_{\mathrm{in}}(\mathbf{r}_2). \tag{18}$$

The $\{\chi_k\}$ are the natural orbitals of $\gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}')$, electron density from wavefunction method is $n^{\mathrm{WF}}(\mathbf{r})=\gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r})$. $\{\lambda_k\}$, $\{f_k(\mathbf{r})\}$ are the eigenvalues and eigenfunctions of the generalized Fock matrix:

$$F(\mathbf{r},\mathbf{r}')=\left(-\frac{\nabla_{\mathbf{r}}^2}{2}+v(\mathbf{r})\right)\gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}')+\int d\mathbf{r}_2\frac{\Gamma_{\mathrm{in}}(\mathbf{r},\mathbf{r}_2;\mathbf{r}',\mathbf{r}_2)}{|\mathbf{r}-\mathbf{r}_2|}. \tag{19}$$

and $n_{i,j,k,l}$ is the orbital occupation number. When a new v_{XC} is obtained from Equation (16), it is inserted into Equation (1) to solve for quantities that will fill in the right-hand-side of Equation (16). Superscripts "WF" and "KS" are used to distinguish the results of a wavefunction calculation (input) from those of a KS calculation (fKS).

2.6 | The Ou–Carter method

Ou and Carter developed a pure iKS method inspired by mRKS.[37] Instead of a cancellation between one equation derived from a wavefunction calculation and one from the KS system, only the KS system is utilized to derive a self-consistent equation for $v_{\mathrm{XC}}(\mathbf{r})$ from the KS equation:

$$v_{\mathrm{XC}}(\mathbf{r})=\overline{\epsilon}^{\mathrm{KS}}(\mathbf{r})-\frac{\tau_L^{\mathrm{KS}}(\mathbf{r})}{n^{\mathrm{KS}}(\mathbf{r})}-v_{\mathrm{ext}}(\mathbf{r})-v_{\mathrm{H}}(\mathbf{r}), \tag{20}$$

where τ_L^{KS} is the KS kinetic energy density and

$$\frac{\tau_L^{\mathrm{KS}}(\mathbf{r})}{n^{\mathrm{KS}}(\mathbf{r})}=\frac{\left|\nabla n^{\mathrm{KS}}(\mathbf{r})\right|^2}{8\left|n^{\mathrm{KS}}(\mathbf{r})\right|^2}-\frac{\nabla^2 n^{\mathrm{KS}}(\mathbf{r})}{4n^{\mathrm{KS}}(\mathbf{r})}+\frac{\tau_P^{\mathrm{KS}}(\mathbf{r})}{n^{\mathrm{KS}}(\mathbf{r})}. \tag{21}$$

By replacing the KS density everywhere with the accurate input density and the external potential $v_{\mathrm{ext}}(\mathbf{r})$ by an effective $\tilde{v}_{\mathrm{ext}}(\mathbf{r})$, the final expression for $v_{\mathrm{XC}}(\mathbf{r})$ is:

$$v_{\mathrm{XC}}(\mathbf{r})=\overline{\epsilon}^{\mathrm{KS}}(\mathbf{r})+\frac{\nabla^2 n_{\mathrm{in}}(\mathbf{r})}{4n_{\mathrm{in}}(\mathbf{r})}-\frac{|\nabla n_{\mathrm{in}}(\mathbf{r})|^2}{8|n_{\mathrm{in}}(\mathbf{r})|^2}-\frac{\tau_P^{\mathrm{KS}}(\mathbf{r})}{n^{\mathrm{KS}}(\mathbf{r})}-\tilde{v}_{\mathrm{ext}}(\mathbf{r})-v_{\mathrm{H}}[n_{\mathrm{in}}](\mathbf{r}). \tag{22}$$

Because in this method there is no error cancellation between wavefunction results and KS-DFT results as in mRKS, using $\tilde{v}_{\mathrm{ext}}(\mathbf{r})$ is necessary to eliminate the errors (mostly numerical) in each component.[48,54] This is achieved by doing one extra calculation with a known (and simple) functional (usually LDA) on the same basis set to handle the error that comes from finite basis sets[48]:

$$\tilde{v}_{\mathrm{ext}}(\mathbf{r})=\overline{\epsilon}^{\mathrm{KS}}(\mathbf{r})-\frac{\tau_L^{LDA}(\mathbf{r})}{n_{\mathrm{in}}^{LDA}}-v_{\mathrm{H}}\left[n_{\mathrm{in}}^{LDA}\right](\mathbf{r})-v_{\mathrm{XC}}^{LDA}\left[n_{\mathrm{in}}^{LDA}\right](\mathbf{r}), \tag{23}$$

where $\frac{\tau_L^{LDA}(\mathbf{r})}{n_{\mathrm{in}}^{LDA}}$ is defined by Equation (21).

3 | THE *n2v* LIBRARY

Our library is written in the high-level programming language Python, allowing us to focus on readability and rapid development. This enables users to quickly learn and use the library as well as the inverse Kohn–Sham methodology. We aim to follow the rapid-prototyping scheme introduced by *Psi4Numpy*,[65] a repository of reference implementations and interactive tutorials. By using this methodology, we provide access to a set of inversion methods that can be learned and modified quickly, leading to faster implementations of new applications and methods.

In a nutshell, *n2v* implements high-level details of the inversion algorithms but relies on other computational chemistry packages to perform other lower-level operations such as computing the orbitals, densities, and energies.

n2v has an upstream dependence on many other canonical Python libraries. For example, *Numpy*[66] is used for general numerical operations; *Scipy*[67] is used to minimize functionals; *opt_einsum*[68] is used for efficient contraction of matrices; *Pylibxc*[69] allows computation of exchange–correlation functionals, and *Matplotlib*[70] is used for visualization.

3.1 | Code overview

From its early design, we have focused on making *n2v* easy to learn, use, maintain and extend. We believe we have accomplished this by focusing on the following points:

- *Readability*. In addition to providing an extensive documentation for every component of our code, we use an automatically generated documentation through the use of Sphinx.[71] This can be accessed through the website: wasserman-group.github.io/n2v/. Additionally, we created a set of tutorials in Jupyter Notebooks, providing instructions into using our code and details of the algorithm for every method. The Notebooks can be accessed through the following repository: github.com/wasserman-group/n2v_examples. All of the results shown in this work are computed in the same fashion as the examples.
- *Accessibility*. Users can run our code in the Windows (through the use of Windows Subsystem), Linux, and macOS operating systems. We recommend the use of conda and the Python Package Index (PyPI). To begin with, at least one engine should be installed (see Section 3.2). For example, *Psi4* can be installed with `conda install -c psi4 psi4`, and *Pyscf* can be installed with `pip install pyscf`. Additionally, libxc is required for the Ou–Carter method. To access libxc through Python, one must obtain *libxc* through conda and proceed to manually install *pylibxc*. More details are available on the repository. Proceed to install *n2v* through github.com/wasserman-group/n2v or by direct install with `pip install ntov`.
- *Modern software development*. We follow the "best practices"[72] paradigm defined by The Molecular Science Software Institute[73] that covers testing, code coverage, and continuous integration.
- *Licensing*. We released our code under the BSD 3-clause license, a permissive license that allows for our code's modifications and redistribution.

3.2 | Code structure

The software ecosystems within the computational molecular sciences (CMS) community are vast, and with the increasing popularity of high-level languages such as Python[74–79] or Julia,[80–82] they will only continue to grow. Nowadays, developers designing software must keep in mind that users do not want to start using a new package to access different features.[83] To aid in lowering the barrier of using *n2v*, we introduce *engines*. An engine is an abstract class in Python that specifies what a computational chemistry code interfaced with *n2v* is expected to provide. These include basic information like geometry and basis set, as well as ingredients needed for each inversion method, such as the orbitals and densities in the atomic orbital basis set and, in some cases, on the grid.

The *Psi4*[84–86] and *PySCF*[87,88] engines are available in the current release. *Psi4* interfaces with our code through PsiAPI, its application programming interface (API). The API allows users to access most of *Psi4*'s core C++ libraries at the Python level. On the other hand, *PySCF* is written in Python and thus can be connected to *n2v* directly. The *Psi4* engine can perform all the available methods, while the *PySCF* engine can perform all except the Ou–Carter and mRKS methods. Given the intrinsic dependence on the grid of these two methods, they are still under development, but they will be ready in a future release.

Two other classes are relevant to understanding the structure of *n2v*. The first one is the `Inverter`, a high-level class that requests tasks from the engine. Additionally, the `Inverter` collects all of the information needed for an inversion, generates a guiding potential and routes it to the requested method. Each of the methods from Section 2 is contained within its class, so the `Inverter` inherits properties from all of them. In this way, once an `Inverter` object has been initialized, the user can access all of the different methods with the same object.

To perform some of the methods and visualize the resulting potentials, the class `Grider` is supplied to each engine. A `Grider` object computes relevant quantities on any given grid, such as density, orbitals and their derivatives, and more. Each of the available engines (*Psi4* and *PySCF*) uses their default spherical grid to compute the DFT kernel on the grid. For example, *Psi4* uses a Lebedev–Treutler grid using 75 radial and 302 spherical points. Although the default is robust enough to express exchange–correlation potentials on the grid, it is advised that users refine the parameters according to each system and method.

We encourage other developers to write their engines and make them available for future releases through pull requests in the repository. Our licensing allows any user to use and modify any code component that is useful in their projects.

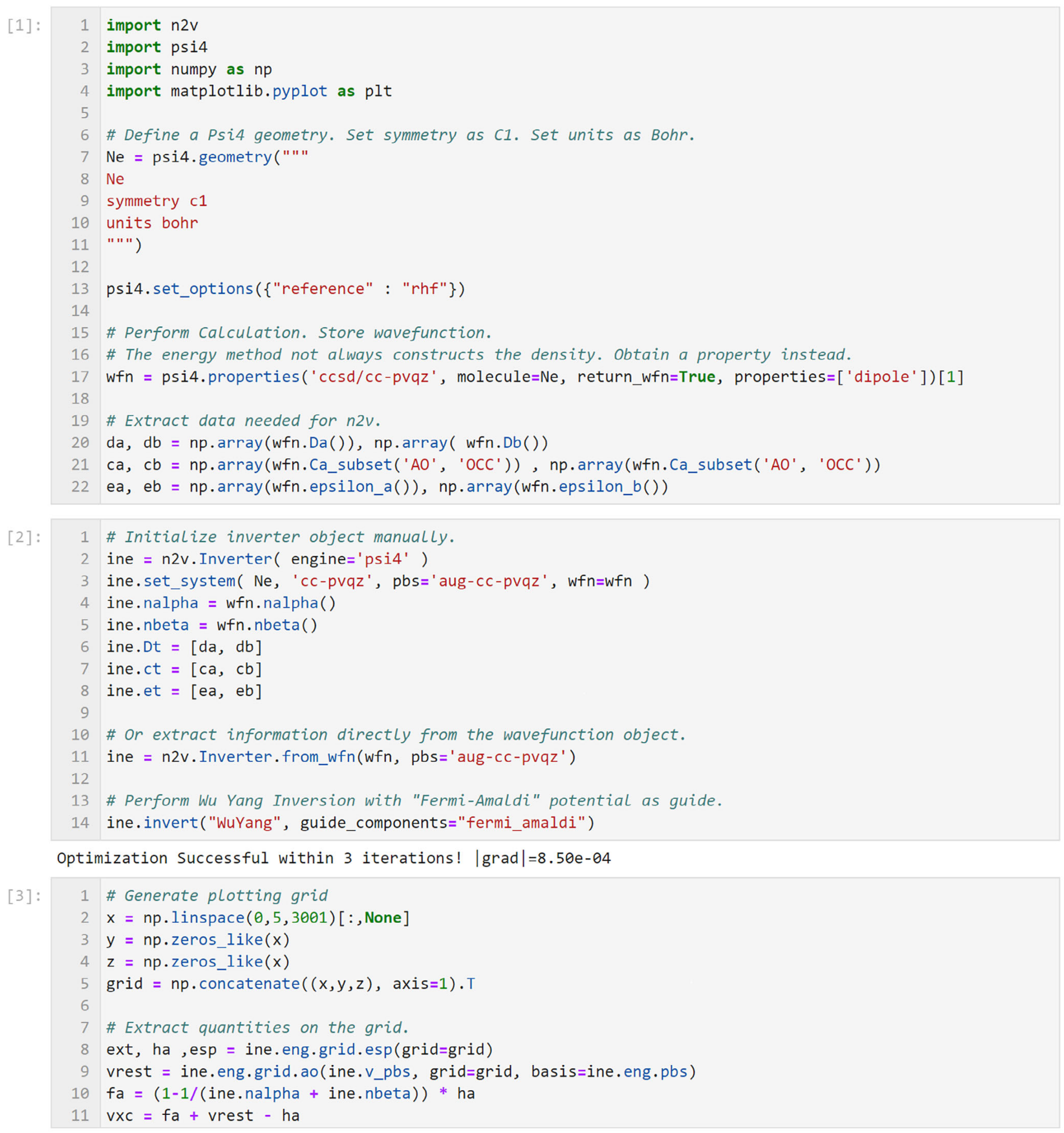
3.3 | Using *n2v*

To illustrate how to perform a calculation using *n2v*, we present a minimal example of a Neon atom using the Wu–Yang inversion method (See Figure 1) using the *Psi4* engine, for a comparison with the *PySCF* engine, please look at Appendix S1. The example below is general enough to provide users with an idea of how to use any of the methods discussed in this paper. This example was run in a Jupyter Notebook and can be accessed through the link: github.com/wasserman-group/n2v_examples. To better describe its usage, the calculation is separated into three groups highlighted by each cell in the Jupyter Notebook. These correspond to (1) the calculation of target density using *Psi4*, (2) defining an `Inverter` object and the inversion itself, and (3) the generation of the potential on the grid handled by the `Grider` and additional data (Figure 1).

1. *Calculation of target density*. We define the geometry as a standard Psi4 geometry. Units should be set to atomic units and symmetry should be set to C1. We are currently working on exploiting the irreducible representations that would allow different symmetries to be used. Our code takes the reference used—restricted or unrestricted—from the Psi4 set option. Except for the mRKS method, every other method is available in the unrestricted scheme. In this example, we obtain the target density using the coupled-cluster with singles and doubles determinants (CCSD) method. Psi4 does not compute the CCSD density from a simple energy calculation. This means that we need to generate a property such as the "Dipole." Other SCF methods will not depend on this step but users should refer to the documentation of each engine for more information.
2. *Inverting the density*. Initialize an `Inverter` object by selecting an engine as well as setting the system and the target components. Alternatively, one can simply initialize it by providing a `wfn` object from the calculation. If no additional basis-set is given, the basis set used will be the same as the energy calculation. Here we specified a larger basis set to be exclusively used for to express the inverted potential. To perform the inversion, we use the method "invert" that requires as basic arguments the "method," followed by the "guide_components" that allows to select a starting point for the inversion. In the example, we use the "Fermi–Amaldi potential" that ensures that the resulting potential has the correct decay as "*r*" tends to infinity.
3. *Visualizing the resulting potential*. To visualize the inverted potential, we require to prepare all quantities on the grid. In this case, we are interested on visualizing a component of the Kohn–Sham effective potential, the exchange–correlation potential. We build a grid by using the `linspace` function in *numpy*. Then we make use of the `esp` function of the `Grider` to calculate the Hartree and Fermi–Amaldi potentials, followed by the actual potential inverted (*rest*). This is necessary since the optimizer did not have to invert the full potential, but only a part of it. By adding all of the components accordingly one can visualize the potential. It should be noted that not every method generates the same output; users should refer to the documentation of each method to ensure that they are visualizing the appropriate potential.

4 | ILLUSTRATIVE RESULTS COMPARING INVERSE METHODS IN *n2v*

We begin by comparing the XC potential of stretched H_2 obtained through iKS from an accurate density. The *exact* XC potential famously features a spike in between the two nuclei.[4,89–91] This spike is a signature of strong static correlation and is missed by most approximate XC functionals in the fKS problem. When using an input density calculated through full configuration interaction on a Universal Gaussian Basis Set (UGBS), Figure 2 shows that four out of the five iKS methods attempt to explore this spike, but its shape and height can differ significantly. Furthermore, the result is



[1]:

```python
import n2v
import psi4
import numpy as np
import matplotlib.pyplot as plt

# Define a Psi4 geometry. Set symmetry as C1. Set units as Bohr.
Ne = psi4.geometry("""
Ne
symmetry c1
units bohr
""")

psi4.set_options({"reference" : "rhf"})

# Perform Calculation. Store wavefunction.
# The energy method not always constructs the density. Obtain a property instead.
wfn = psi4.properties('ccsd/cc-pvqz', molecule=Ne, return_wfn=True, properties=['dipole'])[1]

# Extract data needed for n2v.
da, db = np.array(wfn.Da()), np.array( wfn.Db())
ca, cb = np.array(wfn.Ca_subset('AO', 'OCC')) , np.array(wfn.Ca_subset('AO', 'OCC'))
ea, eb = np.array(wfn.epsilon_a()), np.array(wfn.epsilon_b())
```

[2]:

```python
# Initialize inverter object manually.
ine = n2v.Inverter( engine='psi4' )
ine.set_system( Ne, 'cc-pvqz', pbs='aug-cc-pvqz', wfn=wfn )
ine.nalpha = wfn.nalpha()
ine.nbeta = wfn.nbeta()
ine.Dt = [da, db]
ine.ct = [ca, cb]
ine.et = [ea, eb]

# Or extract information directly from the wavefunction object.
ine = n2v.Inverter.from_wfn(wfn, pbs='aug-cc-pvqz')

# Perform Wu Yang Inversion with "Fermi-Amaldi" potential as guide.
ine.invert("WuYang", guide_components="fermi_amaldi")
```

```
Optimization Successful within 3 iterations! |grad|=8.50e-04
```

[3]:

```python
# Generate plotting grid
x = np.linspace(0,5,3001)[:,None]
y = np.zeros_like(x)
z = np.zeros_like(x)
grid = np.concatenate((x,y,z), axis=1).T

# Extract quantities on the grid.
ext, ha ,esp = ine.eng.grid.esp(grid=grid)
vrest = ine.eng.grid.ao(ine.v_pbs, grid=grid, basis=ine.eng.pbs)
fa = (1-1/(ine.nalpha + ine.nbeta)) * ha
vxc = fa + vrest - ha
```

FIGURE 1 The input as a Jupyter Notebook for calculating the $v_{xc}(\mathbf{r})$ for the Neon atom with a CCSD target density. CCSD, coupled-cluster with singles and doubles determinants method

strongly dependent on the basis set choice: The reader can try *n2v* and see that when UGBS is replaced by a basis even as large as cc-pCVQZ, the spike disappears from both WY and PDE-CO inversions (but not from OC or mRKS). Next, we compare results of five methods available in *n2v* on systems with 10 to almost 100 electrons (the largest system reported by now but not yet the limit for *n2v*), both spin-polarized and spin-unpolarized. We report results for the Ne atom, the NO molecule and the caffeine molecule. Calculations are done on the cc-pCVQZ basis set unless otherwise stated. Exact results are available for the first two molecules, and we use two metrics to assess "quality" and two metrics to assess "efficiency." The quality metrics are the HOMO eigenvalue $\epsilon_{\mathrm{HOMO}} = -I$ and the density errors measured as:

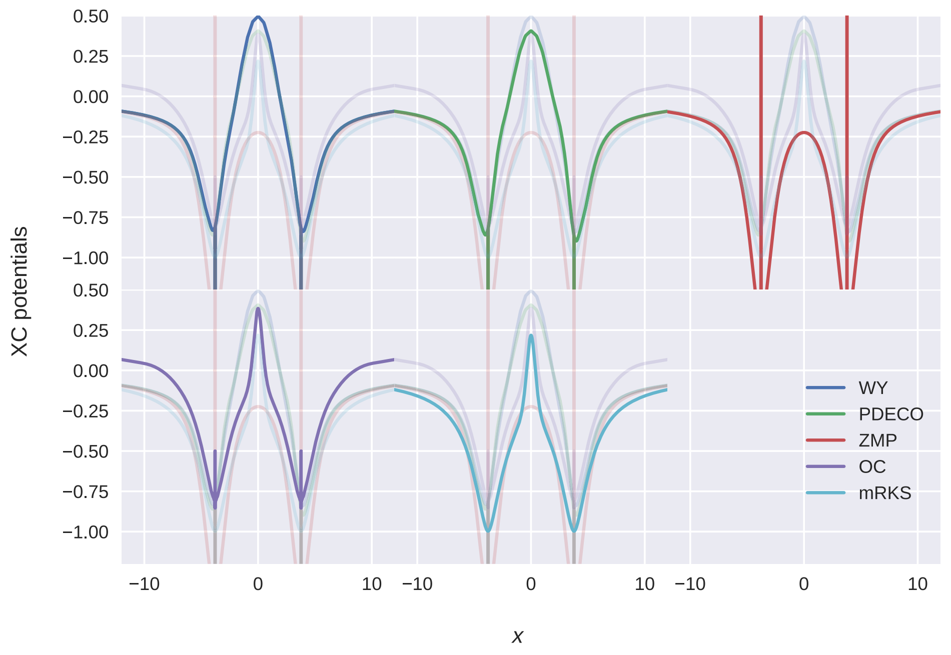


FIGURE 2 The exchange–correlation (XC) potentials of stretched H_2 from five methods inverted from the full configuration interaction density on Universal Gaussian Basis Set. In each subplot, one method is highlighted and compared with others (transparent)

TABLE 1 Comparison for different methods

Ne						
Method	N_{err}	ϵ_{HOMO}[a]	**Peak memory**[b]	**Time**[c]		
WY	1.6E−2	−0.7683	7E2	8E−1	5E−2	2E3[d]
PDE-CO	1.6E−2	−0.7557	9E2	2E2	3E−1	2E3[d]
ZMP	2.3E−2	−0.7961	5E2	5E−1	2E3[e]	
OC	3.4E−2	−0.7586	1E3	3E3		
mRKS	1.7E−2	−0.7823	6E3	2E5		

NO[f]				
Method	N_{err} **FA**	N_{err} **H**	ϵ_{HOMO}[g] **FA**	ϵ_{HOMO}[g] **H**
WY	7.8E−3	9.3E−3	−0.3392	−0.1042
PDE-CO	2.1E−2	2.2E−2	−0.3123	−0.0702
ZMP	1.5E−2	1.6E−2	−0.3248	−0.0696
OC	7.9E−2		−0.3348	

[a]Experimental ionization energy $I = -\epsilon_{\mathrm{HOMO}} = 0.7925$.[92]
[b]Million bytes.
[c]Seconds.
[d]Time: regularization parameter search, main calculation on a basis set, produce potential on the grid.
[e]Time: main calculation, produce potential on the grid.
[f]FA, Fermi–Amaldi; H, Hartree.
[g]Experimental ionization energy $I = -\epsilon_{\mathrm{HOMO}} = 0.3405$.[93]

$$N_{err} = \left(\int d\mathbf{r} |n(\mathbf{r}) - n_{\mathrm{in}}(\mathbf{r})|^2 \right)^{\frac{1}{2}}. \tag{24}$$

We also report the peak memory and run time on a single thread as the efficiency metrics. Note that the run time can vary with multi-threading, CPU clock speeds and many other factors. Also, the peak memory can vary with different numerical details, such as the data structure used and the grid size. Memory and time are only given to the order of

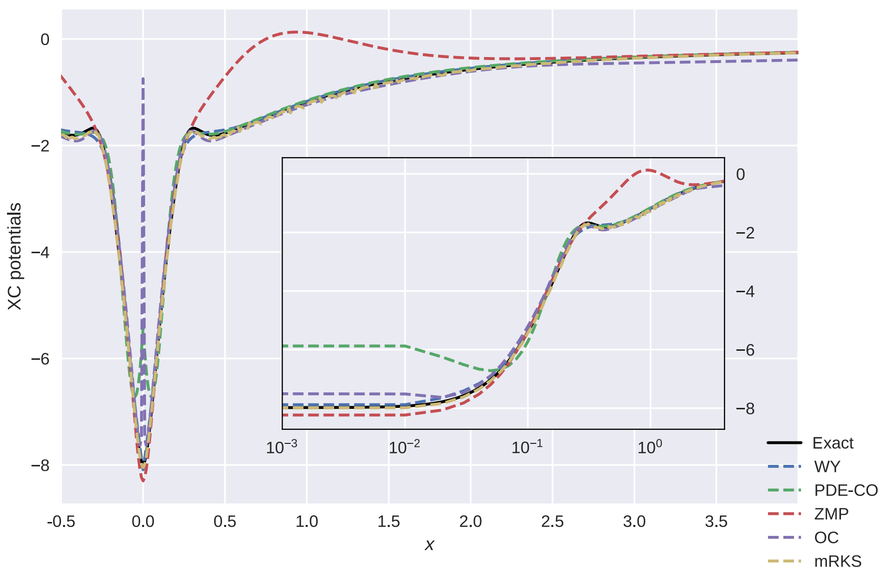


FIGURE 3 Ne atom exchange–correlation (XC) potentials inverted by different methods from configuration interaction singles and doubles densities. The exact one is from the quantum Monte Carlo calculations[94,95]

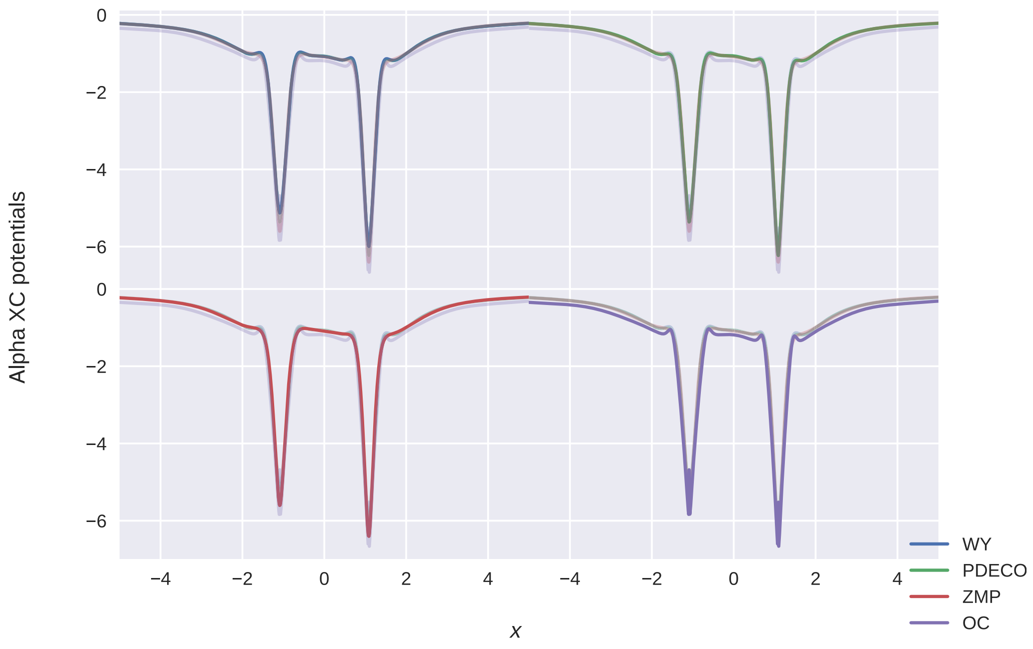


FIGURE 4 NO molecule spin-up exchange–correlation (XC) potentials

magnitude for comparison. Nevertheless, these two factors do illustrate the capability, limitation, difference and general trends for each method. Fermi–Amaldi guide potentials are used for WY, PDE-CO, and ZMP. Table 1 compares these methods for the Ne atom, and Figure 3 shows the corresponding XC potentials. One can see that WY, PDE-CO, and ZMP are much faster than OC and mRKS, even when regularization parameters are searched for. Most of the computational time of WY, PDE-CO, and ZMP is spent in basis transformation to a plotting grid of about 1,000,000 points, specifically the Hartree/Hartree-like potential by density fitting. Thus, a small plotting grid can save most of the time. As an example, a grid of 1000 points will take only about 1 sec. Moreover, for WY and PDE-CO, when the guide potential $v_0(\mathbf{r})$ in Equation (11) is the Hartree potential, $v_{\mathrm{PBS}}(\mathbf{r})$ can be identified directly with $v_{\mathrm{XC}}(\mathbf{r})$ and the Hartree potential on the grid is not needed. But this comes with a

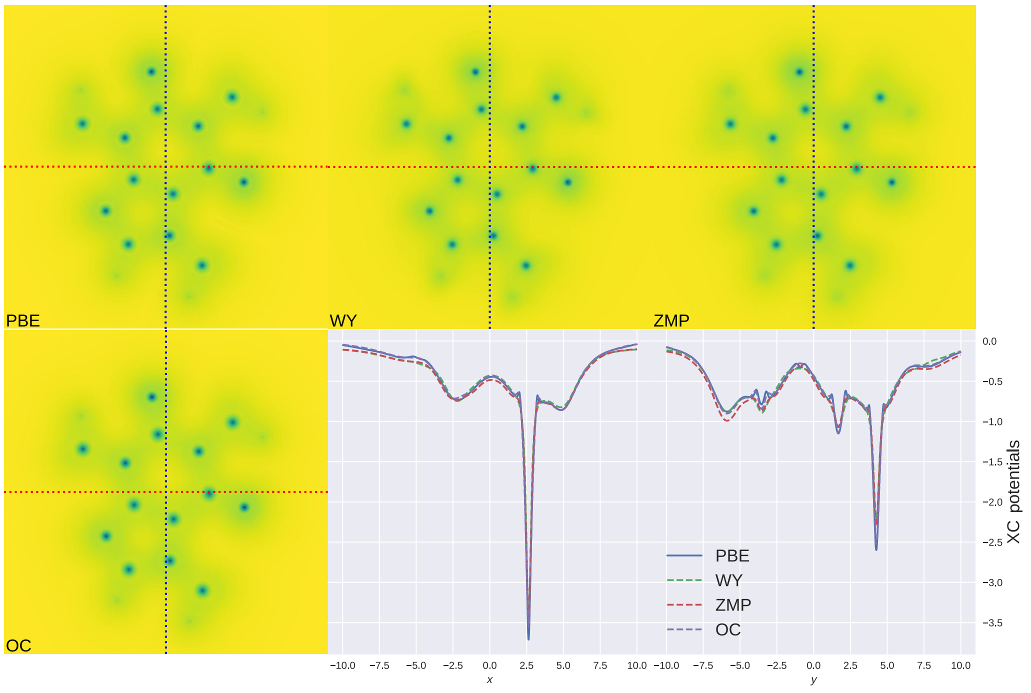


FIGURE 5 Caffeine molecule exchange–correlation (XC) potentials from WY, ZMP, and OC on the *xy*-plane. The last two panels show the potentials along the red and blue dotted lines, respectively. OC, Ou–Carter method; WY, Wu–Yang method; ZMP, Zhao–Morrison–Parr method

cost of accuracy.[45] For OC and mRKS, all the calculations are done on a Lebedev–Treutler grid (about 73,000 points in this case) and are therefore slower. In particular, mRKS needs a double integral for $v_{\mathrm{XC}}^{hole}(\mathbf{r})$ by grid quadratures, which makes it too slow for even medium-sized systems. We also compare spin-polarized cases. Figure 4 shows spin-up XC potentials for the NO molecule. One can again notice that, even though all the methods are capable to reproduce the bulk part of the potentials, small details vary. As mentioned in the introduction, this is largely (if not totally) due to the different numerical features of the methods, as opposed to fundamental differences (i.e., all methods would provide the same numerical XC potentials in the infinite basis-set limit). The spin-down XC potentials can be found in Appendix S1. In Figure 5, the direct PBE XC potential of the caffeine molecule is compared with XC potentials inverted from the PBE density by WY, ZMP, and OC, with $v_0(\mathbf{r})$ as Fermi–Amaldi. However, it should be noted that inverting from the PBE density is an easier job compared to inverting from a density that has been generated through a wavefunction-based method; the former is one of the "inverse crimes" discussed in references 45, 46. More details about the tests can be found in Appendix S1.

5 | CONCLUDING REMARK

The CMS community has built a vast ecosystem for computational chemistry thriving with users and developers who openly share their work. With the release of *n2v*, we are pleased to join a larger community of libraries that add useful functionalities to *Psi4* and *PySCF*.

We have compared the performance of various iKS methods implemented in *n2v* on a few carefully selected systems. For more details on the examples provided, we refer the reader to Appendix S1, which also has a short tutorial explaining how to run an iKS calculation in practice with *n2v*. For a more thorough discussion on the advantages and disadvantages of all the methods implemented into *n2v*, we refer the reader to reference 45.

ACKNOWLEDGMENT
Victor H. Chávez would like to acknowledge the amazing support and helpful conversations provided by Dr. Taylor A. Barnes of The Molecular Sciences Software Institute about the structure of the library.

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

AUTHOR CONTRIBUTIONS

Yuming Shi: Conceptualization (equal); formal analysis (equal); investigation (equal); methodology (equal); resources (equal); software (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal). **Victor H. Chávez:** Conceptualization (equal); formal analysis (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal). **Adam Wasserman:** Conceptualization (equal); formal analysis (equal); methodology (equal); project administration (equal); supervision (equal); writing – original draft (equal); writing – review and editing (equal).

DATA AVAILABILITY STATEMENT

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

ORCID

Victor H. Chávez https://orcid.org/0000-0003-3765-2961
Adam Wasserman https://orcid.org/0000-0002-8037-4453

RELATED WIREs ARTICLES

Subsystem density-functional theory
VeloxChem: A Python-driven density-functional theory program for spectroscopy simulations in high-performance computing environments
DFTpy: An efficient and object-oriented platform for orbital-free DFT simulations

REFERENCES

1. Kohn W, Sham LJ. Self-consistent equations including exchange and correlation effects. Phys Rev. 1965;140:A1133.
2. Perdew JP, Ruzsinszky A, Tao J, Staroverov VN, Scuseria GE, Csonka GI. Prescription for the design and selection of density functional approximations: more constraint satisfaction with fewer fits. J Chem Phys. 2005;123:062201.
3. Verma P, Truhlar DG. Status and challenges of density functional theory. Trends Chem. 2020;2:302–18.
4. Helbig N, Tokatly IV, Rubio A. Exact Kohn–Sham potential of strongly correlated finite systems. J Chem Phys. 2009;131:224105–8.
5. Hodgson MJ, Kraisler E, Schild A, Gross EK. How interatomic steps in the exact Kohn–Sham potential relate to derivative discontinuities of the energy. J Phys Chem Lett. 2017;8:5974–80.
6. Thiele M, Kummel S. Photoabsorption spectra from adiabatically exact time-dependent density-functional theory in real time. Phys Chem Chem Phys. 2009;11:4631–9.
7. Fuks JI, Nielsen SEB, Ruggenthaler M, Maitra NT. Time-dependent density functional theory beyond Kohn–Sham slater determinants. Phys Chem Chem Phys. 2016;18:20976–85.
8. Nam S, Song S, Sim E, Burke K. Measuring density-driven errors using Kohn–Sham inversion. J Chem Theory Comput. 2020;16: 5014–23.
9. Li L, Hoyer S, Pederson R, Sun R, Cubuk ED, Riley P, et al. Kohn–Sham equations as regularizer: building prior knowledge into machine-learned physics. Phys Rev Lett. 2021;126:036401.
10. Naito T, Ohashi D, Liang H. Improvement of functionals in density functional theory by the inverse Kohn–Sham method and density functional perturbation theory. J Phys B. 2019;52:245003.
11. Accorto G, Brandolini P, Marino F, Porro A, Scalesi A, Colò G, et al. First step in the nuclear inverse Kohn–Sham problem: from densities to potentials. Phys Rev C. 2020;101:024315.
12. Accorto G, Naito T, Liang H, Niksic T, Vretenar D. Nuclear energy density functionals from empirical ground-state densities. Phys Rev C. 2021;103:044304.
13. Maitra NT, Zhang F, Cave RJ, Burke K. Double excitations within time-dependent density functional theory linear response. J Chem Phys. 2004;120:5932–7.
14. Maitra NT. Perspective: fundamental aspects of time-dependent density functional theory. J Chem Phys. 2016;144:220901–16.
15. Ramsden J, Godby R. Exact density-functional potentials for time-dependent quasiparticles. Phys Rev Lett. 2012;109:036402.
16. Mancini L, Ramsden J, Hodgson M, Godby R. Adiabatic and local approximations for the Kohn–Sham potential in time-dependent Hubbard chains. Phys Rev B. 2014;89:195114.
17. Kanungo B, Zimmerman PM, Gavini V. A comparison of exact and model exchange–correlation potentials for molecules. J Phys Chem Lett. 2021;12:12012–9.

18. Roncero O, de Lara-Castells M, Villarreal P, Flores F, Ortega J, Paniagua M, et al. An inversion technique for the calculation of embedding potentials. J Chem Phys. 2008;129:184104.
19. Fux S, Jacob CR, Neugebauer J, Visscher L, Reiher M. Accurate frozen-density embedding potentials as a first step towards a subsystem description of covalent bonds. J Chem Phys. 2010;132:164101.
20. Goodpaster JD, Ananth N, Manby FR, Miller TF. Exact nonadditive kinetic potentials for embedded density functional theory. J Chem Phys. 2010;133:084103–084103-10.
21. Nafziger J, Wu Q, Wasserman A. Molecular binding energies from partition density functional theory. J Chem Phys. 2011;135:234101–6.
22. Huang C, Pavone M, Carter EA. Quantum mechanical embedding theory based on a unique embedding potential. J Chem Phys. 2011; 134:154110.
23. Oueis Y, Wasserman A. Exact partition potential for model systems of interacting electrons in 1-D. Eur Phys J B. 2018;91:1–9.
24. Chávez VH, Wasserman A. Towards a density functional theory of molecular fragments. What is the shape of atoms in molecules? Rev Acad Colomb Cienc Exactas Fis Nat. 2020;44:269–79.
25. Hohenberg P, Kohn W. Inhomogeneous electron gas. Phys Rev. 1964;136:B864.
26. Runge E, Gross EK. Density-functional theory for time-dependent systems. Phys Rev Lett. 1984;52:997.
27. Aryasetiawan F, Stott M. Effective potentials in density-functional theory. Phys Rev B. 1988;38:2974.
28. Görling A. Kohn–Sham potentials and wave functions from electron densities. Phys Rev A. 1992;46:3753.
29. Zhao Q, Parr RG. Quantities T s [n] and T c [n] in density-functional theory. Phys Rev A. 1992;46:2337.
30. Zhao Q, Parr RG. Constrained-search method to determine electronic wave functions from electronic densities. J Chem Phys. 1993;98: 543–8.
31. Zhao Q, Morrison RC, Parr RG. From electron densities to Kohn–Sham kinetic energies, orbital energies, exchange–correlation potentials, and exchange–correlation energies. Phys Rev A. 1994;50:2138.
32. Wang Y, Parr RG. Construction of exact Kohn–Sham orbitals from a given electron density. Phys Rev A. 1993;47:R1591.
33. Peirs K, Van Neck D, Waroquier M. Algorithm to derive exact exchange–correlation potentials from correlated densities in atoms. Phys Rev A. 2003;67:012505.
34. Kadantsev ES, Stott M. Variational method for inverting the Kohn–Sham procedure. Phys Rev A. 2004;69:012502.
35. Nafziger J, Jiang K, Wasserman A. Accurate reference data for the nonadditive, noninteracting kinetic energy in covalent bonds. J Chem Theory Comput. 2017;13:577–86.
36. Wagner LO, Baker TE, Stoudenmire E, Burke K, White SR. Kohn–Sham calculations with the exact functional. Phys Rev B. 2014;90: 045109.
37. Ou Q, Carter EA. Potential functional embedding theory with an improved Kohn–Sham inversion algorithm. J Chem Theory Comput. 2018;14:5680–9.
38. Kumar A, Harbola MK. Using random numbers to obtain Kohn–Sham potential for a given density. Chem Phys Lett. 2021;779:138851.
39. Callow TJ, Lathiotakis NN, Gidopoulos NI. Density-inversion method for the Kohn–Sham potential: role of the screening density. J Chem Phys. 2020;152:164114.
40. Wu Q, Yang W. A direct optimization method for calculating density functionals and exchange–correlation potentials from electron densities. J Chem Phys. 2003;118:2498–509.
41. Kananenka AA, Kohut SV, Gaiduk AP, Ryabinkin IG, Staroverov VN. Efficient construction of exchange and correlation potentials by inverting the Kohn–Sham equations. J Chem Phys. 2013;139:074112.
42. Ryabinkin IG, Kohut SV, Staroverov VN. Reduction of electronic wave functions to Kohn–Sham effective potentials. Phys Rev Lett. 2015;115:083001.
43. Ospadov E, Ryabinkin IG, Staroverov VN. Improved method for generating exchange–correlation potentials from electronic wave functions. J Chem Phys. 2017;146:084103.
44. Staroverov VN, Ospadov E. Unified construction of Fermi, Pauli and exchange–correlation potentials. Adv Quantum Chem. 2019;79: 201–19.
45. Shi Y, Wasserman A. Inverse Kohn–Sham density functional theory: progress and challenges. J Phys Chem Lett. 2021;12:5308–18.
46. Jensen DS, Wasserman A. Numerical methods for the inverse problem of density functional theory. Int J Quantum Chem. 2018;118: e25425.
47. Jacob CR. Unambiguous optimization of effective potentials in finite basis sets. J Chem Phys. 2011;135:244102.
48. Gaiduk AP, Ryabinkin IG, Staroverov VN. Removal of basis-set artifacts in Kohn–Sham potentials recovered from electron densities. J Chem Theory Comput. 2013;9:3959–64.
49. Mura ME, Knowles PJ, Reynolds CA. Accurate numerical determination of Kohn–Sham potentials from electronic densities: I. Two-electron systems. J Chem Phys. 1997;106:9659–67.
50. Schipper P, Gritsenko O, Baerends E. Kohn–Sham potentials corresponding to Slater and Gaussian basis set densities. Theor Chem Acc. 1997;98:16–24.
51. Staroverov VN, Scuseria GE, Davidson ER. Optimized effective potentials yielding Hartree–Fock energies and densities. College Park, MD: American Institute of Physics; 2006.
52. Chayes J, Chayes L, Ruskai MB. Density functional approach to quantum lattice systems. J Stat Phys. 1985;38:497–518.
53. Kanungo B, Zimmerman PM, Gavini V. Exact exchange–correlation potentials from ground-state electron densities. Nat Commun. 2019;10:1–9.

54. Kumar A, Singh R, Harbola MK. Accurate effective potential for density amplitude and the corresponding Kohn–Sham exchange–correlation potential calculated from approximate wavefunctions. J Phys B. 2020;53:165002.
55. Kumar A, Singh R, Harbola MK. Universal nature of different methods of obtaining the exact Kohn–Sham exchange–correlation potential for a given density. J Phys B. 2019;52:075007.
56. Kumar A, Harbola MK. A general penalty method for density-to-potential inversion. Int J Quantum Chem. 2020;120:e26400.
57. Schnieders D, Neugebauer J. Accurate embedding through potential reconstruction: a comparison of different strategies. J Chem Phys. 2018;149:054103.
58. Banafsheh M, Adam WT. Nonadditive kinetic potentials from inverted Kohn–Sham problem. Int J Quantum Chem. 2018;118:e25410.
59. Unsleber JP, Dresselhaus T, Klahr K, Schnieders D, Böckers M, Barton D, et al. Serenity: a subsystem quantum chemistry program. J Comput Chem. 2018;39:788–98.
60. Nam S, McCarty RJ, Park H, Sim E. KS-pies: Kohn–Sham inversion toolkit. J Chem Phys. 2021;154:124122.
61. Van Leeuwen R, Baerends E. Exchange–correlation potential with correct asymptotic behavior. Phys Rev A. 1994;49:2421.
62. Gritsenko OV, van Leeuwen R, Baerends EJ. Molecular Kohn–Sham exchange–correlation potential from the correlated ab initio electron density. Phys Rev A. 1995;52:1870.
63. Ayers PW, Morrison RC, Parr RG. Fermi–Amaldi model for exchange–correlation: atomic excitation energies from orbital energy differences. Mol Phys. 2005;103:2061–72.
64. Morrison RC, Zhao Q. Solution to the Kohn–Sham equations using reference densities from accurate, correlated wave functions for the neutral atoms helium through argon. Phys Rev A. 1995;51:1980.
65. Smith DG, Burns LA, Sirianni DA, Nascimento DR, Kumar A, James AM, et al. Psi4NumPy: an interactive quantum chemistry programming environment for reference implementations and rapid development. J Chem Theory Comput. 2018;14:3504–11.
66. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020; 585:357–62.
67. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17:261–72.
68. Daniel G, Gray J. Opt_einsum—a Python package for optimizing contraction order for einsum-like expressions. J Open Source Softw. 2018;3:753.
69. Lehtola S, Steigemann C, Oliveira MJ, Marques MA. Recent developments in libxc—a comprehensive library of functionals for density functional theory. SoftwareX. 2018;7:1–5.
70. Hunter JD. Matplotlib: a 2D graphics environment. IEEE Ann Hist Comput. 2007;9:90–5.
71. Lehmann R. The sphinx project. Universität Potsdam, Project Documentation; 2011.
72. Nash J, Altarawy D, Barnes T, Ellis S, Marin Rimoldi E, Pritchard B, et al. Best practices in Python package development; 2018 [cited March 31, 2022]. Available from: http://education.molssi.org/python-package-best-practices/
73. Krylov A, Windus TL, Barnes T, Marin-Rimoldi E, Nash JA, Pritchard B, et al. Perspective: computational chemistry software and its advancement as illustrated through three grand challenge cases for molecular science. J Chem Phys. 2018;149:180901.
74. Barnes TA, Marin-Rimoldi E, Ellis S, Crawford TD. The MolSSI driver interface project: a framework for standardized, on-the-fly interoperability between computational molecular sciences codes. Comput Phys Commun. 2021;261:107688.
75. Smith DG, Lolinco AT, Glick ZL, Lee J, Alenaizan A, Barnes TA, et al. Quantum chemistry common driver and databases (QCDB) and quantum chemistry engine (QCEngine): automation and interoperability among computational chemistry programs. J Chem Phys. 2021;155:204801.
76. Larsen AH, Mortensen JJ, Blomqvist J, Castelli IE, Christensen R, Dułak M, et al. The atomic simulation environment—a Python library for working with atoms. J Phys Condens Matter. 2017;29:273002.
77. Shao X, Jiang K, Mi W, Genova A, Pavanello M. DFTpy: an efficient and object-oriented platform for orbital-free DFT simulations. WIREs Comput Mol Sci. 2021;11:e1482.
78. Verstraelen T, Adams W, Pujal L, Tehrani A, Kelly BD, Macaya L, et al. IOData: a Python library for reading, writing, and converting computational chemistry file formats and generating input files. J Comput Chem. 2021;42:458–64.
79. Borca CH, Bakr BW, Burns LA, Sherrill CD. CrystaLattE: automated computation of lattice energies of organic crystals exploiting the many-body expansion to achieve dual-level parallelism. J Chem Phys. 2019;151:144103.
80. Poole D, Galvez Vallejo JL, Gordon MS. A new kid on the block: application of Julia to Hartree–Fock calculations. J Chem Theory Comput. 2020;16:5006–13.
81. Herbst MF, Levitt A, Cancès E. DFTK: a Julian approach for simulating electrons in solids. Proceedings of the JuliaCon Conferences. Volume 3. The Open Journal; 2021. p. 69.
82. Magers DB, Chávez VH, Peyton BG, Sirianni DA, Fortenberry RC, Ringer MDA. PSI4EDUCATION: free and open-source programing activities for chemical education with free and open-source software. Teaching programming across the chemistry curriculum. Washington, DC: ACS Publications; 2021. p. 107–22.
83. Govoni M, Whitmer J, de Pablo J, Gygi F, Galli G. Code interoperability extends the scope of quantum simulations. npj Comput Mater. 2021;7:1–10.
84. Smith DG, Burns LA, Simmonett AC, Parrish RM, Schieber MC, Galvelis R, et al. PSI4 1.4: open-source software for high-throughput quantum chemistry. J Chem Phys. 2020;152:184108.
85. Turney JM, Simmonett AC, Parrish RM, Hohenstein EG, Evangelista FA, Fermann JT, et al. Psi4: an open-source ab initio electronic structure program. WIREs Comput Mol Sci. 2012;2:556–65.

86. Parrish RM, Burns LA, Smith DG, Simmonett AC, DePrince AE III, Hohenstein EG, et al. Psi4 1.1: an open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. J Chem Theory Comput. 2017;13:3185–97.
87. Sun Q, Berkelbach TC, Blunt NS, Booth GH, Guo S, Li Z, et al. PySCF: the Python-based simulations of chemistry framework. WIREs Comput Mol Sci. 2018;8:e1340.
88. Sun Q, Zhang X, Banerjee S, Bao P, Barbry M, Blunt NS, et al. Recent developments in the PySCF program package. J Chem Phys. 2020;153:024109.
89. Buijse MA, Baerends EJ, Snijders JG. Analysis of correlation in terms of exact local potentials: applications to two-electron systems. Phys Rev A. 1989;40:4190.
90. Tempel DG, Martinez TJ, Maitra NT. Revisiting molecular dissociation in density functional theory: a simple model. J Chem Theory Comput. 2009;5:770–80.
91. Nafziger J, Wasserman A. Fragment-based treatment of delocalization and static correlation errors in density-functional theory. J Chem Phys. 2015;143:234105.
92. West RC, Astle MJ. CRC handbook of chemistry and physics. Boca Raton, FL: CRC Process; 1979.p. D-71.
93. Reiser G, Habenicht W, Müller-Dethlefs K, Schlag EW. The ionization energy of nitric oxide. Chem Phys Lett. 1988;152:119–23.
94. Umrigar CJ, Gonze X. Accurate exchange–correlation potentials and total-energy components for the helium isoelectronic series. Phys Rev A. 1994;50:3827.
95. Filippi C, Gonze X, Umrigar C. Recent developments and applications of modern density functional theory. Amsterdam: Elsevier; 1996.

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

How to cite this article: Shi Y, Chávez VH, Wasserman A. *n2v*: A density-to-potential inversion suite. A sandbox for creating, testing, and benchmarking density functional theory inversion methods. WIREs Comput Mol Sci. 2022. e1617. https://doi.org/10.1002/wcms.1617