Computer Physics Communications 279 (2022) 108424

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

KSSOLV 2.0: An efficient MATLAB toolbox for solving the Kohn-Sham equations with plane-wave basis set ☆,☆☆

Shizhe Jiao [a], Zhenlin Zhang [a], Kai Wu [a], Lingyun Wan [a], Huanhuan Ma [a], Jielan Li [a], Sheng Chen [a], Xinming Qin [a], Jie Liu [a], Zijing Ding [a], Jinlong Yang [a], Yingzhou Li [b,*], Wei Hu [a,*], Lin Lin [c,d,**], Chao Yang [d,*]

[a] School of Future Technology, Department of Chemical Physics, and Anhui Center for Applied Mathematics, University of Science and Technology of China, Jinzhai Road No.96, Hefei, 230026, PR China
[b] School of Mathematical Sciences, Fudan University, Handan Road No.220, Shanghai, 200433, PR China
[c] Department of Mathematics, University of California, Berkeley, CA 94720, USA
[d] Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Received 5 March 2022
Received in revised form 10 May 2022
Accepted 17 May 2022
Available online 2 June 2022

Keywords:
Kohn-Sham solver
MATLAB
Plane-wave basis set
Density functional theory
Numerical algorithms

ABSTRACT

KSSOLV (Kohn-Sham Solver) is a MATLAB toolbox for performing Kohn-Sham density functional theory (DFT) calculations with a plane-wave basis set. KSSOLV 2.0 preserves the design features of the original KSSOLV software to allow users and developers to easily set up a problem and perform ground-state calculations as well as to prototype and test new algorithms. Furthermore, it includes new functionalities such as new iterative diagonalization algorithms, k-point sampling for electron band structures, geometry optimization and advanced algorithms for performing DFT calculations with local, semi-local, and hybrid exchange-correlation functionals. It can be used to study the electronic structures of both molecules and solids. We describe these new capabilities in this work through a few use cases. We also demonstrate the numerical accuracy and computational efficiency of KSSOLV on a variety of examples.

Program summary

Program title: Kohn-Sham Solver 2.0 (KSSOLV 2.0)
CPC Library link to program files: https://doi.org/10.17632/pp8vgvfcv4.1
Developer's repository link: https://bitbucket.org/berkeleylab/kssolv2.0/src/release/
Licensing provisions: BSD 3-clause
Programming language:: MATLAB
Nature of problem: KSSOLV2.0 is used to perform Kohn-Sham density functional theory based electronic structure calculations to study chemical and material properties of molecules and solids. The key problem to be solved is a constrained energy minimization problem, which can also be formulated as a nonlinear eigenvalue problem.
Solution method: The KSSOLV 2.0 implements both the self-consistent field (SCF) iteration with a variety of acceleration strategies and a direct constrained minimization algorithms. It is written completely in MATLAB and uses MATLAB's object oriented programming features to make it easy to use and modify.

1. Introduction

KSSOLV (Kohn-Sham Solver) [1] is a MATLAB (Matrix Laboratory) toolbox for performing Kohn-sham density functional theory (DFT) [2,3] based electronic structure calculations. It uses the plane-wave basis set to represent electron wavefunctions. One of the original motivations for developing such a software package was to make it easy to prototype and test new algorithms for solving the Kohn-Sham nonlinear eigenvalue problems. KSSOLV leverages the high quality numerical linear algebra

☆ The review of this paper was arranged by Prof. Blum Volker.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding authors.
** Corresponding author at: Department of Mathematics, University of California, Berkeley, CA 94720, USA.
E-mail addresses: yingzhouli@fudan.edu.cn (Y. Li), whuustc@ustc.edu.cn (W. Hu), linlin@math.berkeley.edu (L. Lin), chaoyang@lbl.gov (C. Yang).

https://doi.org/10.1016/j.cpc.2022.108424
0010-4655/

functions and object-oriented features of MATLAB to enable researchers who have minimal knowledge of other electronic structure calculation software written in FORTRAN or C/C++ to quickly modify existing algorithms, as well as to develop and test new ideas. Over the last decade, KSSOLV has become a useful research and teaching tool for studying electronic structures of molecules and solids and developing new methods for solving the Kohn-Sham problem, as evidenced by an increasing number of publications that use KSSOLV to perform numerical experiments required to demonstrate improved convergence or better accuracy of new theoretical methods and numerical algorithms. Examples of such developments include tensor hypercontraction algorithm [4], improved generalized Davidson algorithm [5], elliptic preconditioner in self-consistent field iteration [6], low rank approximation in G_0W_0 calculations [7], perturbation theory [8], quantum embedding theory [9], quantum computation [10], linear-response time-dependent density functional theory [11,12], phonon calculations [13], proximal gradient method [14], trace-penalty minimization method [15].

As KSSOLV becomes more widely used, it also becomes clear that the functionalities supported in the original KSSOLV software package are insufficient. For example, the original KSSOLV could only be used to perform single-point calculations of the ground state energies for molecules placed in a large supercell, and only the local density approximation (LDA) exchange-correlation functional was implemented. Moreover, the pseudopotentials supported in the original KSSOLV did not clearly specify the type of pseudopotentials used and did not allow widely accepted pseudopotential libraries to be easily incorporated. The lack of these functionalities makes the comparison of KSSOLV with other software packages somewhat difficult.

To address these issues, we have recently revamped the development of KSSOLV by adopting more standard pseudopotential types, such as the ONCV (Optimized Norm-Conserving Vanderbilt) [16] and Hartwigsen-Goedecker-Hutter (HGH) [17] pseudopotentials. We have incorporated more recent algorithmic development, and added new functionalities and features without sacrificing the usability of the software. The new software release, KSSOLV 2.0, is an open source software.[1] In addition to being a flexible tool for new algorithm development, KSSOLV 2.0 can also be easily used to study the properties of molecules and solids. It serves as both a research and teaching tool for researchers engaged in the simulation and prediction of chemical and material properties, such as linear-response time-dependent density functional theory [11], many-electron self energy calculations [18], structure optimization [19], photocatalytic materials simulations [20].

We strive to make KSSOLV 2.0 as efficient as possible without sacrificing its readability and usability. In addition to traditional platforms, the software can also be run on heterogeneous architectures with graphics processing units (GPUs) [21]. However, KSSOLV 2.0 is not designed for performing large-scale electronic structure calculations. For these types of calculations, many existing software tools can be used alternatively, such as Gaussian [22], NWChem (NorthWest computational Chemistry) [23], Q-CHEM [24], BDF (Beijing Density Functional program package) [25], and PySCF (Python-based Simulations of Chemistry Framework) [26] within Gaussian-type orbital (GTO) basis set; SIESTA (Spanish Initiative for Electronic Simulations with Thousands of Atoms) [27], HONPAS (Hefei Order-N Packages for Ab initio Simulations) [28–30], FHI-aims (Fritz Haber Institute ab initio molecular simulations) [31] and ABACUS (Atomic-orbital Based Ab-initio Computation at Ustc) [32] within numerical atomic orbital (NAO) basis set; and VASP (Vienna Ab initio Simulation Package) [33], ABINIT [34], QE (QUANTUM ESPRESSO) [35], PWmat [36], PWDFT (Plane-Wave Density Functional Theory) [37] within plane-wave basis set. These DFT codes are often written in languages such as FORTRAN and C++, and parallelized with OpenMP, MPI, and CUDA. The compilation, installation and usage of these software packages often take a significant amount of effort. Our software is more similar to some other recently developed DFT toolboxes such as GPAW (Grid-based Projector Augmented Wave) [38,39], M-SPARC (Matlab-Simulation Package for Ab-initio Real-space Calculations) [49] and PWDFT.jl [41], DFT.jl [42], which are based on higher-level scripting languages such as Python, Julia, and MATLAB. We should point out that the M-SPARC software, which is written in MATLAB, focuses on a real space discretization of the Kohn-Sham problem whereas KSSOLV uses a plane-wave discretization. The main characteristics of these software packages are shown in Table 1. The advantage of KSSOLV is that it is written completely in MATLAB, which is designed to perform linear algebra operations in a straightforward manner. MATLAB also provides an excellent Integrated Development Environment (IDE), which makes the development process much easier than other software tools. Furthermore, the unique profiling capability of MATLAB allows us to easily identify computational bottlenecks.

This work is organized as follows. In the next section, we briefly summarize the main methodology and standard methods implemented in KSSOLV for solving the Kohn-Sham DFT problem, as well as a number of recently developed and more advanced algorithms. We highlight the object-oriented design feature of KSSOLV in section 3, and demonstrate several main features of KSSOLV 2.0 through a number of use cases in section 4. The accuracy and efficiency of KSSOLV 2.0 are reported in section 5 for several small to medium sized benchmark test problems.

2. Methodology

KSSOLV 2.0 is designed to perform Kohn-Sham density functional theory (KS-DFT) based electronic structure calculations. In this section, we briefly describe the main mathematical problem to be solved, namely, the Kohn-Sham nonlinear eigenvalue problem, or equivalently, the Kohn-Sham total energy minimization problem. In KSSOLV 2.0, the eigenfunction to be computed is expanded in a plane-wave basis, which will be discussed briefly in section 2.1.2. A key component of the Kohn-Sham Hamiltonian operator is the exchange-correlation (XC) potential that accounts for many-body effects in a many-electron system. We describe the XC functions implemented in KSSOLV 2.0 in section 2.1.3. KSSOLV 2.0 employs the pseudopotential method which is commonly used to address the weak singularity (cusp) in the nuclei-electron potential. We briefly discuss pseudopotentials used in KSSOLV 2.0 in section 2.1.4. Sections 2.2.1 and 2.2.2 are concerned with several numerical algorithms used in KSSOLV 2.0 to solve the Kohn-Sham and related problems. In particular, we discuss new algorithms that have been added in the latest release of KSSOLV 2.0 in sections 2.2.3, 2.2.4 and 2.3.

2.1. Mathematical formulation

2.1.1. Brief introduction of KS-DFT

The KS-DFT [2,3] is the most widely used methodology to perform first-principles calculations and materials simulations to study the electronic structure of molecules and solids.

The key problem to be solved in KS-DFT based electronic structure calculation of an atomistic system with N_e electrons is a nonlinear eigenvalue problem of the form

$$\hat{H}(\rho)\psi_j = \varepsilon_j \psi_j, \tag{1}$$

[1] Bitbucket repository with documentation: https://bitbucket.org/berkeleylab/kssolv2.0/src/release/.

Table 1
The characteristics of several DFT software packages, including programming language, basis, license, language type and publish year. GTOs: gaussian-type orbital (GTO) basis set, NAOs: numerical atomic orbital basis set, PW: plane-wave basis set. AE: all electronic calculation, NCPPs: norm-conserving pseudopotentials, ECP: effective core potential, PAW: projector augmented wave. GPL: GNU General Public License, ECL-2.0: Educational Community License, BSD: Berkeley Software Distribution License.

Software	Language	Basis	AE/PSP	License	Language type	Year	Reference
Gaussian	Fortran	GTOs	AE/ECP	Commercial	Compiled language	1970	[22]
NWChem	Fortran	GTOs/PW	AE/PAW	Free, ECL-2.0	Compiled language	1994	[23]
QChem	Fortran	GTOs	AE/ECP	Academic, commercial	Compiled language	1997	[24]
BDF	Fortran	GTOs	AE	Free, GPL	Compiled language	2009	[25]
SIESTA	Fortran	NAOs	NCPPs	Free, GPL	Compiled language	1996	[27]
HONPAS	Fortran	NAOs	NCPPs	Free, GPL	Compiled language	2005	[28]
FHI-aims	Fortran	NAOs	AE	Academic, commercial	Compiled language	2009	[31]
ABACUS	Fortran	NAOs/PW	NCPPs	Free, GPL	Compiled language	2016	[32]
VASP	Fortran	PW	PAW	Commercial	Compiled language	1989	[33]
ABINIT	Fortran	PW	NCPPs/PAW	Free, GPL	Compiled language	1998	[34]
QE	Fortran	PW	NCPPs/PAW	Free, GPL	Compiled language	2001	[35]
PWmat	Fortran	PW	NCPPs	Commercial	Compiled language	2013	[36]
PWDFT	C/C++	PW	NCPPs	Free, BSD	Compiled language	2017	[37]
GPAW	Python	PW	PAW	Free, GPL	Interpreted language	2003	[38]
KSSOLV	MATLAB	PW	NCPPs	Free, BSD	Interpreted language	2009	[1]; This work
PySCF	Python	GTOs	AE; NCPPs	Free, BSD	Interpreted language	2014	[26]
M-SPARC	MATLAB	RS	NCPPs	Free, GPL	Interpreted language	2019	[40]
PWDFT.jl	Julia	PW	NCPPs	Free, GPL	Interpreted language	2020	[41]
DFT.jl	Julia	PW	NCPPs	Free, GPL	Interpreted language	2021	[42]

where $j = 1, 2, ..., N_e$, $\varepsilon_1 \le \varepsilon_2 \le \cdots \le \varepsilon_{N_e}$ are N_e eigenvalues of $\hat{H}(\rho)$. They are known as the Kohn-Sham eigenvalues associated with the corresponding eigenfunctions ψ_j's, also known as the *occupied* orbitals or states. The function ρ is the electron density defined (at zero temperature) as

$$\rho = \sum_{j=1}^{N_e} |\psi_j|^2. \tag{2}$$

The Kohn-Sham Hamiltonian $\hat{H}$ to be partially diagonalized is a functional of ρ (and consequently ψ_j's.)

Equation (1) is the first order necessary condition associated with a constrained minimization problem

$$\min_{\langle \psi_i, \psi_j \rangle = \delta_{i,j}} E_{\text{tot}}(\{\psi_i\}), \tag{3}$$

where E_{tot} is a total energy functional that consists of both kinetic and various potential terms [3], i.e.

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{Hartree}} + E_{\text{ion}} + E_{\text{nuc}} + E_{\text{xc}}, \tag{4}$$

where E_{kin} represents the kinetic energy, E_{Hartree} is the potential energy induced by electron-electron repulsion, E_{ion} is the potential energy induced by nucleus-electron attraction, E_{nuc} is the potential energy induced by nucleus-nucleus repulsion, and E_{xc} is the exchange-correlation energy that accounts for the many-body effects unaccounted in the preceding terms. The mathematical expressions for these energy terms can be found in the standard literature [43,44].

As a result, the Kohn-Sham Hamiltonian $\hat{H}$ appeared in (1) can be partitioned accordingly, i.e.

$$\hat{H} = \hat{T} + \hat{V}_{\text{Hartree}} + \hat{V}_{\text{ion}} + \hat{V}_{\text{xc}}, \tag{5}$$

where $\hat{T}$ is the kinetic energy operator, $\hat{V}_{\text{ion}}$ is the ionic potential operator, $\hat{V}_{\text{Hartree}}$ is the Hartree potential operator and $\hat{V}_{\text{xc}}$ is the exchange-correlation potential operator. We again refer readers to standard literature [44] for analytical expressions for each one of these terms.

2.1.2. Plane-wave basis set

The Kohn-Sham Hamiltonian is periodic for solids. For such periodic systems, we solve (1) by focusing on one period, often known as a primitive cell. It follows from the Bloch's theory that a Kohn-Sham orbital $\psi_j(\mathbf{r})$ takes the form

$$\psi_{j,\mathbf{k}} = e^{i\mathbf{k}\mathbf{r}} u_{j,\mathbf{k}}(\mathbf{r}), \tag{6}$$

where $u_{j,\mathbf{k}}(\mathbf{r})$ is periodic and $\mathbf{k}$ is a crystal momentum vector in the first Brillouin zone.

The occupied Kohn-Sham orbitals are indexed by both j and $\mathbf{k}$. The charge density is periodic and defined as

$$\rho(\mathbf{r}) = \frac{|\Omega|}{(2\pi)^3} \int_{BZ} \rho_{\mathbf{k}}(\mathbf{r}) d\mathbf{k}, \tag{7}$$

where $|\Omega|$ is the volume of the primitive cell in the real space, and

$$\rho_{\mathbf{k}}(\mathbf{r}) = \sum_{j=1}^{N_e} \left|\psi_{j,\mathbf{k}}(\mathbf{r})\right|^2.$$

The choice of a periodic unit cell is not unique. When the unit cell is sufficiently large in real space, the corresponding unit cell in first Brillouin zone is so small that the integral in (7) can be approximated the evaluation of $\rho_{\mathbf{k}}(\mathbf{r})$ at a single $\mathbf{k}$-point, e.g., $\mathbf{k} = 0$, also known as the Γ-point.

Because $u_{j,\mathbf{k}}$ is periodic, it can be expanded by plane-wave basis functions, i.e.,

$$u_{j,\mathbf{k}}(\mathbf{r}) = \sum_{\ell=1}^{\infty} c_{j,\ell}^{\mathbf{k}} e^{i\mathbf{g}_\ell^T \mathbf{r}}, \quad c_{j,\ell}^{\mathbf{k}} = \int_{\Omega} u_{j,\mathbf{k}}(\mathbf{r}) e^{-i\mathbf{g}_\ell^T \mathbf{r}} d\mathbf{r}, \tag{8}$$

where $\mathbf{g}_\ell$ is a lattice vector in the reciprocal space. As a result, a Kohn-Sham orbital $\psi_{j,\mathbf{k}}$ can be represented by

$$\psi_{j,\mathbf{k}} = \sum_{\ell=1}^{\infty} c_{j,\ell}^{\mathbf{k}} e^{i(\mathbf{k}+\mathbf{g}_\ell)^T \mathbf{r}}.$$

This is the discretization scheme used in KSSOLV as in other plane-wave based electronic structure calculation software packages.

In practice, the infinite sum in (8) is truncated and approximated by a finite sum. As in all other plane-wave based Kohn-Sham solvers, the truncation of the plane-wave expansion is based on the following criterion

$$|\mathbf{k}+\mathbf{g}_\ell|^2 < 2E_{\text{cut}}, \tag{9}$$

for some energy cut-off value E_{cut}. If the number of $\mathbf{g}$'s that satisfy this criterion is N_g, an approximation to the Kohn-Sham orbital $\psi_{j,\mathbf{k}}$ can be written as

$$\psi_{j,\mathbf{k}}(\mathbf{r}) \approx \sum_{\ell=1}^{N_g} c_{j,\ell}^{\mathbf{k}} e^{i(\mathbf{k}+\mathbf{g}_j)^T\mathbf{r}}. \tag{10}$$

In a plane-wave basis set, the representations of $\hat{T}$ and $\hat{V}_{\text{Hartree}}$ in (5) are particularly simple, i.e., they are diagonal (or local). However, $\hat{V}_{\text{ion}}$ and $\hat{V}_{\text{xc}}$ typically have a more compact representation in real space. As a result, when $\psi_{j,\mathbf{k}}$ are discretized by a plane-wave expansion, the Kohn-Sham Hamiltonian $\hat{H}$ is not constructed or stored explicitly. The multiplication of $\hat{H}$ (which is called an implicit Hamiltonian) with $\psi_{j,\mathbf{k}}$ can be implemented efficiently by working with both the real space and reciprocal space representations of $\psi_{j,\mathbf{k}}$. The change of representation between real space and reciprocal space is facilitated by Fast Fourier Transforms (FFTs). This is a key feature of plane-wave based Kohn-Sham equation solver.

2.1.3. Exchange-correlation functional

The exchange-correlation energy term E_{xc} in (4) and the exchange-correlation Hamiltonian term $\hat{V}_{\text{xc}}$ accounts for the many-body effects of electron interactions. They are particularly significant for KS-DFT. The exact analytical forms of E_{xc} and $\hat{V}_{\text{xc}}$ are unknown. Various approximations have been proposed. These include the local density approximation (LDA) [45], generalized gradient approximation (GGA) [46], and the hybrid functional [47–49]. A hybrid functional includes a fraction of the exact exchange potential from the Hartree-Fock (HF) [50] theory. Three widely used hybrid functionals are shown in (11). In KSSOLV 2.0, all three approximations have been implemented. Both LDA and GGA are local in real space. Hence, applying these potential operators to a wavefunction is relatively straightforward. However, the Hartree-Fock exact exchange term in a hybrid functional is nonlocal, and applying it to a wavefunction is more costly. However, efficient methods for applying this term have been developed [51–54]. We will discuss efficient methods for working with hybrid functional KS-DFT in section 2.3.

$$\begin{aligned} E_{\text{xc}}^{\text{PBE0}} &= \frac{1}{4}E_{\text{x}}^{\text{HF}} + \frac{3}{4}E_{\text{x}}^{\text{PBE}} + E_{\text{c}}^{\text{PBE}} \\ E_{\text{xc}}^{\text{HSE}} &= 0.25E_{\text{x}}^{\text{HF,SR}} + 0.75E_{\text{x}}^{\text{PBE,SR}} + E_{\text{x}}^{\text{PBE,LR}} + E_{\text{c}}^{\text{PBE}} \\ E_{\text{xc}}^{\text{B3LYP}} &= E_{\text{x}}^{\text{LDA}} + 0.2\left(E_{\text{x}}^{\text{HF}} - E_{\text{x}}^{\text{LDA}}\right) + 0.72\left(E_{\text{x}}^{\text{GGA}} - \right. \\ &\quad \left. E_{\text{x}}^{\text{LDA}}\right) + E_{\text{c}}^{\text{LDA}} + 0.81\left(E_{\text{c}}^{\text{GGA}} - E_{\text{c}}^{\text{LDA}}\right), \end{aligned} \tag{11}$$

2.1.4. Pseudopotential

KSSOLV adopts the pseudopotential methodology [55] to model the interaction between nuclei and electrons. In this approach, core electrons are treated as a part of an ionic core represented by a pre-computed effective potential. Only the valence electrons are present in (1) and (2). For a plane-wave DFT code, the pseudopotential method allows us to significantly reduce the computational cost by reducing the number of active electrons and the number of planewaves required to represent eigenfunctions of the Kohn-Sham Hamiltonian. The latter reduction is due to the fact that the use of pseudopotential makes the eigenfunction of the corresponding Kohn-Sham Hamiltonian less oscillatory.

There are two common types of pseudopotentials in modern DFT computation, namely norm-conserving pseudopotential (NCPP) and ultrasoft pseudopotential. In general, the implementation of NCPPs is easier than that for ultrasoft pseudopotentials [56,57], and they produce sufficient accuracy for many systems. Therefore, NCPPs are the supported type of pseudopotentials in KSSOLV.

A pseudopotential typically consists of a local component $V_{\text{loc}}(\mathbf{r})$ and a nonlocal component $V_{\text{NL}}(\mathbf{r},\mathbf{r}')$. By using the Kleinman-Bylander form of an atomic pseudopotential, we can express $V_{\text{NL}}(\mathbf{r},\mathbf{r}')$ in a low rank separable form

$$V_{\text{NL}}(\mathbf{r},\mathbf{r}') = \sum_{lm} \beta_{lm}(\mathbf{r}) v_l \beta_{lm}(\mathbf{r}')^*, \tag{12}$$

where $\beta_{lm}(\mathbf{r})$ is a pseudo atomic wavefunction associated with the quantum numbers l and m, and v_l is a weighting factor that depends on the degree of spherical harmonic used in β_{lm}.

There are many ways to construct pseudopotentials. We refer readers to standard literature on this subject [44], like many other KS-DFT software tools, we use pseudopotentials archived at a URL[2] in KSSOLV. KSSOLV 2.0 can read pseudopotential files in multiple formats, and convert them to suitable real or reciprocal space representations. The local and non-local components are treated differently. The local component is represented in real space and applied as a diagonal matrix. It is constructed by the summing local atomic potentials re-centered at atomic positions. The re-centering and the summation are carried out through Fourier transforms. The nonlocal pseudo wavefunctions are stored and applied in the reciprocal space. For atoms of the same type, their nonlocal pseudo wavefunctions are combined and transformed to reciprocal space via spherical harmonic transform. The pseudo wavefunctions for different types of atoms are stored separately without additional computation.

2.2. Algorithms implemented in KSSOLV 2.0 of conventional calculations

In this section, we describe several standard algorithms implemented in KSSOLV 2.0 for conventional calculations. These include the self consistent field (SCF) iteration and direct energy minimization, matrix diagonalization and geometry optimization. In addition, we describe a method called SCDM (Select Column of the Density Matrix) used to perform orbital localization.

2.2.1. Self consistent field iteration and direct minimization

When LDA or GGA is used in $\hat{V}_{\text{xc}}$, the KS eigenvalue problem can be formulated as a set of nonlinear equations satisfied by the ground state electron density or potential [2], i.e.,

$$\rho = f_{\text{KS}}(\rho), \tag{13}$$

where $f_{\text{KS}}(\cdot)$ is known as the Kohn-Sham map [3]. This formulation suggests that the KS equations can be solved by a quasi-Newton method in which the Jacobian of the Kohn-Sham map is approximated. To be specific, the approximation to ρ can be updated as

$$\rho^{k+1} = \rho^{(k)} - \hat{J}(\rho^{(k)})\left[\rho^{(k)} - f_{\text{KS}}(\rho^{(k)})\right], \tag{14}$$

where $\hat{J}$ is an approximate Jacobian. This approach is generally known as the self-consistent field (SCF) iteration in the physics lit-

[2] pseudopotentials homepage used by KSSOLV: http://pseudopotentials.quantum-espresso.org/legacy_tables.

erature and is implemented in KSSOLV. In this approach, the evaluation of the Kohn-Sham map on the right hand side of (14) requires computing eigenvalues and eigenvectors of the Kohn-Sham Hamiltonian defined at $\rho^{(k)}$, which will be discussed in the next section.

There are many ways to approximate the Jacobian of the Kohn-Sham map. The simplest is to take $\hat{J} = \beta I$, where $0 < \beta < 1$ is a small constant and I is the identity matrix. Such an approximation yields the so-call simple mixing scheme described by

$$\rho^{(k+1)} = \beta\rho^{(k)} + (1-\beta) f_{\mathrm{KS}}(\rho^{(k)}). \tag{15}$$

More sophisticated Jacobian approximate schemes include the Anderson [58] or Pulay [59] mixing, two types of Brodyen's method and Kerker mixing [60], which can also be viewed as a way to accelerate the convergence of the quasi-Newton iteration (14) by preconditioning the nonlinear equation (13) [61]. All these Jacobian approximation and precondition methods have been implemented in KSSOLV 2.0. In earlier work [62], we have demonstrated how new preconditioners can be easily implemented in KSSOLV.

An alternative approach to solving the Kohn-Sham problem is to solve the constrained minimization problem (3) directly. This approach is known as direct minimization. In KSSOLV, we implement a direct constrained minimization algorithm presented in [63]. In each step of the algorithm, a subspace that consists of the current approximation to the Kohn-Sham orbitals, the preconditioned gradient of the Lagrangian and previous search direction is constructed. The update of the wavefunction approximation is obtained by minimizing the total energy (4) within this subspace. Trust region techniques [64] can be used in the DCM algorithm to stabilize the convergence of the iterative minimization procedure. This is particularly useful for metallic systems at low temperature.

2.2.2. Eigensolver

When the SCF iteration is used to solve the Kohn-Sham problem, the most time-consuming part of the computation is the evaluation of the Kohn-Sham map $f_{\mathrm{KS}}(\rho)$. At a finite temperature, the Kohn-Sham map is defined as

$$f_{\mathrm{KS}}(\rho) = \mathrm{diag}\left[\left(I + e^{\frac{H(\rho)-\mu I}{\kappa_B T}}\right)^{-1}\right], \tag{16}$$

where μ is the chemical potential, κ_B is the Boltzmann factor and T is the temperature. The matrix exponential in (16) can be evaluated by a partial spectral decomposition of H. In the limit of $T = 0$, (16) reduces to (2). In this case, we only need to compute the leftmost N_e eigenvalues of $H(\rho)$ and their corresponding eigenvectors. For a finite temperature calculation, we need to compute a few extra eigenvalues ε_j and eigenvectors ψ_j that have non-negligible occupation numbers $1/(1+\exp(\frac{\varepsilon_j-\mu}{\kappa_B T}))$.

Because H is not explicitly stored as a matrix in KSSOLV, iterative eigensolvers are appropriate for computing the desired eigenvalues and eigenvectors of H. In KSSOLV, the default eigensolver employed in a SCF iteration uses the locally optimal block preconditioned conjugate gradient (LOBPCG) method [65]. The method can be viewed as a constrained minimization method for solving the equivalent trace minimization problem

$$\min_{X^T X = I} \mathrm{trace}(X^T H X), \tag{17}$$

where X is a matrix that contains the discretized Kohn-Sham orbitals. Similar to other plane-wave based Kohn-Sham solvers, KSSOLV stores the plane-wave expansion coefficients of each Kohn-Sham orbitals in X. In each LOBPCG iteration, we need to multiply H with a set of vectors. This is done by multiplying the kinetic energy operator $\hat{T}$, the nonlocal part of the ionic pseudopotential operator $\hat{V}_{\mathrm{ion}}$ and the Hartree potential operator $\hat{V}_{\mathrm{Hartree}}$ with the plane-wave expansion coefficients in the reciprocal space and transforming the result to a real space grid (via FFTs) on which the local part of $\hat{V}_{\mathrm{ion}}$ and the local exchange-correlation potential operator $\hat{V}_{\mathrm{xc}}$ are applied.

In addition to LOBPCG, KSSOLV 2.0 also includes an implementation of the Davidson-Liu [66] algorithm. The algorithm can be viewed as a generalization of the LOBPCG method in the sense that the update of the eigenvector is obtained by projecting the $\hat{H}$ into a progressively larger subspace constructed from the juxtaposition of preconditioned gradients of the Lagrangian and the subspace constructed in the previous iteration, and solving the projected eigenvalue problem. When the dimension of the subspace reach a prescribed limit, the procedure is restarted with the most recent approximation of the eigenvectors. Clearly, there is a trade-off between the per iteration cost of the Davidson method, which becomes higher if the maximum allowed dimension of the subspace (m_D) is large, and the number of restarts required to reach convergence, which becomes lower when m_D is larger.

Both the LOBPCG and Davidson solvers are block eigensolver, i.e., in each iteration, the Hamiltonian is applied to a block of vectors, and many other linear algebra operations in the solver can be expressed in terms of level-3 BLAS operations. These features can significantly enhance the concurrency of the computation and take advantage of parallel computer architecture and memory hierarchy.

Another block algorithm that has been demonstrated to be very efficient for solving large-scale Kohn-Sham eigenvalue problem is the Chebyshev filter subspace iteration (CheFSI) [67]. This method is implemented in KSSOLV 2.0 also. The CheFSI method constructs a properly shift and scaled mth degree Chebyshev polynomial T_m to amplify the contribution of the desired eigenvectors when $T_m(H)$ is applied to a set of properly prepared vectors in a subspace iteration. The multiplication of $T_m(H)$ with a block of vectors X can be implemented via a 3-term recurrence. We do not need to solve a projected eigenvalue problem, i.e., we do not need to perform the Rayleigh-Ritz procedure in each subspace iteration. This can significantly reduce the computational cost for the problem with a large number of electrons. The orthonormality of X in each iteration can be maintained by using the Cholesky QR procedure, which is generally efficient. The Rayleigh-Ritz procedure only needs to be performed at the end of subspace iteration to compute the occupation number for each desired eigenvalue.

Two other methods that are designed to reduce the cost of Rayleigh-Ritz calculations in an eigensolver are the project preconditioned conjugated gradient (PPCG) [68] method and the residual minimization method with direct inversion in iterative subspace (RMM-DIIS) [69] acceleration. Both methods are implemented in KSSOLV 2.0 also. Although for most small to medium sized problems to be solved by KSSOLV, the Rayleigh-Ritz cost in the block algorithms discussed above is relatively small, the availability of additional eigensolvers allows us to test and compare convergence properties of these algorithms.

In PPCG, we apply the LOBPCG algorithm to each approximate eigenvector separately, i.e. running the unblocked version of the LOBPCG method for each desired eigenpair for a fixed number of iterations. The Rayleigh-Ritz procedures in these runs only need to solve a set of 3×3 projected eigenvalue problems. A global Rayleigh-Ritz procedure for all desired eigenpairs is only applied periodically at the end of a fixed number of unblocked LOBPCG iterations.

In RMM-DIIS, each approximate eigenpair is updated separately by minimizing the residual (instead of the Rayleigh-quotient) associated with the approximate eigenpair within a progressively larger subspace incrementally constructed from a set of previously ap-

proximations to the same eigenpair. When the initial guess of the desired eigenpairs are sufficiently accurate, no Rayleigh-Ritz procedure is ever needed in RMM-DIIS. This feature of the algorithm makes it ideal for solving large-scale problems that contain many electrons on a parallel computer on which the refinement of each eigenpair can be carried out independently.

One of the key features of the eigenvalue problems solved in each SCF iteration is that the accuracy required for the desired eigenpairs is generally lower in early SCF iterations and higher in later iterations when self-consistency is nearly reached. This is due to the fact that in early iterations of the quasi-Newton algorithm used to solve (13), the residual term $\rho^{(k)} - f_{\mathrm{KS}}(\rho^{(k)})$ on the right-hand side of (14) is relatively large even if $f_{\mathrm{KS}(\rho^{(k)})}$ is evaluated to full accuracy. As a result, we may lower the accuracy requirement for $f_{\mathrm{KS}}(\rho^{(k)})$, and consequently the accuracy requirement for the solution of the eigenvalue problem could be reached. As the SCF iteration converges, a more accurate evaluation of $f_{\mathrm{KS}}(\rho^{(k)})$. Therefore, in KSSOLV 2.0, we use an adaptive strategy to define the convergence criterion for each eigenpair. An approximate eigenpair (θ, ψ) is considered converged if the relative residual norm

$$\|r\|/|\theta| = \|H\psi - \theta\psi\|/|\theta|,$$

is less than a tolerance $\tau^{(k)}$, where

$$\tau = \min(\tau_0, \|\rho^{(k)} - f_{\mathrm{KS}}(\rho^{(k)})\|/\|\rho^{(k)}\|), \tag{18}$$

and τ_0 is a maximal error tolerance set to 10^{-2} by default, but can be changed by a user.

Moreover, at a finite temperature, lower accuracy can be tolerated for partially occupied states with low occupation numbers.

2.2.3. Geometry optimization

In KSSOLV 2.0, we include the functionality to compute atomic forces which are the derivatives of E_{tot} defined in (4) with respect to atomic coordinates. The derivatives can be taken with respect to either the cartesian coordinates or relative coordinates of the atoms [70]. The calculations of these forces make use of the Hellmann-Feynman theorem [71]. The availability of atomic forces allows us to optimize the atomic structure of a molecule or solid. The calculations are often referred to geometry optimization or structure relaxation. The goal of the optimization is to minimize the total energy of the atomistic system with respect to atomic coordinates. The atomic forces simply yield the gradient of the objective function.

KSSOLV 2.0 leverages the standard unconstrained minimization algorithm implemented in MATLAB's optimization box. The user has the option of using MATLAB's `fminunc`(Find minimum of unconstrained multivariable function) function to perform the optimization. By default, `fminunc` uses the BFGS (Broyden–Fletcher–Goldfarb–Shanno) [72] quasi-Newton algorithm which constructs approximations to the Hessian of the energy using gradients computed in successive quasi-Newton iterations. One can also choose the `trust region` algorithm, which is based on the interior-reflective Newton method described in [73].

In addition to algorithms implemented in MATLAB Optimization toolbox, one can also use other algorithms such as the limited memory BFGS [74] algorithm implemented in the HANSO package [75] or the nonlinear conjugate algorithm [76] implemented by Overton [77]. Both algorithms contain a number of parameters that a user can experiment with and adjust. KSSOLV 2.0 also provides ample flexibilities to utilize other optimization algorithms. For example, we also implemented a version of the FIRE (Fast Inertial Relaxation Engine) [78] algorithm.

2.2.4. Orbital localization via selected column of density matrix

It is well known that electrons in insulating systems obey the nearsightedness principle, i.e. local electron properties such as the electron density $\rho(\mathbf{r})$ only depend significantly on the effective potential at nearby points. Mathematically, the nearsightedness principle translates into the decay property of the single-particle density matrix associated with the ground state of the atomistic system, i.e., the magnitude of the matrix elements of the density matrix decays rapidly away from the diagonal. A direct consequence is that the occupied Kohn-Sham orbitals can be rotated to a set of functions that span the same invariant space, but have approximately localized support. These localized orbitals can be used to develop linear scaling methods for solving the Kohn-Sham problem and to develop efficient post-DFT methods [79,80].

There are several ways to construct localized orbitals. One of the most known technique is the maximally localized Wannier functions (MLWFs) proposed by Marzari and Vanderbilt [81]. The MLWF method requires solving a nonlinear optimization problem, whose results can sometimes depend sensitively to the initial guesses. Recently, an alternative method called Selected Column of the Density Matrix (SCDM) [82] has been proposed to construct localized orbital using a simple linear algebraic procedure. Suppose Ψ is an $N \times N_e$ matrix containing N_e approximate Kohn-Sham orbitals on N real space grid points. The SCDM method performs a rank revealing QR factorization of Ψ^* first to yield

$$\Psi^*\Pi = Q\,R, \tag{19}$$

where Π is a column permutation matrix that moves maximally linearly independent columns of Ψ^* (or a row permutation matrix that moves maximally linear independent rows of Ψ) to the leading column (row) positions, Q is a $N_e \times N_e$ unitary matrix and R is $N_e \times N$ matrix with the leading N_e columns being a upper triangular matrix. The magnitudes of the diagonal matrix elements of the leading columns of R are in a decreasing order.

Localized orbitals can be computed simply by performing a matrix multiplication

$$\Phi = \Psi\Psi_C^*, \tag{20}$$

where Ψ_C represents the leading N_e rows of the row permuted Ψ where the permutation is defined by the permutation matrix Π obtained in (19). Note that the localized columns in Φ are not necessarily orthonormal. To obtain an orthonormal set of orbitals $\tilde{\Phi}$ that remain to be localized, we simply perform a Cholesky factorization of the matrix $P_{C,C} = \Psi_C\Psi_C^*$, i.e.,

$$P_{C,C} = LL^*, \tag{21}$$

and solve the following set of linear equations using the Cholesky factor L obtained in (21)

$$\tilde{\Phi}L^* = \Phi.$$

The SCDM method has been implemented in KSSOLV. We refer readers to [82] for the theoretical justification of this method and how localized orbitals constructed from the SCDM procedure can be used to speedup the Hartree-Fock exchange energy calculation [83,84].

2.3. Accelerated algorithms implemented in KSSOLV 2.0 for hybrid functional DFT calculations

In KSSOLV 2.0, we implement several new algorithms to accelerate hybrid functional DFT calculations. The main challenge in performing a hybrid functional DFT calculation is the efficient

treatment of the (screened) Hartree-Fock exchange potential defined as

$$\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}(\mathbf{r},\mathbf{r}') = -\sum_{j=1}^{N_e} \psi_j(\mathbf{r})\psi_j^*(\mathbf{r}')K(\mathbf{r},\mathbf{r}'), \tag{22}$$

where $K(\mathbf{r},\mathbf{r}')$ is either the Coulomb kernel $1/|\mathbf{r}-\mathbf{r}'|$ or a screened Coulomb kernel of the form $\mathrm{efrc}(\mu|\mathbf{r}-\mathbf{r}'|)/|\mathbf{r}-\mathbf{r}'|$. This non-local potential is part of the exchange-correlation potential $\hat{V}_{\mathrm{xc}}$ in a hybrid functional DFT Hamiltonian.

In KSSOLV 2.0, we do not explicitly construct $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$, which is a dense matrix, in either the real or reciprocal space. The $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ operator is applied to a set of wavefunctions $\{\varphi_i\}$ as follows

$$\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}\varphi_i = -\sum_{j=1}^{N_e} \psi_j(\mathbf{r}) \int \varphi_i\left(\mathbf{r}'\right)\psi_j^*\left(\mathbf{r}'\right)K(\mathbf{r},\mathbf{r}')d\mathbf{r}'. \tag{23}$$

The evaluation of the integral on the right hand side of (23) requires solving a set of Poisson equations. This can be done by using FFT based convolution. However, because the summation in (23) is over N_e terms, we need to solve $\mathcal{O}(N_e^2)$ Poisson equations in total per iteration in an iterative eigensolver used to compute the lowest N_e eigenpairs of the hybrid functional Hamiltonian. The excessive number of FFTs used to solve many Poisson equations is the reason that hybrid functional DFT calculation is orders of magnitude more expensive than LDA or GGA DFT calculations in other plane-wave DFT software tools. KSSOLV 2.0 uses several recently developed algorithms to reduce the complexity of hybrid functional DFT calculation. These algorithms include 1) the interpolative separable density fitting (ISDF) method for reducing the number of Poisson equations to be solved; 2) the use of inner and outer iterative schemes in combination with the adaptive compressive exchange (ACE) operator method to further reduce the total number of Poisson equations to be solved; 3) a special projector commutator direct inversion of iterative subspace (PC-DIIS) method for accelerating the outer SCF iteration. We will briefly describe each one of these algorithms below.

2.3.1. ISDF (interpolative separable density fitting decomposition)

If we place the right hand sides of the Poisson equations to be solved in (23) for $i = 1, 2, ..., N_e$ in a matrix Z, defined as

$$Z = \{\varphi_i(\mathbf{r})\psi_j^*(\mathbf{r})\}, \quad i, j = 1, 2, ..., N_e, \tag{24}$$

we can see that the rank Z is less than N_e^2 if $\varphi_i(\mathbf{r})$ and $\psi_j^*(\mathbf{r})$ are discretized on a real space grid with $N_g = \mathcal{O}(N_e)$ grid points, which is the case for systems that are sufficiently large. As a result, we can rewrite Z as

$$Z = \Theta C, \tag{25}$$

where Θ is $N_g \times N_\mu$ and C is $N_\mu \times N_e^2$ and $N_\mu = \mathcal{O}(N_e)$. Columns of Θ can be viewed as a set of numerical auxiliary basis $\{\zeta_\mu(\mathbf{r})\}$, $\mu = 1, 2, ..., N_\mu$ that span the same space defined by the pair product basis $\{\varphi_i(\mathbf{r})\psi_j^*(\mathbf{r})\}$. Consequently, we can evaluate (23) by first computing

$$V_\mu^\zeta = \int K(\mathbf{r},\mathbf{r}')\zeta_\mu(\mathbf{r}')d\mathbf{r}', \tag{26}$$

for all $\mu = 1, 2, ..., N_\mu$, which requires solving N_μ Poisson equations. If we use V^ζ to denote the matrix that contains V_μ^ζ's as its columns, (23) can be then evaluated as

$$\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}\varphi_i = -\sum_j \psi_j(\mathbf{r})V^\zeta c_{ij}, \tag{27}$$

where c_{ij} is the column of C indexed by i and j consistent with the column indexing scheme used in (24).

Although the computational procedure for evaluating (22) now requires solving only $N_\mu = \mathcal{O}(N_e)$ Poisson equations, the overall complexity of the algorithm hinges on an efficient factorization of Z in (25). From an accuracy standpoint, the optimal factorization can be obtained by performing a singular value decomposition (SVD) of Z. However, such a factorization is costly.

In [53], the ISDF technique is used to obtain an approximate factorization that is much more efficient and sufficiently accurate. In ISDF, each entry of the C matrix is chosen to be $\varphi_i(\mathbf{r}_\mu)\psi_j^*(\mathbf{r}_\mu)$ for a set of carefully chosen real space grid points $\mathbf{r}_\mu$. The auxiliary basis vectors in Θ can be obtained by solving a linear least square problem. Due to the separable nature of the pair product basis in C, this least square problem can be solved efficiently. We will refer readers to [53,54] for computational details of the ISDF method. We should note that in this approach N_μ is a parameter that a user needs to choose in advance. Typically, N_μ is a small multiple of N_e, e.g. $2N_e$. As a result, the use of ISDF allows us to reduce the overall computational complexity of $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ related operation to $O(N_e^3)$.

2.3.2. ACE (adaptively compressed exchange)

Due to the high cost associated with the application of the Hartree-Fock exchange operator $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ to a set of wavefunctions, the iterative solution of the Kohn-Sham problem for hybrid functional DFT is separated into inner and outer SCF iterations. At the beginning of each outer SCF iteration, $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ is updated with the most recent approximations to the Kohn-Sham orbitals $\{\psi_j\}$. This $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ is then fixed in the corresponding inner SCF iterations in which only the charge density ρ and potential terms that depends on ρ are updated.

However, as we can see in (23), even when $\{\psi_j\}$ and $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}(\{\psi_j\})$ are fixed, applying $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}(\{\psi_j\})$ to a set of orbitals φ_i, $i = 1, 2, ..., N_e$ is costly. Although we can use ISDF to reduce the number of Poisson solves from $\mathcal{O}(N_e^2)$ to $\mathcal{O}(N_e)$, performing the ISDF procedure and solving $\mathcal{O}(N_e)$ Poisson equations in each inner SCF iteration is still quite costly.

To reduce the computational cost of each inner iteration, Lin [51] proposed the construction of an approximate $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ using a procedure called the Adaptively Compressed Exchange Operator (ACE) algorithm. The approximate $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$, denoted by $\hat{V}_{\mathrm{x}}^{\mathrm{ACE}}$, is constructed to satisfy the condition

$$\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}\psi_j = \hat{V}_{\mathrm{x}}^{\mathrm{ACE}}\psi_j, \tag{28}$$

where ψ_j, $j = 1, 2, ..., N_e$ is the set of approximate Kohn-Sham orbitals available at the beginning of each outer SCF iteration. The ACE construction yields a low-rank operator of the form

$$\hat{V}_{\mathrm{x}}^{\mathrm{ACE}} = -\sum_{i,j=1}^{N_e} W_i(\mathbf{r})B_{ij}W_j^*(\mathbf{r}'), \tag{29}$$

where

$$W_i(\mathbf{r}) = \left(\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}[\{\psi_i\}]\psi_i\right)(\mathbf{r}), \quad i = 1, \ldots, N_e, \tag{30}$$

$B = M^{-1}$ and the (k,l)th element of the overlap matrix M is $M_{kl} = \int \psi_k(\mathbf{r})W_l(\mathbf{r})\mathrm{d}\mathbf{r}$.

By constructing an ACE approximation of $\hat{V}_{\mathrm{x}}^{\mathrm{HSE}}$ in the low rank form (29), we can apply $\hat{V}_{\mathrm{x}}^{\mathrm{ACE}}$ to a set of orbitals $\{\varphi_i\}$ in each SCF inner iteration by using two matrix-matrix multiplications. This type of BLAS3 dense linear algebra operations are extremely efficient on modern high performance computers.

We should note that the construction of $\hat{V}_{\mathrm{x}}^{\mathrm{ACE}}$ in each outer SCF iteration requires solving $\mathcal{O}(N_e^2)$ Poisson equations in (30) just as

$\mathcal{O}(N_e^2)$ Poisson equations need to be solved in (23). The number of Poisson equations to be solved can be reduced to $\mathcal{O}(N_e)$ by using the ISDF technique discussed above. Therefore, by combining ACE with ISDF, we can significantly reduce the complexity of hybrid functional DFT calculation as reported in [85].

2.3.3. PC-DIIS (projected commutator direct inversion in the iterative subspace)

As we indicated in section 2.2.1, when LDA and GGA are used in $\hat{V}_{xc}$, the KS eigenvalue problem can be formulated as a set of nonlinear equations (13) satisfied by the ground state electron density ρ. For hybrid functional DFT, a similar nonlinear equation should be defined in terms of the density matrix $P = \sum_{j=1} \psi_j(\mathbf{r})\psi_j^*(\mathbf{r}')$ at zero temperature. Alternatively, one can define a nonlinear equation in terms of the commutator between $H(P)$ and P. Upon convergence, P satisfies

$$H(P)P - PH(P) = 0. \tag{31}$$

The outer SCF iteration used to solve the hybrid functional KS-DFT problem can be viewed as a quasi-Newton method for finding the solution of (31). When the density matrix can be formed explicitly, one can use the direct inversion of iterative subspace (DIIS) method proposed by Pulay [59] to solve (31). This is the approach often used to solve the Hartree-Fock equation in quantum chemistry. Given a few previous approximations to the density matrix $P^{(i-1)}$, $P^{(i-2)}$,...,$P^{(i-\ell)}$, for some constant $\ell < i$, the DIIS method or commutator DIIS (C-DIIS) method constructs a new approximation to the density matrix in the ith iteration as

$$\tilde{P} = \sum_{k=1}^{\ell} \alpha_k P^{(i-k)}, \tag{32}$$

where the coefficients α_k are chosen to solve the following constrained minimization problem

$$\min_{\sum_k \alpha_k = 1} \|\alpha_k R[P^{(i-k)}]\|_F, \tag{33}$$

where

$$R[P^{(i-k)}] = H[P^{(i-k)}]P^{(i-k)} - P^{(i-k)}H[P^{(i-k)}] \tag{34}$$

and $\|\cdot\|_F$ is the Frobenius norm.

When the Kohn-Sham orbitals ψ_j's are discretized by plane-wave expansions, it is generally not practical to construct the density matrix P explicitly and solve the minimization problem (33) directly because the density matrix dimension is so large($N_g \times N_g$) within a plane-wave basis set. In [52], a projected commutator DIIS (PC-DIIS) method was proposed to solve a projected minimization problem in which the matrix $R[P^{(i-k)}]$ in (33) is replaced by

$$R[P^{(i-k)}]\Phi_{\text{ref}} = H\Psi^{(i-k)}S^{(i-k)} - \Psi^{(i-k)}T^{(i-k)}, \tag{35}$$

where Φ_{ref} is a set of reference orbitals to be defined later and $\Psi^{(i-k)}$ is a matrix that contains approximate Kohn-Sham orbitals ψ_j's obtained in the $(i-k)$th outer SCF iteration, $S^{(i-k)} = \langle\Psi^{(i-k)}, \Phi_{\text{ref}}\rangle$ and $T^{(i-k)} = \langle H\Psi^{(i-k)}, \Phi_{\text{ref}}\rangle$. Note that we dropped the density matrix $P^{(i-k)}$ in the Hamiltonian H above to simplify the notation. The projected residual (35) can be computed without forming $P^{(i-k)}$ explicitly. We will refer readers to [52] for the theoretical justification for using (35) in the objective function of the minimization problem (33). The solution of the alternative minimization problem is used to construct an intermediate set new approximation to Kohn-Sham orbitals as

$$\tilde{\Psi} = \sum_{k=1}^{\ell} \alpha_k \Psi^{(i-k)}.$$

The eigenvectors of $H[\tilde{\Psi}]$ then form the approximate Kohn-Sham orbitals $\Psi^{(i)}$ in the ith SCF iteration. Self-consistency is achieved when the norm of $H[\Psi^{(i)}]\Psi^{(i)} - \Psi^{(i)}\Lambda^{(i)}$ is sufficiently small, where $\Lambda^{(i)}$ is a diagonal matrix containing the corresponding eigenvalues of $H[\tilde{\Psi}]$.

We should note that the reference orbitals in Φ_{ref} can be chosen to be any linearly independent functions that approximates the desired Kohn-Sham orbitals. They can be chosen as a set of Kohn-Sham orbitals obtained in an LDA or GGA calculation. Also, the constrained minimization problem (33) can be easily converted to an unconstrained least square minimization problem by substituting α_1 in the objective function with $1 - \sum_{k=2}^{\ell} \alpha_k$. We will refer readers to [52] for algorithmic and computational details.

3. Object oriented design

Object-oriented programming (OOP) is a modern design paradigm developed to define data and functions together as an object. KSSOLV adopts OOP features in MATLAB and implements many key quantities required in the numerical solution of (1) as classes. In KSSOLV, there are several basic classes and some more advanced classes. The basic classes include the `Atom`, `Molecule`, `Crystal`, `PpData`, `PpVariable`, `Ggrid`, and `IterInfo` classes. The `Atom`, `Molecule` and `Crystal` classes are created to represent and encapsulate all relevant properties of an atom, a molecule and a crystal respectively. All relevant features of an object, *e.g.*, the mass of an atom, the positions of all atoms within a molecule, the energy cut-off used for plane-wave expansion is kept as attributes (member variable) of the object. Since a crystal shares many features with a molecule, the `Crystal` class is defined as a derived class of the `Molecule` class with additional attributes such as the positions of k-point samples and their corresponding weights. The `PpData` and `PpVariable` classes are two classes that encapsulate a variety of information related to the pseudopotential. The `PpData` class is used to encapsulate the raw data read from a pseudopotential file, and the `PpVariable` class stores the actual pseudopotentials associated with all atomic species contained in a molecule (or crystal). The `Ggrid` class is used to provide a compact representation of reciprocal space grid points enclosed within a sphere of a fixed radius determined by the kinetic energy cutoff E_{cut}. Note that in a plane-wave based DFT calculation, the plane-wave coefficients associated with reciprocal grid points outside of this sphere are set to zero, and thus not stored. Finally, the `IterInfo` class is a bookkeeping class used to simply record information related to the SCF/DCM iterations. All these basic classes are designed as data containers to simplify the interfaces in KSSOLV. The member functions in these classes are used to process data within the class but do not interfere with data outside the class.

In the following, we will introduce advanced classes in KSSOLV one by one in detail, *i.e.*, the `Wavefun`, `Ham`, `BlochWavefun` and `BlochHam` classes.

3.1. The Kohn-Sham wavefunction class

Kohn-Sham orbitals are key quantities used and updated throughout a KS-DFT calculation. We created a class, called `Wavefun`, to encapsulate all relevant information contained in these orbitals. This class contains matrix attributes that keep either the values of wavefunctions on a real space grid or plane-wave expansion coefficients on a compressed reciprocal space grid. Standard algebraic operations applied to a `Wavefun` object are overloaded. They include element-wise operations such as the absolute value,

Listing 1 Setting up a `Wavefun` object.

```
% X is a @Wavefun object of size N by k
X = Wavefun(...);

% Q is a @Wavefun of the same size as X,
% and R is a k by k upper triangular matrix
[Q,R] = qr(X,0);

% U is a @Wavefun of the same size as X,
% and S and V are k by k matrices
[U,S,V] = svd(X,0);

% Y is a @Wavefun object of size N by p
% on the same grid points as X
Y = Wavefun(...);

% Z is a @Wavefun object of size N by (k+p)
Z = [X Y];

% T is a submatrix of Z with its odd row index
% and last k column index.
% T is a @Wavefun object of size N/2 by k
% Then the corresponding submatrix of Z is reset
% to a random matrix.
T = Z(1:2:end,end-k+1:end);
Z(1:2:end,end-k+1:end) = randn(N/2,k);
```

the addition, subtraction, pointwise multiplication and pointwise powering. These operations typically return a `Wavefun` object. Other operations such as the matrix norm and the inner product of two sets of wavefunctions return a scalar or a matrix. Other commonly used operations such as the QR factorization, SVD and 3D (inverse) Fast Fourier Transform are overloaded as well. Listing 1 provides some simple examples of overloaded operations on a `Wavefun` object.

Other overloaded functions include the concatenation of `Wavefun` objects, the selection of one or a subset of wavefunctions, which are unique in MATLAB. We allow a `Wavefun` object to be multiplied with a matrix also when the dimension of the matrix contained in the `Wavefun` object is compatible with that of second matrix to be multiplied.

3.2. The Hamiltonian class

Even though the Kohn-Sham Hamiltonian is not stored as a matrix in KSSOLV, it is convenient to create a `Ham` class that allows us to easily apply the Hamiltonian to a `Wavefun` object. The `Ham` class encapsulates the kinetic and potential energy components of the Hamiltonian in either real space or reciprocal space representation as well as the charge density associated with the Hamiltonian. Member functions are created to make it easy to update the Hamiltonian when the charge density is changed. The multiplication of a `Ham` object `H` and a `Wavefun` object `X` can be simply performed as `H*X` with all the details resulting from the conversion from the real space to the reciprocal space and back to the real space representation of the wavefunction hidden from the user. See Listing 2 for how a `Ham` object is created and used. Furthermore, we have also implemented several functions such as the MINRES and GMRES functions for solving the linear system of involving a shifted Kohn-Sham Hamiltonian operator.

3.3. Wavefunction and Hamiltonian classes for solids

For periodic systems, we have created the `BlochWavefun` and `BlockHam` classes to encapsulate data elements required to represent Bloch wavefunctions and Hamiltonian. These classes allow users to specify a k-point sampling and the associated weights. They are containers of the `Wavefun` and `Ham` type variables respectively. Listing 2 contains an example of how these two classes are used.

Listing 2 Setting up a Hamiltonian object.

```
% H is a @Ham object of size N by N
H = Ham(...);

% X is a @Wavefun object of size N by k, and Y is
% a @Wavefun of the same size
X = Wavefun(...);
Y = H*X;

% V is a random matrix of size N by k, and U is
% a matrix of the same size
V = randn(N,k);
U = H*V;

% BH is a @BlochHam object of size N by N for
% m k-points
BH = BlochHam(...);

% BX and BY are a @BlochWavefun objects of size
% N by k for m k-points
BX = BlochWavefun(...);
BY = BlochWavefun(...);

% BH is applied to BX and saved at BY
for i = 1:m
    BY{i} = BH{i}*BX{i};
end
```

4. Use cases

In this section, we will illustrate some key features of KSSOLV through some use cases. The main workflow for using KSSOLV to perform an electronic structure calculation of a molecule or solid involves

1. Setting up the system;
2. Calling an appropriate function to solve the Kohn-Sham problem or perform a geometry optimization;
3. Examining, post-processing and visualizing the results.

We will use a simple example to demonstrate how to perform a basic calculation in section 4.1. One of the key advantages of KSSOLV is that it allows users to try different algorithms and algorithmic parameters. We will illustrate how this can be achieved in KSSOLV by properly setting different options and comparing results. The object-oriented design of KSSOLV enables developers to prototype and implement new algorithms with ease. We will give an example to show some of the key features that make prototyping new algorithms easy in KSSOLV. Finally, the MATLAB performance profiler allows developers to identify the main computational bottleneck of the calculation and develop strategies to improve computational efficiency.

4.1. Setting up and solving a simple problem

Before we perform an electronic structure calculation for a molecule or a solid, we must first set up the system. This step entails selecting the constituent atoms and defining their atomic coordinates. In addition, we must define a sufficiently large unit (super)cell that contains all constituent atoms. The list of atoms and their coordinates as well as the supercell are used to define a `Molecule` object. For example, in Listing 3, we show how a silane molecule (SiH_4) is set up.

In this script, which can be found in the `kssolv2.0/examples` directory, we first choose the pseudopotential type by using `kssolvpptype`. The Optimized Norm-Conserving Vanderbilt (ONCV) [16] is chosen (by default). Changing it to another type of NCPP, e.g. the Hartwigsen-Goedecker-Hutter (HGH) [17] pseudopotential simply involves uncommenting the second line of the code

Listing 3 Setup A System.

```
kssolvpptype('ONCV_PBE-1.0', 'UPF');
%kssolvpptype('pz-hgh', 'UPF');
%
% 1. construct atoms
%
a1 = Atom('Si');
a2 = Atom('H');
atomlist = [a1 a2 a2 a2 a2];
%
% 2. set up a supercell
%
C = 10*eye(3);
%
% 3. define the coordinates the atoms
%
redxyz = [
 0.0     0.0      0.0
 0.161   0.161    0.161
-0.161  -0.161    0.161
 0.161  -0.161   -0.161
-0.161   0.161   -0.161
];
xyzlist = redxyz*C';
%
% 4. Configure the molecule (crystal)
%
mol = Molecule('supercell',C,'atomlist',
      atomlist,'xyzlist',xyzlist, ...
      'ecut',12.5,'name','SiH4' );
```

Listing 4 Building an atom list array.

```
a1 = Atom('Si');
a2 = Atom('H');
atomlist(1) = a1;
for j = 2:5
  atomlist(j) = a2;
end;
```

snippet. In KSSOLV 2.0, users can adopt NCPPs in both UPF file format (used by QUANTUM ESPRESSO) and psp8 file format (used by ABINIT).

We then create two `Atom` objects `a1` and `a2` representing the Si and H atoms. These objects are then placed in an `atomlist` array using one of MATLAB's array creation syntax. For systems containing a large number of atoms, we can also use MATLAB's loop scripting capability to build such an array using, e.g.,

We then specify the Cartesian coordinates for each atom as a 5×3 array `xyzlist`. In this example, these atomic coordinates are calculated from the reduced coordinates specified in (`redxyz`) and the supercell defined by the matrix C. But it is possible to specify these coordinates directly.

In order to solve the Kohn-Sham problem associated with this molecule, we must also specify the kinetic energy cut-off `ecut` to be used for the plane-wave discretization of the Kohn-Sham orbitals. In the Listing 3, `ecut` is set to 12.5 Hartree.

All attributes of the SiH_4 molecule are passed into the function that creates a `Molecule` object as key–value pairs as shown in Listing 3.

Once a molecule object has been properly defined, we can solve the Kohn-Sham problem associated with this molecule by calling the `scf` function as

```
[mol,H,X,info] = scf(mol);
```

Running the `scf` function generates the output shown in Listing 5.

The default output written in the MATLAB command line window shows the convergence history of the SCF iteration. The output contains the dynamically adjusted error tolerance used to terminate iterative solution of a linear eigenvalue problem in each

Listing 5 SCF Output.

```
Beging SCF calculation for SiH4...
SCF iter   1:
eigtol =   1.000e-02
Rel Vtot Err    =            1.024e-01
Total Energy    = -6.2382906512612e+00
......
SCF iter  10:
eigtol =   9.543e-07
Rel Vtot Err    =            1.250e-06
Total Energy    = -6.2542498381078e+00
Elapsed time is 3.132360 seconds.
......
||HX-XD||_F     =            1.144e-08
```

Listing 6 Visualize the electron density.

```
view(3);
isosurface(fftshift(H.rho));
figure;
view(3);
vol3d('cdata',fftshift(H.rho));
```

SCF iteration. It also contains the measurement of self-consistency error defined as

$$\frac{\|V_{\text{in}} - V_{\text{out}}\|}{\|V_{\text{in}}\|}, \tag{36}$$

where V_{in} is the sum of potential terms in (5) that are functional of the electron density or density matrix at the beginning of each SCF iteration, and V_{out} is the corresponding new potential sum evaluated from the solution of the linear eigenvalue problem. The Frobenius norm of the eigenpair residual $HX - X\Lambda$ is also printed out, where X contains all the desired eigenvectors and Λ is a diagonal matrix containing the corresponding eigenvalues.

4.2. Visualization and post-processing

In addition to the interactive output displayed in MATLAB's command line window, the `scf` function also returns a number of output variables that can be further examined and visualized. The returned `Molecule` object (which in the example given here overwrites the input argument `mol` includes the atomic forces computed for each atom at the end of the SCF iteration. We can examine these forces simply by typing

```
mol.xyzforce
```

on the command line, which produces

```
ans =

    0.0000    0.0000    0.0000
    0.0021    0.0021    0.0021
   -0.0021   -0.0021    0.0021
    0.0021   -0.0021   -0.0021
   -0.0021    0.0021   -0.0021
```

The returned Hamiltonian object `H` contains the electron density ρ as one of its attributes, which we can visualize by using a third party volume rendering function `vol3d` included in KSSOLV or simply MATLAB's isosurface rendering function `isosurface` as shown in Listing 6. The `fftshift` function used in the listing is called to re-center ρ to the middle of the unit cell (instead of the origin of the Cartesian grid).

These renderings are shown in Fig. 1.

We can also show a squared amplitude of Kohn-Sham orbital ψ_j. This requires some post-processing of the returned `Wavefun`

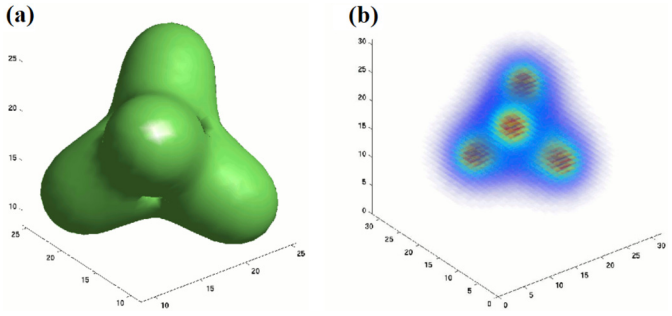


Fig. 1. (a) An isosurface rendering of the converged electron density of SiH_4. (b) A volume rendering of the converged electron density of SiH_4.

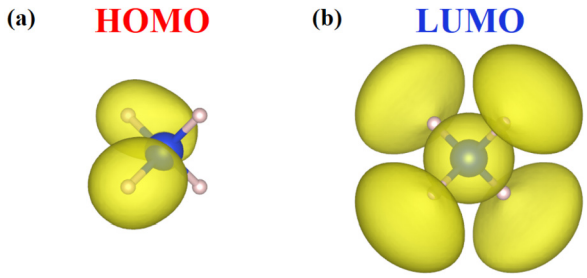


Fig. 2. The isosurfaces of HOMO and LUMO produced by VESTA. (a) HOMO of SiH_4. (b) LUMO of SiH_4.

Listing 7 Wavefunction post-processing and visualization of the HOMO.

```
n1=mol.n1;n2=mol.n2;n3=mol.n3;
homo = mol.nel/2;
lumo = homo + 1;
XG = X.psi(:,homo);
F = KSFFT(mol);
XR = (F'*XG)*sqrt(mol.vol);
X2 = abs(XR).^2;
pos = poscar(mol);
outchg('SiH4_HOMO', pos, reshape(X2,n1,n2,n3));
```

object X. The post-processing involves using FFT to transform the default compact reciprocal space representation of the wavefunction XG to a real space vector representation XR, evaluating its magnitude square as abs(XR).^{2}, and reshaping the resulting vector into a 3D array. KSSOLV provides a utility function poscar to write the magnitude square of the reshaped wavefunction to a text file that can be read by other visualization software tools such as the VESTA [86].

Listing 7 shows how post-processing is performed to write the magnitude square of the highest occupied molecular orbital (HOMO) to a file named SiH4_HOMO. A similar set of commands can be used to write the lowest unoccupied molecular orbital (LUMO) to another file. The HOMO and LUMO can be subsequently visualized by using the VESTA software as shown in Fig. 2.

The returned info argument is a MATLAB structure that contains several fields.

```
>> info

info =

  struct with fields:

      Eigvals: [4{\texttimes}1 double]
      Etotvec: [10{\texttimes}1 double]
    SCFerrvec: [10{\texttimes}1 double]
         Etot: -6.2542
```

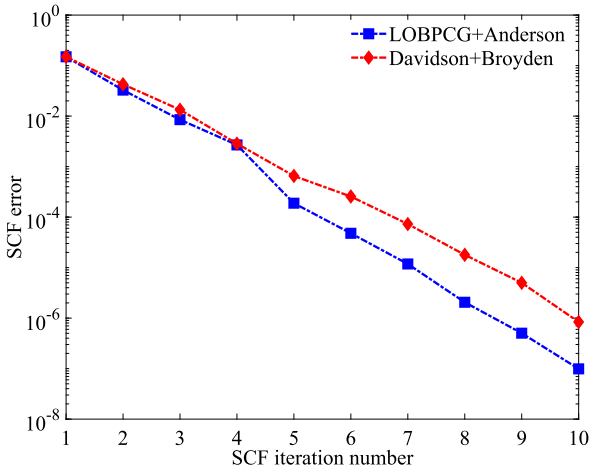


Fig. 3. The change of SCF error (36) with respect to SCF iteration number. Two combinations of diagonalization algorithm and mixing method are given, 1. LOBPCG with Anderson, 2. Davidson with Broyden. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We can plot the convergence history of the SCF iteration by simply using

```
x = [1:length(options.maxscfiter)]
fig1 = semilogy(info.SCFerrvec_lob,'-s'
       ,info.SCFerrvec_dia,'-d');
legend([fig1(1)fig1(2)],{'LOBPCG+Anderson'
       ,'Davidson+Broyden'});
xlabel('SCF iteration number','FontName',
       'Times New Roman')
ylabel('SCF error','FontName',
       'Times New Roman')
set(gca,'XTick',x)
```

We can also use the Eigvals information from the info argument to obtain the DOS (Density of States). The post-processing includes setting some parameters to get the energy range and using either the Gaussian or the Lorentzian spread function to create a smooth DOS curve from info.Eigvals. Listing 8 gives a simple script for carrying out such type of post-processing. The DOS curves produced for four different systems (SiH_4, C_6H_6, Si_{64} and C_{60}) are shown in Fig. 4.

4.3. Modifying options and algorithms

KSSOLV 2.0 allows users to choose and experiment with different algorithms or algorithmic components for solving the Kohn-Sham problem. For example, instead of calling the scf function, one can call the trdcm function, which implements the trust region regularized DCM algorithm discussed in section 2.2.1, as

```
[mol,H,X,info] = trdcm(mol);
```

Both the scf and trdcm functions accept an additional option argument that allows users to alter the default algorithms and parameters used by these functions.

The optional argument can be first created by calling the setksopt function which returns a MATLAB structure that contains default algorithmic choices and parameters listed in Listing 9.

We can change, for example, the algorithm for solving the linear eigenvalue problem in each SCF iteration from LOBPCG to Davidson, and the quasi-Newton algorithm (charge mixing scheme) used to accelerate the SCF iterations from Anderson to Broyden by using the commands given in Listing 10 to modify the options

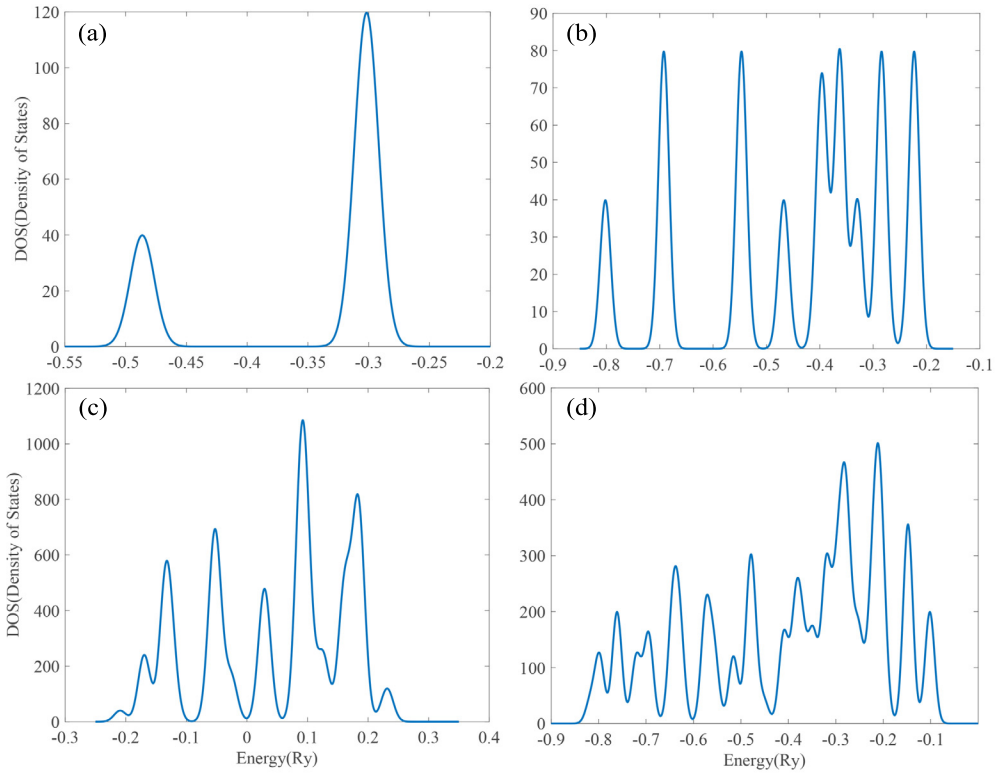


Fig. 4. DOS(Density of States) by using Gaussian function, which describes the proportion of states that are to be occupied by the system at each energy. (a)SiH_4, (b)C_6H_6, (c)Si_{64}, (d)C_{60}.

Listing 8 Energy post-processing and visualization of the DOS (density of states.)

```
ev = info.Eigvals;
nx = 1000;
sigma = 0.01;
% Initialization
[m,n]    = size(ev);
ne       = m;
emin     = -1;%min(ev);
emax     = 1;%max(ev);
xgrid    = (emax - emin)/(nx - 1);
dos      = zeros(nx,2);
% Get the x distribution of the energy
for ix = 1 : nx
  dos(ix,1) = emin + (ix - 1)*xgrid;
end
% Calculate DOS
for ie = 1 : ne
  for ix = 1 : nx
    x = emin + (ix - 1)*xgrid - ev(ie);
    %if(Gaussian)
    dos(ix,2) = dos(ix,2) + 1/(sigma*sqrt(2*pi))
    *exp(-x^2/(2*sigma^2));
    %if(Lorentzian)
    %dos(ix,2) = dos(ix,2) + sigma/(pi*(x^2+sigma^2));
  end
end
```

structure and passing it to the `scf` function along with the `mol` object.

Fig. 3 shows these changes lead to a slightly difference convergence behavior of the SCF iteration (the red curve) although the difference is relatively small in this particular case.

We can see from Fig. 3 that the SCF iteration did not converge to the default accuracy requirement specified by the parameter `options.scftol`, which is set to 10^{-8}. To reach that level of accuracy, we can rerun the `scf` function by using the wavefunction `X` and electron density `rho` returned from the previous run

Listing 9 Option structure returned from the `setksopt` function.

```
          verbose: 'off'
        eigmethod: 'lobpcg'
       maxscfiter: 10
       maxdcmiter: 10
       maxinerscf: 3
        maxcgiter: 10
      maxeigsiter: 300
           scftol: 1.0000e-08
           dcmtol: 1.0000e-08
            cgtol: 1.0000e-09
         eigstol: 1.0000e-10
       what2mix: 'pot'
        mixtype: 'anderson'
         mixdim: 9
        betamix: 0.8000
          brank: 1
             X0: []
           rho0: []
         degree: 10
          force: 1
       ishybrid: 0
         useace: 0
           Vexx: []
     maxphiiter: 5
         phitol: 1.0000e-08
         dfrank: 0
        ncbands: 0
     relaxmethod: 'fminunc'
        relaxtol: 1.0000e-04
  factorOrbitals: 1
       davsteps: 3
        ngbands: 0
```

as the starting guess. This can be achieved by simply setting `options.X0` and `options.rho0` to the previously returned wavefunction and electron density.

```
options.X0 = X;
options.rho0 = H.rho;
```

Listing 10 Choosing a different eigensolver and charge mixing scheme.

```
options = setksopt();
options.eigmethod = 'davidson';
options.mixtype = 'broyden';
[mol1,H1,X1,info1] = scf(mol,options);
```

Listing 11 Convergence is reached after rerunning `scf` with the wavefunction and electron density initialized to the approximation produced from the first `scf` call.

```
Regular SCF for Pure DFT
Beging SCF calculation for SiH4...
SCF iter   1:
eigtol =   1.000e-02
Rel Vtot Err    =            1.146e-07
Total Energy    = -6.2542498381078e+00
...
SCF iter   4:
eigtol =   1.257e-09
Rel Vtot Err    =            4.190e-09
Total Energy    = -6.2542498381079e+00
Convergence is reached!
Elapsed time is 1.005311 seconds.
Etot            = -6.2542498381079e+00
Eone-electron   = -5.3304963587462e+00
Ehartree        =  3.2198596360275e+00
Exc             = -2.5983987591201e+00
Eewald          = -1.5452143562691e+00
Ealphat         =  0.0000000000000e+00
---------------------------------------
Total time used =            4.563e+00
||HX-XD||_F     =            2.562e-09
```

Listing 12 Constructing the ACE operator.

```
W = ApplyVexx(X);
M = X' * W;
M = (M + M')/2;
R = chol(-M);
Y = W / R;
ApplyVexxACE = @(x) -Y * (Y' * x);
```

After calling the `scf` function with the modified option as an input, we can reach convergence as reported in Listing 11.

4.4. Algorithm prototype and modification

KSSOLV is designed to enable researchers to easily modify existing algorithms and prototype new algorithms. To a large extent, this feature is facilitated by the object-oriented programming model supported in MATLAB. By creating Hamiltonian and wavefunction objects and overloading the basic linear algebra operations with these objects as operands, one can literally translate mathematical expressions into MATLAB codes in KSSOLV in a few minutes. To give an example, let us take a look at the implementation of the ACE operator for hybrid functional DFT calculation in KSSOLV which is shown in Listing 12.

The ACE operator is defined by (29) which, once $\psi_j(\mathbf{r})$'s are discretized and represented by columns of the matrix X, can also be written in matrix form as

$$\hat{V}^{\mathrm{ACE}} = -WM^{-1}W^*, \tag{37}$$

where

$$W = \hat{V}^{\mathrm{HSE}}(X)X, \tag{38}$$

with $\hat{V}^{\mathrm{HSE}}(X)$ being the matrix representation of the Hartree-Fock exchange operator and $M = X^*W$. Because $-M$ is Hermitian positive definite, we can rewrite (37) in a symmetric form by performing a Cholesky factorization of $-M$, i.e., $-M = RR^*$ with R being upper triangular, and expressing $\hat{V}^{\mathrm{ACE}}$ as $\hat{V}^{\mathrm{ACE}} = -YY^*$ with $Y = WR^{-1}$.

Listing 13 Profiling for a HSE06 calculation.

```
profile on;
testHSE06;
save profHSE06 p
profile off;
```

In Listing 12, we apply the function `ApplyVexx`, which implements (38), to the `Wavefun` object `X` to obtain another `Wavefun` object `W`. Even though `X` and `W` are `Wavefun` objects, we can treat them as matrices and multiply them together in the second line of Listing 12 to obtain the matrix $M = X^*W$. Line 3 in Listing 12 is used to ensure `M` is numerically Hermitian before the Cholesky factorization function `chol` is applied to `-M`. The inverse of the Cholesky factor `R` is applied to `W` to yield the `Wavefun` object `Y` by solving a set of linear equations using the MATLAB `/` operator. The `Y` object is then used to define a function handle `ApplyVexxACE` that can be applied to any `Wavefun` object of matching dimensions without explicitly forming the ACE operator.

4.5. Performance profiling

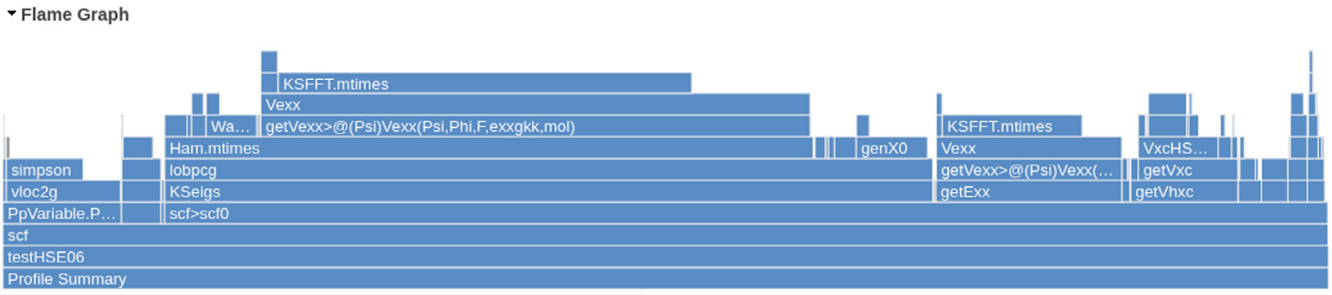
MATLAB provides a convenient performance profiling tool that allows us to easily analyze the performance features of KSSOLV functions and identify potential computational bottlenecks. For example, to profile the performance of the HSE06 calculation contained in a testing script named `testHSE06.m`, we can simply issue the following several commands listed in Listing 13.

MATLAB provides a viewer in the Windows Visual interface that allows us to clearly see the hierarchical relationship among different computational components as well as which function takes most of the time. We can further zoom into the most time-consuming function and identify the line number of the code that takes most of the time within that function.

For example, Fig. 5(a) displays a Flame graph of KSSOLV functions called by `testHSE06`, the most time consuming function is `getVexx`, which is used to calculate the exchange potential. By clicking on the block containing this function name, we obtain a table shown in Fig. 5(b), which lists the line numbers of the most time consuming functions contained in `getVexx`. If we click the line number associated with a particular function, we can step in the code of that function and analyze the computation performed in that function.

Once we have identified the computational bottleneck of HSE06 calculation, which is in the evaluation of the exchange term, we can optimize the performance of KSSOLV by seeking alternative implementations of the functions in question or using alternative algorithms. For example, as we discussed in section 2.3 and 4.4, the use of the ACE algorithm to refactor the exchange operator can significantly reduce the computational complexity of applying the Hartree-Fock exchange operator to a set of wavefunctions in the hybrid functional DFT calculation. Table 2 gives a direct comparison between the cost of hybrid functional DFT calculations with and without the use of ACE. We can clearly see that after using ACE, the total amount of wall clock time used by `scf0` is reduced from 76 seconds to 28 seconds. This is mainly due to a significant reduction in time spent in the `lobpcg` eigensolver used for the outer iteration. With the use of ACE, the function `getVexx` which is used to update the Fock exchange operator, is called only 5 times (in the 5 outer `scf0` iterations), whereas 187 such calls are made in the hybrid functional DFT calculation without using ACE. Furthermore, the use of the ACE allows us to significantly reduce the number of FFTs used to apply the Fock exchange operator to a set of wavefunctions. In the ACE enabled hybrid functional cal-

(a) Profile Summary (Total time: 113.237 s)



(b)

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
testHSE06	1	113.237	0.057	
scf	1	113.179	0.034	
scf>scf0	5	99.320	0.241	
getVexx>@(Psi	187	67.571	0.013	
Vexx	187	67.558	14.452	
KSeigs	60	65.600	0.013	
lobpcg	60	65.587	0.444	
Ham.mtimes	200	57.037	0.269	

(c) Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
15	VexxPsi(:,i) = sum	736	53.489	79.2%	
6	Psi = F' * Y.psi;	187	9.062	13.4%	
12	F2 = KSFFT(mol,1);	187	2.136	3.2%	
20	Y.psi = F * VexxPs	187	1.302	1.9%	
17	VexxPsi = -0.25 *	187	0.569	0.8%	
All other lines			1.000	1.5%	
Totals			67.558	100%	

Fig. 5. Profiling of a HSE06 calculation, (a) The overall profile summary flame graph, (b) Functions hot spot analysis, including the called times of sub-functions, (c) Time corresponding to each line of `getVexx`.

Table 2
Cost comparison between HSE and HSE-ACE calculations. The system calculated here is SiH_4 with E_{cut} set to 20 Hartree.

Function name	Calls numbers	Time(s)
`scf0`(HSE)	5	75.965
`lobpcg`(HSE)	59	48.779
`getVexx`(HSE)	187	49.969
`KSFFT.mtimes`(HSE)	1851	38.448
`scf0`(HSE-ACE)	5	28.055
`lobpcg`(HSE-ACE)	59	13.778
`getVexx`(HSE-ACE)	5	1.554
`KSFFT.mtimes`(HSE-ACE)	55	1.362
`calculateACE`(HSE-ACE)	5	1.554

culation, only 55 times FFTs are used to construct the ACE operator, whereas 1851 FFTs are performed when the Fock exchange operator is applied to a set of wavefunctions in each step of the LOBPCG eigensolver. Table 2 also shows that the overhead incurred in constructing the ACE operator is relatively small, i.e. 1.5 seconds (used by `calculateACE`) out of 28 seconds used by `scf0`.

5. Results and discussion

In this section, we give some examples of a few applications that can be studied with KSSOLV and demonstrate its accuracy and performance. The descriptions of these systems are listed in Table 3.

5.1. Accuracy

We first use KSSOLV to perform ground state total energy and atomic force calculations, band structure analysis and geometry optimization for a few molecules and solids. In all these runs, we set the inner SCF convergence tolerance to 10^{-7} for calculations that use LDA and PBE functionals, and 10^{-6} for outer SCF convergence tolerance when using the HSE06 functional. We use QUANTUM ESPRESSO as the baseline for comparison in assessing the accuracy of KSSOLV.

5.1.1. Total energy and atomic forces

When comparing with the QE (QUANTUM ESPRESSO) results, we compute the total energy difference per atom as well as the maximum difference in atomic forces, which are defined by

$$\Delta E = \left(E_{\text{tot}}^{\text{KSSOLV}} - E_{\text{tot}}^{\text{QE}}\right)/N_{\text{A}},$$

$$\Delta F = \max_{I} \left\| F_I^{\text{KSSOLV}} - F_I^{\text{QE}} \right\|,$$

where $E_{\text{tot}}^{\text{KSSOLV}}$ and $E_{\text{tot}}^{\text{QE}}$ are converged total energy levels returned from KSSOLV and QE respectively, N_A is the total number of atoms in each system, and I is an atom index.

To check the accuracy systematically, we measure ΔE and ΔF for each system at several plane-wave cut-off energy (E_{cut}) levels (from 10 to 100 Hartree). We also use three types of psedopotential and exchange-correlation functional combinations in KSSOLV 2.0, which are LDA-HGH, PBE-ONCV and HSE06, respectively. The total energy differences for test systems are plotted in Fig. 6. The solid black square lines correspond to the LDA-HGH exchange-correlation functional and psedopotential combination, the solid red circle lines correspond to PBE-ONCV, and the solid blue triangle lines correspond to HSE06.

We observe that, in general, the difference between the converged KSSOLV and QE total energies per atom is on the order of between 10^{-6} and 10^{-4} Hartree, which is sufficiently small. For Si_{64}, the energy difference is slightly larger (on the order of 10^{-3} Hartree) at some plane-wave cut-off levels. However, these differences are acceptable since they are around chemical accuracy, which is defined to be 1kcal/mol or 10^{-3} Hartree, and are sufficient for most applications. In previous studies [84], we also compared differences in cohesive energies for several test problems and showed that they match well.

In Fig. 7, we plot the maximum difference in atomic forces between KSSOLV and QE for all test systems. The magnitude of force

Table 3
The performance of KSSOLV on a set of test problems. System: including solid, molecule, nanotube, n_a: number of atoms, Cell dim: the unit cell size with three dimensions, N_e: number of electrons, n_k: number of k-points, Functional: exchange-correlation functional, Ecut: cut-off energy, N_r: grids number in real space, N_g: grids number in reciprocal space, Scftol : the converge limit of inner SCF, Scf-iter: the iteration number of inner SCF, Phitol : the converge limit of outer SCF, Phi-iter: the iteration number of outer SCF, Total time: the total wall clock time of each calculation.

System	n_a	Cell dim	N_e	n_k	Functional	Ecut (Ha)	N_r	N_g	Scftol	Scf-iter	Phitol	Phi-iter	Total time (sec)
SiH_4	5	20	8	1	PBE	20	531441	34265	10^{-7}	14	-	-	17.544
C_6H_6	12	$22.4 \times 24.9 \times 30.2$	30	1	PBE	20	1121302	72079	10^{-7}	18	-	-	154.358
Si_{64}	64	20.52^3	256	1	PBE	20	571,787	37,073	10^{-7}	19	-	-	642.018
C_{60}	60	24.57^3	240	1	PBE	20	970,299	63,317	10^{-7}	20	-	-	1174.029
Si_{216}	216	30.78^3	864	1	PBE	20	1906624	124289	10^{-7}	19	-	-	8769.963
Si_{64}	64	20.52^3	256	1	HSE	20	571,787	37,073	10^{-7}	19	10^{-6}	3	79077.64
Si_{64}	64	20.52^3	256	1	HSE-ACE	20	571,787	37,073	10^{-7}	19	10^{-6}	4	3493.511
C_{60}	60	24.57^3	240	1	HSE	20	970299	63317	10^{-7}	20	10^{-6}	3	184672.234
C_{60}	60	24.57^3	240	1	HSE-ACE	20	970299	63317	10^{-7}	20	10^{-6}	4	5058.289
CNT661	60	$38 \times 38 \times 4.6$	96	1	PBE	20	399475	25485	10^{-7}	19	-	-	146.720
Si_8	8	10.216^3	96	64	PBE	20	74088	4553	10^{-7}	14	-	-	345.123
Cu_4	4	6.8308^3	32	64	PBE	30	39304	2517	10^{-5}	18	-	-	512.685

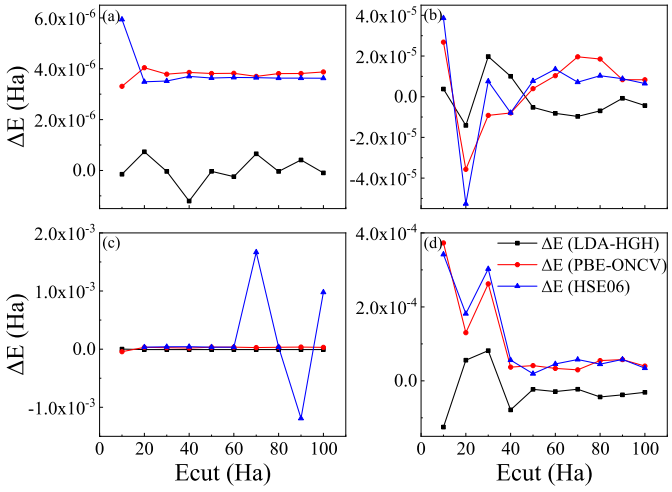


Fig. 6. Total energy difference between KSSOLV and QE for (a) silane (SiH_4) molecule, (b) benzene (C_6H_6) molecule, (c) bulk silicon Si_{64} and (d) fullerene (C_{60}) molecule at different plane-wave cut-off energy levels.

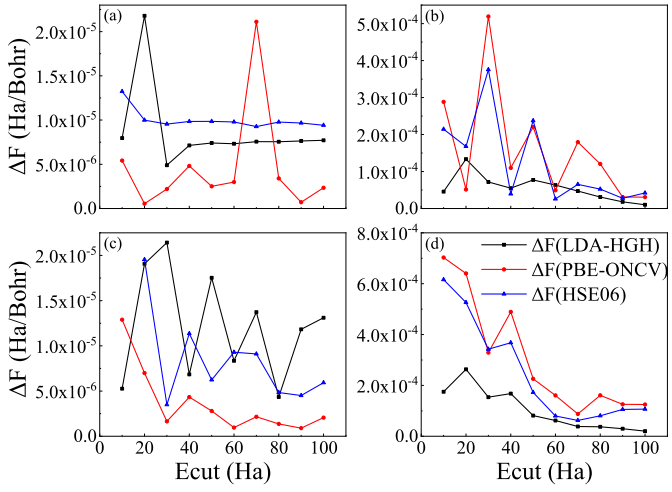


Fig. 7. Maximum difference in atomic forces between KSSOLV and QE for (a) silane (SiH_4) molecule, (b) benzene (C_6H_6) molecule, (c) bulk silicon Si_{64} and (d) fullerene (C_{60}) molecule at different plane-wave cut-off energy levels.

difference is generally small and within the range of 10^{-6} to 10^{-4} Hartree/Bohr. In some cases, the difference is slightly larger when E_{cut} is relatively small, but becomes sufficiently small when E_{cut} reaches 50 Ha or so.

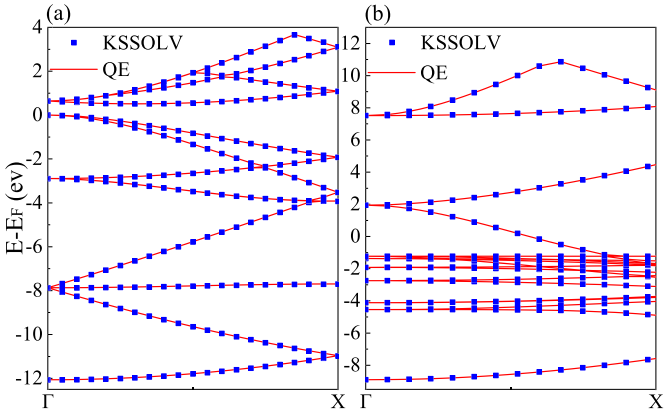


Fig. 8. The band structures calculated by KSSOLV and QE using the PBE functional for (a) bulk silicon Si_8 and (b) bulk copper Cu_4 system. The cut-off energies are 20 Hartree and 30 Hartree for Si_8 and Cu_4, respectively. All energy levels have been shifted to keep the Fermi energy at zero ev.

5.1.2. Band structure

Because KSSOLV 2.0 facilitates k point samplings in the first Brillouin zone for solids, we can use it to compute band structures of solids and compared them with the results obtained from QE also. In Fig. 8, we plot the band structure of Si_8 and Cu_4 respectively between the Γ and X k-points. In both cases, the number of k-points used in the SCF calculation is 64(4x4x4), we observe that the band structures obtained from KSSOLV are in excellent agreement with those obtained from QE, and the average numerical difference of band structure between the two packages is about 10^{-9}. We can clearly see a band gap between the highest valence band and the lowest conducting band for Si_8 which confirms the fact that Si is a semiconductor. No band gap can be seen in Fig. 8(b) for Cu_4, this is consistent with the previous knowledge that Cu is a metal.

5.1.3. Geometry optimization

We use KSSOLV 2.0 to optimize the geometry of an isolated water (H_2O) molecule and compared the optimal H-O bond length, and the optimal angle between two H-O bonds with the corresponding experiment values. The initial bond lengths between the H and O atoms are set to 0.98523 and 1.53953 Å, respectively, and the initial bond angle is set to 38.5624°. These are slightly different from the experiment value [87] of 0.957 Åfor the bond length and 104.5° for the bond angle. The BFGS algorithm implemented in MATLAB Optimization toolbox function `fminunc` is used to perform the optimization. The convergence of bond length and bond angle to the experimentally observed values in the KSSOLV geometry optimization function `relaxatoms` is shown in Fig. 9. We can

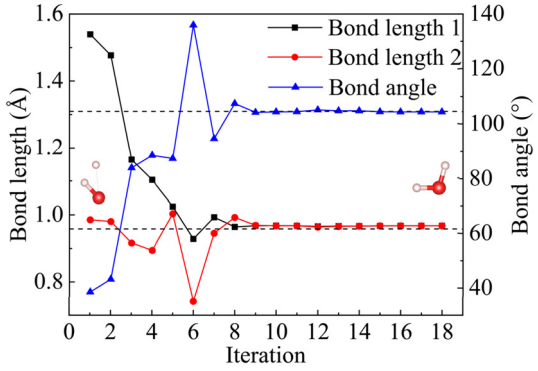


Fig. 9. Geometric optimization for H_2O, the solid black square line, the solid red dotted line, and the blue triangle represent two bond lengths and bond angle respectively, the black dotted line represents the experimental value. The total number of iteration steps is 18 and the initial and final structures are given at the first and the last iteration step. The cut-off energy of the calculation is 60 Hartree.

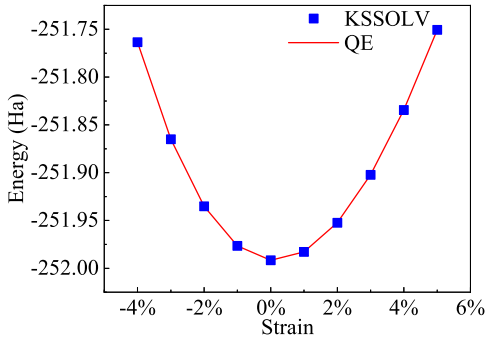


Fig. 10. Geometric optimization for bulk silicon Si_{64} system, the energy change with strain, KSSOLV (the blue square), QE (solid red line), the unit of strain change is one percent of the overall structure, from compressive strain (-4%) to tensile strain ($+5\%$), the cut-off energy of these calculations is 60 Hartree.

clearly see that convergence is reached after 9 geometry optimization steps. And the error of the KSSOLV calculation result differs from the experimental value by only one percent.

To give another example, we perform a geometry optimization Si_{64} with respect to the size of the unit cell. The change in unit cell size corresponds to the change in the strain applied to the solid. To perform the optimization, we sample several unit cell sizes that correspond to $0-6\%$ changes in applied strain, and compute the ground state of Si_{64} contained in these unit cells. The results are compared with those obtained from QE. Fig. 10 shows that the KSSOLV results match well with QE results. These results clearly show the energy is the lowest at zero strain.

5.2. Performance

In this section, we report the performance of KSSOLV by applying it to a set of benchmark problems listed in Table 3. The version of MATLAB we used is R2021a and the benchmark is run on a Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz with 40 maximum threads. We perform ground state calculation for both molecules and solids using either PBE or HSE06 functionals specified in Table 3. For hybrid functional DFT calculations, the HSE exchange-correlation functional and a two-level SCF procedure are used. The standard SCF calculation is used as the inner iteration to reach self consistency in the electron density for a fixed Fock exchange potential. In the outer SCF iteration that we refer to as the Phi-iteration, the Fock exchange potential and energy are updated. Thus, two different convergence criteria are used in the inner and outer iteration.

We report the wall clock time it takes to complete the calculation as well as the number of SCF iterations required to reach convergence. The convergence criterion (i.e., the SCF error tolerance) for each case is listed in the table also.

The first four systems listed in Table 3 were used in the previous subsections to demonstrate the accuracy of KSSOLV. These systems are relatively small and can be solved between tens of seconds to tens of minutes. The number of iterations required to reach convergence and the wall clock time it takes generally increase with the system size. The largest system we tested is the Si_{216} cluster with 216 atoms and 864 electrons. More than 1 million plane-waves are used in this calculation that employs the PBE functional. The entire calculation took more than two hours.

The HSE06 functional is used for Si_{64} and C_{60} to perform hybrid functional calculations for these systems with and without using the ACE method. We can see from Table 3 that, without using ACE, the hybrid functional calculations for these systems are two orders of magnitude more expensive than the corresponding PBE calculations for the same systems. When ACE is used, the hybrid functional calculations are only $4\sim5$ times more expensive than the corresponding PBE calculations. In previous studies [21], we also compared different diagonalization algorithms.

In addition to insulators and semiconductors, we also measure the performance of KSSOLV on a metallic system Cu_4. The calculation, which uses 64 k-points, can be completed in less than 10 minutes.

Because MATLAB can take advantage of multiple threads on a many-core CPU to parallelize many computational kernels, we can speed up KSSOLV calculation on such a CPU without additional parallelization effort. In Fig. 11, we report the parallel scaling of KSSOLV when it is used to compute the ground states of four different systems using the PBE functional. We observe that the total wall clock time can be reduced by a factor of 5 when the number of threads is increased from 1 to 8. Increasing the number of threads further to 16 leads to an additional reduction in wall clock time. However, the reduction factor is much smaller. By default, MATLAB generally will try to use the maximum number of threads available on the machine being used. As a result, KSSOLV can benefit from the maximum shared memory concurrency available on any many-core CPUs.

In addition to reporting the total wall clock time, we also show a breakdown of timing among several key computational components of KSSOLV. The Hamiltonian wavefunction multiplication (labeled by `Ham.mtimes`), which performs `HX = H*X` as an overloaded matrix-matrix multiplication between a `Ham` object `H` and a `Wavefun` object `X` as explained in section 3.2, constitutes the largest cost. This is followed by the cost of `Wavefun.mtimes` which performs dense matrix-matrix multiplications between two `Wavefun` objects or between a `Wavefun` object and a regular matrix. The `Wavefun.subsref` function, which is used to extract a subset of wavefunctions, involves mainly data movement and copying. Such data movement cannot be easily parallelized. Hence, the timing associated with `Wavefun.subsref` does not decrease as the number of threads increases. The `VxcPBE` function, which is used to evaluate the PBE exchange-correlation energy and potential, takes a small fraction of the time. The function `vlov2g` is used to convert local pseudopotential on a non-uniform grid in real-space to a uniform grid in the reciprocal space in the initialization of the pseudopotential. This one-time cost can be relatively large for small systems, but becomes negligible when the system size becomes sufficiently large.

6. Conclusion and outlook

KSSOLV 2.0 preserves the main object-oriented design features of the original KSSOLV software toolbox for solving the Kohn-Sham

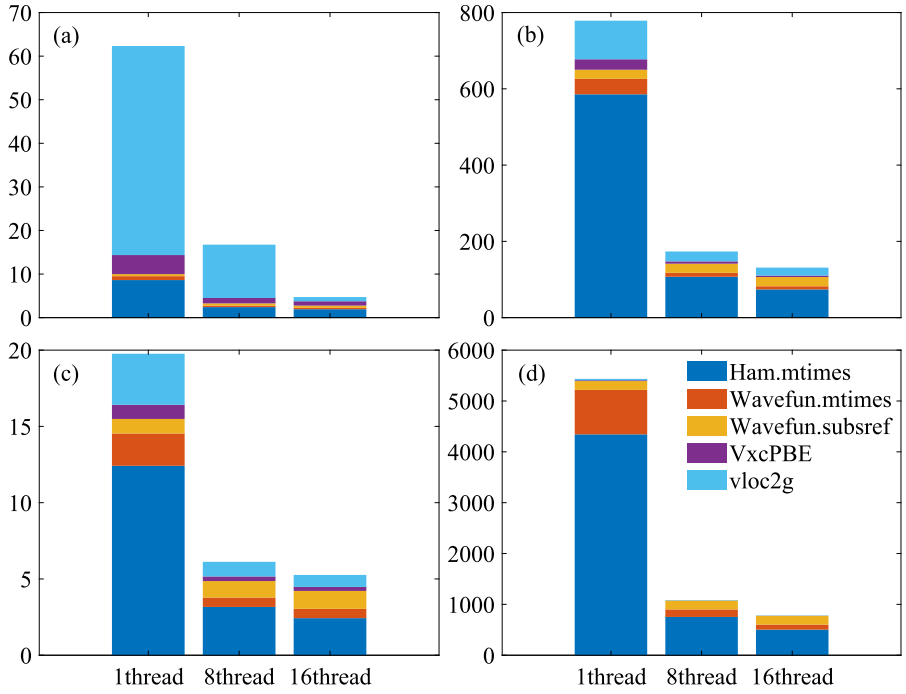


Fig. 11. The profiling results of different systems PBE-calculation within 1, 8, 16 threads, including Ham.mtimes, Wavefun.mtimes, Wavefun.subsref, VxcPBE, vlov2g. Systems: (a) silane (SiH_4) molecule, (b) benzene (C_6H_6) molecule, (c) bulk silicon Si_8, (d) bulk silicon Si_{64}. All of these results come from the whole calculation process.

equations. Such design features make it easy for users to set up a problem and obtain a solution. They also enable developers to easily prototype and test new algorithms. The new version contains more advanced algorithms such as ACE, PC-DIIS, ISDF for hybrid functional DFT calculations, and new functionalities such as geometry optimization. The software produces accurate results that are consistent with those produced by other plane-wave based KS-DFT software such as QE. It is efficient for performing KS-DFT electronic structure calculations for small to medium sized problems. It is a great teaching tool that can help students and researchers quickly learn how to analyze the electronic structure of molecules and solids. At the same time, it can also be a useful research tool in chemical and materials sciences for analyzing properties of interesting materials or chemical systems and for developing more efficient numerical methods. Although KSSOLV 2.0 is designed to perform ground state DFT and geometry optimization calculations, several new developments are already underway to include new functionalities in the next release. In particular, we plan to include functionalities to allow users to perform time-dependent density functional theory (TDDFT) calculations and post-DFT calculations such as computing GW[88] quasi-particle energies and eigenvalues and eigenvectors of the Bethe-Salpeter Hamiltonian [89]. In addition, we will integrate KSSOLV with machine learning tools to accelerate the materials design and discovery process.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service or company that could be construed as influencing the position presented in, or the review of, the manuscript titled "*KSSOLV 2.0: An efficient MATLAB toolbox for solving the Kohn-Sham equations with plane-wave basis set*".

Acknowledgements

This work is partly supported by the National Natural Science Foundation of China under the Grant No. 22173093 (W.H.), 22073086 (L.J.), 21803064 (Z.D.) and 21688102 (J.Y.), by the School of Future Technology under Grant No. KF2020003 (W.H.), by the Chinese Academy of Sciences Pioneer Hundred Talents Program under Grant No. KJ234007002 (W.H.), by the National Key Research and Development Program of China under the Grant No. 2016YFA0200604 (J.Y.), the Anhui Initiative in Quantum Information Technologies under Grant No. AHY090400 (J.Y.), the Center of Chinese Academy Project for Young Scientists in Basic Research under Grant No. YSBR-005 (W.H.), the School of Future Technology under Grant No. SK2340002001 (W.H.), the Fundamental Research Funds for the Central Universities under Grant No. WK2340000091 (W.H.) and WK2060000018 (W.H.) from University of Science and Technology of China. This work was also supported by the Scientific Discovery through Advanced Computing (SciDAC) program and the Center for Applied Mathematics for Energy Research Applications (CAMERA) funded by U.S. Department of Energy Office of Science, Advanced Scientific Computing Research under Contract No. DE-AC02-05CH11231 (L.L., C.Y.). The authors thank the Hefei Advanced Computing Center, the Supercomputing Center of Chinese Academy of Sciences, the Supercomputing Center of USTC and the National Energy Scientific Computing Center (NERSC) for the computational resources. We would like to thank Temo Vekua, Reza Fazel-Rezai, Kajia Ruan and Wei Chen at MathWorks for helpful suggestions.

References

[1] C. Yang, J.C. Meza, B. Lee, L.-W. Wang, ACM Trans. Math. Softw. 36 (2) (2009) 1–35.
[2] P. Hohenberg, W. Kohn, Phys. Rev. 136 (1964) B864–B871.
[3] W. Kohn, L.J. Sham, Phys. Rev. 140 (1965) A1133.
[4] J. Lu, L. Ying, J. Comput. Phys. 302 (2015) 329–335.
[5] C. Kelley, J. Bernholc, E. Briggs, S. Hamilton, L. Lin, C. Yang, J. Comput. Phys. 409 (2020) 109322.
[6] L. Lin, C. Yang, SIAM J. Sci. Comput. 35 (5) (2013) S277–S298.

[7] M. Shao, L. Lin, C. Yang, F. Liu, F.H. Da Jornada, J. Deslippe, S.G. Louie, Sci. China Math. 59 (8) (2016) 1593–1612.
[8] A. Damle, L. Lin, L. Ying, SIAM J. Sci. Comput. 36 (6) (2014) A2929–A2951.
[9] L. Lin, L. Zepeda-Nunez, Multiscale Model. Simul. 17 (4) (2019) 1274–1300.
[10] J.R. McClean, F.M. Faulstich, Q. Zhu, B. O'Gorman, Y. Qiu, S.R. White, R. Babbush, L. Lin, New J. Phys. 22 (9) (2020) 093015.
[11] W. Hu, J. Liu, Y. Li, Z. Ding, C. Yang, J. Yang, J. Chem. Theory Comput. 16 (2) (2020) 964–973.
[12] J. Liu, W. Hu, J. Yang, J. Chem. Phys. 154 (6) (2021) 064101.
[13] L. Lin, Z. Xu, L. Ying, Multiscale Model. Simul. 15 (2017) 29–55.
[14] M. Ulbrich, Z. Wen, C. Yang, D. Klockner, Z. Lu, SIAM J. Sci. Comput. 37 (4) (2015) A1975–A2002.
[15] Z. Wen, C. Yang, X. Liu, Y. Zhang, J. Sci. Comput. 66 (3) (2016) 1175–1203.
[16] D. Hamann, Phys. Rev. B 88 (8) (2013) 085117.
[17] C. Hartwigsen, S. Gredecker, J. Hutter, Phys. Rev. B 58 (7) (1998) 3641.
[18] F. Liu, L. Lin, D. Vigil-Fowler, J. Lischner, A.F. Kemper, S. Sharifzadeh, H. Felipe, J. Deslippe, C. Yang, J.B. Neaton, et al., J. Comput. Phys. 286 (2015) 1–13.
[19] J. Hu, B. Jiang, L. Lin, Z. Wen, Y.-x. Yuan, SIAM J. Sci. Comput. 41 (4) (2019) A2239–A2269.
[20] Y. Gao, C. Fu, W. Hu, J. Yang, J. Phys. Chem. Lett. 13 (2021) 1–11.
[21] Z. Zhang, S. Jiao, J. Li, W. Wu, L. Wan, X. Qin, W. Hu, J. Yang, Chin. J. Chem. Phys. 34 (5) (2021) 552–564.
[22] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, G. Scalmani, V. Barone, G.A. Peters- son, H. Nakatsuji, X. Li, M. Caricato, A.V. Marenich, J. Bloino, W.G. Janesko, R. Gomperts, B. Mennucci, H.P. Hratchian, J.V. Ortiz, A.F. Izmaylov, J.L. Sonnenberg, D. Williams- Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V.G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J.A. Montgomery Jr., J.E. Peralta, F. Ogliaro, M.J. Bearpark, J.J. Heyd, E.N. Brothers, K.N. Kudin, V.N. Staroverov, T.A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A.P. dell Ren-, J.C. Burant, S.S. Iyengar, J. Tomasi, M. Cossi, J.M. Millam, M. Klene, C. Adamo, R. Cammi, J.W. Ochterski, R.L. Martin, K. Morokuma, O. Farkas, J.B. Foresman, D.J. Fox, Gaussian'16 Revision C.01, Gaussian Inc., Wallingford CT, 2016.
[23] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, et al., Comput. Phys. Commun. 181 (9) (2010) 1477–1489.
[24] Y. Shao, Z. Gan, E. Epifanovsky, A.T. Gilbert, M. Wormit, J. Kussmann, A.W. Lange, A. Behn, J. Deng, X. Feng, et al., Mol. Phys. 113 (2) (2015) 184–215.
[25] Y. Zhang, B. Suo, Z. Wang, N. Zhang, Z. Li, Y. Lei, W. Zou, J. Gao, D. Peng, Z. Pu, et al., J. Chem. Phys. 152 (6) (2020) 064113.
[26] 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, et al., WIREs Comput. Mol. Sci. 8 (1) (2018) e1340.
[27] J.M. Soler, E. Artacho, J.D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, J. Phys. Condens. Matter 14 (11) (2002) 2745.
[28] X. Qin, H. Shang, H. Xiang, Z. Li, J. Yang, Int. J. Quant. Chem. 115 (10) (2015) 647–655.
[29] H. Xiang, J. Yang, J. Hou, Q. Zhu, Phys. Rev. Lett. 97 (26) (2006) 266402.
[30] X. Qin, J. Liu, W. Hu, J. Yang, J. Phys. Chem. A 124 (27) (2020) 5664–5674.
[31] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, M. Scheffler, Comput. Phys. Commun. 180 (11) (2009) 2175–2196.
[32] M. Chen, G. Guo, L. He, J. Phys. Condens. Matter 22 (44) (2010) 445501.
[33] G. Kresse, J. Hafner, Phys. Rev. B 47 (1) (1993) 558.
[34] X. Gonze, J.-M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.-M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, et al., Comput. Mater. Sci. 25 (3) (2002) 478–492.
[35] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, et al., J. Phys. Condens. Matter 21 (39) (2009) 395502.
[36] W. Jia, J. Fu, Z. Cao, L. Wang, X. Chi, W. Gao, L.-W. Wang, J. Comput. Phys. 251 (2013) 102–115.
[37] W. Hu, L. Lin, A.S. Banerjee, E. Vecharynski, C. Yang, J. Chem. Theory Comput. 13 (3) (2017) 1188–1198.
[38] J.J. Mortensen, L.B. Hansen, K.W. Jacobsen, Phys. Rev. B 71 (3) (2005) 035109.
[39] J. Enkovaara, C. Rostgaard, J.J. Mortensen, J. Chen, M. Dulak, L. Ferrighi, J. Gavnholt, C. Glinsvad, V. Haikola, H. Hansen, et al., J. Phys. Condens. Matter 22 (25) (2010) 253202.
[40] Q. Xu, A. Sharma, P. Suryanarayana, Software X 11 (2020) 100423.
[41] F. Fathurrahman, M.K. Agusta, A.G. Saputro, H.K. Dipojono, Comput. Phys. Commun. 256 (2020) 107372.
[42] M.F. Herbst, A. Levitt, E. Cances, Proc. JuliaCon. Conf. 3 (2021) 69.
[43] J. Ihm, A. Zunger, M.L. Cohen, J. Phys. C, Solid State Phys. 12 (21) (1979) 4409.
[44] R.M. Martin, Electronic Structure, Basic Theory and Practical Methods, Cambridge University Press, 2020.
[45] J.P. Perdew, A. Zunger, Phys. Rev. B 23 (10) (1981) 5048.
[46] J.P. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (18) (1996) 3865.
[47] A. Stroppa, G. Kresse, New J. Phys. 10 (6) (2008) 063020.
[48] L. Schimka, J. Harl, A. Stroppa, A. Griineis, M. Marsman, F. Mittendorfer, G. Kresse, Nat. Mater. 9 (9) (2010) 741–744.
[49] H. Sun, D.J. Mowbray, A. Migani, J. Zhao, H. Petek, A. Rubio, ACS Catal. 5 (7) (2015) 4242–4254.
[50] E.J. Baerends, D. Ellis, P. Ros, Chem. Phys. 2 (1) (1973) 41–51.
[51] L. Lin, J. Chem. Theory Comput. 12 (5) (2016) 2242–2249.
[52] W. Hu, L. Lin, C. Yang, J. Chem. Theory Comput. 13 (11) (2017) 5458–5467.
[53] W. Hu, L. Lin, C. Yang, J. Chem. Theory Comput. 13 (11) (2017) 5420–5431.
[54] J. Lee, L. Lin, M. Head-Gordon, J. Chem. Theory Comput. 16 (1) (2019) 243–263.
[55] J.C. Phillips, L. Kleinman, Phys. Rev. 116 (2) (1959) 287.
[56] P. Giannozzi, F. De Angelis, R. Car, J. Chem. Phys. 120 (13) (2004) 5903–5915.
[57] M. Schlipf, F. Gygi, Comput. Phys. Commun. 196 (2015) 36–44.
[58] D.G. Anderson, J. Assoc. Comput. Mach. 12 (4) (1965) 547–560.
[59] P. Pulay, Chem. Phys. Lett. 73 (2) (1980) 393–398.
[60] G. Kerker, Phys. Rev. B 23 (6) (1981) 3082.
[61] G. Kresse, J. Furthmtiller, Phys. Rev. B 54 (16) (1996) 11169.
[62] Y. Zhou, J.R. Chelikowsky, X. Gao, A. Zhou, Commun. Comput. Phys. 18 (1) (2015) 167–179.
[63] C. Yang, J.C. Meza, L.-W. Wang, J. Comput. Phys. 217 (2) (2006) 709–721.
[64] C. Yang, J.C. Meza, L.-W. Wang, SIAM J. Sci. Comput. 29 (5) (2007) 1854–1875.
[65] J.A. Duersch, M. Shao, C. Yang, M. Gu, SIAM J. Sci. Comput. 40 (5) (2018) C655–C676.
[66] C. Davidson, J. Comput. Phys. 17 (1975) 87–94.
[67] Y. Zhou, Y. Saad, M.L. Tiago, J.R. Chelikowsky, J. Comput. Phys. 219 (1) (2006) 172–184.
[68] E. Vecharynski, C. Yang, J.E. Pask, J. Comput. Phys. 290 (2015) 73–89.
[69] G. Kresse, J. FurthmUller, Phys. Rev. B 54 (16) (1996) 11169.
[70] R. Car, M. Parrinello, Phys. Rev. Lett. 55 (22) (1985) 2471.
[71] R.P. Feynman, Phys. Rev. 56 (4) (1939) 340.
[72] J.D. Head, M.C. Zerner, Chem. Phys. Lett. 122 (3) (1985) 264–270.
[73] T.F. Coleman, Y. Li, Math. Program. 67 (1) (1994) 189–224.
[74] D.C. Liu, J. Nocedal, Math. Program. 45 (1) (1989) 503–528.
[75] M.L. Overton, HANSO: hybrid algorithm for non-smooth optimization, https://cs.nyu.edu/~overton/software/hanso/index.html, 2006.
[76] J.V. Burke, A.S. Lewis, M.L. Overton, SIAM J. Optim. 15 (3) (2005) 751–779.
[77] M.L. Overton, NLCG: nonlinear conjugate gradient, https://cs.nyu.edu/~overton/software/nlcg/index.html, 2010.
[78] E. Bitzek, P. Koskinen, F. Gahler, M. Moseler, P. Gumbsch, Phys. Rev. Lett. 97 (17) (2006) 170201.
[79] C. Edmiston, K. Ruedenberg, Rev. Mod. Phys. 35 (3) (1963) 457.
[80] N. Marzari, A.A. Mostofi, J.R. Yates, I. Souza, D. Vanderbilt, Rev. Mod. Phys. 84 (2012) 1419–1475.
[81] N. Marzari, D. Vanderbilt, Phys. Rev. B 56 (20) (1997) 12847.
[82] A. Damle, L. Lin, L. Ying, J. Chem. Theory Comput. 11 (4) (2015) 1463–1469.
[83] I. Carnimeo, S. Baroni, P. Giannozzi, Electron. Struct. 1 (1) (2019) 015009.
[84] K. Wu, X. Qin, W. Hu, J. Yang, J. Chem. Theory Comput. (2021).
[85] W. Hu, L. Lin, A.S. Banerjee, E. Vecharynski, C. Yang, J. Chem. Theory Comput. 13 (3) (2017) 1188–1198.
[86] K. Momma, F. Izumi, J. Appl. Crystallogr. 44 (6) (2011).
[87] M.R. Milovanovic, J.M. Zivkovic, D.B. Ninkovic, I.M. Stankovic, S.D. Zaric, Phys. Chem. Chem. Phys. 22 (7) (2020) 4138–4143.
[88] H. Ma, L. Wang, L. Wan, J. Li, X. Qin, J. Liu, W. Hu, L. Lin, L. Yang, J. Yang, J. Phys. Chem. A 125 (34) (2021) 7545–7557.
[89] J. Vinson, J. Rehr, J. Kas, E. Shirley, Phys. Rev. B 83 (11) (2011) 115106.