SCIENCE CHINA
Technological Sciences

• RESEARCH PAPER •

September 2025, Vol. 68, Iss. 9, 1920101:1–1920101:17
https://doi.org/10.1007/s11431-025-2974-2

Hylanemos: An integrated solution for materials simulations based on Kohn-Sham DFT

Jianshu JIE[1,2*], Ming XU[1,2*], Chun WANG[1,2], Shiqiang FAN[1,2], Fan ZHANG[1,2], Haifeng ZHENG[1,2], Yaokun YE[2,3], Ruiqi ZHANG[2,3], Jiahua LIU[2,3], Kangming HU[1,2], Shucheng LI[1,2], Qinghua LIU[1,2], Yipu ZHANG[1,2], Linping SUN[1,2], Xiaohe SONG[1,2], Sibai LI[1,2], Yunxing ZUO[1,2] & Jiaxin ZHENG[2,3*]

[1] *Shenzhen Eacomp Technology Co., Ltd., Shenzhen 518055, China*
[2] *Peking University Shenzhen Graduate School - Eacomp Technology Joint Battery Materials Simulation Laboratory, Shenzhen 518055, China*
[3] *School of Advanced Materials, Peking University, Shenzhen Graduate School, Shenzhen 518055, China*

Received April 14, 2025; accepted June 5, 2025; published online August 22, 2025

Abstract The Kohn-Sham density functional theory (KS-DFT) has played an important role in materials simulation for a long time. To better serve the industry, it is desirable to have an integrated solution that supports different calculation tasks by KS-DFT with different corrections and modifications. In this work, we present Hylanemos, a plane wave pseudopotential (PW-PP) KS-DFT package written entirely in the Julia programming language, which could offer such a solution. First, we analyze the code design to get the flexibility needed to implement such a solution. Then, we show that its accuracy and speed are comparable to widely-used packages. Next, we show its ability to perform common tasks such as single point (SP) calculations, geometry optimization, and transition state calculations. Finally, the LDA+Gutzwiller (LDA+G) method is presented, a feature not commonly found in DFT packages. In addition, we have also developed a set of ultrasoft (US) PP through parameter adjustment and optimization. This set of PP, called Eacomp PP, has a low cutoff energy (<18 Ha) and exhibits excellent performance in our benchmarks. Combining a performant package and optimized potentials will facilitate our in-depth efforts in promoting industrialization.

Keywords *ab initio* software, Kohn-Sham density functional theory, high-performance computing, generalized self-consistent method

Citation: Jie J S, Xu M, Wang C, et al. Hylanemos: An integrated solution for materials simulations based on Kohn-Sham DFT. Sci China Tech Sci, 2025, 68(9): 1920101, https://doi.org/10.1007/s11431-025-2974-2

1 Introduction

Ab initio calculations based on Kohn-Sham (KS) density functional theory (DFT) [1,2] have been widely used both in academia and in industry. In DFT, the total energy is a functional of the electron density, and the KS method introduces a non-interacting reference system (KS system) with the same electron density as the real system. There are two main benefits of using a non-interacting system. First, the wavefunction can be expressed as a single Slater determinant and solved using the rather mature self-consistent field (SCF) method. Second, the energy functional can be easily written out except for the exchange-correlation (XC) term, so we only need to find an approximated functional for the XC energy and the difference between the true system and the KS system instead of an approximated functional for the total energy. It has been proved by practice that even basic KS-DFT with a simple XC functional can result in good agreement with experiments for many simple systems, and KS-DFT has been widely used in academia for a long time.

There are different implementation techniques for the

* Corresponding author (email: jiejianshu@eacomp.com; xuming@eacomp.com; zhengjx@pkusz.edu.cn)

KS-DFT. For real industrial usage on crystal materials, KS-DFT based on plane waves (PW) with pseudopotentials (PP) [3,4] or projector-augmented-wave (PAW) [5] potentials are two common solutions. Corresponding software packages such as CASTEP [6], Quantum ESPRESSO (QE) [7], ABINIT [8], VASP [9], and PWmat [10,11] are widely used. Packages, like QuantumATK [12] and ABACUS [13,14], which originally supported KS-DFT based on atomic orbital basis sets, also began to support PW-PP calculations recently. These packages usually perform not only the simple total energy calculation with KS-DFT but also some tasks based on the total energy calculation, such as geometry optimization and molecular dynamics (MD).

Over the years, there have been many advances in KS-DFT and related algorithms. Two different approaches are adopted to bridge the gap that the basic KS-DFT with simple XC functionals is insufficient to accurately calculate some complex systems. In the first approach, new XC functionals, such as meta-generalized gradient approximation (GGA) functionals [15] and hybrid functionals [16,17], are developed. In the other approach, many corrections are proposed, such as the LDA+U [18–20] method and the LDA+G [21–23] method. Also, some corrections or modifications have been introduced to reduce the computational cost. For example, dipole corrections [24] are introduced so that a smaller vacuum layer can be used, and implicit solvent models [25] are introduced to make adding many solvent molecules into the system unnecessary. Moreover, some modifications are introduced to calculate systems under different physical conditions, such as those with magnetic structure constraints [26]. Some algorithms closely related to DFT are proposed, which usually adopt KS-DFT-like solutions. For example, the density functional perturbation theory (DFPT) [27] and linear-response time-dependent DFT (TDDFT) [28] are also available in many DFT packages. Also, algorithms that are beyond DFT but use DFT as a part are proposed. For example, currently, GW calculations [29,30] usually take the output orbital information of a KS-DFT calculation as input and are available in DFT packages like VASP and QE. New algorithms based on total energy calculations are further proposed, such as the dimer method [31–33] and the nudged elastic band (NEB) method [34].

We believe an integrated solution for materials simulations based on KS-DFT, with many different algorithms out of the box, is of great value to the industry. However, it is difficult to implement such a solution without a flexible design. Many packages created in earlier years have been rewritten either partly or completely to accommodate later advances. In a worse scenario, some features, such as the dimer method and implicit solvent models, are usually absent in many widely used packages.

In this paper, we will first discuss the flexibility required for an integrated solution and how we can achieve such flexibility when designing the code. Next, we will present a new codebase entitled Hylanemos, a Julia implementation of the flexible and integrated framework. We made an arguably comprehensive comparison in both accuracy and efficiency between Hylanemos and some widely used packages. We also show that Hylanemos is capable of running critical tasks in industrial applications and has some features rarely seen in existing packages.

The Julia programming language is chosen for the following reasons. First, the performance of Julia is comparable to that of Fortran, C, or C++, which is crucial in materials simulations due to the high computational cost. Second, with the support of an interactive mode, it is arguably easier to develop and debug Julia code. Third, with the evolution of programming techniques, new concepts such as modules and polymorphism are introduced into old languages like Fortran. These concepts must be introduced in a way that does not break the existing code. As a result, sometimes it is not easy to use them. For example, polymorphism in Fortran is very tricky to use. Being a relatively new language, Julia does not suffer from these historical burdens, which makes it easier to use. Last, though the ecosystem of Julia is not as mature as that of Fortran or C, the ability to directly call C and Fortran libraries in Julia code makes up for this shortcoming. In summary, Julia's combined advantages in performance and easy-to-use make it an ideal choice for implementing our project.

2 Flexibility in an integrated solution

2.1 An integrated solution

In materials simulations, a common computation task involves conducting one or more single-point (SP) calculations. A SP calculation is carried out on a specific structure under defined environmental conditions to obtain properties of interest. The PW-PP KS-DFT method is a notable approach for conducting SP calculations, and it can be used to determine the total energy and electronic structure.

Regarding industrial usage, an all-in-one solution is preferred over a complicated set of tools. An integrated solution should offer flexibility in three key areas: the ability to carry out various tasks based on SP calculations, the option to perform SP calculations using different methods, and the capability to apply different adjustments and modifications in a specific manner to conduct SP calculations.

In practice, the flexibility of the second layer is of less importance. It is acceptable for only one or a few SP calculation methods to be supported. Many KS-DFT packages can be regarded as integrated solutions based on KS-DFT. However, the flexibility in the other two layers is often insufficient in these packages. For instance, in the task layer, the dimer method is absent in many packages, and in the correction and modification layer, the implicit solvation model is typically implemented by third-party software, such as VASPsol [35] for VASP and Environ [36] for QE.

In the following section, we will first inspect the three layers and determine the requirements for achieving sufficient flexibility in each layer. Next, we will analyze the PW-PP KS-DFT method within the correction and modification layer. Lastly, we will explore the important aspects of general module design that are crucial for ensuring flexibility across all three layers.

Before discussing the details of each layer, we would like to emphasize the importance of the overall three-layer architecture. These three layers do exist in most KS-DFT packages, but they are usually neither clearly stated nor separated. An explicit and well-defined three-layer architecture will offer distinct advantages. It enables us to fully use the dependencies between these layers to facilitate the implementation of new algorithms, which will be shown in the following sections.

2.2 Flexibility for different tasks

Task workflow can be categorized into three types based on their relationship to SP calculations, as illustrated in Figure 1. Type I tasks entail a single SP calculation, with the output used directly or after postprocessing. Type II tasks involve the construction of multiple structures, each undergoing SP calculations. This approach is commonly used for tasks related to the finite difference method, such as mechanical properties or phonon spectra calculation.

Type III tasks are the most intricate, as they involve structural evolution using various methods, which differ greatly among different tasks. For instance, in geometry optimization, a single structure is employed. Conversely, the dimer method utilizes two initial structures that evolve together, while the NEB method involves several initial structures that evolve independently.

For Type I tasks, only one single-point calculation is needed, so they do not pose a challenge to code design. Type II tasks can be accomplished using a simple loop, as outlined below.

(1) Build all structures and designate the first structure as the current one.

(2) Perform a single-point calculation on the current structure.

(3) If all structures have been calculated, stop. Otherwise, set the next structure as the current one and return to step (2).

For straightforward Type III tasks, such as MD or geometry optimization using simple algorithms, the same loop can be employed with slight adjustments.

(1) Create the initial structure and designate it as the current one.

(2) Perform a single-point calculation on the current structure.

(3) If certain stopping criteria have been met, stop. If not, modify the structure using an algorithm, designate the new structure as the current one, and return to step (2).

The approach described above is commonly used in many

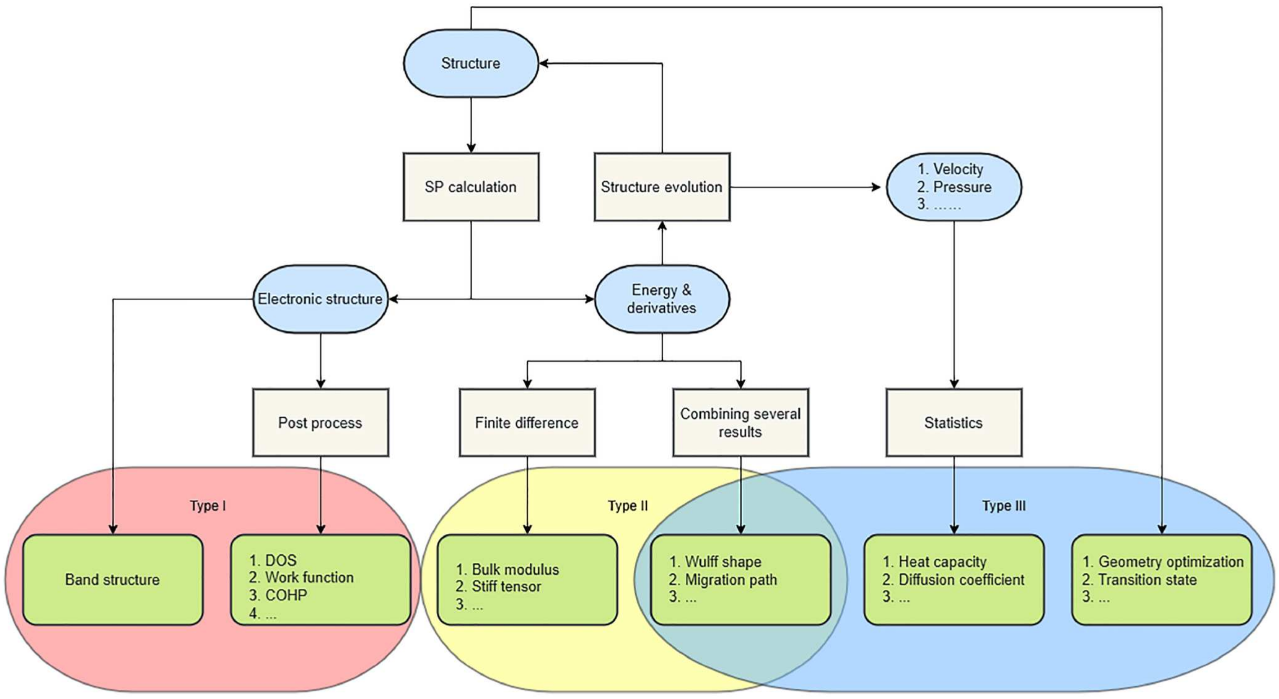

Figure 1 (Color online) The uppermost framework. The blue rounded rectangles denote quantities in the calculation process, the green rounded rectangles denote the results we want, i.e., the tasks, and the rectangles denote actual treatments in the calculation.

existing packages, such as VASP. There may be some considerations for adopting this approach. In this approach, the code for implementing the structure evolution algorithm does not necessarily need to depend on the SP calculation subroutine. Besides, all type II and type III tasks can be implemented within a single function, at least in theory.

However, such considerations are questionable. Firstly, the structure evolution algorithms belong to the task layer, while the subroutines conducting SP calculations belong to the SP layer. This fact indicates a well-behaved dependency between them, which can be used to implement complex algorithms more clearly. Secondly, there are many different algorithms for structure evolution, especially for geometry optimization and transition state search. Implementing all these algorithms in one function is not advisable since implementing complex algorithms in this manner can be challenging.

Let's take geometry optimization as an example, one of the most important tasks for DFT packages. In VASP, a conjugate gradient algorithm with line search is implemented, which combines cubic interpolation, the Brent algorithm, and interval enlargement. We will first examine how to implement such an algorithm in Hylanemos, which is presented below in a simplified version as shown in Function 1. We omit simple functions such as data translation between SP data and optimization data structure and convergence checks for brevity.

Function 1 Hylanemos CG geometry optimization

```
function cg_hylanemos!(com::OptCommData, cg::CGData, br::BrentData,
sp::SPData)
    solve!(sp)
    # status 1
    for iter = 1 : com.iter_max
        find_dir!(cg, com, ld, sp)
        l, s = get_step(cg)
        make_step!(br, l, s)
        solve!(sp)
        # status 2
        l, s = get_cubic(br)
        make_step!(br, l, s)
        solve!(sp)
        if good_enough(br)
            # status 1
            continue
        end
        while !is_bracket(br)
            # status 3
            enlarge_interval!(br)
            solve!(sp)
        end
        while true
            # status 4
            brent!(br)
            solve!(sp)
            if good_enough(br) || quick_exit_brent(br)
                # status 1
                break
            end
        end
    end
end
```

The algorithm is complex but clear. We will first find the direction using the conjugate gradient algorithm at each optimization step. Simultaneously, we obtain a step size. However, we will only use it as an initial guess to get a trial point. We gather sufficient data for a cubic interpolation from the initial and trial points, enabling us to identify the third point within the current optimization step.

If the interpolated point meets the required criteria, we shall stop here and adopt it as the initial point in the next optimization step. Conversely, we will first construct an initial interval from two of the three obtained points. The details of the choice are not included here for brevity. An important subsequent step involves determining whether the interval confidently contains a minimum, which is decided by whether it contains a root for the derivative. This can be done by checking whether the derivatives at the two endpoints of the interval have the opposite signs.

If there is no guarantee that a root lies within the interval, we will first gradually enlarge the interval. This process continues until we can confirm the existence of a root within the interval.

After we have found an interval with a root in it, we perform a modified Brent algorithm to find the root, which is, in fact, a sophisticated combination of bisection, linear interpolation, and inverse quadratic interpolation. The modification is that we will perform a check for each Brent iteration. If some other quick return conditions are satisfied, we will exit the Brent iteration and use the last point as the initial point for the next optimization step.

After understanding the algorithm, let's examine its implementation in VASP. Remember that in VASP, all structure evolution algorithms are written in a simple for-loop way, as shown in Function 2.

As a result, we shall focus on the logic within the cg_vasp! function, as shown in Function 3.

At first sight, there may seem to be no relation between this code and the above code in Hylanemos, but a careful examination will show their equivalence. As shown in the

Function 2 VASP structure evolution

```
function evo_vasp!(evo::EvoData, sp::VaspSPData)
    for iter = 1 : evo.iter_max
        solve!(sp)
        if evo.alg == "cg"
            cg_vasp!(evo.cg, sp)
        elseif evo.alg == "dampMD"
            dampmd_vasp!(evo.dampmd, sp)
        # numerous else ifs
        else
            # some algorithm
        end
    end
end
```

Function 3 VASP CG geometry optimization

```
function cg_vasp!(cg::VaspCGData, sp::VaspSPData)
    if !cg.ltrial
        cg.beta = calc_beta(cg)
        trial_con = cg.lbrent && quick_exit_brent(cg)
        if !trial_con
            make_trial_step!(cg)
            reset_zbrent!(cg)
            cg.ltrial = true
            return
        end
    end
    zbrent!(cg)
    if cg.ltrial
        cg.lbrent = true
    end
    if good_enough(cg)
        cg.lbrent = false
    end
    cg.ltrial = false
end
```

comments in the Hylanemos code, we must be in one of the four statuses after performing an SP calculation and related checks.

(1) Status 1. We are about to enter a new optimization step.

(2) Status 2. We have finished the calculation of the trial point.

(3) Status 3. We are about to perform interval enlargements.

(4) Status 4. We are about to perform a step in Brent's iterations.

Possible hopping among the statuses is as follows.

(1) Status 1 will jump to status 2.

(2) Status 2 may jump to any of the other three statuses. Note that the SP calculation for the cubic interpolated point will be performed for status 2. If the point is good enough, it will jump to status 1. If not, then if the initial interval definitely contains a root, it will jump to status 4. Otherwise, it will jump to status 3.

(3) Status 3 may jump to status 4 or stay at status 3, depending on whether an appropriate interval has been found.

(4) Status 4 may jump to status 1 or stay at status 4, depending on whether a root has been found or whether some quick-returning conditions are satisfied.

We then prove that these statuses and hopping exist in the VASP code, except that status 3 and status 4 are merged into one status.

First, ***cg.ltrial*** means whether we have just finished the SP calculation for the trial point. Thus, when it is true, we are in status 2. In this case, the large ***if*** block will not be executed, and we will call ***zbrent***!. Note that the ***zbrent***! function in VASP contains cubic interpolation and the so-called Brent algorithm, which again contains interval enlargement and the real Brent algorithm. Here we are doing the cubic interpolation. Then, we set ***cg.lbrent*** to be true, indicating we are about to perform the so-called Brent algorithm, i.e., jumping to status 3/4. However, if we find the cubic interpolated point is good enough, we set ***cg.lbrent*** to be false because we should jump to status 1 instead. In either case, we will not stay in status 2, so we set ***cg.ltrial*** to be false.

Then let's consider the situation when ***cg.ltrial*** is false. We may be either in status 1 or 3/4. In either case, we will first calculate beta for the conjugate gradient because the calculation is computationally inexpensive and will not cause any issues even if we do not use its result. After that, we need to determine our current state. When ***cg.lbrent*** is true and the quick-returning conditions are not satisfied, we are in status 3/4, and the following code within the large ***if*** block is not executed. Again, we call the ***zbrent***! function, this time for the interval enlargement or the real Brent algorithm. If the current point (note now it is generated by the Brent algorithm, instead of cubic interpolation) is good enough, we shall jump from status 4 to status 1, so ***cg.lbrent*** should be set to false. Since we are not jumping to status 2, ***cg.ltrial*** should also be set to false.

When ***cg.ltrial*** is false, and the ***trial_con*** is determined to be false, we are in status 1. We then generate the trial point from the CG beta and move to status 2 by setting ***cg.ltrial*** to be true. It should be noted that since the ***zbrent***! function in VASP contains multiple utilities, it must be reset here. This reset ensures that the cubic interpolation utility will be executed next time.

From the above analysis, it can be seen that the two code segments achieve the same functionality. However, the code in VASP is much more obscure. The main reason is that mathematical optimization algorithms are usually designed assuming the gradient is available whenever needed. In our case, the SP calculation must be performed to get the force and stress. Thus, the simple ***for-loop*** architecture in VASP requires that during the execution of optimization functions like ***cg_vasp***!, the gradient for only one point is available, which prohibits the direct translation of most mathematical optimization algorithms. As demonstrated above, we must carefully identify all possible statuses and hopping within the algorithm, and then we use numerous ***if-else*** blocks in the code to handle different possibilities.

We can point out two details where the VASP implementation appears rather idiosyncratic. First, integrating cubic interpolation, interval enlargement, and the real Brent algorithm in one single ***zbrent***! function seems odd. However, separating them would introduce even more branches in the code, further reducing its readability. Besides, after each Brent algorithm, we shall skip subsequent steps if the last point has been good enough or some quick-returning conditions are satisfied. In Hylanemos, these two examinations are placed together. In VASP, the ***good_enough*** function is called near the end, while the ***quick_exit_brent*** function is called near the beginning. There are two reasons for this separation. First, the ***good_enough*** function in VASP serves not only to assess whether a point obtained from the Brent algorithm is satisfactory but also to determine the adequacy of the cubic interpolated point. Thus, it must be put and called near the end. The other reason is that the ***quick_exit_brent*** function requires the gradient at the new point. Therefore, it can only be calculated after the SP calculation, i.e., near the beginning.

From the above analysis, it can be seen that the simple ***for-loop*** architecture becomes unwieldy when implementing complex structure evolution algorithms. Therefore, certain algorithms, such as the backtrack in geometry optimizations, are absent in VASP. Some other algorithms are also implemented awkwardly, similar to the example we have analyzed. For another example, readers can examine the implementation of the dimer method [31–33] in either VASP or VTST and compare it with the algorithm described in the papers. The equivalence can be proved through a similar but even harder analysis of statuses and hopping.

With the emergence of various algorithms in type III tasks, it will become increasingly difficult to accommodate them in one simple for loop. Utilizing the three-layer architecture in Hylanemos, we can directly convert each algorithmic logic into one corresponding function. This approach proves to be significantly more convenient compared to other methods.

2.3 Flexibility for different SP methods

Even though this layer of flexibility is less important and often ignored, it is beneficial for an integrated solution to support it. There are two key points to consider in this regard. First, some methods may rely on other methods. For example, a GW [29,30] calculation usually contains a DFT calculation and another calculation based on the output of DFT. Second, some common treatments are shared among different methods, such as the Ewald summation for the ion-ion interaction calculations, and various dispersion corrections proposed by Grimme [37–39].

A multi-layer design is necessary to accommodate various methods and treatments. The bottom layer provides modules for common treatments, the intermediate layer for different methods, and the top layer is a wrapper module that provides common interfaces for ion-related tasks, such as updating the current structure, performing the actual calculation, and obtaining output energy and force.

When working on ion-related tasks, it is crucial to utilize the common interfaces of the top layer rather than directly manipulating the underlying structures, which can vary among different SP calculation methods. Following this approach, new SP calculation methods can be easily incorporated by adding a new module in the intermediate layer and using it in the top layer, without modifying the code for the ion-related tasks.

2.4 Flexibility in PW-PP KS-DFT

In DFT, the functional for the total energy depends solely on the electron density. However, in KS-DFT, the used XC functional may involve multiple parameters. For instance, some meta-GGA functionals incorporate kinetic energy density, while hybrid functionals involve the wavefunction. These parameters are critical for determining the system's Hamiltonian. Additional corrections may introduce other quantities for Hamiltonian determination, such as occupation matrices in LDA+U. In this paper, all these quantities will be referred to as variables in the Hamiltonian, or simply variables, which should be treated equally.

However, it is important to note that these variables do not need to be treated identically. First, their convergence behaviors differ, so they can be grouped into different sets. Each set can then be iterated in different layers of self-consistent calculations. Second, some variables may be defined in a reduced space or a space with higher dimensions, corresponding to different projections of the Hamiltonian.

The LDA+G method is a good example. Even the basic algorithm of the LDA+G method is too complex to be discussed here, and there are many slightly different implementations. In the Supporting Information, we briefly introduce the basic ideas, and here we only describe two different implementations. In Bunemann's implementation [40], the Gutzwiller-Kohn-Sham (GKS) loop is a loop of the outer minimizations, with each iteration containing a full normal KS SCF and an inner minimization with respect to $\lambda_{I;\Gamma\Gamma'}$, the expansion coefficients for local configurations. The outer minimizations are iterated with respect to $\tilde{\boldsymbol{\rho}}$, the density matrix. In their work, Bunemann mentioned that this implementation may not be the most efficient, but it is rather easy to implement in the QE framework. This proves the need for a more flexible framework. Bunemann proposed another possible implementation, which is used in Hylanemos. In this implementation, the GKS loop is an inner loop within each iteration in the normal KS SCF. Dai has proposed another implementation [41], where the overall process is similar to Bunemann's implementation, except that the variable in the outer loop is the renormalization matrix $\mathcal{R}$.

Consequently, multiple layers of self-consistent loops may be involved in the SCF process instead of one single layer in basic KS-DFT. Moreover, consecutive calculations on different spaces may occur within a single iteration. The introduction of different corrections leads to varying looping behaviors, similar to how different type III tasks require different algorithms.

It should be noted that using a different function for each correction is not ideal for two reasons. First, multiple corrections can be used simultaneously. Second, these corrections do not rely on statuses to determine their actions in each

Function 4 Recursive SCF

```
function Scf!(es::ES, layer)
    while true
        if layer <= MaxLayer(es)
            Scf!(es, layer+1)
        end
        Calc!(es, layer)
        if Isconv(es, layer)
            break
        end
        Mix!(es, layer)
        UpdateHamiltonian!(es, layer)
    end
end
```

iteration. Therefore, a single function capable of handling the SCF process across different layers is preferred.

The key here is to use a unified structure, the electronic status (ES), to collect the variables together, group them into different sets, and store the information of the sets, like the related Hamiltonians. With the recursion technique, the complicated SCF process can be performed similarly to the normal simple SCF, like the following code.

The function ***Calc***! is to perform a single loop of self-consistent calculations and get the new variables in the specific layer. Likewise, ***Isconv*** is to check the convergence of these variables, ***Mix***! is to apply some mixing algorithms to get the input values for these variables in the next iteration, and ***UpdateHamiltonian***! is to update the corresponding Hamiltonian according to new values of these variables. Different computational tasks may need different variables, and the details of these corresponding functions can vary greatly. Thus the ES must be used to wrap these values and hide these differences.

This approach results in a unique SCF process within our framework, which differs from the commonly employed method. Figure 2 illustrates a schematic of our SCF process.

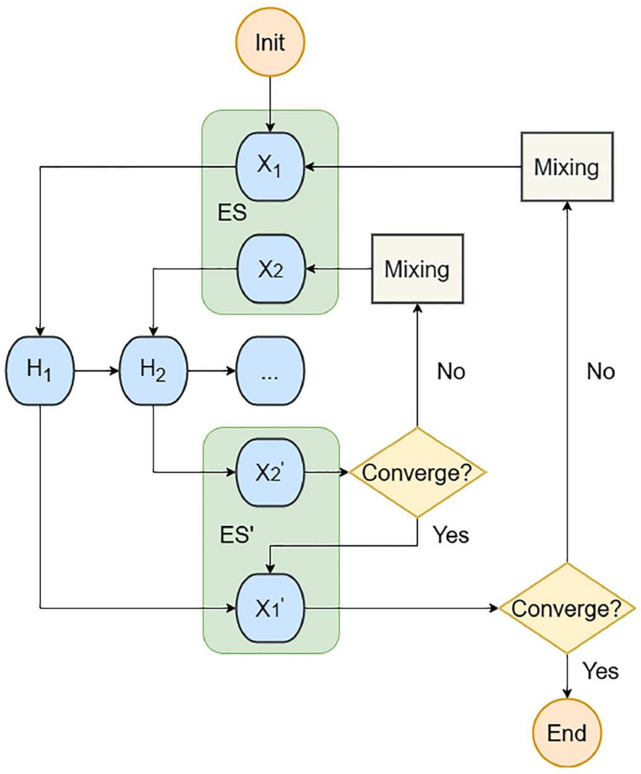

Figure 2 (Color online) The SCF process in our framework. Note that X represents the electronic status (ES), instead of the electron density ρ. Variables in ES can be split into different groups, denoted by X_1 and X_2 in the figure. There may be even more groups, denoted by the ellipsis in the figure. Each group of variables has its corresponding Hamiltonian and its layer of iteration. The whole SCF process exists only when all variables have been converged.

2.5 Flexibility from module design

Modular programming is a widely used software design technique [42] that divides a program's functionality into independent, interchangeable modules. These modules contain data and functions for specific purposes and communicate with each other through interfaces.

It is important to consider what modules are needed and what should be contained in each module. Additionally, the design of module interfaces, or how modules communicate with each other, is crucial.

In this approach, it is natural to have a module for each physical term and each type of numerical algorithm under consideration. For example, separate modules can be created for LDA+U and eigen solvers. Furthermore, having a module for each group of data that shares common computational treatments can also be beneficial.

An example is the local potentials, which contribute to the energy in the form of

$$E = \int V^{\mathrm{loc}}(\mathbf{r})\rho(\mathbf{r})\mathrm{d}\mathbf{r}. \tag{1}$$

Local potentials can be introduced by many different corrections and modifications, such as dipole correction, magnetic constraints, and the implicit solvent model. These potentials have common computational treatments and share contributions to the Hamiltonian and force, as discussed by Laasonen et al. [43] and Chan et al. [44]. Here, we only list the most important conclusions.

When ultrasoft pseudopotentials (USPP) [4] are used, local potentials contribute to two terms in the Hamiltonian: the local term and the non-local term. The contribution to the local term is direct:

$$\hat{V}^{\mathrm{loc}}\psi(\mathbf{r}) = V^{\mathrm{loc}}(\mathbf{r})\psi(\mathbf{r}), \tag{2}$$

while the contribution to the non-local term

$$\hat{V}^{\mathrm{NL}}|\psi\rangle = \sum_{ij}|\beta_i\rangle D'_{ij}\langle\beta_j|\psi\rangle \tag{3}$$

is through the D coefficients:

$$D'^{I}_{ij} = D^{I}_{ij} + \int V^{\mathrm{loc}}(\mathbf{r}) Q^{I}_{ij}(\mathbf{r}) \mathrm{d}\mathbf{r}, \tag{4}$$

D, Q, and β are all parameters defined in the USPP.

For norm-conserving pseudopotentials (NCPP), the augmentation function Q does not exist. Thus, the second term vanishes, and local potentials do not contribute to the nonlocal term.

In both NCPP and USPP, local potentials contribute to a force correction term:

$$F_c = \int \mathrm{d}\mathbf{r}(\delta\rho/\delta\tau)\left(V^{\mathrm{loc}}_{\mathrm{in}} - V^{\mathrm{loc}}\right). \tag{5}$$

Thus, it is advantageous to use individual modules to encapsulate these similarities. This approach allows for new corrections of local potentials by simply adding them to the total local potential instead of reimplementing all the treatments.

The design of module interfaces is critical in modular programming, which requires a balance between simplicity and flexibility. In our implementation, we prioritize high flexibility over simplicity, as future modifications may necessitate this level of flexibility.

In the following examples, we illustrate two distinct scenarios. The first example pertains to the parallelization over **G** vectors, where two different types of parallelization, namely ρ type and ψ type, are implemented using two distinct sets of interfaces instead of one unified interface. The differences are subtle yet critical. There are many different orbitals (indicated by band) for each k point, but only one common charge density for all k points. For instance, when performing ψ-related calculations on 2 k point groups and 2 band groups with 8 processes, **G** vectors can only be split into 2 processes because of the number of k point groups and band groups. On the contrary, when performing ρ-related calculations, **G** vectors can always be split into all 8 processes regardless of the parallelization settings. The calculation efficiency can severely deteriorate without considering the subtle differences.

The second example involves the complexity of the interface when calculating ρ from ψ, where the interface is designed with arguments instead of using modular or global variables. In traditional practices, KS orbitals, orbital occupancies, and electron densities are stored as modular or global variables, allowing the electron density calculation subroutine to function without arguments. However, LDA+G modifies the way electron density is calculated. Theoretically, the electron density should be calculated in a larger space instead of the KS space, while it can still be done in the KS space with certain adjustments in practice. The process involves generating a set of corrected orbitals from the KS orbitals, using these corrected orbitals to calculate an electron density, and then modifying the electron density to obtain the true electron density. It is important to note that the corrected orbitals are only utilized for calculating charge density. One should still use the original orbitals afterwards, and cannot modify the KS orbitals in place.

The subroutine without arguments can not serve this purpose. One approach is to write a completely new subroutine for LDA+G electron density calculation only. However, this would lead to code duplication, as a part of LDA+G electron density calculation is similar to the ordinary electron density calculation. Another approach is integrating judgment within the subroutine to determine whether corrections should be made, which would cause the unreasonable fact that the electron density module depends on the Gutzwiller module. Herein, we prefer to use a single interface that requires values as arguments instead of relying on modular or global variables, and determine in the SCF module whether corrections should be made and what values are to be passed. It consequently results in clean logic, eliminates code duplication, and offers increased flexibility for potential corrections.

3 Applications

3.1 Overall introduction

Considering the discussions, we implemented the Hylanemos (HY) software with the Julia programming language. Hylanemos is designed to be an integrated solution for general materials simulations, while we currently focus on its applications in lithium-ion batteries based on PW-PP KS-DFT. Hylanemos is part of Matter Craft [45], an integrated platform for materials simulation tools, which contains modeling tools and graphical user interface (GUI) tools for Hylanemos and several other simulation packages.

In the following sections, we present several assessments for Hylanemos. The Hylanemos was evaluated for its accuracy in reproducing SCF results of some simple systems calculated by VASP and QE, two widely used KS-DFT packages. The computational efficiency of the three packages is further investigated with a lithium cobalt oxide ($LiCoO_2$, LCO) system. An attempt was also made to apply the three packages to perform several complex tasks on typical lithium-ion batteries-related systems, spanning LCO, layered $LiNi_{1/3}Co_{1/3}Mn_{1/3}O_2$ (NCM), lithium iron phosphate ($LiFePO_4$, LFP), and Li-Graphene. Finally, the LDA+G feature of Hylanemos is presented, which is absent and arguably difficult to implement in the other two packages.

3.2 Calculation details

All calculations were carried out on an HPC cluster, with each node consisting of two Intel Xeon Platinum 8358 Processors. VASP version 5.4.4 was used, and the "makefile.

include.linux_intel" file that comes with the source code was used for compiler options. QE version 7.0 was used, but minor modifications were made to enable the simultaneous use of Libxc [46] and DFT-D3 [38].

Currently, Hylanemos supports NCPP [3] and USPP, VASP focuses on PAW potentials, and QE supports all three. Hylanemos uses in-house generated USPPs named Eacomp PP, with an 18 Ha energy cutoff for wavefunctions and 90 Ha for electron densities. VASP, on the other hand, uses PAW potentials with an energy cutoff for wavefunctions set to 500 eV, or about 18.37 Ha. The choice of 500 eV is based on the cutoff requirement for a specific potential Li_sv when the Li element is present. VASP does not need an energy cutoff for the electron density. In the case of QE, GBRV USPPs [47] are utilized, and while the publication suggests an energy cutoff of 20 Ha for wavefunctions, a higher cutoff (24 Ha) is necessary for systems containing Ni atoms. Therefore, 24/120 Ha cutoffs are used for QE calculations in most benchmark systems.

In most cases, the number of k points is selected so that the spacing is less than 0.4 Å^{-1} and $N_k \times N_{atoms} > 500$. Hylanemos and QE utilize Libxc for the exchange-correlation functionals, while VASP employs its own implementation. All calculations were performed using the Perdew-Burke-Ernzerhof (PBE) functional [48] unless otherwise specified. In the case of structures containing transition metals, LDA+U corrections are applied, with specific U values set for Fe (3.6 eV), Co (5.6 eV), Mn (3.7 eV), and Ni (6.6 eV). Grimme-D2/D3/D4 [27–29] dispersion corrections are employed when required, with the D3 correction used in conjunction with the Becke-Johnson damping function [49].

We believe packages will most likely be used in the industry with parameters set as default or recommended. Thus, we also set parameters in the benchmarks in this way, which may lead to differences among different packages. For example, in VASP, the length of the charge density grid is only 1.5 times that of the wavefunction grid, while it is 2 times in QE and Hylanemos. (Readers are referred to the VASP manual for PREC for more details). Another example is that VASP adopts parallelization over bands, while QE and Hylanemos adopt parallelization over plane waves. No parallelization over k points is used in any benchmarks. The Davidson algorithm is used in all three packages, but the implementation details differ.

3.3 Evaluation of accuracy

As mentioned previously, SP calculations are the basis of many complex tasks, and SCF calculations are the core of SP calculations. Thus, the accuracy is evaluated by the results of SCF calculations, i.e., total energies, forces, and stresses.

The total energies can only be compared when the same potentials are used. Thus, the SCF results of Hylanemos and QE are compared because they both support PP in the unified pseudopotential format (UPF). Table 1 illustrates the differences of total energies, forces, and stresses of several simple systems calculated by Hylanemos and QE with both pseudo dojo [50] (a set of NCPPs) and GBRV (a set of USPPs), and the results agree very well. For dojo, energy cutoffs of 50/200 Ha are used. The convergence threshold for self-consistency is set to 1×10^{-8} Ry.

The three packages were compared for formation energies, regardless of the different potentials used. Table 2 clearly shows that these packages yield similar results, indicating Hylanemos' accuracy in reproducing SCF results calculated by these widely used packages. In the case of bulk systems, the k spacing is chosen to ensure that $N_k \times N_{atoms}$ is greater than 8000. For atomic systems, a cubic cell with a cell length of 15 Å and a 3×3×3 k-point mesh is utilized.

Diverse materials containing bulk, surface, and molecule systems also undergo a more comprehensive validation. In addition to energy, forces, and stress, the magnetic moments calculated by Hylanemos show a high degree of consistency with those obtained by QE. The validation details are listed in the Supporting Information.

3.4 Assessment of computational efficiency

LCO systems containing 48, 96, and 300 atoms were utilized to compare efficiency. The k-point meshes for these systems are 4×4×2, 2×4×2, and 1×1×1, respectively. Different numbers of processes (1, 2, 4, 8, 16, 32, and 64) were employed to assess the parallel performance.

The comparison between different potentials, instead of

Table 1 Differences of SCF results calculated by Hylanemos and QE with the same settings [a)]

	ΔE (Ry)	ΔS_{max} (kbar)	ΔF_{max} (Ry bohr^{-1})
Si-GBRV	$<2.0\times10^{-8}$	0.01 (0.02%)	3.95×10^{-6} (0.00%)
NaCl-GBRV	4.1×10^{-7}	0.07 (1.90%)	-2.15×10^{-6} (−0.03%)
Al-GBRV	$<2.0\times10^{-8}$	0.01 (0.10%)	-3.65×10^{-6} (−0.09%)
Si-dojo	$<2.0\times10^{-8}$	0.01 (−0.01%)	-1.03×10^{-6} (0.00%)
NaCl-dojo	$<2.0\times10^{-8}$	−0.03 (−0.79%)	-1.28×10^{-6} (−0.02%)
Al-dojo	$<2.0\times10^{-8}$	0.00 (−0.01%)	-2.45×10^{-6} (0.03%)

a) ΔE, ΔS_{max}, and ΔF_{max} represent energy difference, maximum stresses difference, and maximum forces difference, respectively

Table 2 Formation energies (kJ mol^{-1}) calculated by Hylanemos, QE, and VASP

	Hylanemos	VASP	QE
Na	122.710	122.960	122.761
Ti	633.541	629.382	635.817
Si	514.587	514.324	514.569
O	324.046	326.551	326.790
Al	355.042	355.142	355.034
Br	150.589	150.567	150.651
Ga	274.099	274.072	274.221
S	328.950	328.804	328.907

packages, is an important factor to consider. The analysis of Figure 3 may cause the illusion that QE yields the worst performance. However, this is primarily due to the variations in potentials. GBRV pseudopotentials have the most valence electrons for transition metal elements, requiring much higher energy cutoffs than the other two types. In contrast, VASP potentials and Eacomp PP are similar in these aspects. Consequently, the computational cost is significantly higher in QE, leading to longer computation times and better parallel performance. In the Supporting Information, we compare QE and Hylanemos, which are both calculated with GBRV pseudopotentials and the same settings. It can be seen that parallel performance is similar. The efficiency of Hylanemos is slightly better for the SCF process and much better for force and stress calculation.

It should be noted that this is not an inherent limitation of USPP. Eacomp PP also falls under USPP but is comparable to VASP PAW potentials. This could be attributed to the fact that GBRV pseudopotentials are designed for high-throughput calculations, generating only one PP for each element. As a result, a single GBRV PP must be accurate for numerous potential structures, some of which require a large number of semi-core electrons, necessitating an overall higher energy cutoff. On the contrary, VASP offers a variety of potentials for each element, each with a different number of valence electrons. As the systems in our benchmarks do not require many semi-core electrons, those with fewer valence electrons are utilized. Eacomp PPs used in the benchmarks are generated with the same valence electron numbers as those used in VASP potentials.

The Hylanemos yields the best performance for systems with 48 and 96 atoms, indicating its comparable computa-

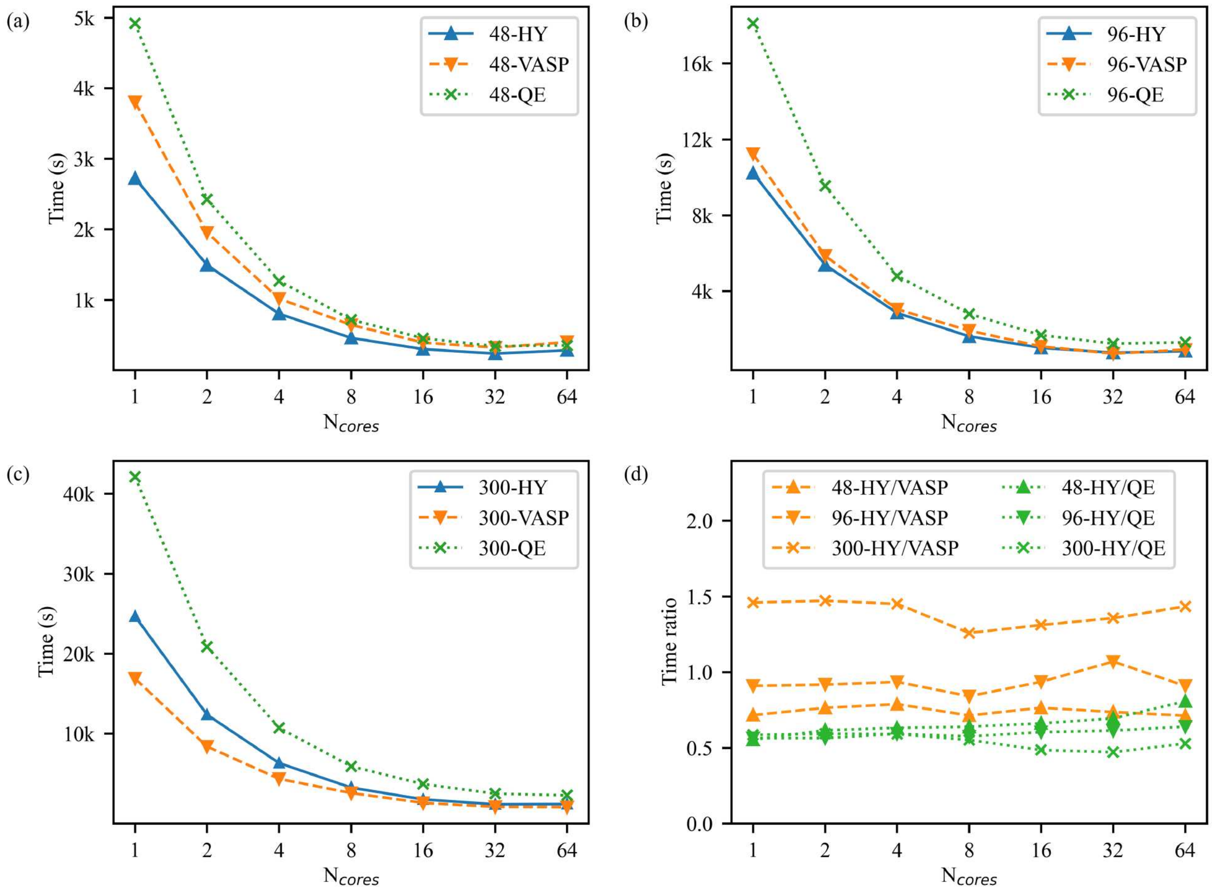

Figure 3 (Color online) Efficiency for different systems with different cores. (a) 48-atom LCO system; (b) 96-atom LCO system; (c) 300-atom LCO system. (d) The ratio of Hylanemos' computation time to that of QE and VASP.

tional efficiency to these widely used packages. VASP demonstrates the best scaling with system size, which can be attributed to its use of real-space calculations for the nonlocal term. In reciprocal space, beta functions are expanded by the entire basis, which grows in size with the system. Conversely, beta functions are local in real space and can be expanded in a nearly fixed-sized grid. As a result, it is efficient to use real space calculations when the system has more than several tens of atoms. Nevertheless, real space calculations can introduce non-negligible errors without careful treatment, which has been addressed by VASP internally. To highlight the importance of real space calculations, we also perform calculations for the non-local term with the reciprocal space method. Our findings reveal a substantial decline in the computational efficiency of VASP. VASP turns out to be slower than both QE and Hylanemos. More details of this comparison are listed in the Supporting Information.

Regarding scaling with the number of processes, QE demonstrates the best performance. This can be largely attributed to the difference in potentials, as previously discussed. When the number of processes is limited to 16, Hylanemos exhibits performance comparable to VASP. However, its performance deteriorates significantly with 32 processes and worse with 64. Notably, for a system with 300 atoms, Hylanemos takes longer to compute using 64 processes than with 32. These findings indicate the need to further optimize the parallelization mechanism in Hylanemos.

It should be noted that the default parallelization scheme in VASP is different from that in QE and Hylanemos. In VASP, the default setting is parallelization over bands, while in QE and Hylanemos, it is parallelization over plane waves. Theoretically, this difference will impact both efficiency and scalability with the number of processes. However, no clear trend can be found after conducting a benchmark test in VASP. The relevant data are included in the Supporting Information.

In the context of computational chemistry software packages, the direct diagonalization in the subspace is a crucial aspect of Davidson-type eigensolvers. However, the computational cost of this subroutine scales cubically with system size and cannot benefit from parallelization over k points, bands, or G points. To address this limitation, parallel linear algebra subroutines such as ScaLapack or ELPA are utilized. Nevertheless, the implementation of these subroutines requires careful preprocessing and postprocessing. In the case of Hylanemos, while it incorporates this feature, it is still in its early stages and only demonstrates a non-negligible effect on large systems, such as those with 300 atoms.

3.5 Presentation of capabilities

3.5.1 Geometry optimization

The three packages provide various algorithms for geometry optimization. In VASP, the residual minimization method with direct inversion in the iterative subspace (RMM-DIIS) is used as the default algorithm in the benchmark. QE utilizes a Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm based on the trust radius procedure (BFGS-tr) as its default algorithm in the benchmark. Hylanemos employs a modified version of the BFGS-tr algorithm from QE as its default, with adjustments to enhance performance on complex systems, such as a change in the criterion for the acceptance of trial points and re-initialization of the basis set when the change of cell is large. Additional algorithms have also been implemented in Hylanemos, including a BFGS algorithm based on line search (BFGS-ls), a two-point steepest descent (TPSD) algorithm, a conjugate gradient (CG) algorithm, and a damped molecular dynamics (DAMPMD) algorithm.

Table 3 demonstrates the lattice parameters of LCO, NCM, and LFP systems calculated by Hylanemos, VASP, and QE. It should be noted that the impact of different dispersion corrections was also evaluated in the LCO system using Hylanemos because of the well-accepted fact that both axes *a* and *c* of the system are overestimated without dispersion corrections. It can be observed that with any dispersion corrections (D2, D3, and D4), axis *a* is still overestimated, albeit to a lesser extent, while axis *c* is underestimated. Consequently, the cell volume aligns much more closely with the experimental value. The results indicate that D3 and D4 corrections perform better than D2 corrections.

The same D3 correction has been applied to the calculations of lattice parameters of LCO, NCM, and LFP systems. As shown, the results calculated by the three packages are close to the experimental values. The lattice shapes remain consistent, with minor distortions observed in NCM, and the cell volume discrepancy is less than 2%.

3.5.2 Electronic structures

The calculated band structures, partial density of states (PDOS), and crystal orbital hamilton population (COHP) by the three packages are shown in Figures 4–6. Due to the arbitrary process of projecting the KS orbitals onto atomic orbitals, there may be variations in the PDOS and COHP results across different packages. However, the band structures are expected to resemble each other.

To demonstrate the band structure, we selected an FCC Al system with 4 atoms. The results, illustrated in Figure 4, exhibit significant overlap. For PDOS and COHP, the LFP system was used, with each package utilizing its own optimized structure. As previously mentioned, there may be slight discrepancies between the structures. Nevertheless, the results displayed in Figures 5 and 6 are consistent with each other.

No COHP results from QE are available. Typically, the LOBSTER [54,55] software is used to calculate COHP from data generated by DFT packages. However, LOBSTER only

Table 3 The comparison of the lattice constants of LCO, NCM, and LFP after optimization by Hylanemos with experimental values and those optimized by VASP (VA) and QE. The numbers of atoms for LCO, NCM, and LFP systems are 12, 108, and 28, respectively

	a (Å)	*b* (Å)	*c* (Å)	*α* (°)	*β* (°)	*γ* (°)
LCO-exp [51]	2.814	2.814	14.048	90.00	90.00	120.00
LCO-HY	2.847 (1.17%)	2.847 (1.17%)	14.149 (0.72%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
LCO-HY d2	2.842 (0.99%)	2.842 (0.99%)	13.681 (−2.61%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
LCO-HY d3	2.829 (0.52%)	2.829 (0.52%)	13.965 (−0.59%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
LCO-HY d4	2.830 (0.57%)	2.830 (0.57%)	13.983 (−0.46%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
LCO-VA d3	2.800 (−0.49%)	2.800 (−0.49%)	13.945 (−0.73%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
LCO-QE d3	2.824 (0.34%)	2.824 (0.34%)	13.950 (−0.70%)	90.00 (0.00%)	90.00 (0.00%)	120.00 (0.00%)
NCM-exp [52]	8.580	8.580	14.227	90.00	90.00	120.00
NCM-HY d3	8.662 (0.96%)	8.665 (0.99%)	14.173 (−0.38%)	90.10 (0.11%)	90.00 (0.00%)	119.99 (−0.01%)
NCM-VA d3	8.559 (−0.25%)	8.562 (−0.21%)	14.120 (−0.75%)	90.08 (0.09%)	90.00 (0.00%)	119.99 (−0.01%)
NCM-QE d3	8.661 (0.94%)	8.664 (0.97%)	14.183 (−0.31%)	90.12 (0.13%)	90.00 (0.00%)	119.99 (−0.01%)
LFP-exp [53]	10.336	6.006	4.693	90.00	90.00	90.00
LFP-HY d3	10.396 (0.58%)	6.046 (0.67%)	4.706 (0.27%)	90.00 (0.00%)	90.00 (0.00%)	90.00 (0.00%)
LFP-VA d3	10.371 (0.33%)	6.015 (0.14%)	4.695 (0.05%)	90.00 (0.00%)	90.00 (0.00%)	90.00 (0.00%)
LFP-QE d3	10.356 (0.19%)	6.040 (0.57%)	4.704 (0.22%)	90.00 (0.00%)	90.00 (0.00%)	90.00 (0.00%)

supports results calculated from PAW potentials. Although QE supports PAW potentials, such a calculation was not carried out. It is worth noting that COHP analysis for PPs is supported in Hylanemos. Therefore, in this instance, we compare the results from Hylanemos with those obtained from VASP+LOBSTER.

3.5.3 Mechanical properties

Due to its precise experimental values, the conventional cell of AgCl with 8 atoms was utilized to study mechanical properties. Because of its high symmetry, only a few entries in the stiffness tensor are non-zero. QE does not offer this capability out of the box, so the ElaStic [56] package is utilized for calculating these mechanical properties in QE.

When comparing QE to Hylanemos and VASP, it is important to note that ElaStic uses a different approach for calculating mechanical properties in two primary aspects. First, Elastic directly computes the "relaxed-ion" elastic tensor through geometry optimization calculations. In contrast, Hylanemos and VASP calculate the "clamped-ion" elastic tensor, internal-strain tensors, and force-constant matrix via SCF calculations and subsequently derive the "relaxed-ion" tensor from these values. For more in-depth information, readers can refer to the research conducted by Hamann et al. [57]. Second, Elastic uses Lagrangian stress and strain, while VASP and Hylanemos use physical stress and strain.

The maximum strain is specified as 0.015 for the three packages; however, the strains are effectively varied because of the different definitions. Additionally, the number of points varied among the packages. A maximum of 5 points in each direction is allowed in VASP, while 7 and 21 points in each direction are used in Hylanemos and QE, respectively.

Table 4 lists the calculated values and experimental values. Despite some variations resulting from the abovementioned disparities, the overall performance remains fairly consistent.

3.5.4 Phonon calculations

In phonon calculations, the primitive cell of the diamond with 2 C atoms was used. Phonopy [59,60] was used for preprocessing and postprocessing for the three packages, and a 2×2×2 supercell was employed for the calculations. As depicted in Figure 7, the results show close similarities across the three packages, with minor differences in the high-frequency region. However, it should be noted that there is a systematic deviation when comparing the results to experi-

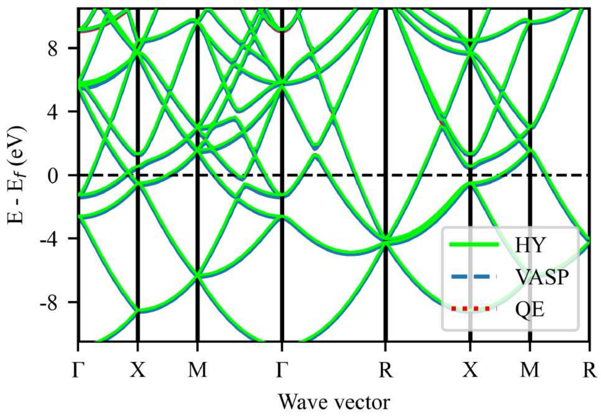

Figure 4 (Color online) Band structures of Al calculated by Hylanemos, VASP, and QE.

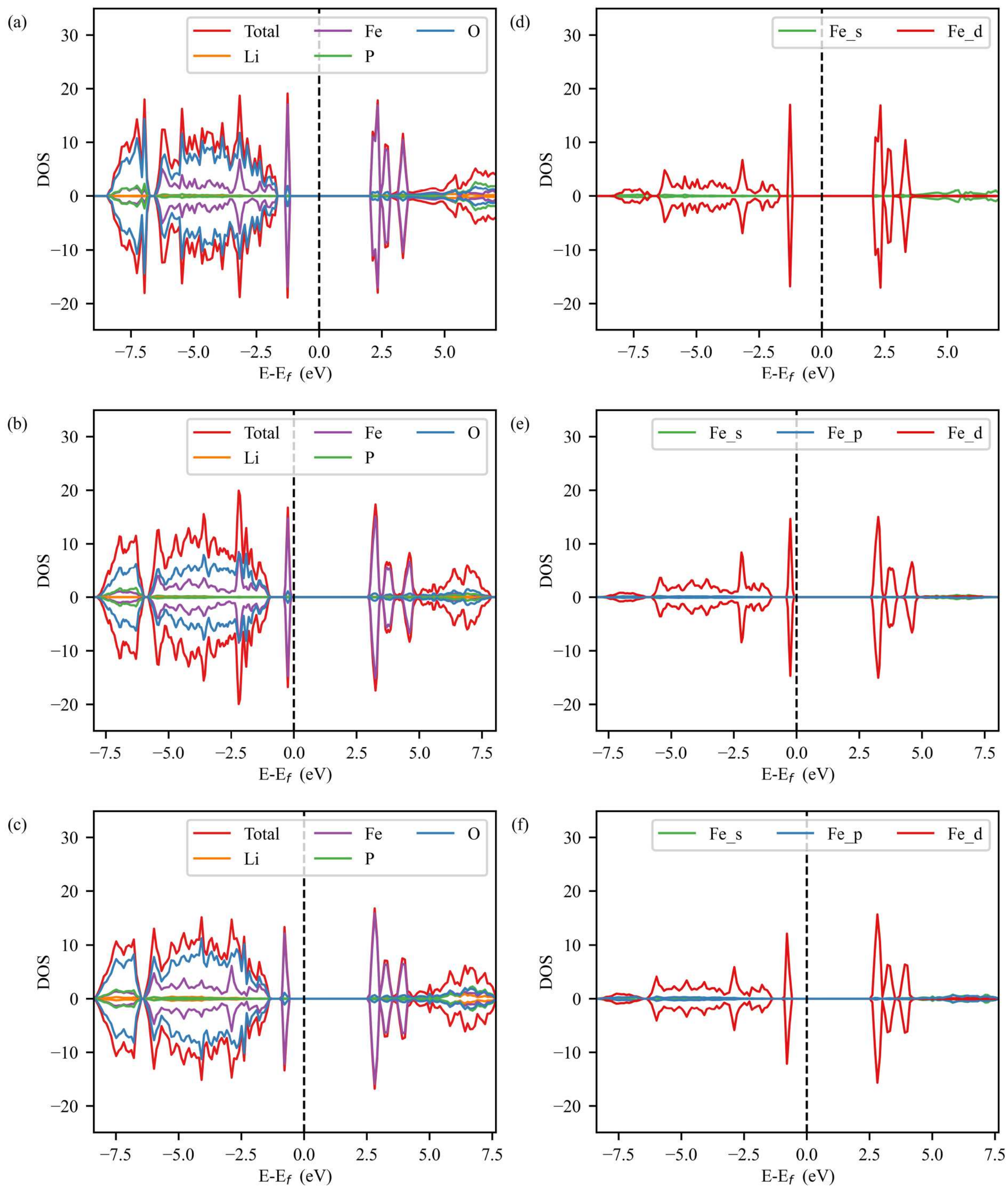

Figure 5 (Color online) Calculated PDOS for LFP. (a)–(c) The total DOS and DOS on each element; (d)–(f) the DOS on different orbitals of Fe. Calculated (a) and (d) by Hylanemos, (b) and (e) by VASP, and (c) and (f) by QE.

mental values [61,62].

3.5.5 *Ion migration*

The nudged elastic band (NEB) method was utilized to calculate the migration of Li on graphene. The system consists of 32 C atoms and one Li atom, with the *z*-axis length of 15 Å. A 6×6×1 k-point mesh was employed, and we specifically focused on the hole-bridge-hole path. The hole site is in the middle of a C_6 unit, while the bridge site is above a C–C bond. The results, depicted in Figure 8, revealed that the calculated diffusion barriers exhibit remarkable similarity.

3.6 LDA+G

The lattice constant of the fcc Ni metal was calculated using both LDA+U and LDA+G methods, with different *U* values.

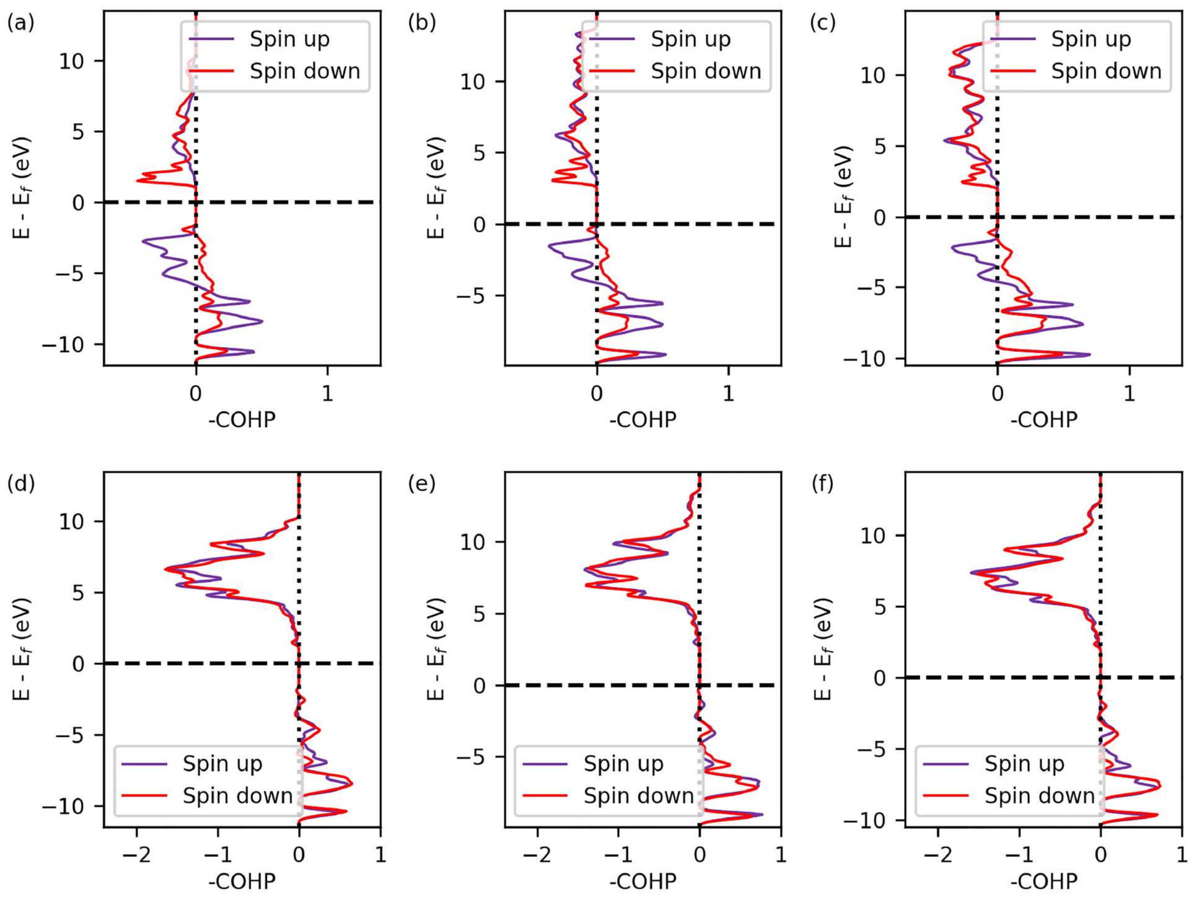

Figure 6 (Color online) Calculated COHP for LFP. (a)–(c) The COHP for the nearest Fe–O, and (d)–(f) for the nearest P–O. Calculated (a) and (d) by Hylanemos with Eacomp PP, (b) and (e) by VASP, and (c) and (f) by Hylanemos with GBRV.

Table 4 Mechanical properties of AgCl calculated by Hylanemos, VASP, and QE, along with experimental values [a)]

AgCl (GPa)	C11	C12	C44	YM	SM	BM
Exp [58]	73.91	39.07	6.94	29.1	10.3	50.7
HY	69.942	41.055	4.821	21.772	7.621	50.684
VASP	70.762	41.578	4.608	21.381	7.473	51.307
QE	73.7	33.9	4.1	23.290	8.214	47.167

a) YM, SM, and BM represent Young's modulus, shear modulus, and bulk modulus, respectively. The outputs of ElaStic for stiffness tensors contain only one digit after the decimal point.

The calculation was based on a conventional cell with 4 Ni atoms, as conducted by Schickling et al. [63]. A k-point mesh of 12×12×12 and the LDA functional, as used in Schickling's work, were employed. PseudoDojo was used because currently, only NCPP is supported.

The experimental value was also obtained from Schickling's work. Figure 9 shows that for U=0, both LDA+U and LDA+G methods produce similar results, as expected. When U=0, both methods have no correction, and their outcome aligns with the basic KS-DFT LDA approach. As the U value increases, the LDA+U method yields a decreasing lattice constant, deviating further from the experimental value. On the contrary, for the LDA+G method, the calculated lattice constant approaches the experimental value as the U value increases. The LDA+G curve is similar to the results from Schickling [53].

4 Conclusions

In this work, we present an analysis of techniques for designing an integrated solution for materials simulations. The core of the proposed approach lies in the flexibility of three layers: the task layer, the SP method layer, and the correction

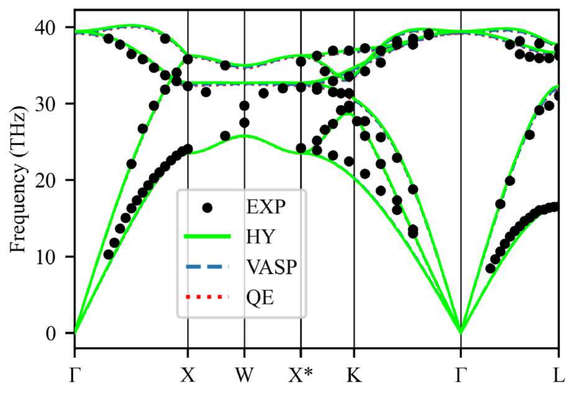

Figure 7 (Color online) Phonon spectrum of diamond calculated by Hylanemos, VASP, and QE, along with experimental results.

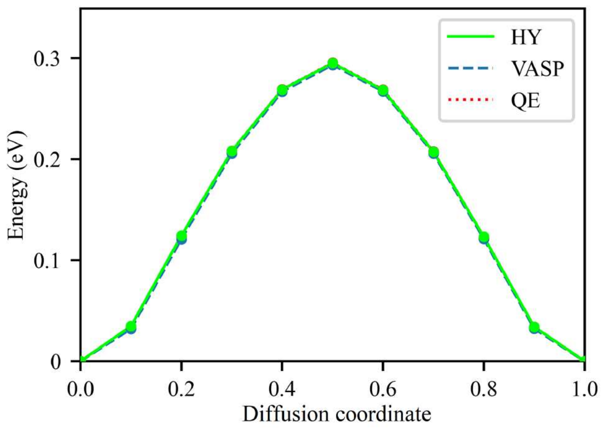

Figure 8 (Color online) Energy barrier of Li diffusion on top of graphene calculated by Hylanemos, VASP, and QE.

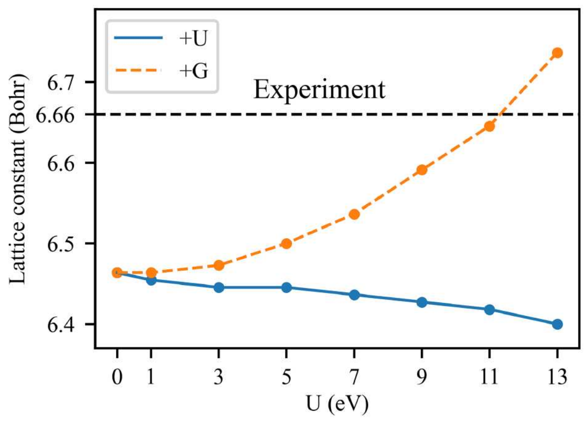

Figure 9 (Color online) Calculated lattice constants of fcc Ni by LDA+U and LDA+G with different *U* values. The dashed line denotes the experimental values.

and modification layer. We demonstrate the use of a distinct function for each new algorithm in the task layer, a multi-layer design for accommodating different SP methods in the second layer, and the utilization of electronic status (ES) to enhance the flexibility of the SCF process in the third layer. With these techniques, we implemented Hylanemos, an integrated solution based on PW-PP KS-DFT. In particular, Hylanemos enables the LDA+G calculations, which is uncommon in the previous KS-DFT packages.

It should be emphasized that the proposed flexible design does not compromise the accuracy or the efficiency of Hylanemos. Besides, we also developed a set of highly optimized USPP called Eacomp PP, which has energy cutoffs and valence electron numbers similar to VASP PAW potentials. A comparison of Hylanemos with VASP and QE on various calculation tasks reveals that the overall performance of Hylanemos is comparable and arguably better than these widely used packages in terms of accuracy and efficiency. The improvements of Eacomp PP over commonly used GBRV USPP are also significant and crucial to achieve efficiency comparable with VASP. Despite the numerous advantages that Hylanemos has demonstrated, it is important to recognize that this is merely the starting point of its journey. The field of computational materials science is evolving at an astonishing pace, and Hylanemos has substantial room for further evolution. Particularly, challenges remain in effectively scaling with the system size and the number of processes. Future work could focus on enhancing the code's parallel efficiency, perhaps by exploring novel parallel algorithms and optimizing the distribution of computational tasks. Additionally, more sophisticated numerical techniques might be introduced to improve the handling of large-scale systems without sacrificing accuracy. We envision that through continuous efforts in research and development, we can bridge these existing gaps, making Hylanemos a more robust and versatile tool for the scientific community and the industry of computational material science applications.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 12426301).

Supporting Information The supporting information is available online at tech.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Hohenberg P, Kohn W. Inhomogeneous electron gas. Phys Rev, 1964, 136: B864–B871
2 Kohn W, Sham L J. Self-consistent equations including exchange and correlation effects. Phys Rev, 1965, 140: A1133–A1138
3 Hamann D R, Schlüter M, Chiang C. Norm-conserving pseudopotentials. Phys Rev Lett, 1979, 43: 1494–1497
4 Vanderbilt D. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. Phys Rev B, 1990, 41: 7892–7895
5 Blöchl P E. Projector augmented-wave method. Phys Rev B, 1994, 50: 17953–17979
6 Clark S J, Segall M D, Pickard C J, et al. First principles methods using CASTEP. Z Kristallogr-Cryst Mater, 2005, 220: 567–570
7 Giannozzi P, Baroni S, Bonini N, et al. QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of

materials. J Phys-Condens Matter, 2017, 21: 395502

8 Gonze X, Amadon B, Antonius G, et al. The Abinitproject: Impact, environment and recent developments. Comput Phys Commun, 2020, 248: 107042

9 Kresse G, Furthmüller J. Efficiency of *ab-initio* total energy calculations for metals and semiconductors using a plane-wave basis set. Comput Mater Sci, 1996, 6: 15–50

10 Jia W, Fu J, Cao Z, et al. Fast plane wave density functional theory molecular dynamics calculations on multi-GPU machines. J Comput Phys, 2013, 251: 102–115

11 Jia W, Cao Z, Wang L, et al. The analysis of a plane wave pseudopotential density functional theory code on a GPU machine. Comput Phys Commun, 2013, 184: 9–18

12 Smidstrup S, Markussen T, Vancraeyveld P, et al. QuantumATK: An integrated platform of electronic and atomic-scale modelling tools. J Phys-Condens Matter, 2020, 32: 015901

13 Chen M, Guo G C, He L. Systematically improvable optimized atomic basis sets for *ab initio* calculations. J Phys-Condens Matter, 2010, 22: 445501

14 Li P, Liu X, Chen M, et al. Large-scale *ab initio* simulations based on systematically improvable atomic basis. Comput Mater Sci, 2016, 112: 503–517

15 Sun J, Ruzsinszky A, Perdew J P. Strongly constrained and appropriately normed semilocal density functional. Phys Rev Lett, 2015, 115: 036402

16 Heyd J, Scuseria G E, Ernzerhof M. Hybrid functionals based on a screened Coulomb potential. J Chem Phys, 2003, 118: 8207–8215

17 Krukau A V, Vydrov O A, Izmaylov A F, et al. Influence of the exchange screening parameter on the performance of screened hybrid functionals. J Chem Phys, 2006, 125: 224106

18 Anisimov V I, Zaanen J, Andersen O K. Band theory and Mott insulators: Hubbard U instead of Stoner I. Phys Rev B, 1991, 44: 943–954

19 Liechtenstein A I, Anisimov V I, Zaanen J. Density-functional theory and strong interactions: Orbital ordering in Mott-Hubbard insulators. Phys Rev B, 1995, 52: R5467–R5470

20 Dudarev S L, Botton G A, Savrasov S Y, et al. Electron-energy-loss spectra and the structural stability of nickel oxide: An LSDA+U study. Phys Rev B, 1998, 57: 1505–1509

21 Gutzwiller M C. Effect of correlation on the ferromagnetism of transition metals. Phys Rev Lett, 1963, 10: 159–162

22 Gutzwiller M C. Effect of correlation on the ferromagnetism of transition metals. Phys Rev, 1964, 134: A923–A941

23 Gutzwiller M C. Correlation of electrons in a narrow s band. Phys Rev, 1965, 137: A1726–A1735

24 Bengtsson L. Dipole correction for surface supercell calculations. Phys Rev B, 1999, 59: 12301–12304

25 Andreussi O, Dabo I, Marzari N. Revised self-consistent continuum solvation in electronic-structure calculations. J Chem Phys, 2012, 136: 064102

26 Dederichs P H, Blügel S, Zeller R, et al. Ground states of constrained systems: Application to cerium impurities. Phys Rev Lett, 1984, 53: 2512–2515

27 Baroni S, Giannozzi P, Testa A. Green's-function approach to linear response in solids. Phys Rev Lett, 1987, 58: 1861–1864

28 Baer R, Neuhauser D. Real-time linear response for time-dependent density-functional theory. J Chem Phys, 2004, 121: 9803–9807

29 Hedin L. New method for calculating the one-particle green's function with application to the electron-gas problem. Phys Rev, 1965, 139: A796–A823

30 Aryasetiawan F, Gunnarsson O. The GW method. Rep Prog Phys, 1998, 61: 237–312

31 Henkelman G, Jónsson H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. J Chem Phys, 1999, 111: 7010–7022

32 Heyden A, Bell A T, Keil F J. Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method. J Chem Phys, 2005, 123: 224101

33 Kästner J, Sherwood P. Superlinearly converging dimer method for transition state search. J Chem Phys, 2008, 128: 014106

34 Jónsson H, Mills G, Jacobsen K W. Nudged elastic band method for finding minimum energy paths of transitions. In: Berne B J, Ciccotti G, Coker D F (eds). Classical and Quantum Dynamics in Condensed Phase Simulations. Singapore: World Scientific, 1998. 385–404

35 Mathew K, Sundararaman R, Letchworth-Weaver K, et al. Implicit solvation model for density-functional study of nanocrystal surfaces and reaction pathways. J Chem Phys, 2014, 140: 084106

36 Giannozzi P, Andreussi O, Brumme T, et al. Advanced capabilities for materials modelling with Quantum ESPRESSO. J Phys-Condens Matter, 2017, 29: 465901

37 Grimme S. Semiempirical GGA-type density functional constructed with a long-range dispersion correction. J Comput Chem, 2006, 27: 1787–1799

38 Grimme S, Antony J, Ehrlich S, et al. A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. J Chem Phys, 2010, 132: 154104

39 Caldeweyher E, Mewes J M, Ehlert S, et al. Extension and evaluation of the D4 London-dispersion model for periodic systems. Phys Chem Chem Phys, 2020, 22: 8499–8512

40 Bünemann J, Gebhard F, Schickling T, et al. Numerical minimisation of Gutzwiller energy functionals. Phys Status Solidi (b), 2012, 249: 1282–1291

41 Lanatà N, Strand H U R, Dai X, et al. Efficient implementation of the Gutzwiller variational method. Phys Rev B, 2012, 85: 035133

42 Milan C. Modern Fortran: Building Efficient Parallel Applications. New York: Manning Publications, 2020. 85

43 Laasonen K, Pasquarello A, Car R, et al. Car-Parrinello molecular dynamics with Vanderbilt ultrasoft pseudopotentials. Phys Rev B, 1993, 47: 10142–10153

44 Chan C T, Bohnen K P, Ho K M. Accelerating the convergence of force calculations in electronic-structure computations. Phys Rev B, 1993, 47: 4771–4774

45 MatterCraft. Version 3.4. Shenzhen: Shenzhen Eacomp Technology Co., Ltd., 2004

46 Lehtola S, Steigemann C, Oliveira M J T, et al. Recent developments in libxc—A comprehensive library of functionals for density functional theory. SoftwareX, 2018, 7: 1–5

47 Garrity K F, Bennett J W, Rabe K M, et al. Pseudopotentials for high-throughput DFT calculations. Comput Mater Sci, 2014, 81: 446–452

48 Perdew J P, Burke K, Ernzerhof M. Generalized gradient approximation made simple. Phys Rev Lett, 1996, 77: 3865–3868

49 Grimme S, Ehrlich S, Goerigk L. Effect of the damping function in dispersion corrected density functional theory. J Comput Chem, 2011, 32: 1456–1465

50 van Setten M J, Giantomassi M, Bousquet E, et al. The PseudoDojo: Training and grading a 85 element optimized norm-conserving pseudopotential table. Comput Phys Commun, 2018, 226: 39–54

51 Jin K H, Seung M O. Crystal structure and electrochemical performance of $LiNi_{1-x}Co_xO_2$ (x = 0.0–1.0) according to Co substitution. J Korean Chem Soc, 2003, 6: 1–5

52 Yin S C, Rho Y H, Swainson I, et al. X-ray/neutron diffraction and electrochemical studies of lithium De/Re-intercalation in $Li_{1-x}Co_{1/3}Ni_{1/3}Mn_{1/3}O_2$ ($x = 0 \rightarrow 1$). Chem Mater, 2006, 18: 1901–1910

53 Janssen Y, Santhanagopalan D, Qian D, et al. Reciprocal salt flux growth of $LiFePO_4$ single crystals with controlled defect concentrations. Chem Mater, 2013, 25: 4574–4584

54 Dronskowski R, Bloechl P E. Crystal orbital Hamilton populations (COHP): Energy-resolved visualization of chemical bonding in solids based on density-functional calculations. J Phys Chem, 1993, 97: 8617–8624

55 Deringer V L, Tchougréeff A L, Dronskowski R. Crystal orbital hamilton population (COHP) analysis as projected from plane-wave basis sets. J Phys Chem A, 2011, 115: 5461–5466

56 Golesorkhtabar R, Pavone P, Spitaler J, et al. ElaStic: A tool for calculating second-order elastic constants from first principles. Comput Phys Commun, 2013, 184: 1861–1873

57 Wu X, Vanderbilt D, Hamann D R. Systematic treatment of displacements, strains, and electric fields in density-functional perturbation theory. Phys Rev B, 2005, 72: 035105

58 Gene S, Herbert W. Single Crystal Elastic Constants and Calculated Aggregate Properties: A Handbook. Cambridge: MIT Press, 1971. 87

59 Togo A, Chaput L, Tadano T, et al. Implementation strategies in phonopy and phono3py. J Phys-Condens Matter, 2023, 35: 353001

60 Togo A. First-principles phonon calculations with phonopy and phono3py. J Phys Soc Jpn, 2023, 92: 012001

61 Warren J L, Yarnell J L, Dolling G, et al. Lattice dynamics of diamond. Phys Rev, 1967, 158: 805–808

62 Warren J L, Wenzel R G, Yarnell J L. Dispersion curves for phonons in diamond. In: Proceedings of the Symposium on Inelastic Scattering of Neutrons. Bombay: International Atomic Energy Agency, 1964. 361–371

63 Schickling T, Bünemann J, Gebhard F, et al. Gutzwiller density functional theory: A formal derivation and application to ferromagnetic nickel. New J Phys, 2014, 16: 093034