Computer Physics Communications 135 (2001) 348–376

Computer Physics Communications

www.elsevier.nl/locate/cpc

A Projector Augmented Wave (PAW) code for electronic structure calculations, Part II: *pwpaw* for periodic solids in a plane wave basis

A.R. Tackett [a], N.A.W. Holzwarth [b,*], G.E. Matthews [b]

[a] *Department of Physics and Astronomy, Vanderbilt University, Nashville, TN 37235, USA*
[b] *Department of Physics, Wake Forest University, Winston-Salem, NC 27109, USA*

Received 21 July 2000; accepted 26 October 2000

Abstract

The *pwpaw* code is a plane wave implementation of the Projector Augmented Wave (PAW) method developed by Blöchl for electronic structure calculations within the framework of density functional theory. In addition to the self-consistent calculation of the electronic structure of a periodic solid, the program has a number of other capabilities, including structural geometry optimization and molecular dynamics simulations within the Born–Oppenheimer approximation.

PACS: 71.15.Ap; 71.15.Hx; 71.15.Mb; 71.15.Nc; 71.15.Pd; 2.70.c; 7.05.Tp

Keywords: Electronic structure calculations; Density functional calculation; Local density approximation; Projector Augmented Wave method; PAW; Calculational methods

PROGRAM SUMMARY

Title of program: *pwpaw*

Catalogue identifier: ADNP

Program Summary URL: http://cpc.cs.qub.ac.uk/summaries/ADNP

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Computers on which code has been tested: DEC Alpha, IBM SP2

Operating systems under which the program has been tested: Unix

Programming language used: Fortran90

Memory required to execute with typical data: 100 Mbytes

No. of bytes in distributed program, including test data, etc.: 16 762 946

Distribution format: gzip tar file

Library packages needed for running this code: BLAS, LAPACK (available from http://www.netlib.org), and FFTW (available from http://www.fftw.org)

* Corresponding author. Address: Department of Physics, Wake Forest University, Winston-Salem, NC 27109, USA.
E-mail addresses: alan.r.tackett@vanderbilt.edu (A.R. Tackett), natalie@wfu.edu (N.A.W. Holzwarth).

0010-4655/01/$ – see front matter
PII: S0010-4655(00)00241-1

Nature of physical problem
The projector augmented wave (PAW) method, developed by Blöchl, is a very powerful tool for performing electronic structure calculations in the framework of density functional theory, combining some of the best features of pseudopotential and all-electron approaches. The *pwpaw* program finds the one-electron eigenfunctions and eigenvalues for a periodic system, and optionally optimizes or performs molecular dynamics on the atomic positions within the unit cell.

Method of solution
The program initializes the wavefunctions with a linear combination of atomic orbitals (LCAO) or with a random number generator and determines the eigenstates of the generalized eigenvalue problem by iterative diagonalization. The atomic forces are calculated, using a modified Feynman–Hellmann approach, from a knowledge of the converged eigenstates.

Restrictions on the complexity of the program
In this version of the code, only serial processing has been implemented. In addition, of the many exchange-correlation functionals, only the local density approximation (LDA) is currently available. Also, relativistic and magnetic effects are not yet coded.

Typical running time
Roughly 3–15 minutes/atom for each geometry on an SP2 computer.

Unusual features of the program
The program sequence is controlled by a keyword input file. A memory management scheme is implemented which enables the user to tune the program to make optimal use of available computer resources.

LONG WRITE-UP

1. Introduction

The projector augmented wave (PAW) method, developed by Blöchl [1], is a very powerful tool for performing electronic structure calculations within the framework of density functional theory [2,3], combining some of the best features of pseudopotential and all-electron approaches. In addition to Blöchl's original paper [1], a number of papers [4–7] have discussed the details of the method. The *pwpaw* code is an implementation of the PAW method for periodic solids using a plane wave basis. It is designed to use output from the *atompaw* [8] program which generates the necessary atom-centered projector and basis functions. The structure of the program was conceived and developed by Tackett [9] in his Ph.D. work on a "real-space" implementation of the PAW formalism. The program is designed to be user-friendly in the sense that the input file makes good use of keywords and allows comments. Also, several options exist to maximize usage of available machine resources. For example, the user can specify the maximum memory usage of the program. If additional storage is needed, the program makes efficient use of disk access rather than relying on virtual memory of the machine. Its modular form also makes the program relatively easy to modify.

2. Formalism

2.1. PAW representation of electronic wavefunctions density

The PAW formalism has been well-described in earlier work [1,4]. For convenience, we present here some of the important equations.

In the PAW formalism, the calculations are performed entirely in terms of the smooth "pseudo" wave functions. For a periodic solid, an eigenstate of the system has a band index n and wave vector $\mathbf{k}$. The corresponding pseudo-wavefunction can be represented in terms of a plane wave expansion:

$$\widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r}) = \sqrt{\frac{1}{\mathcal{V}}} \sum_{\mathbf{G}} A_{n\mathbf{k}}(\mathbf{G}) \mathrm{e}^{\mathrm{i}(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}}, \tag{1}$$

were **G** denotes a reciprocal lattice vector and $\mathcal{V}$ denotes the volume of the unit cell. From a knowledge of the pseudowave function and the PAW basis and projector functions, one can reconstruct the corresponding fully nodal eigenstate of the system according to

$$\Psi_{n\mathbf{k}}(\mathbf{r}) = \widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r}) + \sum_{ai} \left(\phi_i^a(\mathbf{r}-\mathbf{R}^a) - \tilde{\phi}_i^a(\mathbf{r}-\mathbf{R}^a)\right)\left\langle \tilde{p}_i^a | \widetilde{\Psi}_{n\mathbf{k}}\right\rangle, \tag{2}$$

where the projector $|\tilde{p}_i^a\rangle$ and basis functions $|\phi_i^a\rangle$ and $|\tilde{\phi}_i^a\rangle$ are known functions which can be generated using the program *atompaw* [8]. The electron density can be determined as a sum of three different terms:

$$n(\mathbf{r}) = \tilde{n}(\mathbf{r}) + \sum_a \left(n^a(\mathbf{r}-\mathbf{R}^a) - \tilde{n}^a(\mathbf{r}-\mathbf{R}^a)\right). \tag{3}$$

The first term is the pseudo-density which can be represented as a Fourier expansion:

$$\tilde{n}(\mathbf{r}) = \sum_{n\mathbf{k}} f_{n\mathbf{k}} \left|\widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r})\right|^2 = \frac{1}{\mathcal{V}} \sum_{\mathbf{G}} \bar{\tilde{n}}(\mathbf{G}) \mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}}. \tag{4}$$

Here $f_{n\mathbf{k}}$ represents the occupancy weighted by the fractional Brillouin zone sampling volume. The atom-centered density terms can easily be evaluated in terms of the projected occupation coefficients defined according to:

$$W_{ij}^a \equiv \sum_{n\mathbf{k}} f_{n\mathbf{k}} \left\langle \widetilde{\Psi}_{n\mathbf{k}} | \tilde{p}_i^a \right\rangle \left\langle \tilde{p}_j^a | \widetilde{\Psi}_{n\mathbf{k}} \right\rangle. \tag{5}$$

In these terms, the atom-centered density contributions can then be written:

$$n^a(\mathbf{r}) = \sum_{ij} W_{ij}^a \phi_i^{a*}(\mathbf{r}) \phi_j^a(\mathbf{r}) \quad \text{and} \quad \tilde{n}^a(\mathbf{r}) = \sum_{ij} W_{ij}^a \tilde{\phi}_i^{a*}(\mathbf{r}) \tilde{\phi}_j^a(\mathbf{r}). \tag{6}$$

It is also convenient to introduce a compensation charge density as a sum of atom-centered contributions $\hat{n}(\mathbf{r}) \equiv \sum_a \hat{n}^a(\mathbf{r}-\mathbf{R}^a)$, with the atom-centered functions taking the form

$$\hat{n}^a(\mathbf{r}) = \sum_{LM} Q_{LM}^a g_{LM}(\mathbf{r}), \tag{7}$$

where the functions $g_{LM}(\mathbf{r})$ have been defined in [8, Eq. (14)] and Q_{LM}^a represents a moment of compensation charge which can be calculated according to [4, Eq. (A22)].

2.2. PAW representation of the energy and effective Hamiltonian

Using the terms above and the notation of Refs. [4,8], the cohesive energy of the system is then given by [1]

$$-E_{\mathrm{coh}} = \widetilde{E} + \sum_a \left(E^a - \widetilde{E}^a - E_{\mathrm{atom}}^a\right). \tag{8}$$

The first term represents the pseudopotential-like contributions which take the form

$$\begin{aligned}\widetilde{E} = {} & \sum_{n\mathbf{k}} f_{n\mathbf{k}} \left(\sum_{\mathbf{G}} \frac{\hbar^2 |\mathbf{k}+\mathbf{G}|^2}{2m} |A_{n\mathbf{k}}(\mathbf{G})|^2 \right) + \frac{2\pi \mathrm{e}^2}{\mathcal{V}} \sum_{\mathbf{G}\neq 0} \frac{|\bar{\tilde{n}}(\mathbf{G}) + \bar{\hat{n}}(\mathbf{G})|^2}{G^2} \\ & + \frac{1}{\mathcal{V}} \sum_{\mathbf{G}} \bar{\tilde{v}}_{\mathrm{loc}}(\mathbf{G}) \bar{\tilde{n}}^*(\mathbf{G}) + E_{\mathrm{xc}}[\tilde{n}].\end{aligned} \tag{9}$$

[1] As noted in Ref. [8], we set the core tail function defined in Ref. [4] identically to zero in this formulation.

The remaining terms of Eq. (8) are all atom-centered terms. The term E^a_{atom} represents the atomic valence total energy calculated by the *atompaw* program. The other atom-centered terms can be determined from

$$E^a - \widetilde{E}^a = \sum_{ij} W^a_{ij}\big(K^a_{ij} + [v^a_{\text{at}}]_{ij} - [\hat{v}^a]_{ij} + \tfrac{1}{2}[V^a_{\text{H}}]_{ij}\big) - \widehat{E}^a + \big(E_{\text{xc}}[n^a_{\text{core}} + n^a] - E_{\text{xc}}[n^a_{\text{core}}] - E_{\text{xc}}[\tilde{n}^a]\big). \tag{10}$$

In this expression, $K^a_{ij} \equiv K^a_{n_i l_i n_j l_i}\delta_{l_i l_j}\delta_{m_i m_j}$ and $[\hat{v}^a]_{ij} \equiv \langle\tilde{\phi}^a_i|\hat{v}^a|\tilde{\phi}^a_j\rangle$. The matrix elements $K^a_{n_i l_i n_j l_i}$, $\langle\tilde{\phi}^a_i|\hat{v}^a|\tilde{\phi}^a_j\rangle$, and $[V^a_{\text{H}}]_{ij}$ are defined in Eqs. (A10), (A23), and (A26) of Ref. [4], respectively. In addition, $[v^a_{\text{at}}]_{ij} \equiv [v^a_{\text{at}}]_{n_i l_i n_j l_i}\delta_{l_i l_j}\delta_{m_i m_j}$, where $[v^a_{\text{at}}]_{n_i l_i n_j l_i}$ is defined in [8, Eq. (26)]. The term $\bar{\tilde{v}}_{\text{loc}}(\mathbf{G})$ denotes the Fourier transform of the local potential which is a sum of atom-centered contributions of the form given in [8, Eq. (10)]. The exchange-correlation energy terms E_{xc} are currently evaluated using the local density approximation of Perdew and Wang [10], although additional functionals could easily be added. The self-Coulomb repulsion of the compensation charge is evaluated in terms of the tabulated atomic moment terms [8, Eq. (27)] according to

$$\widehat{E}^a \equiv \sum_{LM} \big|Q^a_{LM}\big|^2 \widehat{E}^{aL}. \tag{11}$$

By evaluating the functional variation of the cohesive energy with respect to $|\widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r})\rangle$, Blöchl derived the Kohn–Sham equations [3] for the PAW formalism which take the form of a generalized eigenvalue problem:

$$\big\{\mathbf{H}^{\text{PAW}}(\mathbf{r}) - E_{n\mathbf{k}}\mathbf{O}\big\}\big|\widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r})\big\rangle = 0, \tag{12}$$

where

$$\mathbf{H}^{\text{PAW}} \equiv \widetilde{H}(\mathbf{r}) + \sum_{aij}\big|\tilde{p}^a_i\big\rangle D^a_{ij}\big\langle\tilde{p}^a_j\big| \quad \text{and} \quad \mathbf{O} \equiv \mathbf{1} + \sum_{aij}\big|\tilde{p}^a_i\big\rangle O^a_{ij}\big\langle\tilde{p}^a_j\big|. \tag{13}$$

The local term contribution to the PAW Hamiltonian is given by

$$\widetilde{H}(\mathbf{r}) = -\frac{\hbar^2}{2m}\nabla^2 + \tilde{v}_{\text{eff}}(\mathbf{r}), \tag{14}$$

with the smooth local potential given by

$$\tilde{v}_{\text{eff}}(\mathbf{r}) = \frac{1}{\mathcal{V}}\sum_{\mathbf{G}}\bar{\tilde{v}}_{\text{loc}}(\mathbf{G})\mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}} + \frac{4\pi\mathrm{e}^2}{\mathcal{V}}\sum_{\mathbf{G}\neq 0}\frac{\bar{\tilde{n}}(\mathbf{G}) + \bar{\hat{n}}(\mathbf{G})}{G^2}\mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}} + \mu_{\text{xc}}\big[\tilde{n}(\mathbf{r})\big]. \tag{15}$$

The last term in this expression is the exchange-correlation potential function [10]. The non-local contribution to the PAW Hamiltonian is given by

$$D^a_{ij} = K^a_{ij} + \big[v^a_{\text{at}}\big]_{ij} - \big[\hat{v}^a\big]_{ij} + \big[V^a_{\text{H}}\big]_{ij} + \big[v^a_0\big]_{ij} + \big[V^a_{\text{xc}}\big]_{ij} \tag{16}$$

and

$$O^a_{ij} \equiv \big\langle\phi^a_i\big|\phi^a_j\big\rangle - \big\langle\tilde{\phi}^a_i\big|\tilde{\phi}^a_j\big\rangle = O^a_{n_i l_i n_j l_j}\delta_{l_i l_j}\delta_{m_i m_j}. \tag{17}$$

The matrix elements $[v^a_0]_{ij}$ and $[V^a_{\text{xc}}]_{ij}$ are slightly modified (as indicated in the footnote above) from their definitions in Ref. [4], Eqs. (A27) and (A29), respectively.

2.3. PAW representation of the atomic forces

The generalized eigenvalue problem of Eq. (12) is solved by several iterative techniques and the self-consistent-field (SCF) cycles are iterated at the same time as discussed below. After convergence, the self-consistent densities and matrix elements are used to calculate the effective forces on each atom. Since we are working with eigenstates

of the Hamiltonian and since only the compensation charge density terms, the local potential terms, and the projector matrix elements depend upon the atomic positions, the expression for the atomic force is somewhat simplified from the original derivation of Blöchl [1]. The force on atom a at the site $\mathbf{R}^a$ is given by

$$\mathbf{F}^a \equiv \left\{\nabla_{\mathbf{R}^a}[E_{\text{coh}}]\right\} = \frac{4\pi \mathrm{i} e^2}{\mathcal{V}} \sum_{\mathbf{G}\neq 0} \frac{\mathbf{G}\,\bar{\tilde{n}}^a(\mathbf{G})[\bar{\tilde{n}}^*(\mathbf{G}) + \bar{\tilde{n}}^*(\mathbf{G})]}{G^2} + \frac{\mathrm{i}}{\mathcal{V}} \sum_{\mathbf{G}\neq 0} \mathbf{G}\,\bar{\tilde{v}}^a_{\text{loc}}(\mathbf{G})\bar{\tilde{n}}^*(\mathbf{G}) - \sum_{ij} \left\{\nabla_{\mathbf{R}^a}[W^a_{ij}]\right\} D^a_{ij} + \sum_{ij} \left\{\nabla_{\mathbf{R}^a}[U^a_{ij}]\right\} O^a_{ij}. \tag{18}$$

The first contribution depends on the Fourier transform of the atom-centered compensation charge (Eq. (6)) and the second contribution depends on the Fourier transform of the atom centered local potential [8, Eq. (10)]. The last term of the force equation involves a weighted projected occupation coefficient which we define according to

$$U^a_{ij} \equiv \sum_{n\mathbf{k}} f_{n\mathbf{k}} E_{n\mathbf{k}} \langle \widetilde{\Psi}_{n\mathbf{k}} | \tilde{p}^a_i \rangle \langle \tilde{p}^a_j | \widetilde{\Psi}_{n\mathbf{k}} \rangle. \tag{19}$$

The gradient with respect to the atomic position of both W^a_{ij} and U^a_{ij} depends on the gradient of the matrix elements $\langle \nabla_{\mathbf{R}^a}[\tilde{p}^a_i] | \widetilde{\Psi}_{n\mathbf{k}} \rangle$ which can be conveniently evaluated in Fourier space using [4, Eq. (A20)].

2.4. Algorithms for solving generalized eigenvalue problem and for finding self-consistent electron density.

The art of self-consistent solutions of the Kohn–Sham equations is well developed [11], especially since the innovative ideas of Car and Parrinello [12] showed that numerical methods of optimization can solve these equations efficiently. In order to be able to treat metallic systems and to take advantage of iterative diagonalization methods, we do not use the Car–Parrinello algorithm, but instead use the following procedure.

(1) Start with a set of initial pseudowavefunctions $\{\widetilde{\Psi}^0_{n\mathbf{k}}\}$, and occupancy factors $\{f^0_{n\mathbf{k}}\}$ obtained from previous calculation, from a linear combination of atomic orbital (LCAO) functions, or from a random initial guess.
(2) Calculate E^0_{coh} (Eq. (8)), $\tilde{v}^0_{\text{eff}}$ (Eq. (15)), and $[D^a_{ij}]^0$ (Eq. (16)).
(3) Start iteration loop, $\alpha = 0$.
  (a) Calculate $[\mathbf{H}^{\text{PAW}}]^\alpha$ (Eq. (13)).
  (b) Use modified block Davidson algorithm [13,14] to update $\{\widetilde{\Psi}^{\alpha+1}_{n\mathbf{k}}\}$ and $E^{\alpha+1}_{n\mathbf{k}}$ from knowledge of $\{[\mathbf{H}^{\text{PAW}}]^\alpha - E^\alpha_{n\mathbf{k}}\mathbf{O}\}|\widetilde{\Psi}^\alpha_{n\mathbf{k}}\rangle$.
  (c) From the new band energies, $E^{\alpha+1}_{n\mathbf{k}}$, update the occupancy factors $\{f^{\alpha+1}_{n\mathbf{k}}\}$, using a Gaussian smoothing function (23), described below.
  (d) Calculate $E^{\alpha+1}_{\text{coh}}$, $\tilde{v}^{\alpha+1}_{\text{eff}}$, and $[D^a_{ij}]^{\alpha+1}$.
  (e) Calculate merit function $\mathit{Merit} \equiv |E^{\alpha+1}_{\text{coh}} - E^\alpha_{\text{coh}}|$.
    (i) If $\mathit{Merit} \leqslant \mathit{MeritTol}$, calculation is complete.
    (ii) If $\mathit{Merit} > \mathit{MeritTol}$, update $\tilde{v}^{\alpha+1}_{\text{eff}}$ and $[D^a_{ij}]^{\alpha+1}$ using mixing accelerator [15,16]. Set $\alpha \to \alpha + 1$ and continue iteration loop.

The block Davidson algorithm [17,18] can be described as follows.

For each wave vector $\mathbf{k}$, we start with the current set of $N^\alpha_{\mathbf{k}}$ pseudowavefunctions $\{|\widetilde{\Psi}^\alpha_{n\mathbf{k}}\rangle\}$ and generate $N^\alpha_{\mathbf{k}}$ additional functions defined by

$$\left\{|\tilde{\mathcal{R}}^\alpha_{n\mathbf{k}}\rangle \equiv K\left([\mathbf{H}^{\text{PAW}}]^\alpha - E^\alpha_{n\mathbf{k}}\mathbf{O}\right)|\widetilde{\Psi}^\alpha_{n\mathbf{k}}\rangle\right\},$$

where K represents a suitable preconditioner [11]. Davidson's algorithm consists of seeking the updated wavefunctions as an optimized linear combination of these $2N_{\mathbf{k}}^{\alpha}$ functions. If we label these $2N_{\mathbf{k}}^{\alpha}$ functions as $|\Phi_i\rangle$, then

$$\left|\widetilde{\Psi}_{n\mathbf{k}}^{\alpha+1}\right\rangle = \sum_{i=1}^{2N_{\mathbf{k}}^{\alpha}} C_i^n |\Phi_i\rangle. \tag{20}$$

The coefficients C_i^n are determined as eigenvectors of the $2N_{\mathbf{k}}^{\alpha}$ by $2N_{\mathbf{k}}^{\alpha}$ generalized eigenvalue problem of the form:

$$\sum_j \mathcal{H}_{ij} C_j^n = E_{n\mathbf{k}}^{\alpha+1} \sum_j \mathcal{O}_{ij} C_j^n, \tag{21}$$

where $\mathcal{H}_{ij} \equiv \langle\Phi_i|[\mathbf{H}^{\mathrm{PAW}}]^{\alpha}|\Phi_j\rangle$ and $\mathcal{O}_{ij} \equiv \langle\Phi_i|\mathbf{O}|\Phi_j\rangle$. Since, especially near convergence, the $\mathcal{O}_{ij}$ can be quite singular, the generalized eigenvalue problem is solved by prediagonalizing the $\mathcal{O}_{ij}$ matrix and keeping only the non-singular portion.

3. Description of the program

Fig. 1 shows the basic structure of the *pwpaw* program. There are three types of basic steps.

(1) Load basic information about the system. This can be grouped into five different types of input.
(2) Set up data structures and initialize variables to start calculation. This is accomplished by a call to the subroutine *Initialize_System* followed by the initialization of the electron wave functions, using either the results of a previous calculation, a linear combination of atomic orbitals (LCAO), or a random number generator.
(3) Perform actual calculation according to *commands* in input file.

Each of these steps will be discussed in more detail below.

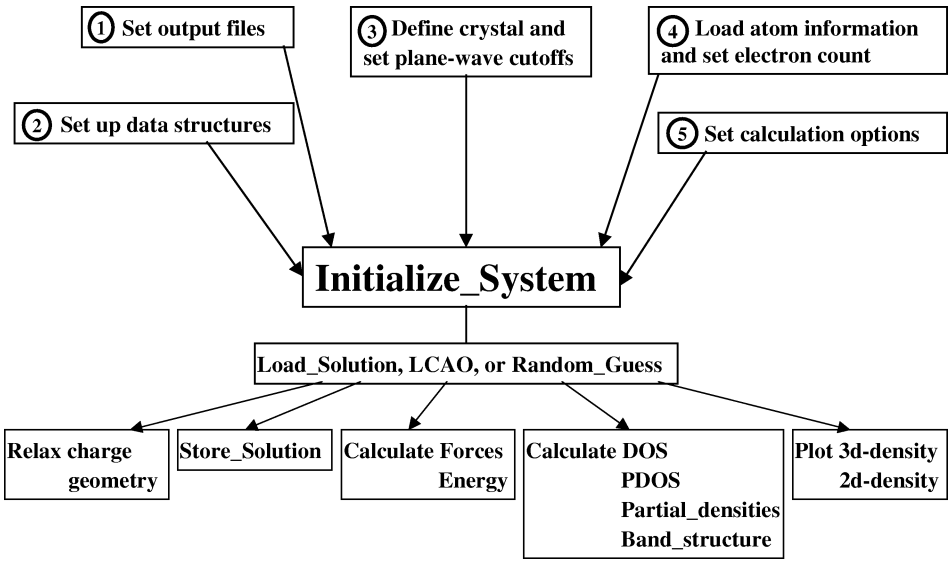

Fig. 1. Flow chart for running *pwpaw* program.

The execution of the program is controlled by a single input file. This input file not only controls the input parameters, but also controls the sequence of calculation steps. Before discussing the details of the keywords and commands used in the input, a few general comments about the input file should be noted.

- Unless otherwise stated all data is entered in Rydberg atomic units.
- Input keywords are not case sensitive.
- A pound, "#", anywhere on a line of the file denotes a comment which extends to the end of that line.
- The input file may access additional files through the use of the "Include" keyword. For example, the appearance of the following line in the input file:

  Include '*filename*'

  has the same effect as inserting the contents of *filename* into the input file. The file *filename* may, in turn, have "Include" files. The current version of *pwpaw* is programed to accept up to 10 nesting levels.
- Characters and numbers are separated by the special delimiter characters:

  , ( ) ␣

  where ␣ means a blank space between characters or numbers. The EOL (end of line character) can also be used as a delimiter.
- Characters and numbers included within single quotes (such as 'single input') are treated as an input unit.
- The program is generally not sensitive to the order of input data, although dimensioning information should generally be listed first so that arrays can be allocated before listing the data associated with those arrays. In addition, the input should be ordered according to the logic of the program. For example, the output files should be specified first so that diagnostic information can be written to those files.

All input data and parameters are associated with predefined keywords which are closely related the program structure. The main keywords used in the program are detailed below.

3.1. Load basic information about the calculation system

3.1.1. Set up output files

The keywords described in Table 1 are associated with the main output data of the program. These are important for monitoring the progress of the program and error information. In fact, the "Open" command can be called multiple times during the program so that the log, error, and output can be written to different files during the course of the calculation. Additional outputs can be specified in Part III of the program.

Table 1
Keywords for output control

Keyword	Data	Default	Description
Print_Level	Character	Normal	Controls amount of output to "log" file. Other recognized character strings are Terse, Commands, and Verbose.
Open log	Character	Screen	This keyword specifies the file name of the diagnostic information output from the program. The default "Screen" means that the output goes to the terminal session.
Open error	Character	Screen	This keyword specifies the file name of the error information output from the program.
Open output	Character	Screen	This keyword specifies the file name of the summary information output from the program.

Table 2
Keywords for main data structures of program

Keyword	Data	Default	Description
Max_AtomTypes	Integer	(none)	The maximum number of different types of atoms that will be used in this calculation. This keyword must come before any atomic data can be loaded.
Max_SpecificAtoms	Integer	(none)	The maximum total number of atoms that will be used in this calculation. This keyword must come before any atomic data can be loaded.
MinPsi	Integer	$1.25 \times N_v$	Calculated default is 25% larger than the total number of valence electrons per unit cell (N_v). This number controls the number of eigenstates per **k**-point that will be calculated. For SCF calculations this number should be larger than $\frac{1}{2}$ the number of valence electrons per unit cell. For density of states or band structure calculations, it should be larger than the total number of bands desired.
Max_TotalPsi	Integer	2×MinPsi	This number should be larger than MinPsi and controls the size of the wavefunction arrays.
Psi_Memory	Real	40	The value represents the number of megabytes available for storing wave function coefficients $A_{n\mathbf{k}}(\mathbf{G})$.
Proj_Memory	Real	40	The value represents the number of megabytes available for storing the projector functions $\tilde{p}_i^a$.
Bloch_Memory	Real	5	The value represents the number of megabytes available for storing phase factors of the form $e^{i\mathbf{G}\cdot\mathbf{R}^a}$.
Ylm_Memory	Real	5	The value represents the number of megabytes available for storing spherical harmonic values $Y_{lm}(\widehat{\mathbf{k}+\mathbf{G}})$ which are needed for calculating projector functions.

3.1.2. Set up data structures

The keywords described in Table 2 must be listed near the beginning of the input file so that data structures can be set up and arrays allocated for additional data input.

3.1.3. Define crystal and set plane-wave cutoffs

The keyword "SuperCell" is associated with the definition of the unit cell lattice parameters, the crystal symmetry, and the Brillouin zone sampling parameters.

SuperCell
  Scale x_s
  A A_x A_y A_z
  B B_x B_y B_z
  C C_x C_y C_z

  Clone_Cell m_A m_B m_C

  Include 'crystal-symmetry.file'
  Include 'k-point-list.file'

BZ_Method GAUSS
Gauss_Width σ

End

Here, the keyword "Scale" is optional and can be used to set a scale factor (x_s) for the lattice vectors. The labels "A", "B", and "C" represent three independent vectors which define the unit cell of the calculation. The values listed after each are the Cartesian components of those vectors in units of Bohr, optionally scaled by the value of x_s. The keyword "Clone_Cell" is optional and can be used to enlarge the unit cell of the calculation so that the lattice vectors of the supercell become $(1 + m_A)\mathbf{A}$, $(1 + m_B)\mathbf{B}$, and $(1 + m_C)\mathbf{C}$. The program was written for $\{m_A, m_B, m_c\}$ being positive integers.

The Include 'crystal-symmetry.file' statement or its contents as described below can be included if the crystal has non-trivial symmetry (that is a space group of order N_{SG} greater than 1). The crystal symmetry is defined by the rotation matrices $\mathcal{R}^i$ and non-primitive translation vectors τ which take a general point within the unit cell $\mathbf{r}$ and map it to a physically equivalent point $\mathbf{r}'$ according to

$$\mathbf{r}' = \mathcal{R}^i\mathbf{r} + \tau^i. \tag{22}$$

The contents of the "crystal-symmetry.file" take the form:

Rot_Size N_{SG}
Matrix i

$$\begin{matrix} \mathcal{R}^i_{xx} & \mathcal{R}^i_{xy} & \mathcal{R}^i_{xz} \\ \mathcal{R}^i_{yx} & \mathcal{R}^i_{yy} & \mathcal{R}^i_{yz} \\ \mathcal{R}^i_{zx} & \mathcal{R}^i_{zy} & \mathcal{R}^i_{zz} \end{matrix}$$

End
Translation i f^i_A f^i_B f^i_C
$\vdots$

Here the index i goes from 1 to N_{SG}, $\mathcal{R}^i_{jk}$ denotes elements of the rotation matrix $\mathcal{R}^i$ in a Cartesian representation, and the translation vector is represented in fractional units of the primitive translation vectors according to $\tau^i \equiv f^i_A\mathbf{A} + f^i_B\mathbf{B} + f^i_C\mathbf{C}$. Alternatively, the inclusion of the keyword 'AUTO_SYMMETRY' calculates the crystal symmetry from the lattice translation and atomic position data. If this keyword is present, the program recalculates the symmetry if the atoms are moved within the run.

The Include 'k-point-list.file' statement or its contents as described below specifies the $\mathbf{k}$-point sampling in the following form.

K-Points_List $N_{\mathbf{k}}$
f_{G_A} f_{G_B} f_{G_C} $W_{\mathbf{k}}$
$\vdots$
End

Here, each of the $N_{\mathbf{k}}$ $\mathbf{k}$-points is specified in fractional units of the primitive reciprocal lattice vectors according to $\mathbf{k} \equiv f_{G_A}\mathbf{G}_A + f_{G_B}\mathbf{G}_B + f_{G_C}\mathbf{G}_C$. The number $W_{\mathbf{k}}$ represents the corresponding relative weighting factor which is normalized to unity within the program. The remaining keywords and constants in this section relate to the method of Brillouin zone integration. In the example, 'BZ_Method GAUSS' means that a Gaussian smoothing function

was used following the approach of Fu and Ho [19], so that the occupancy factor which appears in the calculation of the electron density (Eqs. (4) and (5)) is approximated by

$$f_{n\mathbf{k}} = \frac{W_{\mathbf{k}}}{\sum_{\mathbf{k}'} W_{\mathbf{k}'}} \left\{ 1 + \mathrm{erf}\left(\frac{E_F - E_{n\mathbf{k}}}{\sigma} \right) \right\}, \tag{23}$$

where σ is a smoothing parameter set by the input keyword and constant 'Gauss_Width σ' and E_F is the Fermi level which determined from the total number of valence electrons per unit cell N_v, by solving the transcendental equation [20]

$$\sum_{n\mathbf{k}} f_{n\mathbf{k}} = N_v. \tag{24}$$

Other Brillouin zone schemes could easily be implemented into the code. A short program, *genkpoints*, to read a *pwpaw* input file containing the primitive lattice vectors and symmetry operations and to interactively generate uniformly distributed inequivalent **k**-points (*genkpoints* is included in the package).

The keyword "PlaneWave_Cutoffs" is used to set the range of reciprocal lattice vectors used in the calculation according to the format shown in the following example.

PlaneWave_Cutoffs
  Gcut_LOW *value*
  Gcut_HIGH *value*
  Gcut_PROJ *value*
End

Here, the value of Gcut_LOW determines the truncation of the plane wave expansion of the wavefunction in Eq. (1), and Gcut_HIGH determines the truncation of the plane wave expansion of the density in Eq. (4). The larger cutoff is also used to construct the fast-Fourier-transform grid to perform the evaluation of the pseudo-density efficiently [21]. The keyword 'Gcut_PROJ' is optional. If it is present, the value determines the accuracy of the evaluation of the matrix elements of the smooth wave functions and the projectors $\langle \tilde{p}_i^a | \widetilde{\Psi}_{n\mathbf{k}} \rangle$ if the additional keyword "Real_Space_Projectors" is used (see Table 3).

3.1.4. Load atom information and set electron count

For each atomic species of the calculation, atomic projector and basis functions and associated matrix elements are needed. These are generated by the *atompaw* program in a file [atomic symbol].atomicdata. The format for this file is described in Ref. [8] and can be included in the *pwpaw* input using the format

Include '[atomic symbol].atomicdata'

for each atom.

After the atomic data has been loaded, the electron count is established by specifying the initial configuration in terms of a linear combination of atomic orbital (LCAO) basis. This information is also used to begin the first calculation of a new system. The following keywords are used.

AtomType_Occupancy [atomic symbol]
  Orbitals_Size N_{LCAO}
  Valence_Orbitals $i_1\ i_2\ \ldots i_{N_{\mathrm{LCAO}}}$
  Valence_Occupancy $v_{i_1}\ v_{i_2}\ \ldots v_{i_{N_{\mathrm{LCAO}}}}$

  RS_Scale f_r^a

End

The keyword 'AtomType_Occupancy' takes an atomic symbol argument which must correspond to the atomic symbol defined in the [atomic symbol].atomicdata file. The keyword 'Orbitals_Size' specifies the number of LCAO orbitals which will follow in the specification. The keyword 'Valence_Orbitals' is followed by a list of the i indices corresponding to the $\tilde{\phi}^a_{n_i l_i}(r)$ radial basis functions to be included in the initial configuration. In order to start the calculation with a set of reasonable LCAO functions, it is prudent to include basis functions which correspond to bound atomic states only. The keyword 'Valence_Occupancy' is followed by a list of the initial occupancies ν^a_i of the LCAO bands. Some of the occupancies may be zero, but the sum of all the occupancies and over all atoms should correspond to the total number of electrons per unit cell $\sum_{ai} \nu^a_i = N_v$. The keyword "RS_Scale" is optional. If the "Real_Space_Projectors" keyword is present (see Section 3 below), the cutoff radius for the grid evaluation of $\langle \tilde{p}^a_i | \widetilde{\Psi}_{n\mathbf{k}} \rangle$ is set to be $f^a_r r^a_c$. By default, $f^a_r = 1.5$.

The initial atomic positions can also be specified after the atomic data has been loaded. There are two possible formats for specifying the atomic positions: fractional coordinates or Cartesian coordinates. For the former, the atomic positions are given in terms of the lattice vectors according to $\mathbf{R}^a = f^a_A \mathbf{A} + f^a_B \mathbf{B} + f^a_C \mathbf{C}$ and specified as follows.

```
Atom_List FRAC_POSITION
    [Specific atom label] [Atomic symbol] f_A^a  f_B^a  f_C^a
    ⋮
End
```

Alternatively, the specification in terms of Cartesian coordinates can be given as follows.

```
Atom_List CART_POSITION
    [Specific atom label] [Atomic symbol] R_x^a  R_y^a  R_z^a
    ⋮
End
```

In this case, the atomic position components (R^a_x, R^a_y, R^a_z) are given in Bohr units. In either case, the atomic positions must be consistent with the symmetry of the crystal in the sense that for each atomic position $\mathbf{R}^a$ and each space group element $(\mathcal{R}^i, \tau^i)$,

$$\mathbf{R}^b = \mathcal{R}^i \mathbf{R}^a + \tau^i, \tag{25}$$

where $\mathbf{R}^b$ corresponds to the position of the same or an equivalent atom in the unit cell. The program checks that (25) is satisfied within a tolerance of 10^{-6} Bohr units. If the keyword "Clone_Cell" is set as described above, then additional atomic positions are generated and each is given a "Specific atom label" of the form [Specific atom label]_uvw, where $0 \leqslant u \leqslant m_A$, $0 \leqslant v \leqslant m_B$, and $0 \leqslant w \leqslant m_C$.

3.1.5. Set calculation options

Table 3 lists some of the keywords which are used to control the calculations and execution parameters for the program.

The parameters which control the SCF cycle are included in Table 4. Following the suggestions of Eyert [15], we use the convergence acceleration algorithm of Anderson [16] which uses a mixing parameter to control the stability of the iterations. In order to minimize the possible effects of charge slashing [11], two mixing parameters are given. When the change in the cohesive energy for successive iterations, is small, the first parameter is used. When the change is larger than the value specified with the "Mix_SecondValue" keyword, the second mixing parameter is used. Finally, the keyword "Filter_Potential" and parameter "V_Smooth_Width" implements the algorithm of Wang and Zunger [22] for reducing large Fourier components to the effective potential $\tilde{v}_{\text{eff}}$.

Table 3
Keywords for setting calculation options

Keyword	Data	Default	Description
XC_Type	Character	XC_Perdew	This keyword controls the functional form of the exchange-correlation functional. The currently implemented case is are XC_Perdew [10] which is consistent with exchange-correlation functional form used in the atom program *atompaw*.
Forces_always_calc_H	(none)	Not present	The presence of this keyword ensures that the D^a_{ij} matrix elements are recalculated when the forces are evaluated. If the keyword is not present in the input file, the stored matrix elements are used for evaluating the forces.
Anchor	Character	Not present	The presence of this keyword sets the specific atom indicated with the character data value acts as an anchor to the system in the geometry optimization or in molecular dynamics.
Freeze	Character	Not present	The presence of this keyword sets a calculational flag to ensure that the specific atom indicated with the character data value will not move in the geometry optimization or in molecular dynamics. Unlike the "Anchor" keyword, more than one atom can be fixed in the calculation with the "Freeze" keyword.
Real_Space_Projectors	(none)	Not present	The presence of this keyword specifies that the evaluations of the matrix elements $\langle \tilde{p}^a_j \vert \tilde{\Psi}_{n\mathbf{k}} \rangle$ will be evaluated on a real space grid. The associated keyword "Gcut_Proj" determines the grid spacing from the associated fast-Fourier-transform and the keyword "RS_Scale" determines the effective integration radius. This form evaluates the matrix elements much faster than the Fourier space sum with reasonable accuracy for large unit cells.
No_O_Eigenvalues	(none)	Not present	The presence of this keyword means that the generalized eigenvalue problem is solved by factorizing the overlap matrix, rather than the default algorithm which diagonalizes the overlap matrix. If this procedure returns an error code, the program uses the default algorithm.
Overlap_Tol	Real	10^{-11}	This parameter controls the minimum eigenvalue of the overlap matrix which is used to solve the generalized eigenvalue problem.

3.2. Initialize calculation

The keyword 'Initialize_System' is used to call a series of subroutines which use the loaded data to prepare the data structures which will be used in the program. This initialization step is then followed by the preparation of the initial smooth wave functions by one of the following three possibilities. The default is a call to the subroutine CalcLCAO, which is the default and needs no additional keywords. Alternatively, the results of a previous run of the program can be loaded by using the following form.

Load_Solution *RestartFilename*

The binary file, '*RestartFilename*', generated by the command 'Store_Solution binary' described below in a previous run of the program which may have had different plane wave cutoff parameters or different numbers of wave functions. Alternatively, the keyword 'Random_Guess' can be used to use a random number generator to initialize the smooth wave functions.

Table 4
Keywords for setting parameters which control the convergence of the SCF cycle

Keyword	Data	Default	Description
Mix_V, Mix_Veff, or Mix_Density	(none)	Mix_Veff	This keyword controls how the SCF cycles are updated using the Anderson mixing algorithm discussed above using the Coulomb potential corresponding to $\tilde{n}+\hat{n}$, using v_{eff}, or using $\tilde{n}$, respectively.
V_NewMix	Real Real	0.5 0.25	Determines the initial value of the Anderson mixing procedure for the potential or density updating mode in the SCF cycle. If these values are greater or equal to 0.9999, no mixing is done.
Dij_NewMix	Real Real	1.0 1.0	Determines the initial value of the Anderson mixing procedure for updating the D^a_{ij} matrix elements in the SCF cycle. If these values are greater or equal to 0.9999, no mixing is done.
Mix_SecondValue	Real	0.2	This parameter determines when to switch Anderson mixing parameters from the first (fractional change in cohesive energy is less than parameter value) to the second.
Mix_Size	Integer	5	The parameter determines the maximum number of update data used in the Anderson mixing algorithm.
Filter_Potential	(none)	Not present	This option removes large Fourier components of the potential $\tilde{v}_{\text{eff}}$.
V_Smooth_Width	Real	0.0	If the Filter_Potential option has been chosen, this parameter controls the rate of smoothing of the potential $\tilde{v}_{\text{eff}}$ in Fourier space, following the approach of Wang and Zunger [22]. The value of this parameter corresponds to $1-\beta$ defined in Appendix A of the reference.

3.3. Keyword command control of program

3.3.1. Self-consistent electronic structure

For self-consistent field (SCF) calculations at fixed atomic positions, the keyword command structure is as follows.

Relax Charge *SCFIter MeritTol*

In this case, *SCFIter* denotes the maximum number of iteration steps, described in Section 2.4, that are carried out before the program stops. The parameter *MeritTol* is also defined in Section 2.4.

3.3.2. Store solutions

In order to store the wave functions for later use by the program, the following keyword command can be used.

$$\text{Store\_solution} \begin{cases} \text{binary} \\ \text{text} \end{cases} \quad \textit{RestartFilename}$$

The file *RestartFilename* contains information about the reciprocal lattice cutoffs and the **k**-point sampling in addition to $E_{n\mathbf{k}}$, $f_{n\mathbf{k}}$, and $\{A_{n\mathbf{k}}(\mathbf{G})\}$ for each calculated state. The "binary" form means that the file is stored with unformatted output. In order to transfer data to another operating system, the "text" (formatted) form can be used instead.

For calculating the band structure, the self-consistent local and non-local potential terms $\tilde{v}_{\text{eff}}$ and D^a_{ij} are needed. These can be saved or retrieved by using the keywords

Store_Hamiltonian

or

Load_Hamiltonian

respectively.

3.3.3. Calculate forces or energy

In order to calculate the forces on atoms after an SCF calculation, the following keyword command can be used.

Calculate Forces *ForceOutput*

Here, the parameter *ForceOutput* resets the file name for the output of atomic position and force information. The default name is "paw.forces". The program is written to append the force output to the end of any existing output file. The forces reported are those calculated from Eq. (18).

In order to calculate the cohesive energy and output the result to the "Output_Unit", the command "Calculate Energy" can be used.

3.3.4. Geometry optimization

For calculations which optimize the cohesive energy with respect to the atomic positions on the Born–Oppenheimer surface, the keyword command structure is as follows.

Relax Geometry *MaxSteps Output SCFIter MeritTol ForceTol MaxMove*

In this mode, the program performs an SCF calculation at "MaxSteps" different geometries, or less. For each SCF calculation, as in the "Relax Charge" mode described above, a maximum of "*SCFIter*" iterations are allowed to achieve the convergence of the cohesive energy within the allowance of the "*MeritTol*" parameter. At the end of each SCF calculation, the forces on each atom are calculated according to Eq. (18). If $\sqrt{\sum_a |\mathbf{F}^a|^2}$ is larger than "*ForceTol*", the atoms are moved to new positions in the direction of $\mathbf{F}^a$. The parameter *MaxMove* controls the magnitude of the move. Ideally, this magnitude should be small enough so that the SCF iteration in the new geometry can use the wavefunctions of the previous geometry for a rapidly convergent calculation. The parameter "*Output*" can either list a file name for saving wavefunction information at each geometry step or can be set to "NULL" indicating that the wavefunction is not to be saved. After each force calculation, the position and force information is appended to the force output file which can be renamed by the "Calculate Forces *ForceOutput*" command as explained above.

3.3.5. Calculation of the density of states, partial density of states, electron densities, or band structure

The densities of states or electron densities for selected states can be calculated after any SCF step. The program is used in a 'bandstructure' mode which takes converged results for $\tilde{v}_{\text{eff}}$ and D^a_{ij} and performs iterative diagonalization to solve the generalized eigenvalue equations (12) using the same block Davidson algorithm discussed in Section 2.4. While for the SCF calculation, only occupied states affect the convergence test, for the 'bandstructure' mode of the program, all states with Kohn–Sham eigenvalues $E_{n\mathbf{k}} \leqslant \mathit{EigenMax}$ are included in the definition of the merit function:

$$E^{\alpha}_{\text{merit}} \equiv \sum_{E_{n\mathbf{k}} \leqslant \mathit{EigenMax}} |E_{n\mathbf{k}}|. \tag{26}$$

Therefore, the band merit function is defined to be

$$\mathit{BandMerit} \equiv \left| E^{\alpha+1}_{\text{merit}} - E^{\alpha}_{\text{merit}} \right|. \tag{27}$$

The value of *EigenMax* can be set with a command of the following form.

Set_EigenMax *EigenMax*

In conjunction with setting the value of *EigenMax*, it is necessary to make sure that the data structures for the wavefunctions are dimensioned large enough to accommodate the additional states. This is controlled in the first section of the program, as described in Section 3.1.2 through the 'MinPsi' and 'Max_TotalPsi' parameters.

The commands for calculating the density of states are given by the following.

Calculate DOS *DOSFilename DOSIter DOSTol*

Here, *DOSFilename* specifies the output file name for the density of states information, *DOSIter* determines how many iterative diagonalization steps are allowed, and *DOSTol* denotes the maximum value of *BandMerit* at convergence. The commands for calculating the partial density of states are given by the following.

Calculate PDOS *PDOSFilename P PDOSIter PDOSTol*
ATOM [Specific atom label] S
⋮
label $f_A\ f_B\ f_C$ S
⋮
} P sites
End

In the PDOS mode of the program, the charge $C^p_{n\mathbf{k}}$ associated with each eigenstate $n\mathbf{k}$ of the system within each of P spheres is written out to the file *PDOSFilename*. The number of iterations *PDOSIter* and the convergence tolerance *PDOSTol* are exactly analogous to the DOS case. For each of the P spheres, a radius S is specified. The keyword ATOM indicates that this sphere is associated with an atomic site with the given [Specific atom label]. In this case, $S \geqslant r_c^a$. Any other label can be used to specify a sphere centered at the location $f_A\mathbf{A} + f_B\mathbf{B} + f_C\mathbf{C}$.

The *DOSFilename* and *PDOSFilename* can be processed by the short interactive program *preparepdos* to generate the density of states from the Gaussian smearing function [19]. The partial density of states associated with the pth sphere can be calculated from:

$$N^p(E) = \frac{2}{\sqrt{\pi}\sigma\left(\sum_{\mathbf{k}'} W_{\mathbf{k}'}\right)} \sum_{n\mathbf{k}} C^p_{n\mathbf{k}}\, W_{\mathbf{k}}\, \mathrm{e}^{-(E-E_{n\mathbf{k}})^2/\sigma^2}, \tag{28}$$

where $W_{\mathbf{k}}$ denotes the $\mathbf{k}$-point weight factor defined earlier. In this expression, the δ-function in energy for evaluating the density of states has been replaced by a Gaussian function of width σ which need not be the same as that used (Eq. (23)) for the SCF calculations. [2] The same form can be used for the density of states calculations by setting $C^p_{n\mathbf{k}} \equiv 1$.

The commands for calculating electron densities corresponding to N_d selected energy ranges is given by the following.

Calculate Partial_Densities N_d *PDIter PDtol*
OccFlag PDFilename $E_{\min}$ $E_{\max}$
⋮
} N_d partial densities
End

[2] An advantage of this approach is that it avoids numerical spikes in the evaluation of the density of states and therefore facilitates comparison, as shown, for example, in Ref. [5] in the comparison of the density of states for $CaMoO_4$ calculated using the PAW and LAPW methods.

Here *PDIter* and *PDtol* represent the number of iterative diagonalization steps and convergence tolerance, similar to those variables in the density of states calculations. The parameter *OccFlag* can either be "OCC", meaning that the occupancy factors $f_{n\mathbf{k}}$ calculated in the SCF step from Eq. (23) should be used, or "NOT", meaning that the occupancy should be calculated from the Brillouin zone weight factors, alone, assuming that the bands are fully occupied. That is, calculating $f_{n\mathbf{k}}$ by using Eq. (23) with $E_F \to \infty$. For each energy range, where $E_{\min} \leqslant E_{n\mathbf{k}} \leqslant E_{\max}$, the partial density $n^d(\mathbf{r})$ can be constructed from a knowledge of the Fourier coefficients of the partial pseudo-density

$$\tilde{n}^d(\mathbf{r}) = \sum_{E_{\min} \leqslant E_{n\mathbf{k}} \leqslant E_{\max}} f_{n\mathbf{k}} \left|\widetilde{\Psi}_{n\mathbf{k}}(\mathbf{r})\right|^2 = \frac{1}{\mathcal{V}} \sum_{\mathbf{G}} \bar{\tilde{n}}^d(\mathbf{G}) \mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}}, \tag{29}$$

and the partial projected occupation coefficients

$$W_{ij}^{ad} \equiv \sum_{E_{\min} \leqslant E_{n\mathbf{k}} \leqslant E_{\max}} f_{n\mathbf{k}} \langle \widetilde{\Psi}_{n\mathbf{k}} | \tilde{p}_i^a \rangle \langle \tilde{p}_j^a | \widetilde{\Psi}_{n\mathbf{k}} \rangle. \tag{30}$$

The Fourier coefficients $\bar{\tilde{n}}^d(\mathbf{G})$ and partial projected occupation coefficients W_{ij}^{ad} are calculated and written to the file named *PDFilename*.

In order to construct a bandstructure plot, it is necessary to determine the energy eigenvalues $E_{n\mathbf{k}}$ for a set of wave vectors that are generally different from those used during the SCF step. After an SCF calculation, a Load_Solution keyword, or a Load_Hamiltonian keyword, the following command can be used:

Calculate Band_Structure *Bandfilename BandIter BandTol*

Here *Bandfilename* denotes the name of an input file which contains the list of **k**-points. The band structure output is written to a file named "*Bandfilename*.band". The number of iterations allowed for eigenvalue solver is specified by the integer *BandIter* and *BandTol* specifies the convergence tolerance of the merit function (27). The format of the *Bandfilename* input file is as follows.

Maxbands
f_{G_A} f_{G_B} f_{G_C}
$\vdots$

Here *Maxbands* specifies the maximum number of bands that are expected for any of the **k**-points. This parameter supersedes the value of *Max_TotalPsi* and it should be consistent with the choice of *EigenMax*. The **k**-points are listed in fractional units of the primitive reciprocal lattice vectors. The program processes each new **k**-point until reaching the end of the file. The output file lists **k** and $E_{n\mathbf{k}}$ for each band. A simple program *bandplot* can then be used to convert the fractional wave vectors into a a scalar length along specified directions in the Brillouin Zone for constructing a band diagram.

3.3.6. Contour plots of the electron density

The program is constructed to prepare data for use with IBM's Data Explorer software. The output can be easily modified for use with other plotting software. Both 2- and 3-dimensional plots can be constructed for each of the partial density files *PDFilename*. The keyword commands are designed to first setup the parameters of the plotting area or volume. For the 3-dimensional plots, the setup parameters are given in the following form.

Plot 3DSetup
X X_x X_y X_z
Y Y_x Y_y Y_z
Z Z_x Z_y Z_z

```
    O    Ox Oy Oz
    Grid    Nx Ny Nz
    PlotName    PlotFilename
    Bond    b
    BondTol    u
End
```

Here the keywords "X", "Y", "Z", and "O" are each followed by 3 Cartesian coordinates in Bohr units, which specify the 3 orthogonal vectors **X**–**O**, **Y**–**O**, and **Z**–**O** which define the plotting volume. The keyword "Grid" specifies the uniform grid on which density $n(\mathbf{r})$ will be evaluated within the plotting volume. The keyword "PlotName" specifies the prefix of the output plotting files used with the "Plot 3datom" keyword described below. The keywords "Bond" and "BondTol" are optional keywords which specify stick model bonds that can be drawn. Here, b specifies the largest distance between two atoms for which a bond will be drawn. The parameter u specifies the fractional bond length to plotting cell length ratio which allows atoms outside the plotting cell to be included in the ball and stick plot.

For 2-dimensional plots, the setup parameters are given in a similar form.

```
Plot 2DSetup
    X    Xx Xy Xz
    Y    Yx Yy Yz
    O    Ox Oy Oz
    Grid    Nx Ny
    PlotName    PlotFilename
End
```

Once these parameters have been set up, the plots can be constructed using

```
Plot 3d-density PDFilename
```

or

```
Plot 2d-density PDFilename
```

for the 3- or 2-dimensional plots, respectively. The 3-dimensional case additionally outputs information to construct a ball and stick model for each plot. To output only the ball and stick model information, the following keyword command can be used.

```
Plot 3datom
```

Output is also generated in a format which can be viewed using the program XCrySDen [23].

4. Sample programs

4.1. CaO

This example was discussed in Ref. [8]. The input file to calculate the SCF cycle for this material structure at the lattice constant $a = 4.7$ Å for the projector and basis set we labeled **"sinc",rc=1.4(x2)** is given as follows.

```
#
# Input file for CaO at lattice constant = 4.7 A
#
Print_Level VERBOSE
open Log    'CaO1.log'        # Set output files
Open Error  'CaO1.error'
Open output 'CaO1.out'

Max_AtomTypes 2
MAx_Specific_Atoms 2
Max_TotalPsi 18
MinPsi 16

Psi_Memory 200
Proj_Memory 100
Ylm_Memory 100
Bloch_Memory 100

SuperCell             # Define the crystal
  Scale 1.889725989
  A ( 2.3500000,  2.3500000,  0.0000000)
  B ( 2.3500000,  0.0000000,  2.3500000)
  C ( 0.0000000,  2.3500000,  2.3500000)

  Include 'CaO.crystal-symmetry'
  Include 'CaO.k-point-list'
  Gauss_Width  0.001
  BZ_Method   GAUSS
End

PlaneWave_Cutoffs
 Gcut_LOW 10
 Gcut_HIGH 12
End

Include '../atom/O/sinc4rc1.4/O.atomicdata'
Include '../atom/Ca/Ca.atomicdata'

AtomType_Occupancy O
   Orbitals_Size 2
   Valence_Orbitals 1 3
   Valence_occupancy 2 4
End

AtomType_Occupancy Ca
   Orbitals_Size 3
   Valence_Orbitals 1 2 3
   Valence_occupancy 2 2 6
End

Atom_List Frac_Position
  O   O  (0.5,0.5,0.5)
```

```
  Ca  Ca (0.0,0.0,0.0)
End

XC_Type Perdew-Wang
FORCES_ALWAYS_CALC_H
Mix_Veff
V_NewMix 0.2 .1
Dij_NewMix 0.2  .1
Mix_SecondValue .2

Initialize_System
Load_Solution  'CaO.pwfn1a'

Relax charge 30 1.E-7
Store_Solution binary 'CaO.pwfn1'
Quit
```

In this example, the "Include" files are in different directories, and are specified using the Unix directory conventions. The list of **k**-points file has the form:

```
K-Points_List 10
  0.1250000000000000  0.1250000000000000  0.1250000000000000    2.00000
  0.3750000000000000  0.1250000000000000  0.1250000000000000    6.00000
 -0.3750000000000000  0.1250000000000000  0.1250000000000000    6.00000
 -0.1250000000000000  0.1250000000000000  0.1250000000000000    6.00000
  0.3750000000000000  0.3750000000000000  0.1250000000000000    6.00000
 -0.3750000000000000  0.3750000000000000  0.1250000000000000   12.00000
 -0.1250000000000000  0.3750000000000000  0.1250000000000000   12.00000
 -0.3750000000000000 -0.3750000000000000  0.1250000000000000    6.00000
  0.3750000000000000  0.3750000000000000  0.3750000000000000    2.00000
 -0.3750000000000000  0.3750000000000000  0.3750000000000000    6.00000
End
```

The calculation is restarted with the results of a previous run at smaller plane wave cutoffs which was stored in the file 'CaO.pwfn1a'. The results of the calculation are shown in Fig. 2. In this case, the restarted calculation converged in 8 iterations.

```
 #  SCF results generated on date 05/16/2000, 10:20:57.375
 RelaxElectrons: Converged in 8 iterations with a final Energy error of
                                                                  0.4873608361322112 61E-07
 RelaxElectrons: Cohesive Energy: 1.05488501097457288
 RelaxElectrons: # iterations   : 8 / 50
 RelaxElectrons: Tolerance      : 0.48736083613221261E-07 / 0.999999999999999955E-07
 # Current Fractional Positions
 #Atom_List FRAC_POSITION
 # O           O           5.00000E-01  5.00000E-01  5.00000E-01
 # Ca          Ca          0.00000E+00  0.00000E+00  0.00000E+00
 Energies for cluster 1  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 10  * -2.36959106909795603  * 0.6250000000000000000E-01  * 1
 Energies for cluster 2  (DiskRec, Energy, Occ, Kpnt) * Size: 3
 11  * -1.02263048301959358  * 0.6250000000000000000E-01  * 1
 12  * -1.00683529157566687  * 0.6250000000000000000E-01  * 1
 13  * -1.00683529146213879  * 0.6250000000000000000E-01  * 1
```

Fig. 2. Output file for CaO example.

```
 Energies for cluster 3  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 14  * -0.773623079657166146  * 0.625000000000000000E-01  * 1
 Energies for cluster 4  (DiskRec, Energy, Occ, Kpnt) * Size: 3
 15  * 0.283676485941666723  * 0.625000000000000000E-01  * 1
 16  * 0.307602676693226107  * 0.625000000000000000E-01  * 1
 17  * 0.307602677770904331  * 0.625000000000000000E-01  * 1
 Energies for cluster 5  (DiskRec, Energy, Occ, Kpnt) * Size: 4
 18  * 0.717838616819526432  * 0.000000000000000000E+00  * 1
 19  * 0.767500661111773175  * 0.000000000000000000E+00  * 1
 20  * 0.790069906475341477  * 0.000000000000000000E+00  * 1
 21  * 0.790069941281033139  * 0.000000000000000000E+00  * 1
 Energies for cluster 6  (DiskRec, Energy, Occ, Kpnt) * Size: 2
 22  * 0.911792965109302744  * 0.000000000000000000E+00  * 1
 23  * 0.911793031667607501  * 0.000000000000000000E+00  * 1
 Energies for cluster 7  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 24  * 1.45523265404801361  * 0.000000000000000000E+00  * 1
 Energies for cluster 8  (DiskRec, Energy, Occ, Kpnt) * Size: 3
 25  * 1.89509402235662949  * 0.000000000000000000E+00  * 1
 26  * 1.92648052021326599  * 0.000000000000000000E+00  * 1
 27  * 1.92648060884543959  * 0.000000000000000000E+00  * 1
 Energies for cluster 9  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 28  * -2.36466364750619773  * 0.187500000000000000  * 2
 Energies for cluster 10  (DiskRec, Energy, Occ, Kpnt) * Size: 3
 29  * -1.06484636812383981  * 0.187500000000000000  * 2
 30  * -1.01595534858570451  * 0.187500000000000000  * 2
 31  * -1.00799009742767831  * 0.187500000000000000  * 2
 Energies for cluster 11  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 32  * -0.727530323924740041  * 0.187500000000000000  * 2
 Energies for cluster 12  (DiskRec, Energy, Occ, Kpnt) * Size: 3
 33  * 0.181914652254444420  * 0.187500000000000000  * 2
 34  * 0.251387221971676345  * 0.187500000000000000  * 2
 35  * 0.274027709194861913  * 0.187500000000000000  * 2
 Energies for cluster 13  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 36  * 0.768292789192570935  * 0.000000000000000000E+00  * 2
...
 180  * 1.35040430531786848  * 0.000000000000000000E+00  * 10
 181  * 1.41834909986961488  * 0.000000000000000000E+00  * 10
 Energies for cluster 107  (DiskRec, Energy, Occ, Kpnt) * Size: 1
 182  * 1.64971133783488244  * 0.000000000000000000E+00  * 10
 Exiting PAW program
```

Fig. 2. Continued.

4.2. Diamond

This example shows a SCF calculation for diamond, followed by a calculation of the band structure along the X–Γ–L directions. The input file is given as follows.

```
#
#  Input file for diamond
#
open Log    'diamond.log'        # Set output files
Open Error  'diamond.error'
Open output 'diamond.out'

Max_AtomTypes 1
MAx_Specific_Atoms 2
```

```
Max_TotalPsi 20
MinPsi 4
Psi_Memory 200
Proj_Memory 100
Ylm_Memory 100
Bloch_Memory 100

SuperCell             # Define the crystal
  A (3.35237480000000000, 3.35237480000000000, 0.00000000000000000)
  B (0.00000000000000000, 3.35237480000000000, 3.35237480000000000)
  C (3.35237480000000000, 0.00000000000000000, 3.35237480000000000)

 Include 'diamond.k-point-list'
 Include 'diamond.crystal-symmetry'

  Gauss_Width  0.001
  BZ_Method   GAUSS
End

PlaneWave_Cutoffs
  Gcut_LOW   8
  Gcut_HIGH  10
End

Include '../../atom/C/C.atomicdata'

AtomType_Occupancy C
   Orbitals_Size 2
   Valence_Orbitals 1 2
   Valence_occupancy 1 3
End

Atom_List Frac_Position
   C1 C          ( 0.125, 0.125, 0.125)
   C2 C          (-0.125,-0.125,-0.125)
End

XC_Type Perdew-Wang
FORCES_ALWAYS_CALC_H
Mix_Veff
V_NewMix 1.0  0.2
Dij_NewMix 1.0 0.2
Mix_SecondValue 0.2

Initialize_System

Relax charge 30   1.0E-6
Store_Solution binary 'diamond.wfn1.binary'

Set_Eigenmax 4
Calculate Band_structure diamondXL 40 1.E-5
Quit
```

```
#
# Input file for Ti2Nb6O12 using structural data of  Katya Anokhina
#
open Log    'paw.log'        # Set output files
Open Error  'paw.error'
Open output 'paw.out'

Max_AtomTypes 3
MAx_Specific_Atoms 20

Max_TotalPsi 120
MinPsi 115

Psi_Memory 200
Proj_Memory 100
Ylm_Memory 100
Bloch_Memory 100

SuperCell             # Define the crystal
  A (8.658880719, 0, 9.092038650)
  B (-4.329440361, 7.498810670, 9.092038650)
  C (-4.329440361, -7.498810670, 9.092038650)

  Include 'tinbo.k-point-list'
  Include 'tinbo.crystal-symmetry'

  Gauss_Width  0.001
  BZ_Method   GAUSS
End

PlaneWave_Cutoffs
  Gcut_LOW 8
  Gcut_HIGH 10
End

Include 'O.atomicdata'
Include 'Nb.atomicdata'
Include 'Ti.atomicdata'

AtomType_Occupancy O
   Orbitals_Size 2
   Valence_Orbitals 1 3
   Valence_occupancy 2 4
End
AtomType_Occupancy Nb
   Orbitals_Size 4
   Valence_Orbitals 1 2 3   5
   Valence_occupancy 2 1 6  4
End
AtomType_Occupancy Ti
   Orbitals_Size 4
   Valence_Orbitals 1 2 3  5
   Valence_occupancy 2 2 6  2
End
```

Fig. 3. Input file for $Ti_2Nb_6O_{12}$ SCF calculation.

```
Atom_List Cart_Position
01  O   .3571788298, -2.899790086, 4.726950894
02  O   -.3571788298, 2.899790086, -4.726950894
03  O   2.332702467, 1.759220983, 4.726950894
04  O   -2.332702467, -1.759220983, -4.726950894
05  O   -2.689881296, 1.140569103, 4.726950894
06  O   2.689881296, -1.140569103, -4.726950894
07  O   -3.339297351, 4.512034380, -.01636566957
08  O   3.339297351, -4.512034380, .01636566957
09  O   -2.237887723, -5.147933525, -.01636566957
010 O   2.237887723, 5.147933525, .01636566957
011 O   5.577185072, .6358991448, -.01636566957
012 O   -5.577185072, -.6358991448, .01636566957
Nb1 Nb  2.823661003, -1.303293294, 2.127537044
Nb2 Nb  -2.823661003, 1.303293294, -2.127537044
Nb3 Nb  -.2831453995, 3.097008807, 2.127537044
Nb4 Nb  .2831453995, -3.097008807, -2.127537044
Nb5 Nb  -2.540515603, -1.793715512, 2.127537044
Nb6 Nb  2.540515603, 1.793715512, -2.127537044
Ti1 Ti  0, 0, -7.474564973
Ti2 Ti  0, 0, 7.474564973
End

XC_Type Perdew-Wang
FORCES_ALWAYS_CALC_H
Mix_Veff
V_NewMix 0.20 .1
Dij_NewMix 1.0  0.2
Mix_SecondValue 0.2

Initialize_System

Relax charge 80  1.0E-7
Calculate Forces   'tinbo.force1'
Store_Solution binary 'tinbo.pwfn1'

Quit
```

Fig. 3. Continued.

In this case, the wavefunctions generated during the SCF step are used to generate an initial guess for the first **k**-point of the band calculation. The **k**-point file, named "diamondXL" has the following contents:

```
    20          # maximum number of bands per k-point
0.5  0.5  0     # X point
0.4  0.4  0
0.3  0.3  0
0.2  0.2  0
0.1  0.1  0
0    0    0     # Gamma point
0.1  0.1  0.1
0.2  0.2  0.2
0.3  0.3  0.3
0.4  0.4  0.4
0.5  0.5  0.5   # L point
```

```
K-Points_List 6
  0.1666666666666667  0.1666666666666667  0.1666666666666667    2.00000
  0.4999999999999999  0.1666666666666667  0.1666666666666667    6.00000
 -0.1666666666666667  0.1666666666666667  0.1666666666666667    6.00000
  0.4999999999999999  0.4999999999999999  0.1666666666666667    6.00000
 -0.1666666666666667  0.4999999999999999  0.1666666666666667    6.00000
  0.4999999999999999  0.4999999999999999  0.4999999999999999    1.00000
End
```

Fig. 4. **k**-points file for $Ti_2Nb_6O_{12}$.

```
Rot_Size   6

Matrix   1
   1   0   0
   0   1   0
   0   0   1
 End
Translation   1  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00

Matrix   2
  -1   0   0
   0  -1   0
   0   0  -1
 End
Translation   2  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00

Matrix   3
  -0.5                    .86602540378443864675  0
  -.86602540378443864675  -0.5                   0
   0   0   1
 End
Translation   3  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00

Matrix   4
  -0.5                   -0.86602540378443864675  0
   .86602540378443864675  -0.5                   0
   0   0   1
 End
Translation   4  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00

Matrix   5
   0.5                   -0.86602540378443864675  0
   .86602540378443864675   0.5                   0
   0   0  -1
 End
Translation   5  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00

Matrix   6
   0.5                    0.86602540378443864675  0
  -.86602540378443864675   0.5                   0
   0   0  -1
 End
Translation   6  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
```

Fig. 5. Symmetry file for $Ti_2Nb_6O_{12}$.

```
   ⋮
Max_TotalPsi 220
MinPsi 215
   ⋮
Initialize_System
Load_Solution 'tinbo.pwfn1'

Set_EigenMax 2.0
Calculate PDOS 'tinbo.pdos' 20 40 1.e-6
 ATOM O1  1.51
 ATOM O2  1.51
 ATOM O3  1.51
 ATOM O4  1.51
 ATOM O5  1.51
 ATOM O6  1.51
 ATOM O7  1.51
 ATOM O8  1.51
 ATOM O9  1.51
 ATOM O10 1.51
 ATOM O11 1.51
 ATOM O12 1.51
 ATOM Nb1 2.21
 ATOM Nb2 2.21
 ATOM Nb3 2.21
 ATOM Nb4 2.21
 ATOM Nb5 2.21
 ATOM Nb6 2.21
 ATOM Ti1 2.21
 ATOM Ti2 2.21
End

Store_Solutions 'tinbo.dos.pwfn1'
Quit
```

Fig. 6. Modifications to input file (Fig. 3) needed for calculating partial densities of states if $Ti_2Nb_6O_{12}$.

4.3. $Ti_2Nb_6O_{12}$

A more strenuous demonstration of the program was suggested by colleagues from the Chemistry department. The compound $Ti_2Nb_6O_{12}$ is one of a series of transition metal oxides which are under investigation for their structural properties [24]. It crystallizes in a rhombohedral structure with one formula unit per unit cell. The input file, the **k**-points file, and crystal symmetry files are shown in Figs. 3, 4, and 5, respectively. In this case, the symmetry group has a total of 6 operations including inversion and a rotation of $\pm 120^\circ$ about the z-axis. The atom centered basis and projector functions were calculated using the *atompaw* code [8] using the "sinc" shape function with the basis choice and r_c^a values listed in Table 5. This material has two transition metals with very similar electronegativities, so that the charge transfer between the two is delicately balanced. After the calculation converged, the partial densities of states where calculated using the modified input file shown in Fig. 6. After processing the output (which is in this case written to a file named "tinbo.pdos.pdosout") using the interactive program *preparepdos*, the resultant partial density of states are obtained as shown in Fig. 6. In this case, the results for the individual atoms of each type were averaged to form 3 partial density of states curves — for Ti, Nb, and O. Here we see that this material is a semiconductor with a small band gap. The O states are completely filled and the

Table 5
Atomic basis parameters used for $Ti_2Nb_6O_{12}$ calculations

Atom	$\{n_i l_i\}$	r_c^a (Bohr)
Ti	3s, 4s, 3p, εp, 3d, εd	1.4
Nb	$4s$, $5s$, $4p$, εp, $4d$, εd	1.6
O	$2s$, εs, $2p$, εp	1.5

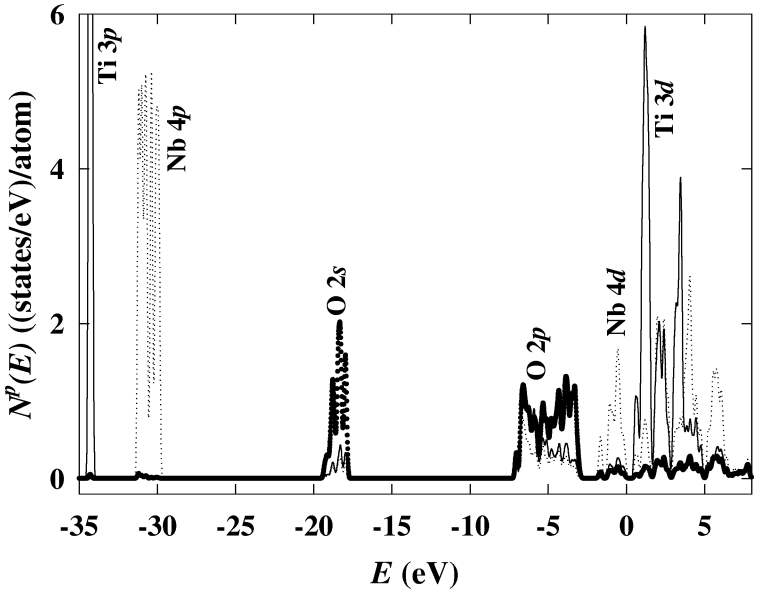

Fig. 7. Plot of partial densities of states for $Ti_2Nb_6O_{12}$. Labels indicate the dominant atomic character associated with nearby peaks. The zero of energy is adjusted to correspond with the last occupied state.

states near the band gap are associated with the d-states of the two transition metals. The top of the valence band has mainly Nd character, while the lowest part of the conduction band has mainly Ti character, according to these results.

In order to see the bonding more clearly, we can construct contour plots of the partial charge densities. The modification to the input file to files with partial charge densities for use with plotting software is shown in Fig. 8. In this case, the wave function file "tinbo.dos.pwfn1" which was generated while calculating the density of states is loaded to speed up the calculations. The example shows the calculation of the "partial_densities" corresponding to 8 different ranges of energy. After these partial densities are calculated, the example shows the calling sequence for the 3-dimensional plotting routines. The plotting could be also done in a separate program run, using the same "partial_densities" files. Fig. 9 shows a 3-dimensional contour plot resulting from one of these calculations obtained by using IBM's Data Explorer software. The example (using the "partial_densities" output file "tinbob") corresponds to the states within a range of 1.5 eV of the top of the valence band and shows a high concentration of charge on the Nb sites, as is consistent with the density of states results.

```
   ⋮
Max_TotalPsi 220
MinPsi 215
   ⋮
Initialize_System
Load_Solution 'tinbo.dos.pwfn1'

Calculate partial_densities 8 40 1.e-7
  OCC tinboa   0.36  0.42  # 2 states
  OCC tinbob   0.42  0.53  # 12 states
  NOT tinboc   0.53  0.584 # 2 states
  NOT tinbod   0.584 0.645 # 10 states
  NOT tinboe   0.645 0.73  # 18 states
  NOT tinbof   0.73  0.79  # 8 states
  NOT tinbog   0.79  0.89  # 17 states
  NOT tinboh   0.89  1.01  # 14 states
End

Plot 3dsetup
 X  17.3177610   0.0000000   0.00000000
 Y   0.0000000  14.9976210   0.00000000
 Z   0.0000000   0.0000000  27.27611600
 O  -8.6588805  -7.4988105 -13.63805800
 Bond   4.4
 Bondtol 0.001
 Grid  36 31 56
End

Plot 3d-density tinboa
Plot 3d-density tinbob
Plot 3d-density tinboc
Plot 3d-density tinbod
Plot 3d-density tinboe
Plot 3d-density tinbof
Plot 3d-density tinbog
Plot 3d-density tinboh

Quit
```

Fig. 8. Modifications to input file (Fig. 3) needed for evaluating electron densities for various energy ranges and for preparing files for contour plots.

5. Future work

In future modifications to the program, we hope to implement additional exchange-correlation functional forms and to introduce the capability of treating relativistic effects.

Acknowledgements

This work was supported by NSF grants DMR-9403009 and DMR-9706575 and a SUR grant from IBM. We would like to thank Ekaterina Anokhina and Abdessadek Lachgar for their help with the $Ti_2Nb_6O_{12}$ calculations,

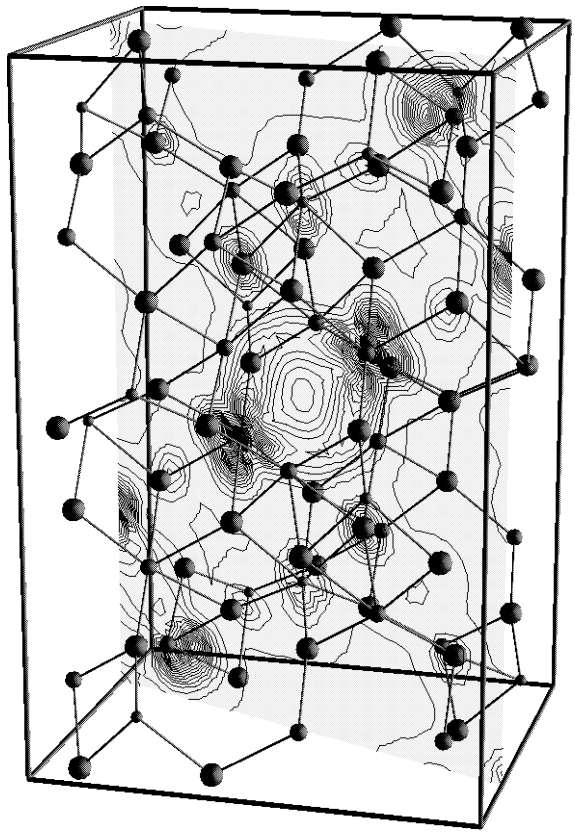

Fig. 9. Ball and stick model of $Ti_2Nb_6O_{12}$ with superimposed plane showing density contours for partial density of upper valence electrons. Balls corresponding to Ti, Nb, and O are denoted with increasing sphere radii.

A. Kokalj for providing a copy of his XCrySDen program, and Yonas Abraham for writing the *bandplot* program. Helpful comments by Kevin Conley and Rodney Dunning are gratefully acknowledged.

References

[1] P.E. Blöchl, Phys. Rev. B 50 (1994) 17 953–17 979.
[2] P. Hohenberg, W. Kohn, Phys. Rev. B 136 (1964) 864–871.
[3] W. Kohn, L.J. Sham, Phys. Rev. A 140 (1965) 1133–1138.
[4] N.A.W. Holzwarth, G.E. Matthews, R.B. Dunning, A.R. Tackett, Y. Zeng, Phys. Rev. B 55 (1997) 2005–2017.
[5] N.A.W. Holzwarth, G.E. Matthews, A.R. Tackett, R.B. Dunning, Phys. Rev. B 57 (1998) 11 827–11 830.
[6] G. Kresse, D. Joubert, Phys. Rev. B 59 (1999) 1758–1775.
[7] M. Valiev, J.H. Weare, J. Phys. Chem. A 103 (1999) 10 588–10 601.
[8] N.A.W. Holzwarth, A.R. Tackett, G.E. Matthews, Comput. Phys. Comm. 135 (2001) 329.
[9] A. Tackett, Ph.D. Thesis, Wake Forest University, May 1998.
[10] J.P. Perdew, Y. Wang, Phys. Rev. B 45 (1992) 13 244–13 249.
[11] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.D. Joannopoulos, Rev. Modern Phys. 64 (1992) 1045–1097.
[12] R. Car, M. Parrinello, Phys. Rev. Lett. 55 (1985) 2471–2474.
[13] D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, SIAM J. Sci. Comput. 20 (1998) 94–125.
[14] G.L.G. Sleijpen, H.A. van der Vorst, SIAM J. Matrix Anal. Appl. 17 (1996) 401–425.
[15] V. Eyert, J. Comput. Phys. 124 (1996) 271–285.

[16] D.G. Anderson, J. ACM 12 (1965) 547–560.
[17] E.R. Davidson, Comput. in Phys. 7 (1993) 519–522.
[18] E.R. Davidson, Comput. Phys. Commun. 53 (1989) 49–60.
[19] C.L. Fu, K.M. Ho, Phys. Rev. B 28 (1983) 5480–5486.
[20] N.A.W. Holzwarth, Y. Zeng, Phys. Rev. B 49 (1994) 2351–2361.
[21] S.G. Louie, K.-M. Ho, M.L. Cohen, Phys. Rev. B 19 (1979) 1774–1782.
[22] L.-W. Wang, A. Zunger, Phys. Rev. B 51 (1995) 17 398–17 416.
[23] A. Kokalj, J. Mol. Graphics Modelling 17 (1999) 176–179, Also A. Kokalj and M. Causa, to be published. The web site for this program is: http://www-k3.ijs.si/kokalj/xc/XCrySDen.html.
[24] E.V. Anokhina, M.W. Essig, C.S. Day, A. Lachgar, J. Amer. Chem. Soc. 121 (1999) 6827–6833.