Computer Physics Communications 183 (2012) 1269–1289

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

# BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures ☆

Jack Deslippe [a,b,*], Georgy Samsonidze [a,b], David A. Strubbe [a,b], Manish Jain [a,b], Marvin L. Cohen [a,b], Steven G. Louie [a,b]

[a] Department of Physics, University of California, Berkeley, CA 94720, United States
[b] Materials Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States

ARTICLE INFO

*Article history:*
Received 27 September 2011
Accepted 2 December 2011
Available online 9 December 2011

*Keywords:*
Many-body physics
GW
Bethe–Salpeter equation
Quasiparticle
Optics
Exciton

ABSTRACT

BerkeleyGW is a massively parallel computational package for electron excited-state properties that is based on the many-body perturbation theory employing the *ab initio* GW and GW plus Bethe–Salpeter equation methodology. It can be used in conjunction with many density-functional theory codes for ground-state properties, including `PARATEC`, `PARSEC`, `Quantum ESPRESSO`, `SIESTA`, and `Octopus`. The package can be used to compute the electronic and optical properties of a wide variety of material systems from bulk semiconductors and metals to nanostructured materials and molecules. The package scales to 10 000s of CPUs and can be used to study systems containing up to 100s of atoms.

**Program summary**

*Program title:* BerkeleyGW
*Catalogue identifier:* AELG_v1_0
*Program summary URL:* http://cpc.cs.qub.ac.uk/summaries/AELG_v1_0.html
*Program obtainable from:* CPC Program Library, Queen's University, Belfast, N. Ireland
*Licensing provisions:* Open source BSD License. See code for licensing details.
*No. of lines in distributed program, including test data, etc.:* 576 540
*No. of bytes in distributed program, including test data, etc.:* 110 608 809
*Distribution format:* tar.gz
*Programming language:* Fortran 90, C, C++, Python, Perl, BASH
*Computer:* Linux/UNIX workstations or clusters
*Operating system:* Tested on a variety of Linux distributions in parallel and serial as well as AIX and Mac OSX
*RAM:* (50–2000) MB per CPU (Highly dependent on system size)
*Classification:* 7.2, 7.3, 16.2, 18
*External routines:* BLAS, LAPACK, FFTW, ScaLAPACK (optional), MPI (optional). All available under open-source licenses.
*Nature of problem:* The excited state properties of materials involve the addition or subtraction of electrons as well as the optical excitations of electron–hole pairs. The excited particles interact strongly with other electrons in a material system. This interaction affects the electronic energies, wavefunctions and lifetimes. It is well known that ground-state theories, such as standard methods based on density-functional theory, fail to correctly capture this physics.
*Solution method:* We construct and solve the Dyson's equation for the quasiparticle energies and wavefunctions within the GW approximation for the electron self-energy. We additionally construct and solve the Bethe–Salpeter equation for the correlated electron–hole (exciton) wavefunctions and excitation energies.
*Restrictions:* The material size is limited in practice by the computational resources available. Materials with up to 500 atoms per periodic cell can be studied on large HPCs.

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

* Corresponding author at: Lawrence Berkeley National Lab, 1 Cyclotron Road Mail Stop 943-256, Berkeley, CA 94720, United States.
*E-mail address:* jdeslip@gmail.com (J. Deslippe).

0010-4655/$ – see front matter 
doi:10.1016/j.cpc.2011.12.006

*Additional comments:* The distribution file for this program is approximately 110 Mbytes and therefore is not delivered directly when download or E-mail is requested. Instead a html file giving details of how the program can be obtained is sent.
*Running time:* 1–1000 minutes (depending greatly on system size and processor number).


## 1. Introduction

Over the last few decades, the *ab initio* GW methodology has been successfully applied to the study of the quasiparticle properties of a large range of material systems from traditional bulk semiconductors, insulators and metals to, more recently, nano-systems like polymers, nano-wires and molecules [1–5]. The GW approach, which is based on approximating the electron self-energy as the first term in an expansion in the screened Coulomb interaction, $W$ [6], has proven to yield quantitatively accurate quasiparticle band gaps and dispersion relations from first principles.

Additionally, the Bethe–Salpeter equation (BSE) approach to the optical properties of materials has proven exceptionally accurate in predicting the optical response of a similarly large class of materials employing an electron–hole interaction kernel derived within the same level of approximations as GW [7–10].

The combined GW-BSE approach is now arguably regarded as the most accurate methodology commonly used for computing the quasiparticle and optical properties of condensed-matter systems. A perceived drawback of the GW methodology is its computational cost; a GW-BSE calculation is usually thought to be an order of magnitude (or worse) more costly than a typical density functional theory (DFT) calculation for the same system. Since the pioneering work of Ref. [1], many GW implementations have been made, but most are limited to small systems of the size of 10s of atoms, and scaling to only small numbers of CPUs on the order of 100.

BerkeleyGW is a massively parallel computer package written predominantly in `FORTRAN90` that implements the *ab initio* GW methodology of Hybertsen and Louie [1] and includes many more recent advances, such as the Bethe–Salpeter equation approach for optical properties [8]. It alleviates the restriction to small numbers of atoms and scales beyond thousands of CPUs. The package is intended to be used on top of a number of mean-field (DFT and other) codes that focus on ground-state properties, such as `PARATEC` [11], `Quantum ESPRESSO` [12], `SIESTA` [13], `PARSEC` [14,15], `Octopus` [49,50] and an empirical pseudopotential code (EPM) included in the package (based on TBPW [16]). More information about BerkeleyGW, the latest source code, and help forums can be found by visiting the website at http://berkeleygw.org/.

## 2. Theoretical framework

The *ab initio* GW-BSE approach is a many-body Green's-function methodology in which the only input parameters are the constituent atoms and the approximate structure of the system [1,8]. Typical calculations of the ground- and excited-state properties using the GW-BSE method can be broken into three steps: (1) the solution of the ground-state structural and electronic properties within a suitable ground-state theory such as *ab initio* pseudopotential density-functional theory, (2) the calculation of the quasiparticle energies and wavefunctions within the GW approximation for the electron self-energy operator, and (3) the calculation of the two-particle correlated electron–hole excited states through the solution of a Bethe–Salpeter equation.

DFT calculations, often the chosen starting point for GW, are performed by solving the self-consistent Kohn–Sham equations with an approximate functional for the exchange-correlation potential, $V_{\text{xc}}$ – common approximations being the local density approximation (LDA) [17] and the generalized-gradient approximation (GGA) [18]:

$$\left[-\frac{1}{2}\nabla^2 + V_{\text{ion}} + V_{\text{H}} + V_{\text{xc}}^{\text{DFT}}\right]\psi_{n\mathbf{k}}^{\text{DFT}} = E_{n\mathbf{k}}^{\text{DFT}}\psi_{n\mathbf{k}}^{\text{DFT}} \tag{1}$$

where $E_{n\mathbf{k}}^{\text{DFT}}$ and $\psi_{n\mathbf{k}}^{\text{DFT}}$ are the Kohn–Sham eigenvalues and eigenfunctions respectively, $V_{\text{ion}}$ is the ionic potential, $V_{\text{H}}$ is the Hartree potential and $V_{\text{xc}}$ is the exchange-correlation potential within a suitable approximation. When DFT is chosen as the starting point for GW, the Kohn–Sham wavefunctions and eigenvalues are used here as a first guess for their quasiparticle counterparts. The quasiparticle energies and wavefunctions (i.e., the one-particle excitations) are computed by solving the following Dyson equation [19, 1] in atomic units:

$$\left[-\frac{1}{2}\nabla^2 + V_{\text{ion}} + V_{\text{H}} + \Sigma\left(E_{n\mathbf{k}}^{\text{QP}}\right)\right]\psi_{n\mathbf{k}}^{\text{QP}} = E_{n\mathbf{k}}^{\text{QP}}\psi_{n\mathbf{k}}^{\text{QP}} \tag{2}$$

where $\Sigma$ is the self-energy operator within the GW approximation, and $E_{n\mathbf{k}}^{\text{QP}}$ and $\psi_{n\mathbf{k}}^{\text{QP}}$ are the quasiparticle energies and wavefunctions, respectively. For systems of dimension less than three, the Coulomb interaction may be replaced by a truncated interaction. The interaction is set to zero for particle separation beyond the size of the system in order to avoid unphysical interaction between the material and its periodic images in a super-cell [20] calculation. The electron–hole excitation states (probed in optical or other measurements) are calculated through the solution of a Bethe–Salpeter equation [8,7] for each exciton state $S$:

$$\left(E_{c\mathbf{k}}^{\text{QP}} - E_{v\mathbf{k}}^{\text{QP}}\right)A_{vc\mathbf{k}}^{S} + \sum_{v'c'\mathbf{k}'}\langle vc\mathbf{k}|K^{\text{eh}}|v'c'\mathbf{k}'\rangle = \Omega^{S}A_{vc\mathbf{k}}^{S} \tag{3}$$

where $A_{vc\mathbf{k}}^{S}$ is the exciton wavefunction (in the quasiparticle state representation), $\Omega^{S}$ is the excitation energy, and $K^{\text{eh}}$ is the electron–hole interaction kernel. We make the Tamm–Dancoff approximation by including only valence → conduction transitions [8,21]. The exciton wavefunction can be expressed in real space as:

$$\Psi(\mathbf{r}_e, \mathbf{r}_h) = \sum_{\mathbf{k},c,v} A_{vc\mathbf{k}}^{S}\psi_{\mathbf{k},c}(\mathbf{r}_e)\psi_{\mathbf{k},v}^{*}(\mathbf{r}_h), \tag{4}$$

and the imaginary part of the dielectric function, if one is interested in optical properties, can be expressed as

$$\epsilon_2(\omega) = \frac{16\pi^2 e^2}{\omega^2}\sum_{S}\left|\mathbf{e}\cdot\langle 0|\mathbf{v}|S\rangle\right|^2\delta\left(\omega - \Omega^{S}\right) \tag{5}$$

where

$$\langle 0|\mathbf{v}|S\rangle = \sum_{vc\mathbf{k}} A_{vc\mathbf{k}}^{S}\langle v\mathbf{k}|\mathbf{v}|c\mathbf{k}\rangle, \tag{6}$$

and $\mathbf{v}$ is the velocity operator along the direction of the polarization of light, $\mathbf{e}$. One may compare this to the non-interacting absorption spectrum:

$$\epsilon_2(\omega) = \frac{16\pi^2 e^2}{\omega^2}\sum_{vc\mathbf{k}}\left|\mathbf{e}\cdot\langle v\mathbf{k}|\mathbf{v}|c\mathbf{k}\rangle\right|^2\delta\left(\omega - E_{c\mathbf{k}}^{\text{QP}} + E_{v\mathbf{k}}^{\text{QP}}\right). \tag{7}$$

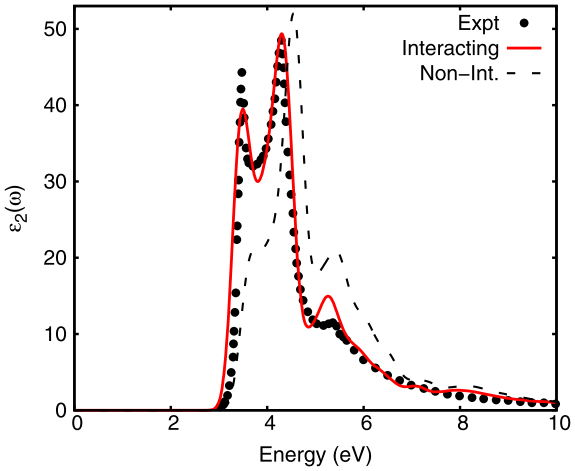


**Fig. 1.** The absorption spectra for silicon calculated at the GW (black dashed) and GW-BSE (red solid) levels using the BerkeleyGW package. Experimental data from [22].

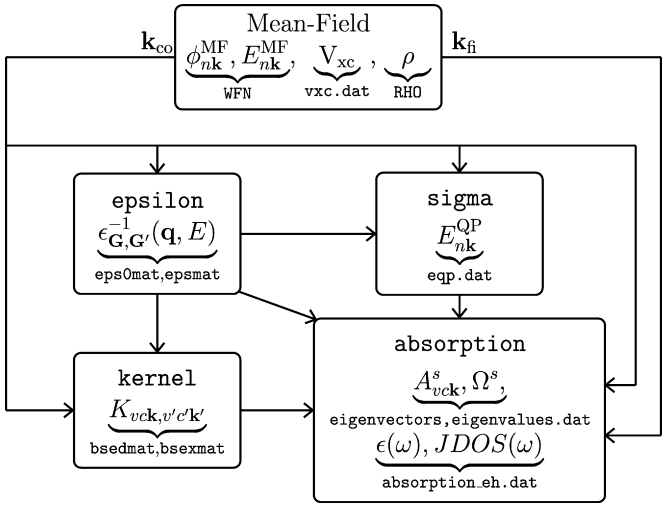


**Fig. 2.** Flow chart of a GW-BSE calculation performed in the BerkeleyGW package.

An example absorption spectrum for silicon computed with the BerkeleyGW package at the GW and GW-BSE levels is shown in Fig. 1. Only when both the quasiparticle effects within the GW approximation and the excitonic effects through the solution of the Bethe–Salpeter equation are included is good agreement with experiment reached.

## 3. Computational layout

### 3.1. Major sections of the code

Fig. 2 illustrates the procedure for carrying out an *ab initio* GW-BSE calculation to obtain quasiparticle and optical properties using the BerkeleyGW code. First, one obtains the mean-field electronic orbitals and eigenvalues as well as the charge density. One can utilize one of the many supported DFT codes [11–13,15,50] to construct this mean-field starting point and convert it to the plane-wave BerkeleyGW format (see Appendix A) using the wrappers included. (Note that norm-conserving pseudopotentials must be used, or else extra contributions would need to be added to our matrix elements.)

The `Epsilon` executable produces the polarizability and inverse dielectric matrices. In the `epsilon` executable, the static or frequency-dependent polarizability and dielectric function are calculated within the random-phase approximation (RPA) using the electronic eigenvalues and eigenfunctions from a mean-field reference system. The main output are the files `espmat` and `eps0mat` that contain the inverse-dielectric matrix.

In the `sigma` executable, the screened Coulomb interaction, $W$, is constructed from the inverse dielectric matrix and the one-particle Green's function, $G$, is constructed from the mean-field eigenvalues and eigenfunctions. We then calculate the diagonal and (optionally) off-diagonal elements of the self-energy operator, $\Sigma = iGW$, as a matrix in the mean-field basis. In many cases, only the diagonal elements are sizable within the chosen mean-field orbital basis; in such cases, in applications to real materials, the effects of $\Sigma$ can be treated within first-order perturbation theory. The `sigma` executable evaluates $\Sigma$ in the form $\Sigma = V_{\rm xc} + (\Sigma - V_{\rm xc})$, where $V_{\rm xc}$ is the independent-particle mean-field approximation to the exchange-correlation potential of the chosen mean-field system. For moderately correlated electron systems, the best available mean-field Hamiltonian may often be taken to be the Kohn–Sham Hamiltonian [17]. However, many mean-field starting points are consistent with the BerkeleyGW package, such as Hartree–Fock, static COHSEX and hybrid functionals. In principle, the process of correcting the eigenfunctions and eigenvalues (which determine $W$ and $G$) could be repeated until self-consistency is reached or the $\Sigma$ matrix diagonalized in full. However, in practice, it is found that an adequate solution often is obtained within first-order perturbation theory on Dyson's equation for a given $\Sigma$ [23,24]. Comparison of calculated energies with experiment shows that this level of approximation is very accurate for semiconductors and insulators and for most conventional metals. The outputs of the `sigma` executable are $E^{\rm QP}$, the quasiparticle energies, which are written to the file `eqp.dat` using the `eqp.py` post-processing utility on the generated `sigma.log` files for each `sigma` run.

The BSE executable, `kernel`, takes as input the full dielectric matrix calculated in the `epsilon` executable, which is used to screen the attractive direct electron–hole interaction, and the quasiparticle wavefunctions, which often are taken to be the same as the mean-field wavefunctions. The direct and exchange part of the electron–hole kernel are calculated and output into the `bsedmat` and `bsexmat` files respectively. The `absorption` executable uses these matrices, the quasiparticle energies and wavefunctions from a coarse **k**-point grid GW calculation, as well as the wavefunctions from a fine **k**-point grid. The quasiparticle energy corrections and the kernel matrix elements are interpolated onto the fine grid. The Bethe–Salpeter Hamiltonian, consisting of the electron–hole kernel with the addition of the kinetic-energy term, is constructed in the quasiparticle electron–hole pair basis and diagonalized yielding the electron–hole amplitude, or exciton wavefunctions, and excitation energies, printed in the file `eigenvectors`. Exciton binding energies can be inferred from the energy of the correlated exciton states relative to the inter-band-transition continuum edge. With the excitation energies and amplitudes of the electron–hole pairs, one then can calculate the macroscopic dielectric function for various light polarizations which is written to the file `absorption_eh.dat`. This may be compared to the absorption spectrum without the electron–hole interaction included, printed in the file `absorption_noeh.dat`.

Example input files for each executable are contained within the source code for the package, as well as complete example calculations for silicon, the $(8,0)$ and $(5,5)$ single-walled carbon nanotubes (SWCNTs), the CO molecule, and sodium metal. There are several post-processing and visualization utilities included in the package that are described in Section 8.

Additionally, sums over **k** and **q** are accompanied by an implicit division by the volume of the super-cell considered, $V_{\rm sc} = N_k V_{\rm uc}$,

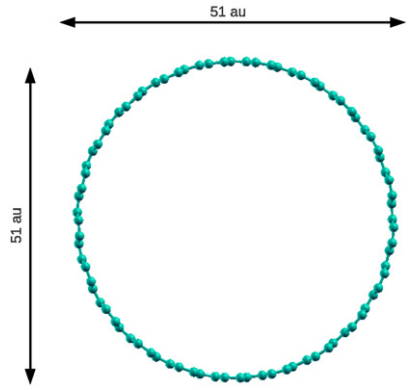


**Fig. 3.** The cross section of the (20, 20) SWCNT used throughout the paper as a benchmark system.

**Table 1**
Breakdown of the CPU and wall-clock time spent on the calculation of the (20, 20) SWCNT with parameters described in the text. The × indicates an additional level of trivial parallelization over the **k**- or **q**-points.

| Step | # CPUs | CPU hours | Wall hours |
|---|---|---|---|
| DFT Coarse | 64 × 32 | 19 000 | 9.1 |
| DFT Fine | 64 × 256 | 29 000 | 1.8 |
| `epsilon` | 1600 × 32 | 61 000 | 1.2 |
| `sigma` | 960 × 16 | 46 000 | 3.0 |
| `kernel` | 1024 | 600 | 0.6 |
| `absorption` | 256 | 500 | 2.0 |

where $N_k$ is the number of points in the **k**-grid and $V_{\text{uc}}$ is the volume of the unit cell in a periodic system.

Throughout the paper, we refer to benchmark numbers from calculations on the (20, 20) SWCNT (Fig. 3). This system has 80 carbon atoms and 160 occupied bands. We use 800 unoccupied bands in all sums requiring empty orbitals. We use a super-cell of size $80 \times 80 \times 4.6$ au$^3$ equivalent to a bulk system of greater than 500 atoms. We use a $1 \times 1 \times 32$ coarse **k**-grid and a $1 \times 1 \times 256$ fine **k**-grid. We calculate the self-energy corrections within the diagonal approximation for 8 conduction and 8 valence bands. The Bethe–Salpeter equation is solved with 8 conduction and 8 valence bands. The relative costs of the various steps in the GW-BSE calculation using the BerkeleyGW package is shown in Table 1. As can be seen from the table, the actual time to solution for the GW-BSE part of the calculation is smaller than that of the DFT parts.

### 3.2. RPA dielectric matrix: `epsilon`

`epsilon` is a standalone executable that computes either the static or dynamic RPA polarizability and corresponding inverse dielectric function from input electronic eigenvalues and eigenvectors computed in a suitable mean-field code. As we discuss in detail below, the input electronic eigenvalues and eigenvectors can come from a variety of different mean-field approximations including DFT within LDA/GGA, generalized Kohn–Sham hybrid-functional approximations as well as direct approximations to the GW Dyson's equation such as the static-COHSEX [19,25] approximation and the Hartree–Fock approximation.

We will first discuss the computation of the static polarizability and the inverse dielectric matrix. The `epsilon` executable computes the static RPA polarizability using the following expression [1]:

$$\chi_{\mathbf{GG}'}(\mathbf{q};0) = \sum_{n}^{\text{occ}} \sum_{n'}^{\text{emp}} \sum_{\mathbf{k}} M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}) M^{*}_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}') \frac{1}{E_{n\mathbf{k}+\mathbf{q}} - E_{n'\mathbf{k}}} \tag{8}$$

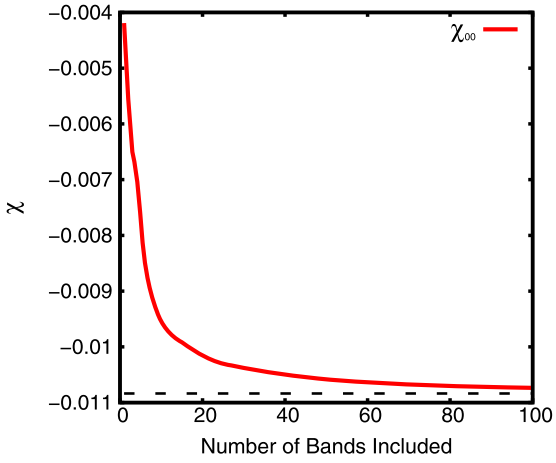


**Fig. 4.** Example convergence output plotted from `chi_converge.dat` showing the convergence of the sum in Eq. (8) for the $\mathbf{G}, \mathbf{G}' = 0$ and $\mathbf{q} = (0, 0, 0.5)$ component of $\chi$ in ZnO.

where

$$M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}) = \langle n\mathbf{k}+\mathbf{q} | e^{i(\mathbf{q}+\mathbf{G})\cdot\mathbf{r}} | n'\mathbf{k} \rangle \tag{9}$$

are the plane-wave matrix elements. Here **q** is a vector in the first Brillouin zone, **G** is a reciprocal-lattice vector, and $\langle n\mathbf{k}|$ and $E_{n\mathbf{k}}$ are the mean-field electronic eigenvectors and eigenvalues. The matrix in Eq. (8) is to be evaluated up to $|G^2| < |E_{\text{cut}}|$ for both **G** and **G**′ where $E_{\text{cut}}$ defines the dielectric energy cutoff. The number of empty states, $n'$, included in the summation must be such that the highest empty state included has an energy corresponding to $E_{\text{cut}}$. There is therefore one, rather than two, convergence parameter in evaluating Eq. (8): one either must choose to converge with empty states or with the dielectric energy cutoff and set the remaining parameter to match the chosen convergence parameter. The `epsilon` code itself reports the convergence of Eq. (8) in an output file called `chi_converge.dat` (plotted in Fig. 4), that presents the computed value of $\chi_{\mathbf{GG}'=0}(\mathbf{q};0)$ and $\chi_{\mathbf{GG}'=\mathbf{G}_{\max}}(\mathbf{q};0)$ using partial sums in Eq. (8) where $\mathbf{G}_{\max}$ is the largest reciprocal-lattice vector included, and the number of empty states is varied between 1 and the maximum number requested in the input file, `epsilon.inp`. A simple extrapolation also is included.

With the expression for $\chi$ above, we can obtain the RPA dielectric matrix as

$$\epsilon_{\mathbf{GG}'}(\mathbf{q};0) = \delta_{\mathbf{GG}'} - v(\mathbf{q}+\mathbf{G})\chi_{\mathbf{GG}'}(\mathbf{q};0) \tag{10}$$

where $v(\mathbf{q}+\mathbf{G})$ is the bare Coulomb interaction defined as:

$$v(\mathbf{q}+\mathbf{G}) = \frac{4\pi}{|\mathbf{q}+\mathbf{G}|^2} \tag{11}$$

in the case of bulk crystals where no truncation is necessary. We discuss in Section 5 how to generalize this expression for the case of nano-systems where truncating the interaction in non-periodic directions greatly improves the convergence with super-cell size.

It should be noted that we use an asymmetric definition of the Coulomb interaction, as opposed to symmetric expressions such as

$$v(\mathbf{G},\mathbf{G}') = \frac{4\pi}{|\mathbf{q}+\mathbf{G}||\mathbf{q}+\mathbf{G}'|}. \tag{12}$$

This causes $\epsilon_{\mathbf{GG}'}(\mathbf{q};0)$ and $\chi_{\mathbf{GG}'}(\mathbf{q};0)$ to be also asymmetric in **G** and **G**′. This asymmetry is resolved when constructing the static screened Coulomb interaction by use of the expression:

$$W_{\mathbf{GG}'}(\mathbf{q};0) = \epsilon^{-1}_{\mathbf{GG}'}(\mathbf{q};0)\, v\left(\mathbf{q}+\mathbf{G}'\right). \tag{13}$$

Here $W$ is symmetric in $\mathbf{G}$ and $\mathbf{G}'$ even though both $v$ and $\epsilon^{-1}$ individually are not.

The computation of $\epsilon^{-1}_{\mathbf{GG}'}(\mathbf{q};0)$ in the `epsilon` code involves three computationally intensive steps: the computation of the matrix elements needed for the summation in Eq. (8), the summation itself and the inversion of the dielectric matrix to yield $\epsilon^{-1}_{\mathbf{GG}'}(\mathbf{q};0)$. The `epsilon` code first computes all the matrix elements $M_{nn'}$ required in the summation for Eq. (8). This step is generally the most time-consuming step in the execution of the `epsilon` code. Naively, one might think this process scales as $N^4$, where $N$ is the number of atoms in the system. This is because both the number of valence and conduction bands needed scales linearly with $N$ and the number of $\mathbf{G}$ vectors scales linearly with the cell volume which itself scales linearly with the number of atoms. Thus, we must calculate $N^3$ matrix elements each of which involves a sum over the plane-wave basis set for the eigenfunctions. We therefore have an $N^4$ scaling. However, we can achieve $N^3 \log N$ scaling by using fast Fourier transforms (FFTs), noting that the expression in Eq. (9) is a convolution in Fourier space [26]. Therefore, Eq. (9) can be written as the Fourier transform of a direct product of the wavefunctions in real space:

$$M_{nn'}\left(\mathbf{k},\mathbf{q},\{\mathbf{G}\}\right) = FFT^{-1}\left(\phi_{n,\mathbf{k}+\mathbf{q}}(\mathbf{r}) * \phi^*_{n',\mathbf{k}}(\mathbf{r})\right). \tag{14}$$

The FFTs are implemented with FFTW [27] and scale as $N \log N$. The computation of all the matrix elements needed for Eq. (8) therefore scales as $N^3 \log N$. We discuss in the following sections that the computation of these matrix elements can be parallelized very trivially up to tens of thousands of CPUs. Given an infinite resource of CPUs, our implementation would have a wall-time scaling of $N \log N$, nearly linear in the number of atoms.

Having computed the individual matrix elements required in Eq. (8), we now turn our attention to the summation involved in the same expression. It should be noted that the formal scaling of this step with the number of atoms is $N^4$ since one must sum over the number of occupied bands and the number of unoccupied bands for every $\mathbf{G}$ and $\mathbf{G}'$ pair – each one of these quantities scales linearly with the number of atoms. This step therefore formally has the worst scaling of the entire GW process – leading many to claim that GW as a whole scales like $N^4$. However, in practice for most systems currently under study within a generalized plasmon-pole (GPP) [1] or other approximation where this sum is done only once for the static polarizability, this step represents less than 10 percent of a typical calculation time even for systems of 100s of atoms because this step can be optimized and parallelized greatly. In particular, Eq. (8) can be written very compactly as a single matrix–matrix product for each $\mathbf{q}$:

$$\chi_{\mathbf{GG}'}(\mathbf{q};0) = \mathbf{M}\left(\mathbf{G},\mathbf{q},\left(n,n',\mathbf{k}\right)\right) \cdot \mathbf{M}^T\left(\mathbf{G}',\mathbf{q}\left(n,n',\mathbf{k}\right)\right) \tag{15}$$

where $(n,n',\mathbf{k})$ represents a single composite index that is summed over as the inner dimension in the matrix–matrix product. The matrices $\mathbf{M}$ can be expressed in terms of the matrix elements $M$ as:

$$\mathbf{M}\left(\mathbf{G},\mathbf{q},\left(n,n',\mathbf{k}\right)\right) = M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}) \cdot \frac{1}{\sqrt{E_{n\mathbf{k}+\mathbf{q}} - E_{n'\mathbf{k}}}}. \tag{16}$$

The single dense matrix–matrix product required in Eq. (15) still scales as $N^4$ since the inner dimension, $(n,n',\mathbf{k})$, scales as $N^2$ and dense matrix multiplication itself scales as $N^2$. However, in the BerkeleyGW package, this single step is still made quite rapid for even systems as large as 100s of atoms. The LEVEL 3 BLAS [28] libraries `DGEMM` and `ZGEMM` and their parallel analogues are used to compute the single matrix product in Eq. (15). As we discuss further in Section 4.1, in the evaluation of Eq. (9), the parallel wall-time scaling is $N^2$ with the number of atoms.

Finally, once we have constructed $\chi_{\mathbf{GG}'}(\mathbf{q};0)$ we can construct the RPA dielectric matrix and inverse dielectric matrix required for the computation of the screened Coulomb interaction, $W$. The dielectric matrix as implemented in the code is expressed in Eq. (10).

Here we require for the first time the Coulomb interaction in reciprocal space $v(\mathbf{q}+\mathbf{G})$, which can be computed trivially from Eq. (11) for the case of bulk crystals, but requires an FFT for the case of nanostructured materials. We discuss this more in Section 5.

There is a clear problem in directly computing $\epsilon_{\mathbf{00}}(\mathbf{q}=\mathbf{0})$ due to the fact that the Coulomb interaction, Eq. (11), diverges as $\mathbf{q}\to 0$ except in the case of box-type truncation schemes (see Section 5). For semiconducting systems, due to orthogonality, the matrix elements (Eq. (9)) themselves go to 0 with the form $|M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}=\mathbf{0})| \propto |q|$. So, $\epsilon(\mathbf{q}\to 0)$ contains a non-trivial $q^2/q^2$ limit. One way to handle this would be to take the limit of Eqs. (8) and (9) analytically via $\mathbf{k}\cdot\mathbf{p}$ perturbation theory, where the perturbation is the momentum operator $-i\nabla$ plus the commutators with the non-local potential of the mean-field Hamiltonian [29,1]. This is analogous to the treatment of the velocity operator in `absorption` (Eq. (45)). The `epsilon` code has implemented a simpler scheme, however, in which we numerically take the limit as $\mathbf{q}\to 0$ by evaluating $\epsilon_{\mathbf{00}}(\mathbf{q}_0)$ at a small but finite $\mathbf{q}_0$ usually taken as approximately 1/1000th of the Brillouin zone. For semiconducting systems, where $\epsilon_{\mathbf{00}}(\mathbf{q}=\mathbf{0}) \to C$, it is sufficient to construct a separate $\mathbf{k}$-grid for the conduction and valence bands shifted by the small vector $\mathbf{q}_0$ in order to compute $M_{nn'}(\mathbf{k},\mathbf{q}_0,\mathbf{G}=\mathbf{0})$, where $n$ is a valence and $n'$ a conduction band, and to evaluate the correct limiting $q^2/q^2$ ratio. For metals, however, intra-band transitions have $|M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}=\mathbf{0})| \propto C$, yielding $\epsilon_{\mathbf{00}}(\mathbf{q}\to 0) \propto C'/q^2$. In this case, the two-$\mathbf{k}$-grid treatment is insufficient, because the proportionality coefficient $C'$ depends sensitively on the density of states (DOS) at the Fermi energy. Therefore a $\mathbf{k}$-grid sampling of the same spacing as $\mathbf{q}_0$ is required, although fewer conduction bands are necessary in the sum since $\epsilon(\mathbf{q}\to 0)$ is dominated by intra-band transitions. Thus we typically calculate $\epsilon(\mathbf{q}\to 0)$ using a single fine wavefunction grid by using the smallest $\mathbf{q}$ consistent with the grid. Note that this treatment of intra-band transitions is still the zero-temperature limit in our code, as the effect of thermal occupations is small in GW except at very large temperatures [30]. Effectively occupations are taken as one below the Fermi level, zero above the Fermi level, and 1/2 at the Fermi level (as needed for graphene at the Dirac point). This is despite any smearing that may have been used in the underlying mean-field calculation.

The inversion of the dielectric matrix required to compute $W$, Eq. (13), is done with LAPACK and ScaLAPACK (for parallel calculations) using `ZGESV`, `DGESV` and their parallel counterparts. The inversion scales like $N^3$ with the number of atoms and, as we discuss below, scales well up to 100s of processors with ScaLAPACK. In general, for systems of up to 100s of atoms, the inversion step represents less than 10 percent of the total computation time for `epsilon`.

We have so far limited ourselves to situations in which only a direct calculation of the static polarizability, Eq. (8), is required, such as in the static-COHSEX approximation [25] or when utilizing a GPP model [1] to extend the dielectric response to non-zero frequencies. However, we can also do a more refined calculation. Options are given in the code so that the dielectric matrix is computed directly at real frequencies without extrapolation, as is formally required in the Dyson equation. We use in the package the advanced and retarded dielectric functions, defined as:

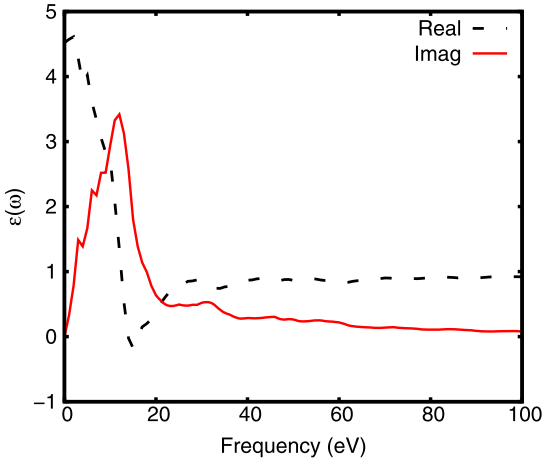


**Fig. 5.** Example output plotted from `EpsDyn` showing the computed $\epsilon_{\mathbf{00}}(\omega)$ in ZnO.

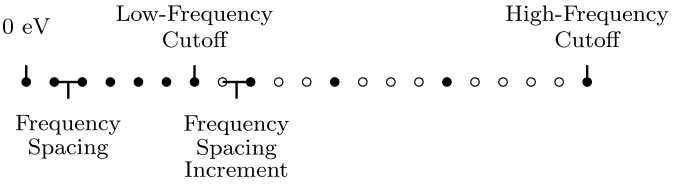


**Fig. 6.** Schematic of the frequency-grid parameters for a full-frequency calculation in `epsilon`. The open circles are a continuation of the uniform grid that are omitted above the low-frequency cutoff.

$$\epsilon^{\mathrm{r/a}}_{\mathbf{GG'}}(\mathbf{q};E) = \delta_{\mathbf{GG'}} - v(\mathbf{q}+\mathbf{G})\sum_{n}^{\mathrm{occ}}\sum_{n'}^{\mathrm{emp}}\sum_{\mathbf{k}} M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G})M^{*}_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G'}) \times \frac{1}{2}\left[\frac{1}{E_{n\mathbf{k}+\mathbf{q}} - E_{n'\mathbf{k}} - E \mp i\delta} + \frac{1}{E_{n\mathbf{k}+\mathbf{q}} - E_{n'\mathbf{k}} + E \pm i\delta}\right] \quad (17)$$

where $E$ is the evaluation frequency and $\delta$ is a broadening parameter chosen to be consistent with the energy spacing afforded by the **k**-point sampling of the calculation, using the upper (lower) signs for the retarded (advanced) function. In principle, one must converge the calculation with respect to increasing the **k**-point sampling and decreasing this broadening parameter (Fig. 5).

In the `epsilon` code, we compute Eq. (17) on a grid of real frequencies, $E$, specified by a frequency spacing, a low-frequency cutoff, a high-frequency cutoff and a frequency-spacing increment. We sample the frequency on the real axis uniformly from 0 to the low-frequency cutoff with a sampling rate given by the frequency spacing. We then increase the frequency spacing by the step increment until we reach the high-frequency cutoff (Fig. 6). In general, one also must refine this frequency grid until convergence is reached, but we find that for the purpose of calculating band gaps of typical semiconductors, a frequency spacing of a few hundred meV and a high-frequency cutoff of twice the dielectric energy cutoff is sufficient, though it should be noted this energy can be quite high (e.g. the case of ZnO [31]).

In cases where the computation of the "full-frequency" dielectric response function is required, the bottleneck of the calculation often does become the $N^4$ summation step of Eq. (15). This is because the computation of the matrix elements, Eq. (9), needs only be done once, whereas the summation must be done for all frequencies separately. Because of this, a full-frequency `epsilon` calculation of between 10–50 frequencies costs only twice the time of a static `epsilon` calculation, but the cost scales linearly with frequencies after this point.

### 3.3. Computation of the self-energy: `sigma`

The `sigma` executable takes as input the inverse epsilon matrix calculated from the `epsilon` executable and a suitable set of mean-field electronic energies and wavefunctions. It computes a representation of the Dyson's equation, Eq. (2), in the basis of the mean-field eigenfunctions through the computation of the diagonal and off-diagonal elements of $\Sigma$:

$$\langle\psi_{n\mathbf{k}}|H^{\mathrm{QP}}(E)|\psi_{m\mathbf{k}}\rangle = E^{\mathrm{MF}}_{n\mathbf{k}}\delta_{n,m} + \langle\psi_{n\mathbf{k}}|\Sigma(E) - \Sigma^{\mathrm{MF}}(E)|\psi_{m\mathbf{k}}\rangle \quad (18)$$

where $E$ is an energy parameter that should be set self-consistently to the quasiparticle eigenvalues, $E^{\mathrm{MF}}_{n\mathbf{k}}$ and $\psi_{n\mathbf{k}}$ are the mean-field eigenvalues and eigenvectors and $\Sigma^{\mathrm{MF}}$ is a mean-field approximation to the electronic self-energy operator, such as $V_{\mathrm{xc}}$ in the case of a DFT starting point.

It is often the case that the mean-field wavefunctions are sufficiently close to the quasiparticle wavefunctions [1] that one may reduce Eq. (18) to include only diagonal matrix elements. In this case the user may ask for only diagonal elements, and the quasiparticle energies will be updated in the following way:

$$E^{\mathrm{QP}}_{n\mathbf{k}} = E^{\mathrm{MF}}_{n\mathbf{k}} + \langle\psi_{n\mathbf{k}}|\Sigma(E) - \Sigma^{\mathrm{MF}}(E)|\psi_{n\mathbf{k}}\rangle. \quad (19)$$

The mean field in Eqs. (19) and (18) can be DFT within the LDA or GGA schemes as well as within a hybrid-functional approach. In the LDA case, for example, $\Sigma^{\mathrm{MF}}(E) = V_{\mathrm{xc}}$, is local and energy-independent. The starting mean-field calculation can also be an approximation to the Dyson's equation, Eq. (2), such as Hartree–Fock (the zero-screening limit) or static COHSEX (the static-screening limit) [32,33,25]. The use of these mean-field starting points for construction of Eqs. (18) and (19) is classified as a one-shot $G_0W_0$ calculation (the 0 subscript means that both $G$ and $W$ are constructed from the mean-field eigenvalues and eigenvectors). One also can start from a previous iteration of GW in an eigenvalue or eigenvector self-consistency scheme [33,34]. In this case, the 'MF' superscripts in Eqs. (19) and (18) should be renamed "previous" to designate the self-consistency process.

The `sigma` executable itself can evaluate the matrix elements of $\Sigma$ in Eqs. (19) and (18) within various approximations: Hartree–Fock, static COHSEX, GW within a GPP model and full-frequency GW.

For GW and static-COHSEX calculations, $\Sigma$ can be broken into two parts, $\Sigma = \Sigma_{\mathrm{SX}} + \Sigma_{\mathrm{CH}}$, where $\Sigma_{\mathrm{SX}}$ is the screened exchange operator and $\Sigma_{\mathrm{CH}}$ is the Coulomb-hole operator [1,6,19]. These are implemented in the `sigma` executable in the following way for a full-frequency calculation:

$$\langle n\mathbf{k}|\Sigma_{\mathrm{SX}}(E)|n'\mathbf{k}\rangle = -\sum_{n''}^{\mathrm{occ}}\sum_{\mathbf{qGG'}} M^{*}_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G'}) \times \left[\epsilon^{\mathrm{r}}_{\mathbf{GG'}}\right]^{-1}(\mathbf{q};E - E_{n''\mathbf{k}-\mathbf{q}})v(\mathbf{q}+\mathbf{G'}) \quad (20)$$

and

$$\langle n\mathbf{k}|\Sigma_{\mathrm{CH}}(E)|n'\mathbf{k}\rangle = \frac{i}{2\pi}\sum_{n''}\sum_{\mathbf{qGG'}} M^{*}_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G'}) \times \int_0^{\infty} dE' \frac{[\epsilon^{\mathrm{r}}_{\mathbf{GG'}}]^{-1}(\mathbf{q};E') - [\epsilon^{\mathrm{a}}_{\mathbf{GG'}}]^{-1}(\mathbf{q};E')}{E - E_{n''\mathbf{k}-\mathbf{q}} - E' + i\delta} v(\mathbf{q}+\mathbf{G'}) \quad (21)$$

where $M$ is defined in Eq. (9) and $\epsilon^{\mathrm{r}}$ and $\epsilon^{\mathrm{a}}$ are the retarded and advanced dielectric matrices defined in Eq. (17) [35]. In practice

the `sigma` executable computes the matrix elements of bare exchange, $\Sigma_X$ and of $\Sigma_{SX}-\Sigma_X$, where the matrix elements of $\Sigma_X$ are obtained by replacing $[\epsilon^r_{GG'}]^{-1}(\mathbf{q};E-E_{n''\mathbf{k}-\mathbf{q}})$ with $\delta_{\mathbf{G},\mathbf{G}'}$ in Eq. (20) (as given by Eq. (31) below). The integral in Eq. (21) over frequency is done numerically on the frequency grid used in the `epsilon` executable (Fig. 6).

For GPP calculations, the corresponding expressions used in the code are:

$$\begin{aligned}&\langle n\mathbf{k}|\Sigma_{SX}(E)|n'\mathbf{k}\rangle\\&=-\sum_{n''}^{\text{occ}}\sum_{\mathbf{q}GG'}M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}')\\&\quad\times\left[\delta_{GG'}+\frac{\Omega^2_{GG'}(\mathbf{q})(1-i\tan\phi_{GG'}(\mathbf{q}))}{(E-E_{n''\mathbf{k}-\mathbf{q}})^2-\tilde{\omega}^2_{GG'}(\mathbf{q})}\right]v(\mathbf{q}+\mathbf{G}')\end{aligned}\tag{22}$$

and

$$\begin{aligned}&\langle n\mathbf{k}|\Sigma_{CH}(E)|n'\mathbf{k}\rangle\\&=\frac{1}{2}\sum_{n''}\sum_{\mathbf{q}GG'}M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}')\\&\quad\times\frac{\Omega^2_{GG'}(\mathbf{q})(1-i\tan\phi_{GG'}(\mathbf{q}))}{\tilde{\omega}_{GG'}(\mathbf{q})(E-E_{n''\mathbf{k}-\mathbf{q}}-\tilde{\omega}_{GG'}(\mathbf{q}))}v(\mathbf{q}+\mathbf{G}')\end{aligned}\tag{23}$$

where $\Omega_{GG'}(\mathbf{q})$, $\tilde{\omega}_{GG'}(\mathbf{q})$, $\lambda_{GG'}(\mathbf{q})$ and $\phi_{GG'}(\mathbf{q})$ are the effective bare plasma frequency, the GPP mode frequency, the amplitude and the phase of the renormalized $\Omega^2_{GG'}(\mathbf{q})$ [1,36] defined as:

$$\Omega^2_{GG'}(\mathbf{q})=\omega_p^2\frac{(\mathbf{q}+\mathbf{G})\cdot(\mathbf{q}+\mathbf{G}')}{|\mathbf{q}+\mathbf{G}|^2}\frac{\rho(\mathbf{G}-\mathbf{G}')}{\rho(\mathbf{0})},\tag{24}$$

$$\tilde{\omega}^2_{GG'}(\mathbf{q})=\frac{|\lambda_{GG'}(\mathbf{q})|}{\cos\phi_{GG'}(\mathbf{q})},\tag{25}$$

$$|\lambda_{GG'}(\mathbf{q})|e^{i\phi_{GG'}(\mathbf{q})}=\frac{\Omega^2_{GG'}(\mathbf{q})}{\delta_{GG'}-\epsilon^{-1}_{GG'}(\mathbf{q};0)}.\tag{26}$$

Here, $\rho$ is the electron charge density in reciprocal space and $\omega_p^2=4\pi\rho(\mathbf{0})e^2/m$ is the classical plasma frequency. In this case, the integral over energy that is necessary in the full-frequency expression, Eq. (21), is reduced to a single term using an analytical approximation to the frequency dependence of the dielectric matrix requiring only the static dielectric matrix $\epsilon^{-1}_{GG'}(\mathbf{q};0)$ in Eq. (25). The analytical approximation is done using the $f$-sum rule for each $GG'$ pair as described in Ref. [1]. This reduces the computational cost of evaluating the $\Sigma$ matrix elements by a factor of the number of frequencies. It is important to note that for systems without inversion symmetry, $\rho$ in Eq. (24) and $V_{xc}$ in Eqs. (18) and (19) are complex functions in reciprocal space (even though these are real functions when transformed to real space). For systems with inversion symmetry, $\Omega^2_{GG'}(\mathbf{q})$ and $\tilde{\omega}^2_{GG'}(\mathbf{q})$ are real, $\phi_{GG'}(\mathbf{q})=0$ or $\pi$ and Eqs. (22)–(26) reduce to a simpler form [1].

In computing the sums in Eqs. (22) and (23) we drop terms in certain circumstances to save time and improve numerical precision. We neglect the terms for which $|\delta_{GG'}-\epsilon^{-1}_{GG'}(\mathbf{q};0)|$, $|\lambda_{GG'}(\mathbf{q})|$ or $|\cos\phi_{GG'}(\mathbf{q})|$ are less than a given tolerance, since these terms have a vanishing contribution to the matrix elements of the self-energy. This avoids ill-conditioned limits due to some of the intermediate quantities here being undefined. Another case is when for an occupied state $n''$, $E-E_{n''\mathbf{k}-\mathbf{q}}-\tilde{\omega}_{GG'}(\mathbf{q})\approx 0$, in which case the GPP factors in $\Sigma_{SX}$ and $\Sigma_{CH}$ each diverge, although the sum

$$-\delta_{GG'}+\frac{\Omega^2_{GG'}(\mathbf{q})(1-i\tan\phi_{GG'}(\mathbf{q}))}{2\tilde{\omega}_{GG'}(E-E_{n''\mathbf{k}-\mathbf{q}}+\tilde{\omega}_{GG'}(\mathbf{q}))}\tag{27}$$

remains finite. In this situation, we do not calculate these terms in $\Sigma_{SX}$ and $\Sigma_{CH}$ separately, but assign the sum of the contributions to $\Sigma_{SX}$. When $n''$ is unoccupied there is only a $\Sigma_{CH}$ contribution which diverges. Similarly, there are divergent contributions to $\Sigma_{SX}$ when $E-E_{n''\mathbf{k}-\mathbf{q}}+\tilde{\omega}_{GG'}(\mathbf{q})\approx 0$. In the full-frequency integrals in Eqs. (20) and (21), we can see that the contributions around a pole of $\epsilon^{-1}_{GG'}$ in this case vanish, so the correct analytic limit of these terms is zero [1].

For static COHSEX calculations, the expressions used in the code are:

$$\begin{aligned}\langle n\mathbf{k}|\Sigma_{SX}(0)|n'\mathbf{k}\rangle=&-\sum_{n''}^{\text{occ}}\sum_{\mathbf{q}GG'}M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}')\\&\times\epsilon^{-1}_{GG'}(\mathbf{q};0)v(\mathbf{q}+\mathbf{G}')\end{aligned}\tag{28}$$

and

$$\begin{aligned}&\langle n\mathbf{k}|\Sigma_{CH}(0)|n'\mathbf{k}\rangle\\&=\frac{1}{2}\sum_{n''}\sum_{\mathbf{q}GG'}M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}')\\&\quad\times\left[\epsilon^{-1}_{GG'}(\mathbf{q};0)-\delta_{GG'}\right]v(\mathbf{q}+\mathbf{G}')\end{aligned}\tag{29}$$

$$=\frac{1}{2}\sum_{\mathbf{q}GG'}M_{nn'}(\mathbf{k},\mathbf{q}=\mathbf{0},\mathbf{G}'-\mathbf{G})\left[\epsilon^{-1}_{GG'}(\mathbf{q};0)-\delta_{GG'}\right]v(\mathbf{q}+\mathbf{G}')\tag{30}$$

where Eqs. (28) and (29) can be derived formally from Eqs. (22) and (23) by setting $(E-E_{n''\mathbf{k}-\mathbf{q}})$ to zero. Using the completeness relation for the sum over empty states, Eq. (29) can be written in a closed form given by Eq. (30), which now does not involve the empty orbitals.

For Hartree–Fock calculations, we compute the matrix elements of bare exchange:

$$\begin{aligned}&\langle n\mathbf{k}|\Sigma_X|n'\mathbf{k}\rangle\\&=-\sum_{n''}^{\text{occ}}\sum_{\mathbf{q}GG'}M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}')\delta_{GG'}v(\mathbf{q}+\mathbf{G}').\end{aligned}\tag{31}$$

In principle, the inner and outer orbitals used in Eqs. (20)–(31) originate from the same mean-field solution. However, there is an option in the `sigma` executable to use a different mean-field solution for the inner and outer states. This is useful if one wishes to construct the $\Sigma$ operator within one mean field but expand the $\Sigma$ matrix using different orbitals, i.e., in order to evaluate matrix elements in a different basis than the mean-field wavefunctions when the quasiparticle wavefunctions are significantly different. This is also useful for verifying the accuracy of the linearization approximation as given by Eq. (32) below.

Eq. (19) depends on the evaluation energy parameter $E$. This parameter should be the quasiparticle energy $E^{QP}_{n\mathbf{k}}$, determined self-consistently. In principle, what one may do is start by setting $E=E^{MF}_{n\mathbf{k}}$ and find $E^0_{n\mathbf{k}}$ using Eq. (19). One can then set $E=E^0_{n\mathbf{k}}$ and solve Eq. (19), arriving at a new quasiparticle energy $E^1_{n\mathbf{k}}$. One can then repeat this process until convergence is reached. This process can be achieved using the different set of inner and outer states as described in the previous paragraph – where the outer-state eigenvalues are updated after each step, and the eigenfunctions are left unchanged. In many cases, one can avoid this process by computing $\Sigma(E)$ on a grid of energies and interpolating or extrapolating to $E^{QP}_{n\mathbf{k}}$. In particular, in many systems, $\Sigma(E)$ is a nearly

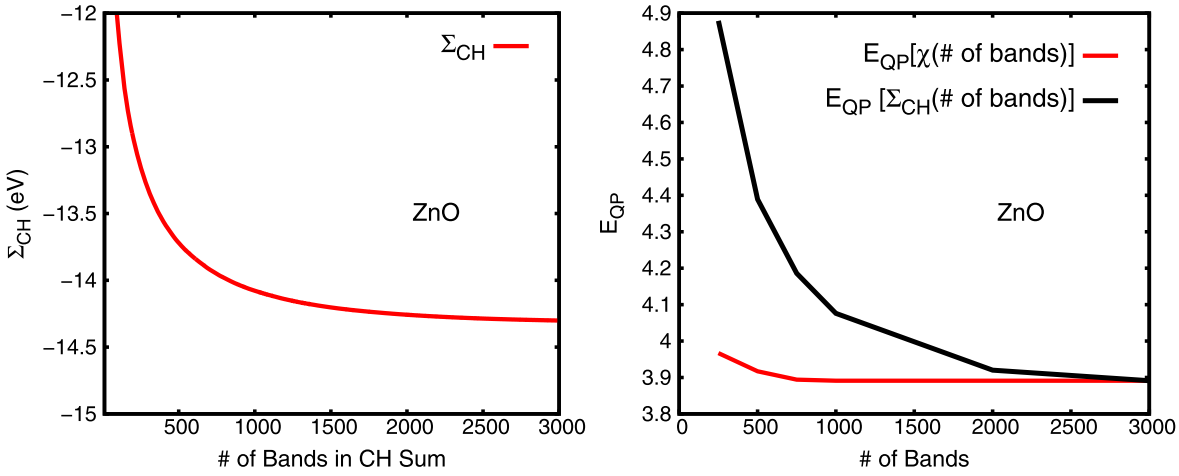


**Fig. 7.** ZnO convergence of the VBM within the Hybertsen and Louie GPP model. (Left panel) Example output from file `ch_convergence.dat` showing the Coulomb-hole sum value vs. the number of bands included in the sum. (Right panel) The convergence of $E_{QP}$ with respect to empty states in the polarizability sum, Eq. (8), and with respect to empty states in the Coulomb-hole sum, Eq. (23). The red curve shows the VBM $E_{QP}$ in ZnO using a fixed 3000 bands in the Coulomb-hole summation and varying the number of bands included in the polarizability summation. The black curve shows the VBM $E_{QP}$ in ZnO using a fixed 1000 bands in the polarizability summation and varying the number of bands included in the Coulomb-hole summation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

linear function of $E$ so one may compute $\Sigma(E)$ for two grid points and evaluate the self-consistent $E^{QP}_{n\mathbf{k}}$ using Newton's method [1]:

$$E^{QP}_{n\mathbf{k}} = E^{0}_{n\mathbf{k}} + \frac{d\Sigma/dE}{1 - d\Sigma/dE}\left(E^{0}_{n\mathbf{k}} - E^{MF}_{n\mathbf{k}}\right). \tag{32}$$

The derivative that appears here is also related to the quasiparticle renormalization factor:

$$Z = \frac{d\Sigma/dE}{1 - d\Sigma/dE}. \tag{33}$$

For full-frequency calculations, Eqs. (20) and (21) are evaluated on a frequency grid, $E$ (not to be confused with the frequency grid over which the integrals are carried out) specified by the user. One then has access directly to Re $\Sigma(\omega)$ and to Im $\Sigma(\omega)$, printed in the file `spectrum.dat`, which can be used to construct the spectral function:

$$A_{\mathbf{k}}(\omega) = \frac{1}{\pi} \cdot \sum_{n} \frac{|\operatorname{Im} \Sigma_{n\mathbf{k}}(\omega)|}{(\omega - E^{MF}_{n\mathbf{k}} - \operatorname{Re} \Sigma_{n\mathbf{k}}(\omega) + V^{n\mathbf{k}}_{xc})^2 + \operatorname{Im} \Sigma_{n\mathbf{k}}(\omega)^2}, \tag{34}$$

where we are using the mean-field exchange-correlation matrix element $V^{n\mathbf{k}}_{xc} = \langle n\mathbf{k}|V_{xc}|n\mathbf{k}\rangle$. This quantity can be used to compare directly with the quasiparticle spectrum from photo-emission experiments and various other measurements of the band-structure.

The plane-wave matrix elements required in Eqs. (20)–(31) are similar to those of Eq. (9) required for the construction of the irreducible polarizability matrix. In the current case, however, we require additional matrix elements between valence–valence band pairs as well as conduction–conduction band pairs. As was the case in the `epsilon` executable, the matrix elements are computed using FFTs utilizing the FFTW library [27]. For each pair of outer states, $n$ and $n'$, we sum over all occupied and unoccupied inner states, $n''$, included in the calculation (typically states of energy up to the dielectric energy cutoff). Therefore, the computational cost of computing all the necessary matrix elements scales as $N^2 \log N$, where $N$ is the number of atoms (a factor of $N \log N$ comes from the FFTs). If one is interested in all the diagonal matrix elements, Eq. (19), in a given energy range (as opposed to just a fixed small number of states – e.g. VBM and CBM) then an additional factor of $N$ is included in the scaling which becomes $N^3 \log N$. If one requires both diagonal and off-diagonal elements within a given energy window (such as in a self-consistent GW scheme), then the scaling becomes $N^4 \log N$.

Once the plane-wave matrix elements have been computed, the summations in the Coulomb-hole terms of Eqs. (20)–(31) for a particular $n$, $n'$ pair scale individually as $N^3$. Again, if all diagonal or off-diagonal matrix elements of $\Sigma$ in a given energy window must be computed an additional factor of $N$ or $N^2$ respectively is added to the scaling.

It is important to point out that the Coulomb-hole summations in terms of Eqs. (20)–(31) converge exceptionally slowly with respect to the number of empty states included in the sums. The highest empty state included should have energy of at least the dielectric energy cutoff. Additionally, the convergence of the sums in (20)–(31) should be tested with respect to the dielectric energy cutoff. As shown in Fig. 7, the convergence with respect to the dielectric energy cutoff, and the corresponding number of empty states, is very slow in many cases. This problem is similar to convergence issues with respect to empty states in the `epsilon` executable, as was discussed above. However, one finds that in many cases (particularly for bulk systems) the final $E^{QP}$ converges much more slowly with respect to the number of empty states in the Coulomb-hole expression than in the polarizability expression [31] – see Fig. 7 for a comparison of these two rates in ZnO when using the Hybertsen and Louie GPP model. The partial sums of the Coulomb-hole matrix elements with respect to number of states included in the sum is written to the file `ch_converge.dat`. Example output from `ch_converge.dat` is plotted in Fig. 7.

### 3.4. Optical properties: `BSE`

The optical properties of materials are computed in the Bethe–Salpeter equation (BSE) executables. Here the eigenvalue equation represented by the BSE, Eq. (3), is constructed and diagonalized yielding the excitation energies and wavefunctions of the correlated electron–hole excited states. There are two main executables: `kernel` and `absorption`. In the former, the electron–hole interaction kernel is constructed on a coarse **k**-point grid, and in the latter the kernel is (optionally) interpolated to a fine **k**-point grid and diagonalized.

The `kernel` executable constructs the second term of the left-hand side of Eq. (3) which is referred to as the electron–hole

kernel. The kernel, $K$, as implemented in the package, is limited to the static approximation, and contains two terms, a screened direct interaction and a bare exchange interaction, $K^{\mathrm{eh}} = K^{\mathrm{d}} + K^{\mathrm{x}}$, defined in the following way [8]:

$$\langle vc\mathbf{k}|K^{\mathrm{d}}|v'c'\mathbf{k}'\rangle = -\int d\mathbf{r}\,d\mathbf{r}'\,\psi_c^*(\mathbf{r})\psi_{c'}(\mathbf{r})W(\mathbf{r},\mathbf{r}')\psi_{v'}^*(\mathbf{r}')\psi_v(\mathbf{r}') \tag{35}$$

and

$$\langle vc\mathbf{k}|K^{\mathrm{x}}|v'c'\mathbf{k}'\rangle = \int d\mathbf{r}\,d\mathbf{r}'\,\psi_c^*(\mathbf{r})\psi_v(\mathbf{r})v(\mathbf{r},\mathbf{r}')\psi_{v'}^*(\mathbf{r}')\psi_{c'}(\mathbf{r}'). \tag{36}$$

These matrices are constructed on a coarse grid of $\mathbf{k}$-points, in most cases the same grid used within the GW calculation because one must have previously constructed the dielectric matrix $\epsilon^{-1}(\mathbf{q})$ for $\mathbf{q} = \mathbf{k} - \mathbf{k}'$. We calculate these matrices in $\mathbf{G}$-space using the prescription of Rohlfing and Louie [8]:

$$\langle vc\mathbf{k}|K^{\mathrm{d}}|v'c'\mathbf{k}'\rangle = \sum_{\mathbf{G}\mathbf{G}'} M_{cc'}(\mathbf{k},\mathbf{q},\mathbf{G})W_{\mathbf{G}\mathbf{G}'}(\mathbf{q};0)M_{vv'}^*(\mathbf{k},\mathbf{q},\mathbf{G}') \tag{37}$$

and

$$\langle vc\mathbf{k}|K^{\mathrm{x}}|v'c'\mathbf{k}'\rangle = \sum_{\mathbf{G}\mathbf{G}'} M_{cv}(\mathbf{k},\mathbf{q},\mathbf{G})v(\mathbf{q}+\mathbf{G})\delta_{GG'}M_{c'v'}^*(\mathbf{k},\mathbf{q},\mathbf{G}') \tag{38}$$

where $M$ is defined in Eq. (9) and calculated using FFTs as described above in Section 3.2.

For each $\mathbf{k}$ and $\mathbf{k}'$, we must therefore calculate all the matrix elements $M_{vv'}$, $M_{cc'}$, and $M_{vc}$. The number of valence and conduction bands required to calculate the absorption spectrum within a given energy window each scales linearly with the number of atoms $N$. So, formally we again have $N^3 \log N$ scaling with the use of FFTs. The summations involved in Eqs. (37) and (38), however, formally scale as $N^6$ since there are $N^4$ terms to compute and each involves a sum over $\mathbf{G}\mathbf{G}'$. In practice, though, except for the largest systems considered, the summations require less time than the matrix elements, and $N_v$ and $N_c$ remain small compared to the values required in the GW step, for example where states with energy up to the dielectric cutoff were required. Usually the energy window used in solving the BSE is approximately 10 eV, giving a spectrum converged beyond the visible region. As we discuss below, within the BerkeleyGW package, the parallel wall-time scales as $N^2$ for this step. However, the $N^6$ scaling will present a considerable challenge when applying the code to systems of size greater than 100s of atoms.

As was the case for GW code, the $\mathbf{q} \to 0$ limit must be handled carefully and differently depending on the type of screening in the system. For the exchange kernel, we zero out all $\mathbf{G} = \mathbf{G}' = 0$ contributions to the kernel matrix elements, as discussed in Ref. [37] which gives directly $\mathrm{Im}\,\epsilon_{\mathrm{M}}$ where $\epsilon_{\mathrm{M}}$ is the macroscopic dielectric constant. For the direct term, however, we must handle the $\mathbf{G} = 0$ case specially. For these purposes, the $\mathbf{G} = 0$ and $\mathbf{G}' = 0$ terms are removed from Eq. (37) and treated separately. For each $(\mathbf{k}cv, \mathbf{k}'c'v')$ we save three terms: the body term, which contains the result of the sum in Eq. (37) with the $\mathbf{G} = 0$ or $\mathbf{G}' = 0$ terms removed; the wing term, which contains all the sum of all the remaining terms in the sum with the exception of the single term where $\mathbf{G} = \mathbf{G}' = 0$; the head term, which contains the remaining term from the sum where $\mathbf{G} = \mathbf{G}' = 0$. For metallic systems, as we discussed above, $\epsilon^{-1}(\mathbf{q}, \mathbf{G} = \mathbf{G}' = 0) \propto 1/v(\mathbf{q})$ so that $W(\mathbf{q}, \mathbf{G} = \mathbf{G}' = 0) \propto C$, thus the head term in the kernel remains well behaved. For semiconductors however, $W(\mathbf{q}, \mathbf{G} = \mathbf{G}' = 0) \propto 1/q^2$ so that the head term in the kernel actually diverges as $1/q^2$ when $\mathbf{q} \to 0$. Similarly, the wing term diverges as $1/q$ when $\mathbf{q} \to 0$ for semiconductors, while it again remains well behaved for metals. These limits are summarized in Table 2.

Because exciton binding energies and absorption spectra depend sensitively on quantities like the joint density of states, it is essential in periodic systems to sample the $\mathbf{k}$-points on a very fine grid. Directly calculating the kernel on this fine grid in the `kernel` executable would be prohibitively expensive, so instead we interpolate the kernel in the `absorption` executable before diagonalization. For semiconductors, the head and wing kernel terms are not smooth functions of $\mathbf{k}$ and $\mathbf{k}'$ (as we have shown above, they diverge for $\mathbf{q} = \mathbf{k} - \mathbf{k}' \to 0$). Therefore, the quantities that we interpolate are $q^2 \cdot K^{\mathrm{d}}_{\mathrm{head}}$, $q \cdot K^{\mathrm{d}}_{\mathrm{wing}}$ and the body term directly as they are now smooth quantities [8]. For metals, we interpolate directly the kernel without any caveats because all the contributing terms are smooth functions of $\mathbf{k}$ and $\mathbf{k}'$. As in GW, we treat metals with zero-temperature occupations.

The `absorption` executable requires both coarse- and fine-grid wavefunctions as input. The interpolation is done through a simple expansion of the fine-grid wavefunction in terms of nearest coarse-grid wavefunction:

$$u_{n\mathbf{k}_{\mathrm{fi}}} = \sum_{n'} C^{\mathbf{k}_{\mathrm{co}}}_{n,n'} u_{n'\mathbf{k}_{\mathrm{co}}} \tag{39}$$

where $\mathbf{k}_{\mathrm{co}}$ is the closest coarse-grid point to the fine-grid point, $\mathbf{k}_{\mathrm{fi}}$, and the coefficients $C^{\mathbf{k}_{\mathrm{fi}}}_{n,n'}$ are defined as the overlaps between the coarse-grid and fine-grid wavefunctions:

$$C^{\mathbf{k}_{\mathrm{co}}}_{n,n'} = \int d\mathbf{r}\,u_{n\mathbf{k}_{\mathrm{fi}}}(\mathbf{r})u^*_{n'\mathbf{k}_{\mathrm{co}}}(\mathbf{r}). \tag{40}$$

The coefficients $C^{\mathbf{k}_{\mathrm{co}}}_{n,n'}$ are normalized so that $\sum_{n'} |C^{\mathbf{k}_{\mathrm{co}}}_{n,n'}|^2 = 1$. It should be noted that for a given set of fine bands one can improve the interpolation systematically by including more valence and conduction bands in the coarse grid due to the completeness of the Hilbert space at each $\mathbf{k}$. It should also be noted that we do restrict $n$ and $n'$ to be either both valence or both conduction bands – this is acceptable due to the different character of the conduction and valence bands in most systems.

Using these coefficients we interpolate the kernel with the following formula:

$$\langle vc\mathbf{k}_{\mathrm{fi}}|K|v'c'\mathbf{k}'_{\mathrm{fi}}\rangle = \sum_{n_1,n_2,n_3,n_4} C^{\mathbf{k}_{\mathrm{co}}}_{c,n_1} C^{*\mathbf{k}_{\mathrm{co}}}_{v,n_2} C^{*\mathbf{k}'_{\mathrm{co}}}_{c',n_3} C^{\mathbf{k}'_{\mathrm{co}}}_{v',n_4} \langle n_2 n_1 \mathbf{k}_{\mathrm{co}}|K|n_4 n_3 \mathbf{k}'_{\mathrm{co}}\rangle \tag{41}$$

where $K$ is one of the head, wing, body or exchange kernel terms. As in the case of `epsilon`, this summation can be performed compactly as a set of matrix–matrix multiplications. We utilize the Level 3 BLAS calls `DGEMM` and `ZGEMM` to optimize the performance.

One can improve on the interpolation systematically by using the closest four coarse-grid points to each fine point and using a linear interpolation layer in addition to the wavefunction-based interpolation described above. This is done by default for the interpolation of the first term of Eq. (3) for the quasiparticle self-energy corrections $E^{\mathrm{QP}} - E^{\mathrm{MF}}$:

$$E^{\mathrm{QP}}_n(\mathbf{k}_{\mathrm{fi}}) = E^{\mathrm{MF}}_n(\mathbf{k}_{\mathrm{fi}}) + \left\langle \sum_{n'} \left|C^{\mathbf{k}_{\mathrm{co}}}_{n,n'}\right|^2 \left(E^{\mathrm{QP}}_{n'}(\mathbf{k}_{\mathrm{co}}) - E^{\mathrm{MF}}_{n'}(\mathbf{k}_{\mathrm{co}})\right)\right\rangle_{\mathbf{k}_{\mathrm{co}}} \tag{42}$$

where the brackets indicate linear interpolation using the tetrahedron method. In this case, the wavefunction-based interpolation

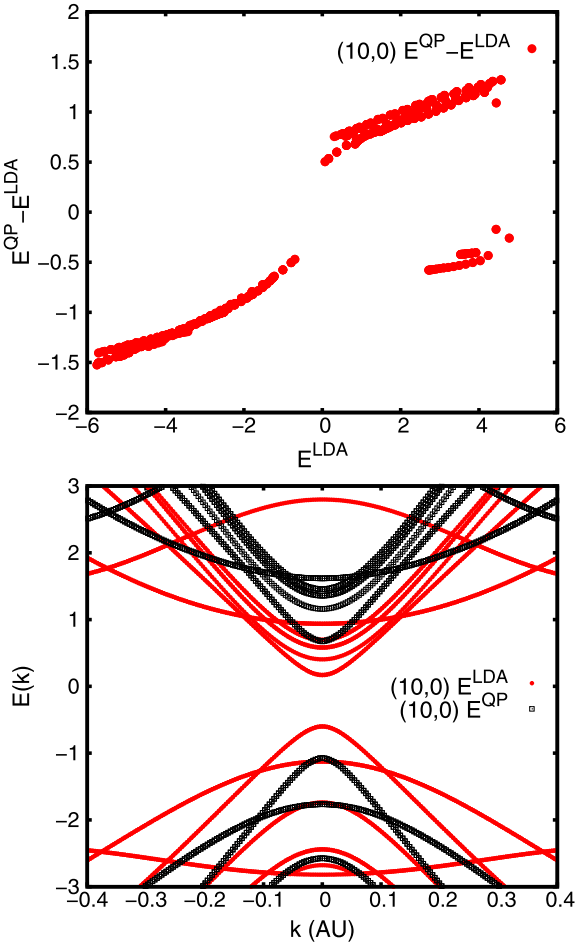


**Fig. 8.** Top: GW quasiparticle self-energy corrections, $E^{\mathrm{QP}} - E^{\mathrm{LDA}}$ vs. the LDA energy for (10, 0) SWCNT. Both a rigid opening of the band gap and a non-linear energy scaling are present. Bottom: The fine-grid quasiparticle band-structure using the interpolated self-energy corrections (thick black open) and the LDA uninterpolated band-structure (thin red). 256 points are used to sample the Brillouin zone. in the Coulomb-hole summation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

layer guarantees that the band crossings are properly handled, and the linear interpolation layer ensures that we correctly capture the energy dependence of the self-energy corrections. In this way, we can construct $E^{\mathrm{QP}}$ on the fine grid, or any arbitrary point, given $E^{\mathrm{MF}}$ on the fine grid and $E^{\mathrm{QP}}$ and $E^{\mathrm{MF}}$ on the coarse grid (Fig. 8).

As an alternative to calculating the quasiparticle corrections on the coarse grid and interpolating them to the fine grid, the user may choose a less refined method of specifying the corrections using a three-parameter model involving a scissor-shift parameter $\Delta E$ to open the energy gap at the Fermi energy, a zero energy $E_0$ (typically the band edge), and an energy-scaling parameter $C$ changing the bandwidth (the parameters are specified separately for valence and conduction bands):

$$E^{\mathrm{QP}} = E^{\mathrm{MF}} + \Delta E + C\left(E^{\mathrm{MF}} - E_0\right). \tag{43}$$

Having constructed the kernel on the fine grid, we now consider the diagonalization of the kernel. The kernel matrix is of dimension $N_c \cdot N_v \cdot N_k$ where $N_k$ is the number of **k**-points on the fine grid. Formally, the kernel dimension scales as $N$ for periodic systems with small unit cells and $N^2$ for large systems, where $N$ is the number of atoms. For bulk systems with small unit cells, $N_k \propto 1/N$, but the reduction with increased cell size saturates quickly for large systems and large molecules where $N_k = O(1)$ (to compute a smooth continuum absorption onset, it is necessary to include some level of **k**-point sampling even for isolated systems). The matrix can be diagonalized exactly within LAPACK (`zheevx`) or ScaLAPACK (`pzheevx`). The diagonalization therefore scales as $N^3$ for periodic systems with small unit cells and $N^6$ for large systems.

The result of the diagonalization is the set of exciton eigenvalues $\Omega^S$ and eigenfunctions $A^S_{cv\mathbf{k}}$ which can be used to construct the joint density of states (JDOS) or the absorption spectrum (or $\mathrm{Im}\,\epsilon_2(\omega)$) using Eq. (5). There are a number of post-processing tools in the package, such as `PlotXct`, which plots the exciton wavefunction in real space according to Eq. (4) to analyze the exciton states. The `absorption` executable can produce both the singlet and triplet eigenvalues and eigenvectors. In the latter case, the exchange term is set to zero when diagonalizing the kernel [8]. (Note that for triplets the oscillator strengths calculated in the code are evaluated without considering spin overlap. In some cases this "oscillator strength of the corresponding singlet" may be useful. The true physical oscillator strength of course is zero for triplets.) Additionally, one may compute the eigenvalues and eigenvectors with only the exchange interaction; the resulting spectrum should be the same as the one obtained within RPA with local-field effects included [37].

The $N^6$ scaling for large systems in the diagonalization is, in practice, more limiting than the $N^6$ step in the construction of the kernel. This is because the latter step can be parallelized very efficiently, while diagonalization, even with the use of ScaLAPACK, typically saturates at $O(1000)$ CPUs. Often one is only interested in the absorption spectrum or the JDOS, and not all of the correlated exciton eigenfunctions and eigenvalues. For such systems, we use the Haydock recursion method [38,39]. This is an iterative method based on spectral decomposition and requires only matrix–vector products, which can be parallelized efficiently. This method gives the absorption spectrum directly, the equivalent of Eq. (5). In principle, one can get eigenvalues and eigenvectors for a small energy range of interest using iterative Lanczos algorithms [39].

As mentioned above, the electron–hole kernel should be constructed with a sufficient number of valence and conduction bands to cover the energy window of interest – typically all bands within the desired energy window from the Fermi energy should be included so that the energy window of the bands included in the calculation is at least twice that of the desired absorption energy window. The `absorption` executable computes the percent deviation from the $f$-sum rule [1]:

$$\int_0^\infty \epsilon_2(\omega)\omega\, d\omega = -\frac{\pi\omega_{\mathrm{p}}^2}{2}. \tag{44}$$

One should converge this quantity with both the number of valence and conduction bands included. The absorption spectrum (or $\epsilon_2$) in the energy window of interest converges much more quickly than $\epsilon_1$ if high-energy transitions outside of the window of interest contribute greatly to the sum rule, since $\epsilon_1(\omega)$ is related to an integration over all frequencies of $\epsilon_2(\omega)$ via the Kramers–Kronig relation.

Finally, the transition matrix elements in Eq. (6) are printed in the file `eigenvalues.dat`. (These are related to the oscillator strengths.) The matrix elements may be computed in two ways. First, exactly, using the velocity operator:

$$\begin{aligned}\langle v\mathbf{k}|\mathbf{v}|c\mathbf{k}\rangle &= \langle v\mathbf{k}|i[H,\mathbf{r}]|c\mathbf{k}\rangle = i(E_v - E_c)\langle v\mathbf{k}|\mathbf{r}|c\mathbf{k}\rangle \\ &\approx i(E_v - E_c)\lim_{\mathbf{q}\to 0}\frac{\langle v\mathbf{k}+q|e^{i\mathbf{q}\cdot\mathbf{r}} - 1|c\mathbf{k}\rangle}{iq} \\ &= (E_v - E_c)\lim_{\mathbf{q}\to 0}\frac{\langle v\mathbf{k}+q|e^{i\mathbf{q}\cdot\mathbf{r}}|c\mathbf{k}\rangle}{q},\end{aligned} \tag{45}$$

where **q** is along the direction of the polarization of light. In practice, we evaluate the limit using finite differences for a small value of **q**, similarly to how the limit is treated in the `Epsilon` code. As an alternative to the finite-difference approach, one may approximate the velocity operator by the momentum operator $-i\nabla$. This yields inexact oscillator strengths due to neglect of commutators between **r** and the non-local part of the Hamiltonian [8,40]. The exciton energies do not involve the velocity or momentum operator.

## 4. Parallelization and performance

### 4.1. `epsilon`

The parallelization of `epsilon` is characterized by two distinct schemes for the two main sections of the code: (1) the computation of matrix elements (Eq. (9)), and (2) the matrix multiplication (Eq. (15)) and inversion.

For the computation of the matrix elements in Eq. (9), the code is parallelized with nearly linear scaling up to $N_v \cdot N_c$ processors, where $N_v$ and $N_c$ are the number of valence and conduction bands respectively used in the sum of Eq. (8). Each processor owns an approximately equal fraction of the total number of $(v, c)$ pairs for all **k**, and performs serial FFTs to compute the matrix elements, Eq. (9), for all **G** and **k** associated with the pair. Note that for large systems, $N_v$ is on the order of 100s and $N_c$ is on the order of 1000s or more, so that this section of the code scales well up to 100 000 CPUs.

All wavefunctions are stored in memory unless the optional `comm_disk` flag is given. Each processor holds in memory the wavefunctions for all the pairs it owns. If `comm_disk` is specified (as opposed to the default `comm_mpi` option), the distribution of pairs is the same, but each processor saves the conduction wavefunctions it needs on disk and reads the wavefunctions back into memory one pair at time for the purposes of computation. Using `comm_disk` can therefore reduce the amount of memory required for the computation, but comes with a substantial performance reduction.

The processors are distributed into valence and conduction band pools in order to minimize the memory required using a complete search algorithm. For example, if one calculates a material with few valence and many conduction bands, all the processors will be in one or two valence pools (holding all or half the valence bands in memory each) but spread over a large number conduction pools because the relative cost of holding all the valence bands in memory is much smaller than holding all the conduction bands in memory. In such a scheme, the amount of memory required per processor drops linearly with small numbers of processors and decreases as $1/\sqrt{N_{\text{proc}}}$ for large numbers of processors (Fig. 9).

In the second section of the `epsilon` code, we switch from a parallelization over bands to a parallelization scheme over $\mathbf{G}\mathbf{G}'$ for the polarizability (Eq. (8)) and dielectric matrices (Eq. (10)). We use the ScaLAPACK block-cyclic layout [41] in anticipation of utilizing the ScaLAPACK libraries for the inversion of the dielectric matrix. The transition between the band distribution of the matrix elements in Eq. (9) and the block-cyclic layout of the polarizability matrix is achieved naturally in the process of doing the parallel matrix–matrix multiplication involved in Eq. (15). There is, however, a significant amount of communication involved at this step. To minimize this communication we have two options for the parallel multiplication.

In the first scheme, corresponding to the use of the `gcomm_matrix` flag in `epsilon.inp`, we loop over processors (for simplicity, we label the loop index $i$) who own a piece of the polarizability matrix $\chi(\mathbf{G}, \mathbf{G}')$. Each processor does the fraction of the matrix multiplication in Eq. (15) relating to the $(n, n')$ pairs it owns and for submatrix $\chi(\mathbf{G}_i, \mathbf{G}'_i)$ that the $i$th processor stores. The processors then MPI-reduce their contribution to the $i$th processor. Thus the total communication in this scheme is an eventual reduction of the entire $\chi(\mathbf{G}, \mathbf{G}')$ to the processors that store it. It is important to note that we must do this one processor at time (or at most in chunks of processors – chosen often as the number of CPUs per node) because no single processor can hold in memory the entire $\chi(\mathbf{G}, \mathbf{G}')$ matrix for large systems.

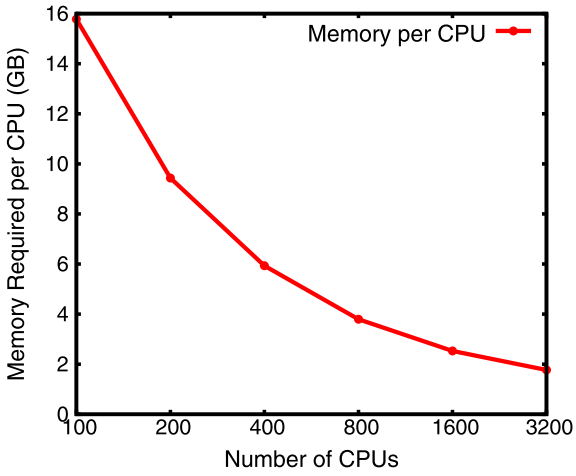


**Fig. 9.** The memory required per CPU vs. the number of CPUs used for an `epsilon` calculation on the (20, 20) nanotube. See text for parameters used.

In the second scheme, corresponding to the use of the `gcomm_elements` flag in `epsilon.inp`, we again loop over processors, but this time, we have the $i$th processor MPI-broadcast to all processors that hold a piece of the polarizability matrix the set of matrix elements for all the $(v, c)$ pairs it owns. Each processor then uses these matrix elements to compute the contribution of the matrix–matrix product, Eq. (15), for the submatrix of $\chi(\mathbf{G}, \mathbf{G}')$ it stores. In this scheme, all the matrix elements (Eq. (9)) are eventually broadcast.

Whether the use of `gcomm_elements` or `gcomm_matrix` is optimal depends on whether it is faster to reduce the $\chi(\mathbf{G}, \mathbf{G}'; \omega)$ or broadcast all the matrix elements, $M_{nn'}(\mathbf{k}, \mathbf{q}, \{\mathbf{G}\})$. In particular if $N_{\mathbf{G}} \cdot N_{\text{freq}} < N_v \cdot N_c \cdot N_k$, where $N_{\text{freq}}$ is the number of frequencies in a full frequency calculation, then it is cheaper to use `gcomm_matrix`. If no flag is specified, the `epsilon` code will make this choice for the user based on the above criteria.

Because we use the block-cyclic layout, the memory required to store the $\chi(\mathbf{G}, \mathbf{G}')$ decreases linearly with the number of CPUs. However, the cost of the inversion utilizing ScaLAPACK can saturate at 100s of CPUs and the cost of the summation can saturate with a few thousand CPUs, see Fig. 10. In general, the number of CPUs used for the block-cyclic distribution of $\chi$ can be tuned.

Beyond the more sophisticated level of parallelization described above, there is a more trivial level of parallelization available to small systems requiring large numbers of **k**-points: Eq. (8) is completely separable as a function of **q**. One may run a separate `epsilon` calculation for each **q** required and merge the dielectric matrices – in such a way, a user can obtain perfectly linear artificial scaling with CPUs to $N_k$ times the number CPUs mentioned above.

The scaling of memory and computation time with respect to the number of CPUs used per **q**-point in `epsilon` for the example (20, 20) SWCNT calculation is shown in Figs. 9 and 10. We find nearly linear scaling up to 3200 CPUs per **q**-point. Since there are 32 **q**-points in this calculation that are trivially parallelized, we find nearly linear scaling of the `epsilon` computation up to $\sim$ 100 000 CPUs.

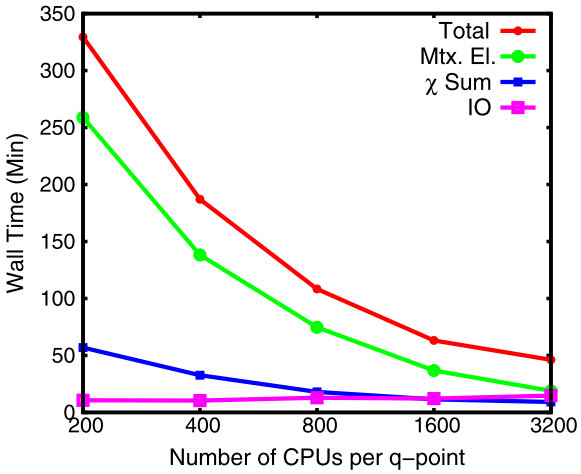


**Fig. 10.** The wall-time required vs. the number of CPUs per **q**-point used for a `epsilon` calculation on the (20, 20) single-walled carbon nanotube. There is near linear scaling up to 1600 CPUs. Since there is an additional layer of trivial parallelization over the 32 **q**-points required, the `epsilon` calculation scales to over 50 000 CPUs. See text for parameters used.

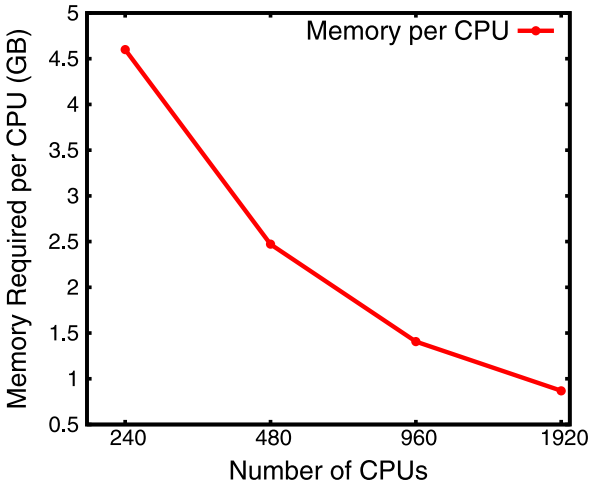


**Fig. 11.** The memory required per CPU vs. the number of CPUs used for a `sigma` calculation on the (20, 20) nanotube. See text for parameters used.

### 4.2. `sigma`

Within a `sigma` calculation, one computes a requested number of diagonal, Eq. (19), or off-diagonal, Eq. (18), $\Sigma$ matrix elements. For each matrix element there are two computationally intensive steps. The first is to calculate all the plane-wave matrix elements $M_{nn''}$ and $M_{n'n''}$, Eq. (9), for the outer states of interest, $n$ and $n'$, and for all occupied and empty states, $n''$. Secondly, we compute the sum over states, $n''$, as well as **G**, **G**′ and **q** in the expressions in Eqs. (20)–(31).

The `sigma` execution is parallelized over both outer bands, $n$ and $n'$, and inner bands, $n''$. As in the case of `epsilon`, this is done by defining pools and distributing the $n$, $n'$ pairs evenly among the pools. We then distribute the $n''$ bands evenly within the pools. As was the case for `epsilon`, we define the number of pools using a complete search algorithm to minimize the amount of memory per CPU required to store the inner and outer wavefunctions.

As described above, the CPU time required for the computation of all plane-wave matrix elements, $M_{nn''}$ and $M_{n'n''}$, scales as $N^2 \log N$, where $N$ is the number of atoms, for each $\Sigma$ matrix element of interest. As described above, the outer-state pairs are parallelized over pools and the inner states are parallelized over the CPUs within each pool. The wall-time for the computation of all the plane-wave matrix elements required for every $\Sigma$ matrix element scales as $N \log N$ with unlimited CPU resources. As was the case in the `epsilon` executable, each CPU computes the plane-wave matrix elements between all the $(n, n'')$ and $(n', n'')$ pairs it owns for all **G** through serial FFTs using FFTW [27].

The summations required in Eqs. (20)–(31) are parallelized by again distributing the outer-state pairs over the pools and then distributing the inner states over the CPUs within each pool. The wall-time for the summations, therefore, scales as $N^2$ (for the sums over **G** and **G**′) regardless of the number of diagonal or off-diagonal elements requested, given unlimited CPU resources.

As was the case for `epsilon`, the wavefunctions are distributed in memory, with each CPU owning only the $n$, $n'$ and $n''$ wavefunctions that it needs for the computations described above. The dielectric matrix, $\epsilon^{-1}_{\mathbf{G},\mathbf{G}'}(\mathbf{q}; E)$ for each **q** and $E$, is distributed globally over the matrix rows, **G**.

The scaling of memory and computation time with respect to the number of CPUs used per **k**-point in `sigma` for the example (20, 20) SWCNT calculation is shown in Figs. 11 and 12. We find nearly linear scaling up to 1600 CPUs per **k**-point. Since there are 16 irreducible **k**-points in this calculation that are trivially parallelized, we find nearly linear scaling of the `sigma` computation up to 25 000 CPUs.

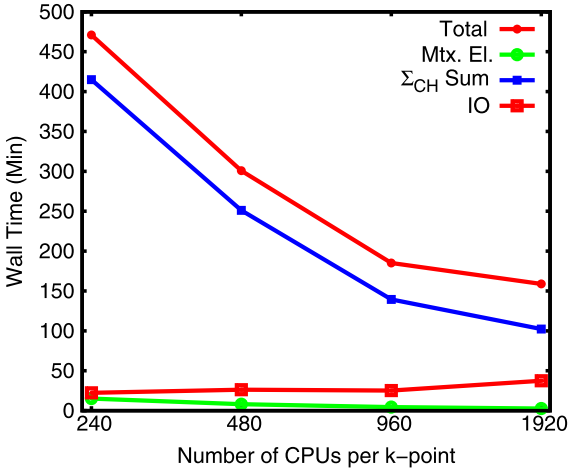


**Fig. 12.** The wall-time required vs. the number of CPUs per **k**-point used for a `sigma` calculation on the (20, 20) single-walled carbon nanotube. There is near linear scaling up to 1920 CPUs. Since there is an additional layer of trivial parallelization over the 16 **k**-points required, the `sigma` calculation scales to over 30 000 CPUs. See text for parameters used.

### 4.3. `BSE`

As mentioned in the previous sections, in the `kernel` executable, for each **k** and **k**′, we must calculate all the matrix elements $M_{vv'}$, $M_{cc'}$, and $M_{vc}$ and then perform the summations involved Eqs. (37) and (38) for each $(vc\mathbf{k}, v'c'\mathbf{k}')$ pair. BerkeleyGW automatically parallelizes this in different schemes depending on the system and number of CPUs provided.

If the number of CPUs is less than $N_k^2$, the square of the number of coarse **k**-points, we distribute the $(\mathbf{k}, \mathbf{k}')$ pairs evenly over the CPUs and each CPU calculates all the matrix elements, $M_{vv'}$, $M_{cc'}$, and $M_{vc}$, required for the **k**-point pairs it owns through serial FFTs. It then computes the sums in Eqs. (37) and (38) for all of its pairs.

If the number of CPUs is greater than $N_k^2$, as is often the case for large systems and molecules, but less than $N_k^2 \cdot N_c^2$, we distribute the $(c\mathbf{k}, c'\mathbf{k}')$ pairs evenly among the processors, first distributing the processors evenly over **k**-point pairs and then cre-

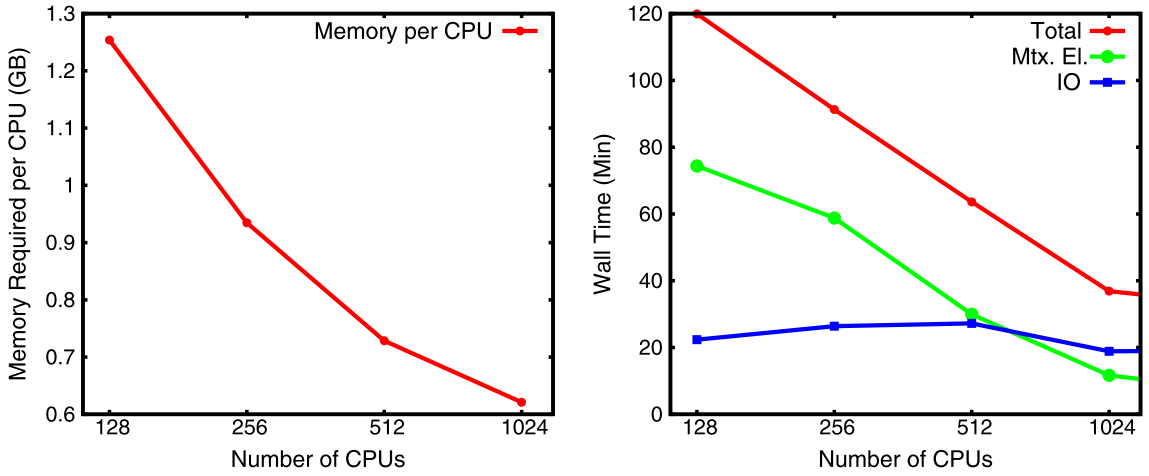


**Fig. 13.** (Left) Memory per CPU required vs. the number of CPUs for a `kernel` calculation on the (20, 20) SWCNT. (Right) The wall-time required vs. the number of CPUs used for a `kernel` calculation on the (20, 20) SWCNT. The parameters used are described in the text.

ating pools to distribute the $(c, c')$ evenly among the pools. In this scheme, each CPU computes $M_{cc'}$ for only the $(c\mathbf{k}, c'\mathbf{k}')$ it owns but computes $M_{vv'}$ and $M_{vc}$ for all $v$ and $v'$ at each $(c\mathbf{k}, c'\mathbf{k}')$ pairs it owns. Each CPU does the summations in Eqs. (37) and (38) for all $v$, $v'$ for the $(c\mathbf{k}, c'\mathbf{k}')$ it owns.

If the number of CPUs is greater than $N_k^2 \cdot N_c^2$, we distribute the entire set of $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs out evenly among the processors, first distributing the processors evenly over $(c\mathbf{k}, c'\mathbf{k}')$ pairs and then creating pools to distribute the $(v, v')$ evenly among the pools. In this scheme, each CPU computes only the $M_{vv'}$, $M_{cc'}$, and $M_{vc}$ for the $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs it owns. Additionally, each CPU does the summations in Eqs. (37) and (38) for only the pairs it owns.

In the last scheme the calculation of the matrix elements has a parallel wall-time scaling of $N$ and the summations scale as $N^2$ (accounting for the sum over $\mathbf{GG}'$) if a limitless number of CPU resources is assumed.

The large arrays that must be stored in memory are the dielectric matrix, the wavefunctions and the computed kernel itself. The computed kernel is distributed evenly in memory among the processors and computed directly by the CPUs which own the various $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs. The dielectric matrix is distributed, as in the `sigma` executable, over its rows $\mathbf{G}$ and must be broadcast during each calculation of the sums in Eqs. (37) and (38). It is for the purposes of minimizing the communication of the dielectric matrix that we use the three different parallelization schemes above – i.e. so that we might work on the biggest blocks of $(v, v')$ and $(c, c')$ at once. For example, in the first scheme, where the number of CPUs is less than $N_k^2$, we do the sums in Eqs. (37) and (38) for all $(vc, v'c')$ at once, so that we need only broadcast the dielectric matrix one time.

The wall-time scaling for the example `kernel` (20, 20) SWCNT calculation is shown in Fig. 13. We see nearly linear scaling up to 1024 CPUs – which is the square of the number of $\mathbf{k}$-points, 32.

In the `absorption` executable, as described above, the first computational challenge is the interpolation of the kernel from the coarse $\mathbf{k}$-grid onto the fine $\mathbf{k}$-grid. The computation of the interpolation coefficients, Eq. (40), is done by distributing the fine-grid $\mathbf{k}$-points evenly among the processors. For molecules or other large systems, this does not represent a problem because the computation of the coefficients is very quick in these cases regardless of the lack of parallelization.

The parallelization of the kernel interpolation is different from the parallelization scheme in the `kernel` executable. However, the parallelization is also described by three different schemes:

First, if the number of CPUs is less than $N_k$, where $N_k$ is the number of fine-grid $\mathbf{k}$-points, we distribute the $N_k$ $\mathbf{k}$-vectors on the fine grid evenly among the CPUs. Each processor owns all the $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs consistent with the $\mathbf{k}$-vectors it was assigned. It then performs the interpolation in Eq. (41) serially by replacing the simple loops that represent the sums with four matrix–matrix multiplications. For example, for each $n_1$ and $n_3$, one can write the sum over $n_2$ as a matrix–matrix product between the coarse kernel matrix (whose outer dimension is $n_4$) and the $[C^{\mathbf{k}_{\mathrm{co}}}_{v,n_2}]^*$ matrix (whose outer dimension is $N_v$ on the fine grid). We write the remaining sums as a similar matrix–matrix product.

If the number of CPUs is greater than $N_k$ but less than $N_k \cdot N_c$, we distribute the $N_k \cdot N_c$ $\mathbf{k}$-point and conduction-band pairs evenly among the processors. Again, each processor owns all the $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs consistent with the $(\mathbf{k}, c)$ it was assigned. In the previous scheme, we utilized matrix–matrix products to interpolate many kernel elements at once. This required the allocation of an array of size $N_v^2 \cdot N_c^2$ as described above. In the present scheme, we avoid storing large intermediate arrays by doing more of the sums in Eq. (41) as simple loops rather than matrix products. By default we do two simple loops of two matrix products. However, if the user selects the `low_memory` option, we do all four summations as four nested loops without the aid of matrix–matrix multiplications, obtaining one fine-grid matrix element at a time without the need for any intermediate matrices.

If the number of CPUs is greater than $N_k \cdot N_c$, we distribute the $N_k \cdot N_c \cdot N_v$ $\mathbf{k}$-point and conduction- and valence-band pairs evenly among the processors. Each processor owns all the $(vc\mathbf{k}, v'c'\mathbf{k}')$ pairs consistent with the $(\mathbf{k}, c, v)$ it was assigned. The interpolation is done exactly as in the previous case.

Once the fine-grid kernel has been constructed, in the `absorption` executable, the matrix is diagonalized using ScaLAPACK with a block-cyclic layout [41]. This diagonalization scales well to $O(1000)$ CPUs but saturates quickly beyond this point. In order to calculate the absorption spectrum as per Eq. (5), we compute all the necessary matrix elements, Eq. (6), by distributing the fine-grid $\mathbf{k}$-points evenly over processors. In order to diagonalize large matrices (e.g., graphene on a $256 \times 256$ $\mathbf{k}$-point grid [42]), we turn to iterative methods, in particular Haydock recursion [38,39]. Since it requires only matrix–vector products, this method scales well to larger numbers of processors. It should be pointed out that the kernel matrix is not sparse in general, so methods designed for the diagonalization of sparse matrices are not appropriate here.

## 5. Coulomb interaction

The bare Coulomb interaction is used in many places throughout the code. In all cases, the 1D array $v(\mathbf{q} + \mathbf{G})$ is generated from

a single `vcoul_generator` routine in the `Common` directory. However, there is a lot that can be specified about the Coulomb interaction in the code. The Coulomb interaction can be truncated to eliminate the spurious interaction between periodic images of nano-systems in a super-cell calculation. One can implement a cell-averaging technique whereby the value of the interaction at each $\mathbf{q}$-point (or $\mathbf{q} \to 0$ in particular) can be replaced by the average of $v(\mathbf{q}+\mathbf{G})$ in the volume the $\mathbf{q}$-point represents. Finally, this average can be made to include the $\mathbf{q}$-dependence of the inverse dielectric function also if $W$ is the final quantity of relevance for the application – such as in the evaluation of $W$ for the self-energy.

The BerkeleyGW package contains five general choices for the Coulomb interaction. Firstly, one may choose to use the bulk, untruncated value expressed in Eq. (11). There are in addition 4 choices of Coulomb interaction that truncate the interaction beyond a certain cutoff in real space, of generic form

$$v_t(\mathbf{r}) = \frac{\Theta(f(\mathbf{r}))}{r} \tag{46}$$

where $f$ is some function that describes the geometry in which the interaction is truncated. The four choices available implement the methods of Ismail-Beigi [43]: Wigner–Seitz slab truncation, Wigner–Seitz wire truncation, Wigner–Seitz box truncation, and spherical truncation. The Wigner–Seitz box truncation and the spherical truncation truncate the Coulomb interaction in all three spatial directions, yielding a finite value of $v_t(\mathbf{q}=0)$. All the Wigner–Seitz truncation schemes truncate the interaction at the edges of the Wigner–Seitz cell in the non-periodic directions. Slab truncation is intended for nano-systems with slab-like geometry. The Coulomb interaction is truncated at the edges of the unit cell in the direction ($z$) perpendicular to the slab plane ($xy$). Wire truncation is intended for nano-systems with wire-like geometry. The Coulomb interaction is truncated at the edges of the unit cell in the two directions ($xy$) perpendicular to the wire axis ($z$). Spherical truncation allows the user to specify manually a spherical truncation radius outside of which the Coulomb interaction will be truncated.

Like the untruncated interaction, the slab-truncation and spherical-truncation schemes have the benefit that $v_t(\mathbf{q}+\mathbf{G})$ can be constructed analytically:

$$v_t^{\text{sph}}(\mathbf{q}) = \frac{4\pi}{q^2} \cdot \left(1 - \cos(r_c \cdot q)\right), \tag{47}$$

$$v_t^{\text{slab}}(\mathbf{q}) = \frac{4\pi}{q^2} \cdot \left(1 - e^{-q_{xy} \cdot z_c} \cos(q_z \cdot z_c)\right) \tag{48}$$

where $r_c$ and $z_c$ are the truncation distances in the radial and perpendicular directions, respectively. The wire-truncation and box-truncation on the other hand are computed numerically through the use of FFTs. First, the truncated interaction, ($2K_0(|q_z|\rho)$ for wire geometry where $\rho = \sqrt{x^2+y^2}$ and $K_0$ is the modified Bessel function and $1/r$ for box geometry), is constructed on a real-space grid in the Wigner–Seitz cell and folded into the traditional unit cell. The real-space grid is typically more dense than the charge density grid used in DFT calculations. The density of points on the real-space grid relative to the charge density grid in $xy$ directions for wire-truncation and in $xyz$ directions for box-truncation is set to be a factor of 4 and 2, respectively. The origin of the Coulomb potential is offset from the origin of the coordinate system by half a grid step to avoid the singularity. We then FFT to yield the $v_t(\mathbf{q})$ directly. The FFT is done in parallel. For wire-truncation, $xy$-planes are evenly distributed among processors and each processor performs 2D-FFTs in $xy$-planes it owns. For box-truncation, the parallel 3D-FFT is performed as follows: $xy$-planes are distributed and 2D-FFTs in $xy$-planes are carried out the same way as for wire-truncation; the data is then transferred from $xy$-planes to $z$-rods which also are evenly distributed among processors; finally, each processor performs 1D-FFTs in $z$-rods it owns. After the FFT, the origin of the Coulomb potential is shifted back to the origin of the coordinate system by multiplying $v_t(\mathbf{G})$ with $\exp(i2\pi\mathbf{G}\cdot\frac{1}{2}\mathbf{d})$ where $\mathbf{d}$ is the real-space grid displacement vector.

**Table 2**
Top: $\mathbf{q} \to \mathbf{0}$ limits of the head, $\epsilon^{-1}_{\mathbf{00}}(\mathbf{q})$, wing, $\epsilon^{-1}_{\mathbf{G0}}(\mathbf{q})$ and wing', $\epsilon^{-1}_{\mathbf{0G'}}(\mathbf{q})$, of the inverse dielectric matrix. Bottom: $\mathbf{q} \to \mathbf{0}$ limits of the head and wings of the screened Coulomb interaction, $W_{\mathbf{GG'}}(\mathbf{q})$.

| $\epsilon^{-1}_{\mathbf{GG'}}$ | Semiconductor | Semiconductor truncation | Metal | Metal truncation |
|---|---|---|---|---|
| Head | Constant | 1 | $q^2$ | $\frac{1}{V_0^T(q)}$ |
| Wing | $q$ | $q\epsilon^{-1}_{00q}$ | $q^2$ | $\frac{1}{V_0^T(q)}$ |
| Wing' | $\frac{1}{q}$ | $q\epsilon^{-1}_{00q}V_0^T(q)$ | Constant | Constant |

| $W_{\mathbf{GG'}}$ | Semiconductor | Semiconductor truncation | Metal | Metal truncation |
|---|---|---|---|---|
| Head | $\frac{1}{q^2}$ | $V_0^T(q)\epsilon^{-1}_{00q}$ | Constant | Constant |
| Wing | $\frac{1}{q}$ | $q\epsilon^{-1}_{00q}V_0^T(q)$ | Constant | Constant |
| Wing' | $\frac{1}{q}$ | $q\epsilon^{-1}_{00q}V_0^T(q)$ | Constant | Constant |

As mentioned above, for all interaction choices with the exception of cell-box and spherical truncation, the Coulomb interaction diverges as $\mathbf{q} \to 0$. For the case of no truncation, $v(\mathbf{q}\to 0) \propto 1/q^2$; for slab truncation, $v_t^{\text{slab}}(\mathbf{q}\to 0) \propto 1/q$; for wire truncation, $v_t^{\text{wire}}(\mathbf{q}\to 0) \propto -\ln(q)$. As we mentioned in Section 3.2, this divergence is handled in `epsilon` by taking a numerical limit – that is, evaluating $\epsilon$ at a small but finite $\mathbf{q}_0$. For `sigma` and `absorption`, on the other hand, we are interested in evaluating directly $W(\mathbf{q},\mathbf{GG'})$ matrix elements and the appropriate treatment is to replace the divergent (for non-metals) $W(\mathbf{q}\to 0)$ with an average over the volume in reciprocal space that $\mathbf{q}=0$ represents:

$$W^{\text{avg}}\left(\mathbf{q}=0,\mathbf{GG'}=0\right) = \frac{N_{\mathbf{q}}\cdot \text{Vol}}{2\pi}\int_{\text{cell}} d^n q\, W(\mathbf{q}) \tag{49}$$

where "cell" represents the volume in reciprocal space closer to $\mathbf{q}=0$ than any other $\mathbf{q}$-points, and $d^n q$ represents the appropriate dimensional differential for the truncation scheme (e.g., $d^2q$ for slab truncation). Eq. (49) yields a finite number in all truncation schemes even when $W(\mathbf{q}\to 0)$ itself is divergent because of the reduced phase-space around $\mathbf{q}\to 0$.

For metallic systems, it is particularly important to use Eq. (49) to average $W$, as opposed to averaging $v(\mathbf{q})$ and $\epsilon^{-1}(\mathbf{q})$ separately, since for metals $\epsilon(\mathbf{q}) \propto v(\mathbf{q})$ at small $\mathbf{q}$, yielding a constant limit: $W^{\text{metal}}(\mathbf{q}\to 0) = C$. The user can tell the code which model to use for the $\mathbf{q}$-dependence of $\epsilon^{-1}$ by specifying one of the screening flags in the input files. The $\mathbf{q}\to\mathbf{0}$ limits for the inverse dielectric function and screened Coulomb interaction are enumerated in Table 2 for the semiconductor and metallic screening types.

As shown in Fig. 14, including $\epsilon^{-1}$ in the cell-averaging scheme also is important when using a truncated interaction where $\epsilon(\mathbf{q}=0)=1$ but quickly rises by the first non-zero $\mathbf{q}$-point [43]. The figure shows $\epsilon^{-1}(q)$ along the tube axis in the $(14,0)$ SWCNT. $\epsilon^{-1}(0)=1$ but decreases nearly by half by the first non-zero grid point included in a $1\times 1\times 32$ sampling of the first Brillouin zone. $\epsilon^{-1}(0)=1$ is a general property of truncated systems with semiconductor-type screening since the $q^2$ dependence of the polarizability approaches 0 faster than the Coulomb interaction diverges at $q\to 0$. BerkeleyGW uses this $W$-averaging procedure by default for cases with truncated Coulomb interaction.

In general, using an extension of Eq. (49) even for $\mathbf{q}, \mathbf{GG'} \neq 0$ can speed up the convergence of a `sigma` or `absorption` cal-

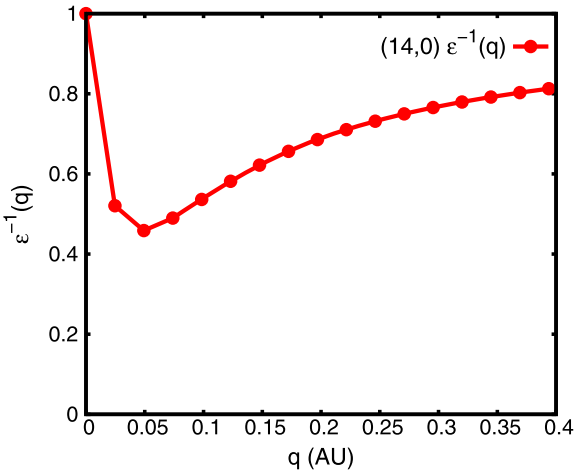


**Fig. 14.** $\epsilon^{-1}(q)$ in the $(14, 0)$ single-walled carbon nanotube for **q** along the tube axis as reported in the `epsilon.log` output file. The circles represent **q**-grid points included in a $1 \times 1 \times 32$ sampling of the first Brillouin zone.

culation with respect to the number of **q**-points required in the calculation. This can be explained easily by the fact that one is replacing a finite sum over **q**-points with an integral – mimicking a calculation on a much larger set of **q**-points or a much larger unit cell. The user can ask that BerkeleyGW use the cell-averaged $W$ for all **q** and **G** below an energy cutoff specified in the input file.

The averaging is implemented in the code using a Monte Carlo integration method with 2 500 000 random points in each cell.

### 5.1. Grid uniformity

Another important consideration in performing integrals over **q** of the Coulomb interaction in `sigma` and `kernel` is the sampling of the grid on which the integral is done. If the sampling in different directions is very different, then the result of the integrals will not go to the correct limit, since it will resemble a 1D or 2D integration rather than a 3D integration. This issue is important for calculation of nano-systems without truncation, or for highly anisotropic crystals. We determine the effective sampling in each direction as follows: take the vectors $b_i/N_i$, where $b_i$ is a reciprocal lattice vector and $N_i$ is the number of **q**-points in that direction. Find the shortest vector. Orthogonalize the next shortest vector to that one. Orthogonalize the remaining vector to the first two. (It is important to use this order since orthogonalization always makes the vector shorter.) Now compare the lengths of these orthogonalized vectors: if the ratio between the longest and shortest is too large (we use a factor of 2 as a tolerance), the grid is non-uniform and may give incorrect answers. The code will write a warning in this case, and the user should try to use a more nearly uniform grid, or check the convergence of results against the cell-averaging cutoff. Note that the sampling in any direction in which the Coulomb interaction is truncated is irrelevant when checking for grid uniformity.

## 6. Symmetry and degeneracy

### 6.1. Mean field

As was mentioned in Section 1, the largest cost when performing a GW calculation with the `BerkeleyGW` package is the generation of the input mean-field states. In order to reduce this cost, all the codes allow the user to input the wavefunctions in only the reduced Brillouin zone and construct the wavefunctions in the full zone by the following relation:

$$\phi_{\mathbf{R(k)}}(\mathbf{G}) = \phi_{\mathbf{k}}\left(\mathbf{R}^{-1}(\mathbf{G})\right)e^{-i\mathbf{G}\cdot\tau} \tag{50}$$

where the symmetry operation is defined by a rotation matrix **R** and a fractional translation $\tau$.

### 6.2. Dielectric matrix

The wavefunctions in the full zone then are used for the sum over **k** in Eq. (8) in `epsilon`, `sigma`, `kernel` and `absorption`, which requires not only the wavefunctions in the full zone but also the dielectric matrix. While in principle one could construct the dielectric matrices at all the **q**-points required in Eqs. (20)–(31) and (37), in practice one can use symmetry to reduce the required **q**-points. The `epsilon` code requires the user to calculate the dielectric matrices on a reduced set of **q**-points and the other codes generate the dielectric matrices in the full zone. If one defines $\mathbf{q}_1 = \mathbf{R}(\mathbf{q}) + \mathbf{G}_R$, where $\mathbf{G}_R$ is a **G**-vector chosen to ensure that **q** and $\mathbf{q}_1$ are in the first Brillouin zone, and **R**, $\tau$ are rotation matrix and translation respectively, then one can use the relation [1,44]:

$$\epsilon^{-1}_{\mathbf{GG'}}(\mathbf{q}_1; E) = e^{-i(\mathbf{G}-\mathbf{G'})\cdot\tau}\epsilon^{-1}_{\mathbf{G}_1\mathbf{G}'_1}(\mathbf{q}; E) \tag{51}$$

where $\mathbf{G}_1 = \mathbf{R}^{-1}(\mathbf{G} + \mathbf{G}_R)$. Given $\epsilon^{-1}$ in the reduced zone, this allows one to construct it in the full zone.

### 6.3. Truncation of sums

Both the dielectric matrix and the self-energy operator involve infinite sums over unoccupied states, which must in practice be truncated in the `epsilon` and `sigma` codes. The dielectric matrix and self-energy operator only retain the full symmetry of the system if the truncation does not cut through any degenerate subspaces. Consider a subspace of states belonging to a degenerate representation of a symmetry operation. Only the whole subspace is invariant under that operation, while just a part of it is not necessarily. As a result, a calculation using only part of the subspace will produce self-energies that break degeneracies due to that operation. Moreover, the actual values obtained are not well defined because the states used are arbitrary linear combinations in the subspace, which could even differ from run to run of a DFT code depending on how the calculation is initialized. These considerations are particularly acute for sums over a small number of states, since the contribution of the last few bands may be significant. Therefore, the `epsilon` and `sigma` codes check that the highest band requested for the sum is not degenerate with the next one, and block calculations that will break degeneracy. However, it is also possible to override this behavior with the flag `degeneracy_check_override`, for testing purposes and because in some cases there may be overlapping degenerate subspaces on different **k**-points that make it difficult to find acceptable numbers of bands; for large numbers of bands, the effect of truncation in a degenerate subspace will be small.

### 6.4. Self-energy operator

Degeneracy is also important from the point of view of the states on which self-energies are calculated, as opposed to those appearing in the sum. Since the self-energy operator has the full symmetry of the system, the matrix elements between states belonging to different representations are zero by symmetry. In the presence of high symmetry, this consideration can make the matrix quite sparse. To take advantage fully of symmetry here would require a careful analysis of each wavefunction's behavior under various symmetry operations and comparison to character tables of space groups. Users certainly can do this in deciding which

off-diagonal self-energy matrix elements to calculate. The `Sigma` takes a very simple approach to identify some of the elements which are zero by symmetry, based on degeneracy. The multiplicity of the degenerate subspace to which each states belongs is counted (1, 2, or 3 for the standard space groups), and clearly two states in subspaces of different multiplicity must belong to different representations, and their matrix element can be set to zero without calculation. This saves time and enforces symmetry.

Application of symmetry in a degenerate subspace can also speed up calculation of diagonal elements of the self-energy operator. The expressions for the exchange, screened exchange, and Coulomb-hole parts contain a sum over $\mathbf{q}$. In general, this must be done over the whole Brillouin zone, but to calculate the sum of the self-energies within a degenerate subspace it is sufficient to use the irreducible part of the Brillouin zone. Each part of $\Sigma$, in the various approximations, has the generic form

$$\begin{aligned}&\langle n\mathbf{k}|\,\Sigma\,|n'\mathbf{k}\rangle\\ &\quad=-\sum_{n''}\sum_{\mathbf{q}\mathbf{G}\mathbf{G}'}\langle n\mathbf{k}|e^{i(\mathbf{q}+\mathbf{G})\cdot\mathbf{r}}|n''\mathbf{k}-q\rangle\langle n''\mathbf{k}-q|e^{-i(\mathbf{q}+\mathbf{G}')\cdot\mathbf{r}}|n'\mathbf{k}\rangle\\ &\qquad\times F(\mathbf{q},\mathbf{G},\mathbf{G}').\end{aligned}\tag{52}$$

The summand is invariant under application of a symmetry operation $O$ in the subgroup of $\mathbf{k}$ provided that $n=n'$ and $n$ and $n''$ are non-degenerate, since in that case the action of the operation simply introduces a phase: $O|mk\rangle=e^{i\theta}|mk\rangle$ (degenerate states may instead transform into linear combinations in the degenerate subspace). These phases are canceled by the fact that each state appears also with its complex conjugate. If the states $n''$ in the sum are degenerate, the summand is not invariant but the sum is, if the whole degenerate subspace is summed over, since then we are taking the trace of the projector matrix $|n''\mathbf{k}\rangle\langle n''\mathbf{k}|$ in that subspace, which is invariant [44]. If $n$ is degenerate, then $\langle n\mathbf{k}|\Sigma|n\mathbf{k}\rangle$ is not invariant, but the trace of the self-energy in the degenerate subspace, $\sum_n\langle n\mathbf{k}|\Sigma|n\mathbf{k}\rangle$, is invariant. Therefore, to calculate diagonal elements for a whole degenerate subspace, for each state we sum only over $\mathbf{q}$ in the irreducible zone, with weight $W_{\mathbf{q}}$ from the number of $\mathbf{q}$-vectors related to $\mathbf{q}$ by symmetry. We then symmetrize by assigning the average to each:

$$\begin{aligned}\langle m\mathbf{k}|\,\Sigma\,|m\mathbf{k}\rangle&=\frac{1}{N_{\mathrm{deg}}}\sum_{n}^{\mathrm{deg}}\langle n\mathbf{k}|\,\Sigma\,|n\mathbf{k}\rangle\\ &=-\sum_{n}^{\mathrm{deg}}\sum_{n''}\sum_{\mathbf{G}\mathbf{G}'}\sum_{\mathbf{q}}^{\mathrm{irr}}W_{\mathbf{q}}\langle n\mathbf{k}|e^{i(\mathbf{q}+\mathbf{G})\cdot\mathbf{r}}|n''\mathbf{k}-q\rangle\\ &\quad\times\langle n''\mathbf{k}-q|e^{-i(\mathbf{q}+\mathbf{G}')\cdot\mathbf{r}}|n\mathbf{k}\rangle F(\mathbf{q},\mathbf{G},\mathbf{G}').\end{aligned}\tag{53}$$

(This averaging over degenerate bands is also done to enforce symmetry even when we use the full $\mathbf{q}$-sum, since the results may differ slightly due to limited precision in the wavefunctions from the mean-field calculation.) If we are calculating only part of a degenerate subspace, this trick does not work, and we must perform the complete sum. For diagonal elements, the code by default uses the irreducible $\mathbf{q}$-sum and will write an error if the calculation requires the full sum because of degeneracy, directing the user to enable via the flag `no_symmetries_q_grid`, or include all states in the degenerate subspace. For off-diagonal elements ($n\neq n''$), even if both are non-degenerate, application of the symmetry operation, in general, introduces different phases from the two states, which are not canceled. Thus the contributions from different $\mathbf{q}$-points related by symmetry differ, so that the full sum must always be used.

### 6.5. Bethe–Salpeter equation

Degeneracy must be considered in BSE calculations as well, when choosing the subspace in which to work. If the set of occupied or unoccupied states includes only part of a degenerate subspace, then the solutions found by `absorption` will break symmetry and can give qualitatively incorrect results. For example, an excitation that should have zero oscillator strength by symmetry, due to interference between transitions to two degenerate states, may not be dark if only one of those transitions is included. This issue is quite general and applies to the choice of active spaces in other theories as well, such as configuration interaction [45].

### 6.6. Degeneracy utility

We provide a utility called `degeneracy_check.x` which reads wavefunction files and writes out a list of acceptable numbers of bands. Multiple wavefunction files can be checked at once, for example the shifted and unshifted grids in `Epsilon` or shifted, unshifted, coarse, and fine grids for Bethe–Salpeter equation calculations, in which case the utility will identify numbers of bands which are consistent with degeneracy for every file.

### 6.7. Real and complex flavors

The component executables come in two "flavors," real and complex, specified at compile time and denoted by the suffix `.real.x` or `.cplx.x`. When the system has inversion and time-reversal symmetry, we can choose the wavefunctions to be real in reciprocal space. The plane-wave expansions are:

$$u(\mathbf{r})=\sum_{\mathbf{G}}u_{\mathbf{G}}e^{i\mathbf{G}\cdot\mathbf{r}},\tag{54}$$

$$u(-\mathbf{r})=\sum_{\mathbf{G}}u_{\mathbf{G}}e^{-i\mathbf{G}\cdot\mathbf{r}},\tag{55}$$

$$u^*(\mathbf{r})=\sum_{\mathbf{G}}u^*_{\mathbf{G}}e^{-i\mathbf{G}\cdot\mathbf{r}}.\tag{56}$$

The symmetry conditions mean that wavefunctions can be chosen to satisfy $u(-\mathbf{r})=au(\mathbf{r})$ (inversion symmetry) and $u^*(\mathbf{r})=bu(\mathbf{r})$ (time-reversal, equivalent to taking the complex conjugate of the Schrödinger equation), with $a,b$ each equal to $\pm1$ depending on whether the wavefunction belongs to an odd or even representation. Thus we can choose $u(-\mathbf{r})=cu^*(\mathbf{r})$ with $c=ab$ also equal to $\pm1$. Combining this with the plane-wave expansions,

$$\sum_{\mathbf{G}}u_{\mathbf{G}}e^{-i\mathbf{G}\cdot\mathbf{r}}=c\sum_{\mathbf{G}}u^*_{\mathbf{G}}e^{-i\mathbf{G}\cdot\mathbf{r}},\tag{57}$$

$$u_{\mathbf{G}}=cu^*_{\mathbf{G}}.\tag{58}$$

The choice $c=1$ corresponds to real coefficients; $c=-1$ corresponds to pure imaginary coefficients. Most plane-wave electronic-structure codes always use complex coefficients, and so the coefficients will in general not be real, even in the presence of inversion and time-reversal symmetry. For a non-degenerate state, the coefficients will be real times an arbitrary global phase, determined by the initialization of the solution procedure. We must divide out this global phase to make the coefficients real. In a degenerate subspace, the states need not be eigenstates of inversion, and so in general they may not just be real times a global phase. Instead, in each subspace of degeneracy $n$ we take the $2n$ vectors given by the real and imaginary parts of each wavefunction, and then use a Gram–Schmidt process to find $n$ real orthonormal wavefunctions

spanning the subspace. The density and exchange-correlation potential are real already in the presence of inversion symmetry and there is no arbitrary phase possible.

The real-space density is always real: $\rho(\mathbf{r}) = \rho^*(\mathbf{r})$. With inversion symmetry, we also have $\rho(\mathbf{r}) = \rho(-\mathbf{r})$. In reciprocal space,

$$\rho(\mathbf{r}) = \sum_{\mathbf{G}} \rho_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}}, \tag{59}$$

$$\rho^*(\mathbf{r}) = \sum_{\mathbf{G}} \rho^*_{\mathbf{G}} e^{-i\mathbf{G}\cdot\mathbf{r}}, \tag{60}$$

$$\rho(-\mathbf{r}) = \sum_{\mathbf{G}} \rho_{\mathbf{G}} e^{-i\mathbf{G}\cdot\mathbf{r}}. \tag{61}$$

Together, these relations imply $\rho_{\mathbf{G}} = \rho^*_{\mathbf{G}}$, i.e. the reciprocal-space coefficients are real. Precisely the same equations apply for the exchange-correlation potential.

The wavefunction, density, and exchange-correlation potential are then all stored as real coefficients, saving disk space (for the files), memory, and operations compared to the complex representation. Usually, only the lack of inversion symmetry of the lattice and basis would require the use of complex wavefunctions, but if spin-orbit coupling or magnetic fields are present, then time-reversal symmetry is lost and complex wavefunctions again are required.

## 7. Computational issues

### 7.1. Memory estimation

In the beginning of each run, all the major code components print the amount of memory available per CPU and an estimate of memory required per CPU to perform the calculation. If the latter exceeds the former, the job is likely to fail with a memory allocation error. The amount of memory required is estimated by determining the sizes of the largest arrays after reading in the parameters of the system from the input files. A straightforward approach to estimating the amount of available memory is to allocate memory by incremental amounts until the allocation call returns with an error. Unfortunately, in many implementations the allocation call returns without an error even if the requested amount of memory is not physically available, but the system fails when trying to access this "allocated" memory. We implement another approach based on the Linux `/proc` file system. First, each CPU opens file `/proc/meminfo` and reads in the values of `MemFree` and `Cached`. The sum of these two values gives the amount of memory available per node. This approach works on almost all modern high-performance computing systems where the Linux `/proc` File System is accessible. (However, for BSD-based MacOS which lacks `/proc/meminfo`, we read the page size and number of free and speculative pages from the command `vm_stat`.) Second, each CPU calls an intrinsic Fortran routine that returns the host name which is unique for each node. By comparing host names reported by different CPUs we identify the number of CPUs per node. The amount of memory available per CPU is then given by the ratio of the amount of memory available per node to the number of CPUs per node.

### 7.2. Makefiles

The main codes are in the `Epsilon`, `Sigma`, `BSE`, `PlotXct`, and `MeanField` directories. Routines used by all parts are in the `Common` directory, and routines common to some of the `MeanField` codes are in the `Symmetry` directory. The `Makefiles` are designed for GNU Make, and enable targets in a directory to be built from any level of the directory hierarchy. They contain a full set of dependencies, including those between directories, to ensure that the build is correct after any changes to source, for ease in development and modification. This also enables use of parallel make on large numbers of processors for rapid builds – any omissions in the dependencies generally cause a failure for a parallel `make`. The special `make` target `all-j` (i.e. `make -j all-j`) begins by using all processes to build the `Common` and `Symmetry` directories, which contain files required by files in a large number of directories; otherwise, the build would fail due to attempts by multiple processes to read and write the same files in the `Common` and `Symmetry` directories. Commonly, Fortran Makefiles are set up with object files depending on other object files. However, the real situation is that object files depend on module files (`.mod`) for the modules they use, and only executables depend on object files. Therefore we have dependencies directly on the module files to ensure the required files are present for compilation, particularly for parallel builds.

### 7.3. Installation instructions

The code can be installed via the following steps:

```
cp [flavor_real.mk/flavor_cplx.mk] flavor.mk
ln -s config/[mysystem].mk arch.mk
make all
make check[-jobscript]
```

First a flavor is selected by copying the appropriate file to `flavor.mk`. Then a configuration file must be put as `arch.mk`. Configurations appropriate for various supercomputers as well as for using standard Ubuntu or Macports packages are provided in the `config` directory. Appropriate paths, libraries, and compiler flags can be set in a new `arch.mk` for other systems. Finally the testsuite should be run to confirm that the build is working. In serial, the command is `make check`; for parallel builds, it is `make check-parallel`. On machines with a scheduler, a job script should be created to run this command. For the architectures supported in `config`, job scripts to run the testsuite are provided in the `testsuite` directory, and can be used via `make check-jobscript`.

### 7.4. Validation and verification

The importance of verification and validation of complicated scientific software packages is receiving increasing attention. We use standard open-source tools for code development, following accepted best practices [46]. Development is done with the subversion (SVN) version-control system [47] and Trac, an issue-tracking system and interface to SVN [48]. All code runs identify the version and revision number used in the output for traceability of results, implemented via a special source file called `svninfo.f90` which all SVN revisions must modify (enforced via a pre-commit hook). Debug mode can be enabled via `-DDEBUG` in the `arch.mk` file, which performs extra checking including of dynamic memory allocation and deallocation. A macro enables a check of the status returned by the system after an allocation attempt, and reports failures, identifying the array name, size, source file, and line number, as well as which processor failed to allocate the array. Additionally, it keeps track of the amount of memory dynamically allocated and deallocated, so the code can report at the end of each run how much memory remains allocated, and the maximum and minimum memory 'high-water-mark' among the processors. In debug mode, a stack trace also can be enabled, either on just the root processor, or on all processors (causing the code to run much slower), which can be used to locate where problems such

as segmentation faults are occurring (possibly on only one processor). Verbose mode can be enabled via `-DVERBOSE` which writes extra information as the calculation proceeds.

The package contains a comprehensive testsuite to test the various executables, run modes, and options, in the `testsuite` directory. Calculations of several different physical systems, with mean-field, `epsilon`, `sigma`, and BSE calculations, are carried out (including use of `PlotXct` and some utilities), detecting any run-time errors and showing any warnings generated. Then selected results are extracted from the output and compared to reference information within a specified tolerance. The actual calculated values, as well as timing for each step, are displayed. Each match is shown as either `OK` or `FAIL`, and a final summary is written of failures. The calculations are small and generally underconverged, to make them quick enough for routine testing and rapid feedback. The mean-field steps are either EPM (quick serial calculations) or stored compressed output from DFT calculations. The `Epsilon`, `Sigma`, and BSE calculations are run either in serial or on 4 processors (for parallel builds).

The testsuite has numerous uses. It is useful for users to verify the success of a new build of the code on their platform (failures could be due to library problems, excessive optimizations, etc.). It is used for developers to verify that the code is giving reproducible answers, ensure consistency between serial/parallel runs, as well as real/complex and spin-polarized/unpolarized runs, and check that the code works with new compilers or libraries. On a routine basis, the testsuite is also useful for developers to check that changes to the code do not introduce problems. The driver scripts (`run_testsuite.sh` and `run_regression_test.pl`) and specifications for the files defining the test steps are originally based on, and developed in conjunction with, those of the Octopus code [49,50]. This framework is quite general and can be used easily for constructing a testsuite for another code. It can be run in serial with the command `make check` (or `make check-save` to retain the working directories from the runs), or in parallel with `make check-jobscript` (or `make check-jobscript-save`). The system configuration file `arch.mk` can specify how to submit an appropriate jobscript for parallel execution on a supercomputer using a scheduler. Scripts are provided in the `testsuite` directory for some supercomputers. The testsuite is used with a continuous-integration system, the open-source tool BuildBot [51], to ensure the integrity of the code during development. Each commit to the SVN repository triggers a build of the code on each of 10 "buildslaves," which have different configurations with respect to serial/parallel, compilers, and libraries. After the build, the testsuite is run. BuildBot will report to the developers if the either the build or test runs failed, so the problem can be quickly remedied. Use of the various different buildslave configurations helps ensure that the code remains portable across different platforms and in accordance with the language standards. Two of the buildslaves are on a supercomputer with a scheduler, a situation for which standard BuildBot usage is problematic. We provide a Perl script `buildbot_pbs.pl` that can submit jobs, monitor their status, capture their output for BuildBot, and determine success or failure. This script is general for any PBS scheduler and can be used for other codes too.

### 7.5. Supported operating systems, compilers, and libraries

With the testsuite, we have tested the code extensively with various configurations, and support the following compilers and libraries:

- Operating systems: Linux, MacOS, AIX
- Fortran compilers (required): `pgf90`, `ifort`, `gfortran`, `g95`, `openf90`, `sunf90`, `pathf90`, `crayftn`, `af90` (Absoft), `nagfor`, `xlf90` (experimental)
- C++ compilers (optional): `pgCC`, `icc`, `g++`, `openCC`, `pathCC`, `crayCC`
- MPI implementation (optional): OpenMPI, MPICH1, MPICH2, MVAPICH2
- LAPACK/BLAS implementation (required): NetLib, ATLAS, Intel MKL, ACML, Cray LibSci
- ScaLAPACK/BLACS implementation (required by BSE if MPI is used): NetLib, Cray LibSci, Intel MKL, AMD
- FFTW (required): versions 2.1.5, 2.1.5.1, 2.1.5.2

## 8. Utilities

### 8.1. Visualization

Several visualization tools designed to simplify work with the DFT codes and the BerkeleyGW package are provided in directory `Visual`. These include the `surface.x` code and Matter library.

Surface is a C++ code for generating an isosurface of a volumetric scalar field (such as the wavefunction, charge density, or local potential). The scalar field is read from a Gaussian Cube or XCrySDen XSF file, the surface triangulation is performed using the marching-cubes or marching-tetrahedra algorithms, and the isosurface is written in the POV-Ray scripting language. The final image is rendered using the ray-tracing program POV-Ray [52]. Running the surface code requires a fairly complicated input parameter file. A brief description of the input file parameters is given in the header of `surface.cpp`.

Matter is a Python library for manipulating atomic structures with periodic boundary conditions. It can translate and rotate the atomic coordinates, generate super-cells, assemble atomic systems from fragments, and convert between different file formats. The supported file formats are `mat`, `paratec`, `vasp`, `espresso`, `siesta`, `xyz`, `xsf`, and `povray`. `mat` is the native file format of the library. `paratec`, `vasp`, `espresso`, and `siesta` represent the formats used by different plane-wave and local-orbital DFT codes. `xyz` is a simple format supported by many molecular viewers, and `xsf` is the internal format of XCrySDen [53]. `povray` stands for a scripting language used by the ray-tracing program POV-Ray. The core of the library consists of files `common.py`, `matrix.py`, and `matter.py`. Script `convert.py` is a command-line driver that performs the basic operations supported by the library. Script `link.py` is a molecular assembler that can be used to rotate and link two molecules together. Script `gsphere.py` generates a real-space grid and a sphere of **G**-vectors given lattice parameters and a kinetic-energy cutoff. This helps to estimate the number of unoccupied states needed in GW calculations. Script `average.py` takes an average of the scalar field on the faces or in the volume of the unit cell. This is used to determine the vacuum level in DFT calculations. Script `volume.py` converts the a3Dr file produced by `PARATEC` or `BerkeleyGW` to Gaussian Cube or XCrySDen XSF format. Each script requires a different set of command-line arguments. Running individual scripts without arguments displays a list of all possible command line arguments and a short description of each argument.

### 8.2. Band-structure interpolation

To plot the quasiparticle band-structure of a system, we provide two methods. The first, `sig2wan` is a utility that uses Wannier interpolation of the band-structure [54,55] and is based on the `Wannier90` [56] package. This utility does not construct

the Wannier functions; the user has to do that using the package used to construct the mean-field eigenfunctions, PARATEC or Quantum ESPRESSO. Once the Wannier functions have been constructed, this utility replaces the mean-field eigenvalues in the wannier.eig file (generated by Wannier90) with the quasiparticle eigenvalues. The then can rerun the Wannier90 executable to generate band-structures along arbitrary directions. Because one replaces the mean-field eigenvalues with quasiparticle eigenvalues, this approach does not work well if one replaces only some of the mean-field eigenvalues with quasiparticle ones for entangled bands. The second method for plotting quasiparticle band-structure, the inteqp utility, uses Eq. (42) to construct the band-structure along arbitrary directions. In this method, we interpolate directly the quasiparticle corrections, which are significantly smoother functions of $\mathbf{k}$ and $E$ than the quasiparticle eigenvalues themselves. Therefore, this method requires both the mean-field eigenvalues and eigenfunctions along the desired band-structure direction.

### 8.3. Other

We provide a general-purpose utility called mf_convert.x which can convert between binary and ASCII formats of wavefunction, density, and exchange-correlation potential files. The real/complex flavor is determined by reading the file header, and if the utility is called through mf_convert_wrapper.sh, the binary/ASCII format is detected via the grep command and need not be specified. This converter is useful for moving such files between different platforms, since the binary files are more compact and the form read by the code, but are not necessarily portable between different platforms, whereas the ASCII files are.

The image-charge model (ICM) is implemented in a utility called icm.x, based on the Surface code. For a molecule weakly coupled to a metallic surface, the self-energy correction to a state can be well approximated by the sum of the self-energy correction of that state in the isolated molecule and an additional term due to screening from the metal [57]. This screening term is modeled as the electrostatic energy of the charge density of the wavefunction and its induced image-charge distribution in the metal. Let the operator $R$ be a reflection across an image plane. Then

$$\Delta\Sigma = \pm\frac{1}{2}\iint \psi(\mathbf{r})\psi^*(\mathbf{r}')\frac{1}{|\mathbf{r}-R\mathbf{r}'|}\psi(\mathbf{r}')\psi^*(\mathbf{r})\,d\mathbf{r}\,d\mathbf{r}' \tag{62}$$

where the plus sign applies for occupied orbitals and the minus sign for unoccupied orbitals. This approximation is useful for modeling scanning-tunneling spectroscopy of molecules absorbed on metal surfaces [58] and for quantum-transport calculations of molecular junctions [59].

## Acknowledgements

We acknowledge the following people for their contributions to earlier version of the package: Xavier Blase, Andrew Canning, Eric K. Chang, Mark S. Hybertsen, Sohrab Ismail-Beigi, Je-Luen Li, Jeff Neaton, Cheol-Hwan Park, Filipe J. Ribeiro, Gian-Marco Rignanese, Catalin D. Spataru, Murilo L. Tiago, Li Yang and Peihong Zhang. We'd also like to thank the following beta users for their feedback, bug-reports, patches and general assistance while developing the code: Sangkook Choi, Peter Doak, Felipe Jornada, Brad D. Malone, Sahar Sharifzadeh, Isaac Tamblyn and Derek Vigil and the various other members of the Louie and Cohen groups at the University of California, Berkeley.

J.D. and M.J. acknowledge support from the Director, Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division, U.S. Department of Energy under Contract No. DE-AC02-05CH11231. G.S. acknowledges support under National Science Foundation Grant No. DMR10-1006184. D.A.S. acknowledges support from the NSF Graduate Fellowship Program. Computational resources have been provided by NSF through TeraGrid resources at NICS and by DOE at Lawrence Berkeley National Laboratory's NERSC facility.

## Appendix A

### A.1. Specification of file formats

Wavefunction files are needed by all parts of the code, with various filenames. The epsilon executable uses an unshifted grid (WFN) and a shifted grid (WFNq). The sigma executable uses WFN_inner to construct the self-energy operator and evaluates matrix elements with WFN_outer. The kernel executable constructs kernel matrix elements with a coarse unshifted (WFN_co) and shifted grid (WFNq_co). The absorption and plotxct executables uses fine unshifted (WFN_fi) and shifted grids (WFNq_fi). Additionally, the sigma executable needs the charge-density RHO for GPP calculations, and needs the exchange-correlation potential VXC (unless its pre-computed matrix elements are supplied in a vxc.dat file). These files all share a common format, which begins with a header. Parts in italics are only for wavefunction files, not charge-density or exchange-correlation potential files. Each bullet represents a record in the file. The utility wfn_rho_vxc_info.x can read the information from the header and report it in a comprehensible format to the user.

- [WFN/RHO/VXC]-[Real/Complex] date time
- number of spins, number of G-vectors, number of symmetries, [0 for cubic symmetry/1 for hexagonal symmetry], number of atoms, charge-density cutoff (Ry), *number of k-points, number of bands, maximum number of G-vectors for any k-point, wavefunction cutoff (Ry)*
- FFT grid(1:3), *k-grid(1:3), k-shift(1:3)*
- real-space cell volume (a.u.), lattice constant (a.u.), lattice vectors(1:3, 1:3) in units of lattice constant, real-space metric tensor(1:3, 1:3)
- reciprocal-space cell volume (a.u.), reciprocal lattice constant (a.u.), reciprocal lattice vectors(1:3, 1:3) in units of reciprocal lattice constant, reciprocal-space metric tensor(1:3, 1:3)
- symmetry rotation matrices(1:3, 1:3, 1:number of symmetries) in reciprocal-lattice basis
- symmetry fractional translations(1:3, 1:number of symmetries) in units of lattice vectors times $2\pi$
- atomic positions(1:3, 1:number of atoms) in units of lattice constant, atomic numbers(1:number of atoms)
- *number of G-vectors for each k-point(1:number of k-points)*
- *k-point weights(1:number of k-points) from 0 to 1*
- *k-point coordinates(1:3, 1:number of k-points) in crystal coordinates*
- *index of lowest band to use on each k-point (1:number of k-points)*
- *index of highest occupied band on each k-point(1:number of k-points)*
- *energy eigenvalues(1:number of bands, 1:number of k-points, 1:number of spins) (Ry)*

- `occupations(1:number of bands, 1:number of k-points, 1:number of spins) from 0 to 1`

In the body of a file, **G**-vectors are listed as (1:3, 1:ng), expressed as integers in reciprocal lattice units, and data is listed as (1:ng, 1:number of spins). **G**-vector components should be chosen in the interval $[-n/2, n/2)$ where $n$ is the FFT grid. A full sphere must be used, not a half sphere as in the Hermitian FFT representation for a real function. Each set is preceded by an integer specifying how many records the **G**-vectors or data is broken up into, for ease of writing files from a code parallelized over **G**-vectors. Wavefunction files follow the header with a listing of all the **G**-vectors; for each **k**-point, there is first a list of **G**-vectors, and then the wavefunction coefficients for each band. `RHO` and `VXC` files have instead just one listing of **G**-vectors and coefficients after the header. The wavefunction coefficients must be normalized so that the sum of their squares is 1. The `RHO` and `VXC` coefficients are normalized such that their $\mathbf{G} = 0$ component is the average value in real space. `RHO` is in atomic units, while `VXC` is in Ry.

The recommended scheme is to use pre-computed exchange matrix elements in an ASCII file `vxc.dat`, because `VXC` is only applicable to a functional that consists of a local potential plus some fraction of exchange. Some hybrid functionals, and certainly self-consistent GW calculations, do not fall in this category. Matrix elements are in eV and are always written with real and imaginary parts (even in the real version of the code). The file may contain any number of **k**-points in any order. It contains a certain number of diagonal elements (`ndiag`) and a certain number of off-diagonal elements (`noffdiag`). Each **k**-point block begins with the line:

*kx*, *ky*, *kz* [*crystal coordinates*], *ndiag*nspin*, *noffdiag*nspin*

There are then `ndiag*nspin` lines of the form

*ispin*, *idiag*, *Re* $\langle idiag|V|idiag\rangle$, *Im* $\langle idiag|V|idiag\rangle$

There are then `noffdiag*nspin` lines of the form

*ispin*, *ioff1*, *ioff2*, *Re* $\langle ioff1|V|ioff2\rangle$, *Im* $\langle ioff1|V|ioff2\rangle$

## References


[1] M.S. Hybertsen, S.G. Louie, Electron correlation in semiconductors and insulators: Band gaps and quasiparticle energies, Phys. Rev. B 34 (1986) 5390.
[2] S.G. Louie, Conceptual Foundations of Materials: A Standard Model for Ground- and Excited-State Properties, Contemporary Concepts of Condensed Matter Science, Elsevier, 2006.
[3] C.D. Spataru, S. Ismail-Beigi, L.X. Benedict, S.G. Louie, Excitonic effects and optical spectra of single-walled carbon nanotubes, Phys. Rev. Lett. 92 (2004) 077402.
[4] C.D. Spataru, S. Ismail-Beigi, L.X. Benedict, S.G. Louie, Quasiparticle energies, excitonic effects and optical absorption spectra of small-diameter single-walled carbon nanotubes, Appl. Phys. A 78 (2004) 1129.
[5] J. Deslippe, C.D. Spataru, D. Prendergast, S.G. Louie, Bound excitons in metallic single-walled carbon nanotubes, Nano Lett. 7 (2007) 1626.
[6] L. Hedin, New method for calculating the one-particle Green's function with application to the electron-gas problem, Phys. Rev. 139 (3A) (1965) A796–A823.
[7] G. Strinati, Application of the Green's functions method to the study of the optical properties of semiconductors, Riv. Nuovo Cimento 11 (1988) 1.
[8] M. Rohlfing, S.G. Louie, Electron–hole excitations and optical spectra from first principles, Phys. Rev. B 62 (2000) 4927.
[9] S. Albrecht, L. Reining, R. Del Sole, G. Onida, Ab initio calculation of excitonic effects in the optical spectra of semiconductors, Phys. Rev. Lett. 80 (20) (1998) 4510–4513.
[10] L.X. Benedict, E.L. Shirley, R.B. Bohn, Optical absorption of insulators and the electron–hole interaction: An ab initio calculation, Phys. Rev. Lett. 80 (20) (1998) 4514–4517.
[11] http://www.nersc.gov/projects/paratec/.
[12] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, A.D. Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matt. 21 (39) (2009) 395502.
[13] J.M. Soler, E. Artacho, J.D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, The siesta method for ab initio order-*N* materials simulation, J. Phys.: Condens. Matt. 14 (2002) 2745.
[14] J.R. Chelikowsky, N. Troullier, Y. Saad, Finite-difference-pseudopotential method: Electronic structure calculations without a basis, Phys. Rev. Lett. 72 (8) (1994) 1240–1243.
[15] M.M.G. Alemany, M. Jain, L. Kronik, J.R. Chelikowsky, Real-space pseudopotential method for computing the electronic properties of periodic systems, Phys. Rev. B 69 (2004) 075101.
[16] R.M. Martin Electronic, Structure: Basic Theory and Practical Methods, Cambridge University Press, 2004.
[17] W. Kohn, L.J. Sham, Self-consistent equations including exchange and correlation effects, Phys. Rev. 140 (1965) A1133.
[18] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, Phys. Rev. Lett. 77 (1996) 3865.
[19] L. Hedin, S. Lundqvist, Effects of electron–electron and electron–phonon interactions on the one-electron states of solids, in: F. Seiz, D. Turnbull, H. Ehrenreich (Eds.), Advances in Research and Applications, in: Solid State Physics, vol. 23, Academic Press, 1970, pp. 1–181.
[20] M.L. Cohen, M. Schlüter, J.R. Chelikowsky, S.G. Louie, Self-consistent pseudopotential method for localized configurations: Molecules, Phys. Rev. B 12 (1975) 5575.
[21] A.L. Fetter, J.D. Walecka, Quantum Theory of Many-Body Systems, McGraw–Hill, San Francisco, 1971.
[22] J.G.E. Jellison, M.F. Chisholm, S.M. Gorbatkin, Optical functions of chemical vapor deposited thin-film silicon determined by spectroscopic ellipsometry, Appl. Phys. Lett. 62 (25) (1993) 3348–3350.
[23] B. Holm, U. von Barth, Fully self-consistent *GW* self-energy of the electron gas, Phys. Rev. B 57 (4) (1998) 2108–2117.
[24] F. Aryasetiawan, O. Gunnarsson, The *GW* method, Rep. Prog. Phys. 61 (3) (1998) 237.
[25] M. Jain, J. Deslippe, G. Samsonidze, M.L. Cohen, S.G. Louie, $G_0W_0$ diagonalization using the static COHSEX approximation, in preparation.
[26] A. Fleszar, Ph.D. thesis, University of Trieste, 1985.
[27] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93 (2) (2005) 216–231, special issue on "Program Generation, Optimization, and Platform Adaptation".
[28] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd edn., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[29] A. Baldereschi, E. Tosatti, Mean-value point and dielectric properties of semiconductors and insulators, Phys. Rev. B 17 (12) (1978) 4710–4717.
[30] L.X. Benedict, C.D. Spataru, S.G. Louie, Quasiparticle properties of a simple metal at high electron temperatures, Phys. Rev. B 66 (8) (2002) 085116.
[31] B.-C. Shih, Y. Xue, P. Zhang, M.L. Cohen, S.G. Louie, Quasiparticle band gap of ZnO: High accuracy from the conventional $G_0W_0$ approach, Phys. Rev. Lett. 105 (2010) 146401.
[32] P. Rinke, A. Qteish, J. Neugebauer, C. Freysoldt, M. Scheffler, Combining *GW* calculations with exact-exchange density-functional theory: an analysis of valence-band photoemission for compound semiconductors, New J. Phys. 7 (2005) 126.
[33] F. Bruneval, N. Vast, L. Reining, Effect of self-consistency on quasiparticles in solids, Phys. Rev. B 74 (2006) 045102.
[34] M. van Schilfgaarde, T. Kotani, S. Faleev, Quasiparticle self-consistent *GW* theory, Phys. Rev. Lett. 96 (2006) 226402.
[35] C.-D. Spataru, Electron excitations in solids and novel materials, Ph.D. thesis, University of California, Berkeley, 2004.
[36] S.B. Zhang, D. Tománek, M.L. Cohen, S.G. Louie, M.S. Hybertsen, Evaluation of quasiparticle energies for semiconductors without inversion symmetry, Phys. Rev. B 40 (1989) 3162.
[37] W. Hanke, Dielectric theory of elementary excitations in crystals, Adv. Phys. 27 (2) (1978) 287–341.
[38] R. Haydock, The recursive solution of the Schrödinger equation, Comput. Phys. Commun. 20 (1) (1980) 11–16.
[39] L.X. Benedict, E.L. Shirley, Ab initio calculation of $\epsilon_2(\omega)$ including the electron–hole interaction: Application to GaN and $CaF_2$, Phys. Rev. B 59 (8) (1999) 5441–5451.
[40] S. Ismail-Beigi, E.K. Chang, S.G. Louie, Coupling of nonlocal potentials to electromagnetic fields, Phys. Rev. Lett. 87 (2001) 087402.
[41] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[42] L. Yang, J. Deslippe, C.-H. Park, M.L. Cohen, S.G. Louie, Excitonic effects on the optical response of graphene and bilayer graphene, Phys. Rev. Lett. 103 (18) (2009) 186802.
[43] S. Ismail-Beigi, Truncation of periodic image interactions for confined systems, Phys. Rev. B 73 (2006) 233103.
[44] M.S. Hybertsen, S.G. Louie, Ab initio static dielectric matrices from the density-functional approach. I. Formulation and application to semiconductors and insulators, Phys. Rev. B 35 (11) (1987) 5585–5601.
[45] G.P. Zhang, D.A. Strubbe, S.G. Louie, T.F. George, First-principles prediction of optical second-order harmonic generation in the endohedral $NC_{60}$ compound, Phys. Rev. A 84 (2) (2011) 023837.
[46] Z. Mirali, … ERROR … why scientific programming does not compute, Nature 467 (2010) 775–777.
[47] http://subversion.tigris.org/.
[48] http://trac.edgewall.org/.
[49] A. Castro, H. Appel, M. Oliveira, C.A. Rozzi, X. Andrade, F. Lorenzen, M.A.L. Marques, E.K.U. Gross, A. Rubio, octopus: A tool for the application of time-dependent density functional theory, Phys. Status Solidi B 243 (11) (2006) 2465–2488.
[50] M.A.L. Marques, A. Castro, G.F. Bertsch, A. Rubio, octopus: A first-principles tool for excited electron-ion dynamics, Comput. Phys. Commun. 151 (1) (2003) 60–78.
[51] http://buildbot.net/.
[52] http://www.povray.org/.
[53] A. Kokalj, Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale, Comp. Mater. Sci. 28 (2003) 155, http://www.xcrysden.org/.
[54] N. Marzari, D. Vanderbilt, Maximally localized generalized Wannier functions for composite energy bands, Phys. Rev. B 56 (20) (1997) 12847–12865.
[55] I. Souza, N. Marzari, D. Vanderbilt, Maximally localized Wannier functions for entangled energy bands, Phys. Rev. B 65 (3) (2001) 035109.
[56] A.A. Mostofi, J.R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt, N. Marzari, wannier90: A tool for obtaining maximally-localised Wannier functions, Comput. Phys. Commun. 178 (9) (2008) 685–699.
[57] J.B. Neaton, M.S. Hybertsen, S.G. Louie, Renormalization of molecular electronic levels at metal–molecule interfaces, Phys. Rev. Lett. 97 (21) (2006) 216405.
[58] C. Tao, J. Sun, X. Zhang, R. Yamachika, D. Wegner, Y. Bahri, G. Samsonidze, M.L. Cohen, S.G. Louie, T.D. Tilley, R.A. Segalman, M.F. Crommie, Spatial resolution of a type II heterojunction in a single bipolar molecule, Nano Lett. 9 (2009) 3963.
[59] S.Y. Quek, D.A. Strubbe, H.J. Choi, S.G. Louie, J.B. Neaton, First-principles approach to charge transport in single-molecule junctions with self-energy corrections, in preparation.