Octopus + BerkeleyGW

e-h matrix linear response

BuildBot, debugging


www.berkeleygw.org
BerkeleyGW: workflow for calculations

Many-body perturbation theory in a plane-wave basis
What can the code do? Dielectric function

Figure 5: Example output plotted from EpsDyn showing the computed $\epsilon_{00}(\omega)$ in ZnO.
Figure 8: Top: GW quasiparticle self-energy corrections, $E^{QP} - E^{LDA}$ 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 bandstructure using the interpolated self-energy corrections (black open) and the LDA uninterpolated bandstructure (red). 256 points are used to sample the Brillouin zone.
What can the code do? Optical absorption

Figure 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].
Figure 2: The electronic component of the excited state wavefunction (Eq. 1) for pentacene (red) and PTCDA (blue), with the hole located at the center of a molecule, as indicated by a green arrow. Insets show the same for the gas-phase molecule. For all cases, a common isosurface value is taken to be 1% of the maximum.

Theoretical framework: mean field

Mean field = One-electron energies and wavefunctions

(like mean-field theory for Ising model vs. correlated treatment)

Could be DFT (LDA, GGA, hybrids), Hartree-Fock, etc.

\[
\left[-\frac{1}{2} \nabla^2 + V_{\text{ion}} + V_H + V_{xc}^{\text{DFT}}\right] \psi_{nk}^{\text{DFT}} = E_{nk}^{\text{DFT}} \psi_{nk}^{\text{DFT}}
\]

Supported codes: PARATEC, ESPRESSO, SIESTA, PARSEC, Octopus.
Must be norm-conserving pseudopotentials (no PAW, ultrasoft).

Need also charge density and XC matrix elements.

Empirical pseudopotential method (EPM) code based on TBPW from Richard Martin’s *Electronic Structure* book, with extensive bugfixes and improvements.
Binary input to BerkeleyGW

- EPM, paratecSGL, ESPRESSO: write wavefunction, density, $V_{xc}$
- Epsilon: read wfns on unshifted (and shifted) grid
- Sigma: read wfns for constructing operator and for taking matrix elements, read density for GPP, read $V_{xc}$ to subtract from self-energy
- Kernel: read wfns on coarse unshifted (and shifted) grid
- Diag/Haydock: read wfns on coarse and fine unshifted (and shifted) grid
## Specification of header

### Scalars

- [WFN/RHO/VXC]-[Real/Complex] date time
- nspin
- ng
- ntran
- cell_symmetry
- nat
- ecutrho
- nk
- nbands
- ngkmax
- ecutwfc

### Arrays of fixed size

- \( k_{\text{max}}(1:3) \)
- \( k_{\text{grid}}(1:3) \)
- \( k_{\text{shift}}(1:3) \)
- celvol
- alat
-avec(1:3, 1:3)
- adot(1:3, 1:3)
- recvol
- blat
- bvec(1:3, 1:3)
- bdot(1:3, 1:3)
- mtrx(1:3, 1:3, 1:ntran)
- tnp(1:3, 1:ntran)

### Allocatable arrays

- apos(1:3, 1:nat)
- atyp(1:3, 1:nat)
- ngk(1:nk)
- kweights(1:nk)
- kpt(1:3, 1:nk)
- ifmin(1:nspin, 1:nk)
- ifmax(1:nspin, 1:nk)
- energies(1:nbands, 1:nk, 1:nspin)
- occupations(1:nbands, 1:nk, 1:nspin)

### Scalars

- real space
- reciprocal space
- symmetry operations
- fractional translations

### Allocatable arrays

- atoms
- G-vector for each k-point
- k-weights
- k-coordinates
- lowest occupied band
- highest occupied band
- eigenvalues
- occupations
Specification of body

WFN:

\[ \text{gvec}(1:3, 1:\text{ng}) \quad \text{global list} \]
\[ \text{gvec}(1:3, 1:\text{ngk}(\text{ik})) \quad \text{for each k-point} \]
\[ \text{data}(1: \text{ngk}(\text{ik}), 1:\text{nspin}) \]

RHO/VXC:

\[ \text{gvec}(1:3, 1:\text{ng}) \]
\[ \text{data}(1: \text{ng}, 1:\text{nspin}) \]
Driver routines

[read/write][binary/format/][header
[read/write][binary/format/][g.vectors
[read/write][binary/format/][real/complex/][data

Generated by preprocessor from only 3 underlying routines.

HDF5 in progress (Jamal Mustafa, UC Berkeley)

Interfaces to other codes welcome!
M4 configure script, Fortran90 modules
Sort of like ETSF_IO.

Current status: more or less works in Octopus.
Exchange for silane agrees between paratec, paratec -> BerkeleyGW, octopus -> BerkeleyGW. But not with octopus …
Velocity matrix elements for optical spectrum

\[ f_S = \frac{2|e \cdot \langle 0|v|S\rangle|^2}{\Omega^S} \quad (44) \]

We compute the velocity matrix element via the commutator of the many-body Hamiltonian, as follows [43]:

\[ \langle 0|v|S\rangle = \langle 0|i[H,r]|S\rangle = i(E_0 - E_S)\langle 0|r|S\rangle \]
\[ = -i\Omega^S \sum_{vck} A^S_{vck} \langle vk|r|ck\rangle \quad (45) \]

In a periodic system, we cannot calculate matrix elements of the position operator, but we can use a \( q \to 0 \) limit [8]:

\[ \langle vk|r|ck\rangle = \lim_{q \to 0} \frac{\langle vk + q|e^{iq\cdot r} - 1|ck\rangle}{iq} \]
\[ = -i \lim_{q \to 0} \frac{\langle vk + q|e^{iq\cdot r}|ck\rangle}{q} \quad (46) \]
CV(2) theory solves for density of excited state. Approximations: kernel expanded around gs density, finite number of unoccupied states, energies only to second order. 4th-order theory has same eigenvectors.
Density-matrix-based TDDFT approaches

\[ QF_I = \Omega_I^2 F_I, \] (1)

given here in a spin-independent form for simplicity. The matrix elements of \( Q \) are given by

\[ Q_{ij,kl} = \delta_{i,k} \delta_{j,l} \omega_{kl}^2 + 2\sqrt{\lambda_{ij} \omega_{ij}} K_{ij,kl} \sqrt{\lambda_{kl} \omega_{kl}}, \] (2)

\[ K_{ij,kl} = 2 \int \int \psi_i^*(r) \psi_j^*(r) \left( \frac{1}{|r-r'|} + \frac{\delta E_{xc}}{\delta \rho(r) \delta \rho(r')} \right) \psi_k(r') \psi_l(r') drdr'. \]

where \( \lambda_{ij} = n_j - n_i \) is the difference between the occupation numbers of the \( i \) and \( j \) single-particle states, \( \omega_{ij} = \varepsilon_i - \varepsilon_j \) is the difference between the \( i \) and \( j \) eigenvalues, and \( K \) is a coupling matrix whose elements in the adiabatic approximation

\[ f_{\uparrow \uparrow}^{\uparrow \uparrow} - f_{\uparrow \uparrow}^{\uparrow \downarrow} \]
Density-matrix-based TDDFT approaches

\[
K_{ij,kl} = 2 \int \int \psi_i^*(r) \psi_j^*(r) \left( \frac{1}{|r-r'|} + \frac{\delta^2 E_{xc}}{\delta \rho(r) \delta \rho(r')} \right) \times \psi_k(r') \psi_l(r') drdr'.
\]

\[
Q_{ij,kl} = \delta_{i,k} \delta_{j,l} \omega_{kl}^2 + 2 \sqrt{\lambda_{ij} \omega_{ij}} K_{ij,kl} \sqrt{\lambda_{kl} \omega_{kl}},
\]

\[
\begin{bmatrix}
-\Delta E - \kappa & -\kappa \\
\kappa & \Delta E + \kappa
\end{bmatrix} - \omega I = \begin{bmatrix}
-B^{(+\omega, re)} \\
-B^{(-\omega, re)}
\end{bmatrix} = 0.
\]

Eps\_diff: \( K = 0 \)

Petersilka: only diagonal part of \( K \)

Tamm-Dancoff: \( B = 0 \), no \sqrt{s}

CV(2): no \sqrt{s}

\[
\begin{pmatrix}
A & B \\
B^* & A^*
\end{pmatrix}
\begin{pmatrix}
X \\
Y
\end{pmatrix} = \omega \begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
X \\
Y
\end{pmatrix},
\tag{21}
\]

with

\[
A_{ai\sigma , bj\tau} = \delta_{ij} \delta_{ab} \delta_{\sigma\tau} (\varepsilon_{a\sigma} - \varepsilon_{i\tau}) + (a_{\sigma} i_{\sigma} | j_{\tau} b_{\tau}) + (a_{\sigma} i_{\sigma} | w_{\sigma\tau} | j_{\tau} b_{\tau}),
\tag{22}
\]

\[
B_{ai\sigma , bj\tau} = (a_{\sigma} i_{\sigma} | b_{\tau} j_{\tau}) + (a_{\sigma} i_{\sigma} | w_{\sigma\tau} | b_{\tau} j_{\tau}).
\tag{23}
\]
### Results for $N_2$

Comparing theory levels for singlet excitations (eV)
9 unoccupied states

<table>
<thead>
<tr>
<th>Index</th>
<th>IPA</th>
<th>Petersilka</th>
<th>Tamm-Dancoff</th>
<th>CV(2)</th>
<th>Casida</th>
<th>Exp’t</th>
</tr>
</thead>
</table>

Tamm-Dancoff occupations: 8.5 eV (mix of two degenerate versions of #1)
1.997, 2.000, 1.999, 1.999, 1.005, 0.499, 0.499, 0.000, 0.002
Excited-state forces within time-dependent density-functional theory: A frequency-domain approach

Amit Sitt and Leeor Kronik*
Department of Materials and Interfaces, Weizmann Institute of Science, Rehovoth 76100, Israel

Sohrab Ismail-Beigi
Department of Applied Physics, Yale University, New Haven, Connecticut 06520, USA

James R. Chelikowsky
Center for Computational Materials, Institute for Computational Engineering and Science, Departments of Physics and Chemical Engineering, University of Texas, Austin, Texas 78712, USA
(Received 28 May 2007; published 2 November 2007)
Excited-state forces

$$\Omega_I^2 = F_I^\dagger \Omega_I^2 F_I = F_I^\dagger Q F_I,$$

$$\partial_{R_a} \Omega_I = \frac{\partial_{R_a} (F_I^\dagger Q F_I)}{2 \Omega_I}.$$ 

$$D_{\alpha\beta,\gamma\delta} = \int \int \partial_{R_a}[\psi_\alpha^*(r)] \psi_\beta^*(r) \left( \frac{1}{|r-r'|} \right) \psi_\gamma(r') \psi_\delta(r') dr dr'.$$

$$P_{\alpha\beta,\gamma\delta} = \int \int \partial_{R_a}[\psi_\alpha^*(r)] \psi_\beta^*(r)$$

$$\times \left( \frac{\delta^2 E_{xc}}{\delta \rho(r) \delta \rho(r')} \right) \psi_\gamma(r') \psi_\delta(r') dr dr'.$$

$$= \int \psi_i^*(r) \psi_j^*(r) \left( \frac{\delta^3 E_{xc}}{\delta \rho^3(r)} \right) \frac{\partial \rho}{\partial R_a} \psi_k(r) \psi_l(r) dr.$$
Excited-state forces

For the proof-of-concept calculations presented below, all pertinent derivatives were calculated by explicit small nucleus displacements with all derivatives of ground-state quantities calculated by finite difference. While this approach is straightforward and relatively easy to implement, it is clearly computationally inefficient. However, as shown below, it suffices for testing the above formalism. We expect the use of density-functional perturbation theory [23] to increase the efficiency substantially.
**Debugging**

Here are several strategies that can be used.

- **Compile with** `-g` `-Wall` `-Mbounds` `-Werror` **all etc. depending on compiler to check array bounds etc.**

- **Valgrind** CPU emulator to find memory leaks (available as Ubuntu package) [http://valgrind.org](http://valgrind.org). Compile with `-g`, then run executable in valgrind.

- **GDB** GNU debugger (available as Ubuntu package) [http://sourceware.org/gdb](http://sourceware.org/gdb). On supercomputers, packages such as Totalview or DDT are available for parallel debugging. Compile with `-g`, then run executable in debugger.

- **Electric-fence** (available as Ubuntu package). Link code with `-lelfence` and code will die with segmentation fault at the origin of memory corruption errors. Be careful, it uses a ton of extra memory. [http://manpages.ubuntu.com/manpages/intrepid/man3/libelfence.3.html](http://manpages.ubuntu.com/manpages/intrepid/man3/libelfence.3.html) written at Pixar!


- **Link code with** `-ffpe-trap=invalid,zero,overflow,underflow` **(gcc only)** to catch floating-point exceptions [http://gcc.gnu.org/onlinedocs/gcc-4.5.3/gfortran/Debugging-Options.html](http://gcc.gnu.org/onlinedocs/gcc-4.5.3/gfortran/Debugging-Options.html) so you can see where NaN's are created.

**General reference:** [http://flylib.com/books/en/1.381.1.37/1/](http://flylib.com/books/en/1.381.1.37/1/)

Optimization: erroneous results at `-O3` probably indicate latent problems at `-O0`. 