BerkeleyGW manual (version 1.0.6)

Contents

About

This manual is assembled automatically from the various documentation files in the BerkeleyGW distribution. The headings are the paths and filenames. For each executable, consult the corresponding .inp file for information about input parameters. The homepage is http://www.berkeleygw.org.

Preliminaries

README

BerkeleyGW - The Berkeley GW (And More) Package
http://www.berkeleygw.org

Version 1.0   (Aug, 2011)
J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain

Version 0.5   (July, 2008)
J. Deslippe, G. Samsonidze, L. Yang, F. Ribeiro

```
  Version 0.2   M. L. Tiago     (2001)  Original BSE Code, PlotXct
                S. Ismail-Beigi (2002)  FFT, Documentation, General Improvements
                C. Spataru      (2005)  Cylindrical Coulomb Truncation, Transform (obsolete)
  Version 0.1   X. Blase        (1998)  Implemented F90, Preprocessing, Dynamical Memory
Allocation
                G. M. Rignanese                                 "                "
                E. Chang                                        "                "
  Version 0.0   M. Hybertsen    (1985)  Original GW Code
```

To report current bugs or problems, contact Jack Deslippe: jdeslip@civet.berkeley.edu

----

See more details in the README and .inp files in individual directories.

Compilation

We have tested the code extensively with various configurations, and support the following compilers and libraries:
* Operating systems: Linux, AIX, MacOS
* Fortran compilers (required): pgf90, ifort, gfortran, g95, openf90, sunf90, pathf90, crayftn, af90 (Absoft), nagfor, xlf90 (experimental)
* C compilers (optional): pgcc, icc, gcc, opencc, pathcc, craycc, clang
* C++ compilers (optional): pgCC, icpc, g++, openCC, pathCC, crayCC, clang++
* MPI implementation (optional): OpenMPI, MPICH1, MPICH2, MVAPICH2, Intel MPI
* 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.x

1. Architecture-specific Makefile-include files appropriate for various supercomputers as well as
   for using standard Ubuntu or Macports packages are provided in the config directory. Copy or

```
     link one to the top directory. Example:

     cp config/lonestar.tacc.utexas.edu.mk arch.mk

  2. Edit it to fit your needs. Refer to config/README for details.

  3. Copy flavor_real.mk or flavor_cplx.mk to flavor.mk to set flavor.
     Complex may always be used. Real may be used for systems that
     have both inversion (about the origin) and time-reversal symmetry, and will be
     faster and use less memory.

  4. Stay in the root directory and run the following commands
     to compile the various codes:

     MAIN:
     make epsilon
     make epsilon-all               Epsilon + utilities
     make sigma
     make sigma-all                 Sigma + utilities
     make bse
     make bse-all                   BSE + utilities

     MISC:
     make plotxct
     make meanfield
     make epm
     make sapo
     make siesta2bgw
     make bgw2para
     make kgrid
     make icm
     make surface

     EVERYTHING:
     make all                       All codes/packages (for selected flavor)
     make -j all-j                  parallel build (for selected flavor)
     make all-flavors               everything, for both flavors
     make -j all-flavors            parallel build of everything, for both flavors
     make install INSTDIR=          install binaries, library, examples, testsuite into
specified prefix

     These commands will generate executables with extension .x in
     the source directory and symbolic links with the same name in
     the bin directory.

  5. Test your build. See testsuite/README for more info.
```

Development

Spirit of contributions:

```
  0. Follow guidelines at https://civet.berkeley.edu/BerkeleyGW/wiki/CodingStandards
  1. These codes are to be used on various machine architectures
     employing various compilers and libraries. Therefore, try not
     to write machine-specific code or quick fixes that will have to
     be reversed later.
  2. Limit compilation changes as much as possible to arch.mk and ensure
     that this is suitable for all codes in Epsilon/, Sigma/, and BSE/.
     Hopefully individual Makefiles will only require the addition of
     new source-code files.
  3. Test the codes using the examples/ and please augment the examples/
     with your own test cases (provided they are not too computationally
     expensive).
```

license.txt

CONTRIBUTORS

```
Xavier Blase (XAV)
Andrew Canning (AC)
Eric K. Chang (EKC)
Sangkook Choi (SKC)
Jack R. Deslippe (JRD)
Peter W. Doak (PWD)
Mark S. Hybertsen (MSH)
Sohrab Ismail-Beigi (SIB)
Manish Jain (MJ)
Felipe H. Jornada (FHJ)
Je-Luen Li (JLL)
```

```
Andrei S. Malashevich (ASM)
Brad D. Malone (BDM)
Jeffrey B. Neaton (JBN)
Cheol-Hwan Park (CHP)
David G. Prendergast (DGP)
Filipe J. Ribeiro (FJR)
Gian-Marco Rignanese (GMR)
Georgy Samsonidze (GSM)
Sahar Sharifzadeh (SS)
Catalin D. Spataru (CDS)
David A. Strubbe (DAS)
Murilo L. Tiago (MLT)
Derek W. Vigil (DWV)
Li Yang (LY)
```

LITERATURE.html

Please let us know when your paper is published with BerkeleyGW for inclusion here (or any other additions or corrections) in the Forum, under the "Literature" topic.

I. Papers on the implementation of BerkeleyGW

1. Jack Deslippe, Georgy Samsonidze, David A. Strubbe, Manish Jain, Marvin L. Cohen, and Steven G. Louie, "BerkeleyGW: A Massively Parallel Computer Package for the Calculation of the Quasiparticle and Optical Properties of Materials and Nanostructures," Comput. Phys. Commun. 183, 1269 (2012) (most up-to-date on arXiV)
2. Mark S. Hybertsen and Steven G. Louie, "Electron correlation in semiconductors and insulators: Band gaps and quasiparticle energies," Phys. Rev. B 34, 5390 (1986) [GW, GPP, COHSEX] Errata: Eq. 11 should have E' instead of E in the numerator. Eq. 32 should have Ω^2 rather than Ω in the numerator. Eq. 34a should have $\delta_{\mathbf{G},\mathbf{G}}$ instead of 1 in the parentheses.
3. Sheng Bai Zhang, David Tománek, Marvin L. Cohen, Steven G. Louie, and Mark S. Hybertsen, "Evaluation of quasiparticle energies for semiconductors without inversion symmetry," Phys. Rev. B 40, 3162 (1989) [GPP for systems without inversion symmetry]
4. Michael Rohlfing and Steven G. Louie, "Electron-hole excitations and optical spectra from first principles," Phys. Rev. B 62, 4927 (2000) [BSE] Errata: Eqs. 26 and 27 should have $8\pi^2$ instead of 16π in the prefactor. Eq. 44 should be a sum over $\mathbf{G} \neq 0$. Eq. 45 should have $\varepsilon^{-1}_{\mathbf{G},\mathbf{G}'}$ instead of $\varepsilon^{-1}_{\mathbf{G},0}$.
5. Je-Luen Li, Gian-Marco Rignanese, Eric K. Chang, Xavier Blase, and Steven G. Louie, "GW study of the metal-insulator transition of bcc hydrogen," Phys. Rev. B 66, 035102 (2002) [spin-polarized GW]
6. Sohrab Ismail-Beigi, "Truncation of periodic image interactions for confined systems," Phys. Rev. B 73, 233103 (2006) [slab and wire truncation]
7. Jeffrey B. Neaton, Mark S. Hybertsen, and Steven G. Louie, "Renormalization of Molecular Electronic Levels at Metal-Molecule Interfaces," Phys. Rev. Lett. 97, 216405 (2006) [ICM] Erratum: p. 3, left col, last paragraph. Should be $1/2|z-z_0|$ instead of $1/4|z-z_0|$.
8. R. Haydock, "The recursive solution of the Schrödinger equation," Comput. Phys. Commun. 20, 11 (1980)
9. Loren X. Benedict and Eric L. Shirley, "Ab initio calculation of $\varepsilon_2(\omega)$ including the electron-hole interaction: Application to GaN and CaF_2," Phys. Rev. B 59, 5441 (1999) [Haydock recursion in BSE]
10. Georgy Samsonidze, Manish Jain, Jack Deslippe, Marvin L. Cohen, and Steven G. Louie, "Simple approximate physical orbitals for GW quasiparticle calculations," Phys. Rev. Lett. 107, 186404 (2011) [SAPO]

II. Review articles on the GW approximation and Bethe-Salpeter equation approaches

1. Lars Hedin and Stig Lundqvist, "Effects of Electron-Electron and Electron-Phonon Interactions on the One-Electron States of Solids," Solid State Phys. 23, 1 (1970)
2. Mark S. Hybertsen and Steven G. Louie, "Theory and Calculation of Quasiparticle Energies and Band Gaps," Comments on Cond. Mat. Phys. 13, 223 (1987)

3. G. Strinati, "Application of the Green's functions method to the study of the optical properties of semiconductors," Riv. Nuovo Cimento 11, 1 (1988)
4. Steven G. Louie, "Quasiparticle Theory of Electron Excitations in Solids," in Quantum Theory of Real Materials, eds. James R. Chelikowsky and Steven G. Louie, (Kluwer Press, Boston, 1996), p. 83
5. Steven G. Louie, "First-Principles Theory of Electron Excitation Energies in Solids, Surfaces, and Defects," in Topics in Computational Materials Science, ed. Ching-Yao Fong (World Scientific, Singapore, 1998) p. 96
6. F. Aryasetiawan and O. Gunnarsson, "The GW method," Rep. Prog. Phys. 61, 237 (1998)
7. Lars Hedin, "On correlation effects in electron spectroscopies and the GW approximation," J. Phys.: Condens. Matter 11, R489 (1999)
8. Wilfried G. Aulbur, Lars Jönsson, and John W. Wilkins, "Quasiparticle calculations in solids," Solid State Phys. 54, 1 (1999)
9. Giovanni Onida, Lucia Reining, and Angel Rubio, "Electronic excitations: density-functional versus many-body Green's-function approaches," Rev. Mod. Phys. 74, 601 (2002)
10. Steven G. Louie, "Predicting Materials and Properties: Theory of the Ground and Excited State," in Conceptual Foundations of Materials: A Standard Model for Ground- and Excited-State Properties, vol. eds. Steven G. Louie and Marvin L. Cohen (Elsevier, Amsterdam, 2006) p. 9

III. Papers using BerkeleyGW

1. Mark S. Hybertsen and Steven G. Louie, "First Principles Theory of Quasiparticles: Calculation of Band Gaps in Semiconductors and Insulators," Phys. Rev. Lett. 55, 1418 (1985)
2. Mark S. Hybertsen and Steven G. Louie, "Electron Correlation and the Band Gap in Ionic Crystals," Phys. Rev. B 32, 7005(R) (1985) [Erratum: Phys. Rev. B 35, 9308 (1987)]
3. Mark S. Hybertsen and Steven G. Louie, "Many-body Calculation of Surface States: As on Ge(111)," Phys. Rev. Lett. 58, 1551 (1987)
4. John E. Northrup, Mark S. Hybertsen, and Steven G. Louie, "Theory of Quasiparticle Energies in Alkali Metals," Phys. Rev. Lett. 59, 819 (1987)
5. Steven G. Louie and Mark S. Hybertsen, "Theory of Quasiparticle Energies: Band Gaps and Excitation Spectra in Solids," Int. J. Quant. Chem.: Quant. Chem. Symp. 21, 31 (1987)
6. Sheng Bai Zhang, David Tománek, Steven G. Louie, Marvin L. Cohen, and Mark S. Hybertsen, "Quasiparticle Calculation of Valence Band Offset of AlAs-GaAs(001)," Solid State Comm. 66, 585 (1988)
7. Mark S. Hybertsen and Steven G. Louie, "Theory of Quasiparticle Surface States in Semiconductor Surfaces," Phys. Rev. B 38, 4033 (1988)
8. Michael P. Surh, John E. Northrup, and Steven G. Louie, "Occupied Quasiparticle Bandwidth of Potassium," Phys. Rev. B 38, 5976 (1988)
9. Xuejun Zhu, Stephen Fahy, and Steven G. Louie, "Ab Initio Calculation of Pressure Coefficients of Band Gaps of Silicon: Comparison of the Local-Density Approximation and Quasiparticle Results," Phys. Rev. B 39, 7840 (1989) [Erratum: Phys. Rev. B 40, 5821 (1989)]
10. John E. Northrup, Mark S. Hybertsen, and Steven G. Louie, "Quasiparticle excitation spectrum for nearly-free-electron metals," Phys. Rev. B 39, 8198 (1989)
11. Sheng Bai Zhang, David Tománek, Marvin L. Cohen, Steven G. Louie, and Mark S. Hybertsen, "Evaluation of Quasiparticle Energies for Semiconductors without Inversion Symmetry," Phys. Rev. B 40, 3162 (1989)
12. Sheng Bai Zhang, Mark S. Hybertsen, Marvin L. Cohen, Steven G. Louie, and David Tománek, "Quasiparticle Band Gaps for Ultrathin GaAs/AlAs(001) Superlattices," Phys. Rev. Lett. 63, 1495 (1989)
13. Xuejun Zhu, Sheng Bai Zhang, Steven G. Louie, and Marvin L. Cohen, "Quasiparticle Interpretation of Photoemission Spectra and Optical Properties of GaAs(110)," Phys. Rev. Lett. 63, 2112 (1989)
14. Sheng Bai Zhang, Marvin L. Cohen, Steven G. Louie, David Tománek, and Mark S. Hybertsen, "Quasiparticle Band Offset at the (001) Interface and Band Gaps in Ultrathin Superlattices of GaAs-AlAs Heterojunctions," Phys. Rev. B 41, 10058 (1990)
15. Hélio Chacham, Xuejun Zhu, and Steven G. Louie, "Metal-Insulator Transition in Solid Xenon at High Pressures," Europhys. Lett. 14, 65 (1991)
16. Hélio Chacham and Steven G. Louie, "Metallization of Solid Hydrogen at Megabar Pressures: A First-Principles Quasiparticle Study," Phys. Rev. Lett. 66, 64 (1991)
17. John E. Northrup, Mark S. Hybertsen, and Steven G. Louie, "Many-Body Calculation of the Surface State Energies for Si(111)2×1," Phys. Rev. Lett. 66, 500 (1991)

18. Michael P. Surh, Steven G. Louie, and Marvin L. Cohen, "Quasiparticle Energies for Cubic BN, BP, and BAs," Phys. Rev. B 43, 9126 (1991)
19. Xuejun Zhu and Steven G. Louie, "Quasiparticle Surface Band Structure and Photoelectric Threshold of Ge(111)-2×1," Phys. Rev. B 43, 12146(R) (1991)
20. Xuejun Zhu and Steven G. Louie, "Quasiparticle Band Structure of Thirteen Semiconductors and Insulators," Phys. Rev. B 43, 14142 (1991)
21. Michael P. Surh, Steven G. Louie, and Marvin L. Cohen, "Band Gaps of Diamond under Anisotropic Stress," Phys. Rev. B 45, 8239 (1992)
22. Hélio Chacham, Xuejun Zhu, and Steven G. Louie, "Pressure-Induced Insulator-Metal Transitions in Solid Xenon and Hydrogen: A First-Principles Quasiparticle Study," Phys. Rev. B 46, 6688 (1992) [Erratum: Phys. Rev. B 48, 2025(E) (1993)]
23. Eric L. Shirley, Xuejun Zhu, and Steven G. Louie, "Core-Polarization in Semiconductors: Effects on Quasiparticle Energies," Phys. Rev. Lett. 69, 2955 (1992)
24. Eric L. Shirley and Steven G. Louie, "Electron Excitations in Solid C_{60}: Energy Gap, Band Dispersions, and Effects of Orientational Disorder," Phys. Rev. Lett. 71, 133 (1993)
25. Angel Rubio, Jennifer L. Corkill, Marvin L. Cohen, Eric L. Shirley, and Steven G. Louie, "Quasiparticle Band Structure of AlN and GaN," Phys. Rev. B 48, 11810 (1993)
26. Steven G. Louie and Eric L. Shirley, "Electron Excitation Energies in Fullerites: Many-Electron and Molecular Orientational Effects," J. Phys. Chem. Solids 54, 1767 (1993)
27. Xavier Blase, Xuejun Zhu, and Steven G. Louie, "Self-Energy Effects on the Surface State Energies of H-Si(111) 1×1," Phys. Rev. B 49, 4973 (1994)
28. J. A. Carlisle, L. J. Terminello, A. V. Hamza, E. A. Hudson, Eric L. Shirley, F. J. Himpsel, D. A. Lapiano-Smith, J. J. Jia, T. A. Callcott, R. C. C. Perera, D. K. Shuh, Steven G. Louie, J. Stöhr, M. G. Samant, and D. L. Ederer, "Occupied and Unoccupied Orbitals of C_{60} and C_{70}," Mol. Cryst. Liq. Cryst. 256, 819 (1994)
29. Oleg Zakharov, Angel Rubio, Xavier Blase, Marvin L. Cohen, and Steven G. Louie, "Quasiparticle Band Structures of Six II-VI Compounds: ZnS, ZnSe, ZnTe, CdS, CdSe, and CdTe," Phys. Rev. B 50, 10780 (1994)
30. Xavier Blase, Angel Rubio, Steven G. Louie, and Marvin L. Cohen, "Quasiparticle band structure of bulk hexagonal boron nitride and related systems," Phys. Rev. B 51, 6868 (1995)
31. Michael P. Surh, Hélio Chacham, and Steven G. Louie, "Quasiparticle Excitation Energies for the F-Center Defect in LiCl," Phys. Rev. B. 51, 7464 (1995)
32. Eric L. Shirley and Steven G. Louie, "Photoemission and Optical Properties of C_{60} Fullerites," in Quantum Theory of Real Materials, eds. J.R. Chelikowsky and Steven G. Louie (Kluwer Press, Boston, 1996), p. 515.
33. Balazs KrÃ¡lik, Eric K. Chang, and Steven G. Louie, "Structural Properties and Quasiparticle Band Structure of Zirconia," Phys. Rev. B 57, 7027 (1998)
34. Michael Rohlfing and Steven G. Louie, "Electron-hole Excitations in Semiconductors and Insulators," Phys. Rev. Lett. 81, 2312 (1998)
35. Michael Rohlfing and Steven G. Louie, "Optical Excitations in Conjugated Polymers," Phys. Rev. Lett. 82, 1959 (1999)
36. Michael Rohlfing and Steven G. Louie, "Excitons and Optical Spectrum of the Si(111)-(2×1) Surface," Phys. Rev. Lett. 83, 856 (1999)
37. Eric K. Chang, Michael Rohlfing, and Steven G. Louie, "Excitons and Optical Properties of Alpha-Quartz," Phys Rev. Lett. 85, 2613 (2000)
38. Eric K. Chang, Michael Rohlfing, and Steven G. Louie, "First-Principles Study of Optical Excitations in Alpha-Quartz," in The Optical Properties of Materials, MRS Symp. Proceed. Vol. 579, eds. Eric L. Shirley, James R. Chelikowsky, Steven G. Louie, and Gérard Martinez (Materials Research Society, Warrendale, 2000), p. 3
39. Peihong Zhang, Vincent H. Crespi, Eric K. Chang, Steven G. Louie, and Marvin L. Cohen, "Computational Design of Direct-Bandgap Semiconductors that Lattice-Match Silicon," Nature 409, 69 (2001)
40. Jeffrey C. Grossman, Michael Rohlfing, Lubos Mitas, Steven G. Louie, and Marvin L. Cohen, "High Accuracy Many-Body Calculational Approaches for Excitations in Molecules," Phys. Rev. Lett. 86, 472 (2001)
41. Gian-Marco Rignanese, Xavier Blase, and Steven G. Louie, "Quasiparticle Effects on Tunneling Currents: A Study of C2H4 Adsorbed on the Si(001)-(2×1) Surface," Phys. Rev. Lett. 86, 2110 (2001)
42. Eric K. Chang, Xavier Blase, and Steven G. Louie, "Quasiparticle Band Structure of Lanthanum Hydride,"

Phys. Rev. B 64, 155108 (2001)

43. Catalin D. Spataru, M. A. Cazalilla, Angel Rubio, Loren X. Benedict, Pedro M. Echenique, and Steven G. Louie, "Anomalous Quasiparticle Lifetime in Graphite: Band Structure Effects," Phys. Rev. Lett. 87, 246405 (2001)
44. Je-Luen Li, Gian-Marco Rignanese, Eric K. Chang, Xavier Blase, and Steven G. Louie, "GW Study of the Metal-Insulator Transition of bcc Hydrogen," Phys. Rev. B 66, 035102 (2002)
45. Weidong Luo, Sohrab Ismail-Beigi, Marvin L. Cohen, and Steven G. Louie, "Quasiparticle Band Structure of ZnS and ZnSe," Phys. Rev. B 66, 195215 (2002)
46. Loren X. Benedict, Catalin D. Spataru, and Steven G. Louie, "Quasiparticle Properties of a Simple Metal at High Electron Temperatures," Phys. Rev. B 66, 085116 (2002)
47. J. A. Alford, Mei-Yin Chou, Eric K. Chang, and Steven G. Louie, "First-principles Studies of Quasiparticle Band Structures of Cubic YH_3 and LaH_3," Phys. Rev. B 67, 125110 (2003)
48. Murilo L. Tiago, John E. Northrup, and Steven G. Louie, "Ab initio calculation of the electronic and optical properties of solid pentacene," Phys. Rev. B 67, 11 5212 (2003)
49. Sohrab Ismail-Beigi and Steven G. Louie, "Excited-State Forces within a First-Principles Green's Function Formalism," Phys. Rev. Lett. 90, 076401 (2003)
50. Catalin D. Spataru, Sohrab Ismail-Beigi, Loren X. Benedict, and Steven G. Louie, "Excitonic effects and optical spectra of single-walled carbon nanotubes," Phys. Rev. Lett. 92, 077402 (2004)
51. Catalin D. Spataru, Sohrab Ismail-Beigi, Loren X. Benedict, and Steven G. Louie, "Quasiparticle energies, excitonic effects and optical absorption spectra of small-diameter single-walled carbon nanotubes," Appl. Phys. A 78, 1129 (2004)
52. Murilo L. Tiago, Sohrab Ismail-Beigi, and Steven G. Louie, "Effect of Semicore Orbitals on the Electronic Band Gaps of Si, Ge, and GaAs Within the GW Approximation," Phys. Rev. B 69, 125212 (2004)
53. Catalin D. Spataru, Loren X. Benedict, and Steven G. Louie, "Ab Initio Calculation of Band-Gap Renormalization in Highly Excited GaAs," Phys. Rev. B 69, 205204 (2004)
54. Murilo L. Tiago, Michael Rohlfing, and Steven G. Louie, "Bound Excitons and Optical Properties of Bulk Trans-Polyacetylene," Phys. Rev. B 70, 193204 (2004)
55. Je-Luen Li, Gian-Marco Rignanese, and Steven G. Louie, "Quasiparticle Energy Bands of NiO in the GW Approximation," Phys. Rev B 71, 193102 (2005)
56. Catalin D. Spataru, Sohrab Ismail-Beigi, Loren X. Benedict, and Steven G. Louie, "Excitonic Effects and Optical Spectra of Single-Walled Carbon Nanotubes," 27th Conference on the Physics of Semiconductors, AIP Conference Proceedings 772, 1061 (2005)
57. Jeffrey B. Neaton, Koonghong Khoo, Catalin D. Spataru, and Steven G. Louie, "Electronic Transport and Optical Properties of Carbon Nanostructures from First Principles," Comput. Phys. Commun. 169, 1 (2005)
58. Sohrab Ismail-Beigi and Steven G. Louie, "Self-Trapped Excitons in Silicon Dioxide: Mechanism and Properties," Phys. Rev. Lett. 95, 156401 (2005)
59. Catalin D. Spataru, Sohrab Ismail-Beigi, Rodrigo B. Capaz, and Steven G. Louie, "Theory and Ab Initio Calculation of Radiative Lifetime of Excitons in Semiconducting Carbon Nanotubes," Phys. Rev. Lett. 95, 247402 (2005)
60. Steven G. Louie and Angel Rubio, "Quasiparticle and Optical Properties of Solids and Nanostructures: The GW-BSE Approach," Handbook of Materials Modeling, ed. S. Yip (Springer, Dordrecht, The Netherlands, 2005), p. 215
61. Cheol-Hwan Park, Catalin D. Spataru, and Steven G. Louie, "Excitons and Many-Electron Effects in the Optical Response of Single-Walled Boron Nitride Nanotubes," Phys. Rev. Lett. 96, 126105 (2006)
62. Jeffrey B. Neaton, Mark S. Hybertsen, and Steven G. Louie, "Renormalization of Molecular Electronic Levels at Metal-Molecule Interfaces," Phys. Rev. Lett. 97, 216405 (2006)
63. Su Ying Quek, Jeffrey B. Neaton, Mark S. Hybertsen, E. Kaxiras, and Steven G. Louie, "First-principles Studies of the Electronic Structure of Cyclopentene on Si(001): Density Functional Theory and GW Calculations," Phys. Status Solidi (b) 243, 2048 (2006)
64. Rodrigo B. Capaz, Catalin D. Spataru, Sohrab Ismail-Beigi, and Steven G. Louie, "Diameter and Chirality Dependence of Exciton Properties in Carbon Nanotubes," Phys. Rev. B 74, 121401 (2006)
65. Takashi Miyake, Peihong Zhang, Marvin L. Cohen, and Steven G. Louie, "Quasiparticle Energy of Semicore d-electrons in ZnS: Combined LDA+U and GW approach," Phys. Rev. B 74, 245213 (2006)
66. Li Yang, Catalin D. Spataru, Steven G. Louie, and Mei-Yin Chou, "Enhanced Electron-hole Interaction and Optical Absorption in a Silicon Nanowire," Phys. Rev. B 75, 201304(R) (2007)

67. Li Yang, Cheol-Hwan Park, Young-Woo Son, Marvin L. Cohen, and Steven G. Louie, "Quasiparticle Energies and Band Gaps of Graphene Nanoribbons," Phys. Rev. Lett. 99, 186801 (2007)
68. Li Yang, Marvin L. Cohen, and Steven G. Louie, "Excitonic Effects in the Optical Spectra of Graphene Nanoribbons," Nano Lett. 7, 3112 (2007)
69. Feng Wang, David J. Cho, Brian Kessler, Jack Deslippe, P. James Schuck, Steven G. Louie, Alex Zettl, Tony F. Heinz, and Y. Ron Shen, "Observation of Excitons in One-Dimensional Metallic Single-Walled Carbon Nanotubes," Phys. Rev. Lett. 99, 227401 (2007)
70. Su Ying Quek, Jeffrey B. Neaton, Mark S. Hybertsen, Efthimios Kaxiras, and Steven G. Louie, "Negative Differential Resistance in Transport through Organic Molecules on Silicon," Phys. Rev. Lett. 98, 066807 (2007)
71. Catalin D. Spataru, Sohrab Ismail-Beigi, Rodrigo B. Capaz, and Steven G. Louie, "Quasiparticle and Excitonic Effects in the Optical Response of Nanotubes and Nanoribbons," in Carbon Nanotubes: Advanced Topics in the Synthesis, Structure, Properties and Applications, A. Jorio, M. S. Dresselhaus, G. Dresselhaus (eds.), Topics in Applied Physics 111 (Springer-Verlag, Heidelberg, Germany 2008), p. 195
72. Jack Deslippe, Catalin D. Spataru, David Prendergast, and Steven G. Louie, "Bound excitons in metallic single-walled carbon nanotubes," Nano Lett. 7, 1626 (2007)
73. Cheol-Hwan Park, Feliciano Giustino, Marvin L. Cohen, and Steven G. Louie, "Velocity Renormalization and Carrier Lifetime in Graphene from the Electron-Phonon Interaction," Phys. Rev. Lett. 99, 086804 (2007)
74. Brad D. Malone, Jay D. Sau, and Marvin L. Cohen, "Ab initio survey of the electronic structure of tetrahedrally bonded phases of silicon," Phys. Rev. B 78, 035210 (2008)
75. Jack Deslippe and Steven G. Louie, "Excitons and Many-electron Effects in the Optical Response of Carbon Nanotubes and Other One-dimensional Nanostructures," Proc. SPIE 6892, 68920U-1 (2008)
76. Emmanouil Kioupakis, Peihong Zhang, Marvin L. Cohen, and Steven G. Louie, "GW Quasiparticle Corrections to the LDA+U/GGA+U Electronic Structure of bcc Hydrogen," Phys. Rev. B 77, 155114 (2008)
77. Jay D. Sau, Jeffrey B. Neaton, Hyoung Joon Choi, Steven G. Louie, and Marvin L. Cohen, "Electronic Energy Levels of Weakly Coupled Nanostructures: C_{60} Metal Interfaces," Phys. Rev. Lett. 101, 026804 (2008)
78. Li Yang, Marvin L. Cohen, and Steven G. Louie, "Magnetic Edge-state Excitons in Zigzag Graphene Nanoribbons," Phys. Rev. Lett. 101, 186401 (2008)
79. Cheol-Hwan Park, Feliciano Giustino, Jessica L. McChesney, Aaron Bostwick, Taisuke Ohta, Eli Rotenberg, Marvin L. Cohen, and Steven G. Louie, "Van Hove Singularity and Apparent Anisotropy in the Electron-phonon Interaction in Graphene," Phys. Rev. B 77, 113410 (2008)
80. Cheol-Hwan Park, Feliciano Giustino, Marvin L. Cohen, and Steven G. Louie, "Electron-phonon Interactions in Graphene, Bilayer Graphene, and Graphite," Nano Lett. 8, 4229 (2008)
81. Cheol-Hwan Park, Feliciano Giustino, Catalin D. Spataru, Marvin L. Cohen, and Steven G. Louie, "First-principles Study of Electron Linewidths in Graphene," Phys. Rev. Lett. 102, 076803 (2009) [Erratum: Phys. Rev. Lett. 102, 189904(E) (2009)]
82. Jack Deslippe, Mario Dipoppa, David Prendergast, M. V. O. Moutinho, Rodrigo B. Capaz, and Steven G. Louie, "Electron-hole Interaction in Carbon Nanotubes: Novel Screening and Exciton Excitation Spectra," Nano Lett. 9, 1330 (2009)
83. Li Yang, Jack Deslippe, Cheol-Hwan Park, Marvin L. Cohen, and Steven G. Louie, "Excitonic effects on the optical response of graphene and bilayer graphene," Phys. Rev. Lett. 103, 186802 (2009)
84. Chenggang Tao, J. Sun, X. Zhang, Ryan Yamachika, Daniel Wegner, Yasaman Bahri, Georgy Samsonidze, Marvin L. Cohen, Steven G. Louie, T. Don Tilley, Rachel A. Segalman, and Michael F. Crommie, "Spatial resolution of a type II heterojunction in a single bipolar molecule," Nano Lett. 9, 3963 (2009)
85. Cheol-Hwan Park, Feliciano Giustino, Catalin D. Spataru, Marvin L. Cohen, and Steven G. Louie, "Angle-resolved Photoemission Spectra of Graphene from First-principles," Nano Lett. 9, 4234 (2009)
86. Emmanouil Kioupakis, Murilo L. Tiago, and Steven G. Louie, "Quasiparticle electronic structure of bismuth telluride in the GW approximation," Phys. Rev. B 82, 245203 (2010)
87. Feliciano Giustino, Steven G. Louie, and Marvin L. Cohen, "Electron-phonon renormalization of the direct band gap of diamond," Phys. Rev. Lett. 105, 265501 (2010)
88. Brad D. Malone, Steven G. Louie, and Marvin L. Cohen, "Electronic and optical properties of body-centered-tetragonal Si and Ge," Phys. Rev. B 81, 115201 (2010)
89. Bi-Ching Shih, Yu Xue, Peihong Zhang, Marvin L. Cohen, and Steven G. Louie, "Quasiparticle band gap of ZnO: High accuracy from the conventional G_0W_0 approach," Phys. Rev. Lett. 105, 146401 (2010)

90. Victor W. Brar, Sebastian Wickenburg, Melissa Panlasigui, Cheol-Hwan Park, Tim O. Wehling, Yuanbo Zhang, R. Decker, Caglar Girit, A. V. Balatsky, Steven G. Louie, Alex Zettl, and Michael F. Crommie, "Observation of Carrier-Density-Dependent Many-Body Effects in Graphene via Tunneling Spectroscopy," Phys. Rev. Lett. 104, 036805 (2010)
91. G. S. Do, J. Kim, Seung-Hoon Jhi, Cheol-Hwan Park, Steven G. Louie, and Marvin L. Cohen, "Ab Initio Calculations of Pressure-induced Structural Phase Transitions of GeTe," Phys. Rev. B 82, 054121 (2010)
92. Li Yang, "First-principles study of the optical absorption spectra of electrically gated bilayer graphene," Phys. Rev. B 81, 155445 (2010)
93. David A. Siegel, Cheol-Hwan Park, C. Hwang, Jack Deslippe, A. V. Federov, Steven G. Louie, and Alessandra Lanzara, "Many-body Interactions in Quasi-freestanding Graphene," Proc. Natl. Acad. Sci. U.S.A. 108, 11365 (2011)
94. H. C. Hsueh, G. Y. Guo, and Steven G. Louie, "Excitonic Effects in the Optical Properties of a SiC Sheet and Nanotubes," Phys. Rev. B 84, 85404 (2011)
95. Georgy Samsonidze, Manish Jain, Jack Deslippe, Marvin L. Cohen, and Steven G. Louie, "Simple approximate physical orbitals for GW quasiparticle calculations," Phys. Rev. Lett. 107, 186404 (2011)
96. Georgy Samsonidze, Marvin L. Cohen, and Steven G. Louie, "Compensation-doped silicon for photovoltaic applications," Phys. Rev. B 84, 195201 (2011)
97. Manish Jain, James R. Chelikowsky, and Steven G. Louie, "Quasiparticle Excitations and Charge Transition Levels of Oxygen Vacancies in Hafnia," Phys. Rev. Lett. 107, 216803 (2011)
98. Manish Jain, James R. Chelikowsky, and Steven G. Louie, "Reliability of Hybrid Functionals in Predicting Band Gaps," Phys. Rev. Lett. 107, 216806 (2011)
99. Isaac Tamblyn, Pierre Darancet, Su Ying Quek, Stanimir A. Bonev, and Jeffrey B. Neaton, "Electronic energy level alignment at metal-molecule interfaces with a GW approach," Phys. Rev. B 84, 201402(R) (2011)
100. Li Yang, "Excitons in intrinsic and bilayer graphene," Phys Rev. B 83, 085405 (2011)
101. Li Yang, "Excitonic Effects on Optical Absorption Spectra of Doped Graphene," Nano Lett. 11, 3844 (2011)
102. Oleg V. Yazyev, Emmanouil Kioupakis, Joel E. Moore, and Steven G. Louie, "Quasiparticle effects in the bulk and surface-state bands of Bi_2Se_3 and Bi_2Te_3 topological insulators," Phys. Rev. B 85, 161101(R) (2012)
103. Sahar Sharifzadeh, Ariel Biller, Leeor Kronik, and Jeffrey B. Neaton, "Quasiparticle and Optical Spectroscopy of Organic Semiconductors Pentacene and PTCDA from First Principles," Phys. Rev. B 85, 125307 (2012)
104. Jesse Noffsinger, Emmanouil Kioupakis, Chris G. Van de Walle, Steven G. Louie, and Marvin L. Cohen, "Phonon-Assisted Optical Absorption in Silicon from First Principles," Phys. Rev. Lett. 108, 167402 (2012)
105. Cheol-Hwan Park, Feliciano Giustino, Catalin D. Spataru, Marvin L. Cohen, and Steven G. Louie, "Inelastic Carrier Lifetime in Bilayer Graphene," Appl. Phys. Lett. 100, 032106 (2012)
106. Sahar Sharifzadeh, Isaac Tamblyn, Peter Doak, Pierre Darancet, and Jeffrey B. Neaton, "Quantitative Molecular Orbital Energies within a G_0W_0 Approximation", Eur. Phys. J. B 85, 323 (2012)
107. Brad D. Malone and Marvin L. Cohen, "Quasiparticle semiconductor band structures including spin-orbit interactions," J. Phys.: Condens. Matter 25, 105503 (2013)

---

config/README

```
# Information about how to set the various options here.
# Ideally you can use a file already created for your platform.
# Otherwise, you can work from the one most similar that is here.

# Select the Fortran compiler you are using.
# if you have a different one than these, modifications in Common/compiler.h
# and Common/common-rules.mk will be necessary.
COMPFLAG  = -D{INTEL, PGI, GNU, PATH, XLF, G95, ABSOFT, NAG, OPEN64, SUN, CRAY}

# Use -DMPI to compile Fortran in parallel with MPI. Otherwise is serial.
PARAFLAG  = -DMPI
# -DUSESCALAPACK enables usage of ScaLAPACK (required in parallel for BSE).
# -DUSEESSL uses ESSL instead of LAPACK in some parts of the code.
# -DUNPACKED uses unpacked rather packed representation of the Hamiltonian
# in EPM. Packed LAPACK operations give bad results eventually in Si-EPM kernel
# test for ACML and Cray LibSci sometimes.
```

```
MATHFLAG  = -DUSESCALAPACK -DUSEESSL -DUNPACKED
# For Fortran and C++. -DDEBUG enables extra checking. -DVERBOSE writes extra information.
DEBUGFLAG = -DDEBUG -DVERBOSE
# Same, but for C. Currently, no use of VERBOSE in the code.
C_DEBUGFLAG = -DDEBUG

# Command for the preprocessor. -ansi should generally be used.
# add -P if the Fortran compiler will not accept indication of line numbers.
# Use gcc's (often in /lib/cpp) or a C compiler (even mpicc) with "-E". Some need "-x c" as
well.
FCPP    = cpp -ansi
# Fortran compiler and its flags for F90 and f90 files (free source form)
# With openmp flags you can use threaded libraries.
F90free = {mpif90, mpiifort, ftn, ifort, pgf90, gfortran, ...}
# Fortran compiler and any flag(s) required for linking
LINK    = {mpif90, mpiifort, ftn, ifort, pgf90, gfortran, ...}
# Fortran optimization flags. The levels below will generally work.
FOPTS   = {-fast, -O3}
# Fortran optimization flags for epsilon_main and sigma_main which can have
# trouble in high optimization due to their long length. If below does not
# work, try -O2.
FNOOPTS = $(FOPTS)
# Fortran compiler flag to specify location where module should be created.
# different for each compiler, refer to examples.
MOD_OPT =
# Fortran compiler flag to specify where to look for modules.
INCFLAG = -I

# Use this to compile C++ in parallel with MPI. Leave blank for serial.
# MPICH, MVAPICH, Intel MPI require also -DMPICH_IGNORE_CXX_SEEK, or an error may occur such
as:
# "SEEK_SET is #defined but must not be for the C++ binding of MPI. Include mpi.h before
stdio.h"
C_PARAFLAG  = -DPARA
# C++ compiling command and any needed flags
CC_COMP = {mpiCC, mpicxx, mpiicpc, CC, icpc, pgCC, g++, ...}
# C compiling command and any needed flags
C_COMP  = {mpicc, mpiicc, cc, icc, pgcc, gcc, ...}
# C/C++ link command and any needed flags
C_LINK  = {mpiCC, mpicxx, mpiicpc, CC, icpc, pgCC, g++, ...}
# C/C++ optimization flags
C_OPTS  = -fast
# Note: for Intel compilers, it is equivalent to use icc or icpc on C++ files.

# command to remove, for make clean
REMOVE  = /bin/rm -f

# Math Libraries
# path for FFTW2 library, used in lines below
FFTWPATH     =
# link line for FFTW2 library. sometimes needs to be -ldfftw
FFTWLIB      = -L$(FFTWPATH)/lib -lfftw
# where include file fftw_f77.i from FFTW is located.
# Sometimes supercomputer installations will not have it.
# If not, find it online and copy it
# (e.g. http://www.fifi.org/doc/fftw-dev/fortran/fftw_f77.i)
FFTWINCLUDE  = $(FFTWPATH)/include

# Different styles will apply depending on the package used.
# Must provide BLAS as well as LAPACK: some packages will do this in one library file,
# others will not. See examples in this directory for how to link with MKL, ACML,
# ATLAS, Ubuntu packages, etc. For MKL, if you will use ScaLAPACK, it is most
# convenient to include BLACS here too. For further info on linking with MKL, see
# the link-line advisor at http://software.intel.com/sites/products/mkl/
LAPACKLIB    =
# BLACS must be provided here too, if it is not provided in LAPACKLIB above. See
# examples in this directory for different packages, as with LAPACKLIB. Note that
# you may have problems if ScaLAPACK was not built with the same LAPACK used above.
SCALAPACKLIB =

# Flags for performance profiling with an external tool such as IPM on NERSC
PERFORMANCE  =

# Command to submit testsuite job script from testsuite directory.
# qsub is for PBS. If no scheduler, put make check-parallel here.
# In serial, delete this line.
TESTSCRIPT = qsub {architecture}.scr
```

---

testsuite/README

```
BerkeleyGW testsuite
David Strubbe 2009-2011

== Running tests ==

* To run the testsuite in serial, type "make check" in this directory or the main directory.
"make check-save" will do the same, but retain the working directories for debugging.
* To run in parallel without a scheduler, you can just use "make check-parallel" if you do
not have a scheduler (e.g. PBS). Set environment variable SAVETESTDIRS=yes to save working
directories. The tests use 4 cores, so be sure that at least that many are available, or use
BGW_TEST_NPROCS (see below).
* To run in parallel with a scheduler, type "make check-jobscript" or "make check-jobscript-
save", which will execute the job script for the machine you are using, as specified in
arch.mk by the line TESTSCRIPT. See example job scripts for various machines in this
directory called *.scr. The tests use 4 cores, so be sure to request at least that many in
the job script, or use BGW_TEST_NPROCS (see below).
* Environment variables:
  - TEMPDIRPATH: sets the scratch directory where temporary working directories will be
created. Default is /tmp, but on some machines you may not have permission to write there.
  - MPIEXEC: sets the command for parallel runs. Default is `which mpiexec`. Note that
mpiexec and mpirun are generally equivalent. Set this if you are using a different command
such as 'ibrun' or 'aprun', if you need to use a different mpiexec than the one in your
path, or if some options need to be appended to the command.
  - BGW_TEST_NPROCS: set to overrule the number of processors listed in the test files.
Useful for testing correctness of parallelization, if you don't have 4 cores, or to run the
testsuite faster with more cores.
  - MACHINELIST: if set, will be added to command line for MPI runs. This is needed in some
MPI implementations.
  - EXEC: if set, will be added before name of executable. Used to run code through
valgrind.

Contents:
(1) run_testsuite.sh -- this script runs tests from filenames ending in *.test in this
directory.
(2) run_regression_test.pl -- called from run_testsuite.sh for each individual test.
(3) *.scr -- job scripts for running the testsuite on various parallel machines.
(4) interactive*.sh -- scripts to run in parallel, in interactive mode. You have to follow
the instructions in the script, not just run it.
(5) buildbot_pbs.pl -- a Perl script used by the BuildBot for parallel tests with a PBS
scheduler, which submits a job and watches the queue to see when it finishes.
(6) test directories.
(8) no_hires.sh -- Some clusters will not have the Perl package Time::HiRes installed which
is needed for timing in run_regression_test.pl. You can just deactivate the usage with this
script.

== Writing tests ==

The test files consist of lines beginning with one of a set of tags, parsed by the Perl
script. Comment lines beginning with '#' will be ignored.
Test : title
```

```
  Write a title to output to identify the test. Should be the first tag in the file.
Banner : text
  Write a heading in a box to identify a specific part of a test. Helpful when input files
are generated by sed and do not have a distinct name.
Enabled : Yes/No
  If Yes, will be run; if No, will not be run. Use to turn off a test without deleting it
from the repository. Should be the second tag in the file.
TestGroups : group-name [group-name2 [...]]
  The run_testsuite.sh script can be run with argument "-g" and a list of groups. Then tests
will only be run if there is a match between the argument list and the list in TestGroups.
Examples of groups that could be used: parallel, serial, long. [This tag is actually read by
run_testsuite.sh rather than run_regression_test.pl.]
Executable : program.x
  Name of executable to be run (path is bin directory). Persists for multiple runs, until
next Executable tag.
Processors : integer
  Number of processors to use. Ignored if mpiexec is not available. May not exceed the
number in the reservation in the job scripts.
  Set to special value "serial" to run without mpiexec.
Command : shell-command
  Specify a shell command, which will be executed in the working directory. Useful for
renaming log files that would be overwritten by a subsequent run, or preprocessing input
files by sed or other utilities (after putting in working directory with Copy).
Copy : file.in [file_newname.in]
  Copy a file from test directory to run directory. If new name is not supplied, original
name will be used. Useful if you want to sed a file before it is run.
Unpack : data.tar.gz
  The file data.tar.gz in the test directory will have "tar xzf" run on it, with resulting
files going to the working directory.
Output : file.out
  Output will be piped into the file named here, in the working directory.
Arguments : arg1 arg2
  The current Executable will be executed (in serial) with the text given here as command-
line argument(s). No files will be copied to working directory. Use redirection with ">" to
capture the output if necessary.
Input : file.in [file_newname.in/PIPE/CMDLINE/NONE]
  The file named here will be copied from the test directory to the working directory, for
use as input. If it is followed by "PIPE", the file will be piped into the executable. If it
is followed by "CMDLINE", the file will be given as a command-line argument to the
executable. If it is followed by "NONE", no file will be copied (useful if an input file was
already generated by sed in the working directory). If it is followed by another name, the
file's name in the working directory will be the new name. This tag causes actual execution
of the run that has been set up by previous tags.
Precision : 1e-5
  A floating point number, the tolerance for testing whether a match has passed or failed.
Persists until next Precision tag. Default is 1e-4.
match ; name ; COMMAND(..); reference-value
  Extracts a calculated number from a run and tests it against the reference value. The name
is an identifier printed to output. The number is extracted as the standard output from the
listed COMMAND, which is one of this set:
  . GREP(filename, 'search-text', field, offset)
      Finds the first line in file containing 'search-text' (which MAY NOT contain a comma),
and returns the specified field of that line. "Field" is meant in the sense used by 'awk',
i.e. the line is parsed into white-space separated groups, indexed starting with 1. The
optional 'offset' directs the use of the Mth line after the line containing 'search-text'.
No offset is equivalent to offset = 0. This is the most robust of the commands and should be
used when possible in preference to the others.
  . LINE(filename, line, field)
      Returns the specified field of the specified line number from the file. Use GREP
instead if possible.
  . SHELL(shell-command)
      The result is the standard output of the listed command. Deprecated; use GREP or LINE
unless absolutely necessary.
STOP TEST
  For debugging purposes, to halt test after the part you are studying has run.
```

library/README

```
-----------------------------------------------------------------
----------  BerkeleyGW library  ---------------------------------
-----------------------------------------------------------------

To build other codes with BerkeleyGW output support, type 'make library'
to create libBGW_wfn.a and wfn_rho_vxc_io_m.mod (and dependent modules),
and then compile with -I library/ and link library/libBGW_wfn.a with
the other code.

An m4 macro for configure scripts is provided in this directory, for
use in linking to this library. Codes linking the library should 'use'
the module 'wfn_rho_vxc_io_m'.

To generate real wavefunctions, a Gram-Schmidt process should be used.
The appropriate parallelization scheme etc. will be dependent on the code,
and cannot be easily handled in a general manner here, but examples can
be found in MeanField/SIESTA/real.f90 and MeanField/ESPRESSO/pw2bgw.f90
(routine real_wfng).
```

---

bin/README

```
Brief descriptions of the scripts in this directory:

----Energies-----

qp_shifts.py - from sigma_hp.log file, writes two files which contain the LDA
               energies and the GW shifts (E_qp-E_lda) for that energy. Useful
               for determining scissor shifting parameters ecdel, evdel, etc.

eqp.py       - extracts QP energies from sigma_hp.log file and generates file
               eqp.dat. This file can be read in directly from the BGW
               code if you want to shift the energies for each kpoint and band
               exactly instead of relying on the scissor shift.

----Kpoints-----


kptlist.pl   - extracts a formatted list of k-points from PARATEC file for use in the Sigma
code

qptlist.pl   - extracts a formatted list of q-points from PARATEC file for use in the
Epsilon code
```

---

examples/README

```
Note: ESPRESSO examples are compatible with Quantum ESPRESSO 5.0 and later versions.
If you want to run them with Quantum ESPRESSO 4.3.2 or earlier version, you should modify
the input files for pw.x.
Change "calculation = 'bands'" to "calculation = 'nscf'" and "CELL_PARAMETERS" to
"CELL_PARAMETERS cubic".

This directory contains example runs for a variety of physical systems:
silicon: a bulk semiconductor (includes PREBUILT and reference results; only example for
EPM)
sodium: a bulk metal
swcnt_5-5: (5,5) single-walled carbon nanotube, a metallic 1D system
```

```
swcnt_8-0: (8,0) single-walled carbon nanotube, a semiconducting 1D system
CO: a small molecule
benzene: a slightly larger molecule (includes G-sphere, SIESTA)

Only silicon is available for EPM.
Only benzene has a run with G-sphere utility or SIESTA.
All the DFT examples may be done with either ESPRESSO or PARATEC.
The silicon example additionally has a tarball of DFT inputs from PARATEC in the PREBUILT
directory,
and sample output files from the GW/BSE results as a reference.
Each directory's README tells you how to run it and suggests number of processors and
walltime for parallel calculations.
Needed pseudopotentials and some useful scripts for them are contained in DFT/pseudos.
The Si2_bs DFT example shows how to use inteqp.x to obtain a GW bandstructure for silicon by
interpolation.

There are also examples in the Visual directory for the scripts there, as well as plotxct.
The testsuite runs may also be consulted, although they are not realistic calculations.
```

MeanField

MeanField/README

```
------------------------------------------------------------------
----------  MeanField -------------------------------------------
------------------------------------------------------------------

Description:

BerkeleyGW uses a mean-field starting point, typically from DFT.
Currently, BerkeleyGW supports two plane-wave DFT codes,
PARATEC and Quantum ESPRESSO; one localized-orbital DFT code, SIESTA;
and two real-space codes, Octopus and PARSEC.
You can find details on how to use BerkeleyGW with these codes in
their respective directories.

For testing purposes and for crude calculations of large systems,
BerkeleyGW can be executed on top of the empirical pseudopotential
plane-wave code. See MeanField/EPM for details.


You can find a simple code for computing the Coulomb integral within an
image-charge model in MeanField/ICM.

------------------------------------------------------------------

Tricks and hints:

1. Vxc0

Vxc0 is the G=0 component of the exchange-correlation potential Vxc.
Vxc0 is added to the eigenvalues in the wavefunction files produced
by PARATEC and ESPRESSO. Vxc0 is also included in the VXC and vxc.dat
files produced by PARATEC and ESPRESSO (the VXC file contains Vxc(G)
and the vxc.dat file contains matrix elements of Vxc). The two Vxc0
terms cancel out when calculating the quasiparticle corrections
in the Sigma code.

2. Vacuum level

To correct the DFT eigenvalues for the vacuum level take the average
of the electrostatic potential on the faces of the unit cell in the
```

```
non-periodic directions and subtract it from the DFT eigenvalues.
The electrostatic potential is defined as Velec = Vbare + Vhartree,
while the total potential is Vtot = Vbare + Vhartree + Vxc, hence
Vtot contains Vxc0 and Velec does not. The average of Velec is
fed into the Sigma code using keywords avgpot and avgpot_outer.
The potentials can be generated with PARATEC and ESPRESSO.
In PARATEC, use elecplot for Velec or potplot for Vtot, then
convert from a3dr to cube using the Visual/volume.py script.
In ESPRESSO, use iflag=3, output_format=6, and plot_num=11
for Velec or plot_num=1 for Vtot. The averaging is done with
the Visual/average.py script.

3. Unit cell size

If you truncate the Coulomb interaction, make sure that the size
of the unit cell in non-periodic directions is at least two times
larger than the size of the charge distribution. This is needed
to avoid spurious interactions between periodic replicas but at
the same time not to alter interactions within the same unit cell.
Run Visual/surface.x to plot an isosurface that contains 99% of the
charge density (see Visual/README for instructions on how to do this).
The code will print the size of the box that contains the isosurface
to stdout. Multiply the box dimensions in non-periodic directions
by two to get the minimum size of the unit cell.

4. Inversion symmetry

When using the real flavor of the code, make sure the inversion
symmetry has no associated fractional translation (if it does,
shift the coordinate origin). Otherwise WFN, VXC, RHO (and VSC
in SAPO) have non-vanishing imaginary parts which are simply
dropped in the real flavor of the code. This won't do any
good to your calculation...
```

ESPRESSO

MeanField/ESPRESSO/README

```
Quantum ESPRESSO is available for download at http://www.quantum-espresso.org/

The interface between Quantum ESPRESSO and BerkeleyGW consists of three programs,
kgrid.x, pw2bgw.x and bgw2pw.x. kgrid.x is compiled with BerkeleyGW package and
pw2bgw.x and bgw2pw.x are compiled with Quantum ESPRESSO.
The files here are only compatible with version 4.3.2.
Starting with version 5.0, compatible files of pw2bgw.f90 and bgw2pw.f90 are
distributed with Quantum ESPRESSO.

--------------------------------------------------------------------------------

kgrid.x

pw.x can automatically generate a uniform grid of k-points, either unshifted
or shifted by half a grid step (Monkhorst-Pack grid), and reduce it to the
irreducible wedge of the Brillouin Zone with the symmetries of the point group
of the Bravais lattice and optionally with time-reversal symmetry. The same
functionality (and more) is provided by kgrid.x in this directory. Additionally,
kgrid.x can use the symmetries of the space group of the crystal instead of the
symmetries of the point group of Bravais lattice, which is needed for BerkeleyGW.
kgrid.x is not limited to the unshifted/half-step shifted Monkhorst-Pack grids.
It can generate an asymmetrically shifted fine grid used to improve the
convergence in absorption calculations. Also, kgrid.x can generate a grid
```

of k-points with a small q-shift used in Epsilon calculation to avoid the divergence of the Coulomb interaction. The list of k-points generated by kgrid.x must be manually added to the input file of pw.x. Note that time-reversal symmetry should NOT be used for BerkeleyGW (noinv = .false. if generating the k-grid in pw.x).

The format of the input file for kgrid.x (along with an example for Si) is found in kgrid.inp. See also more information in the header of kgrid.f90. You can find the input files for kgrid.x in examples/DFT in the ESPRESSO subdirectories of each example.

----

pw2bgw.x

Converts the output files produced by pw.x to the input files for BerkeleyGW.

You cannot use USPP, PAW, or spinors in a pw.x run for BerkeleyGW.

You cannot use "K_POINTS gamma" in a pw.x run for BerkeleyGW.
Use "K_POINTS { tpiba | automatic | crystal }" even for the
Gamma-point calculation.

The format of the input file for pw2bgw.x is described
in files MeanField/ESPRESSO/version-4.3.2/pw2bgw.inp and
MeanField/ESPRESSO/version-5.0/INPUT_pw2bgw.html (copy of espresso-5.0.3/PP/Doc/INPUT_pw2bgw.html).
You can find the input files for pw2bgw.x in examples/DFT
in the ESPRESSO subdirectories of each example.

Version 4.3.2 and earlier:

It is recommended to run a pw.x "nscf" calculation with "K_POINTS crystal"
and a list of k-points produced by kgrid.x.

Sometimes pw.x generates additional k-points in a "nscf" run with an explicit list of k-points. If this is the case for your calculation, there are several ways to go around this problem:

* Apply the patch provided with BerkeleyGW. It will prevent pw.x from generating additional k-points if they are provided explicitly, and take care of the normalization of the weights of k-points in a "bands" calculation.
* Do not specify the atomic positions in the input file of kgrid.x (set number of atoms = 0). Then pw.x will generate additional k-points which are the correct ones. Also set noinv = .true. in the input file of pw.x if time-reversal symmetry was not used in kgrid.x.
* Run a pw.x "bands" calculation instead of "nscf". In this case you have to explicitly specify the occupations in the input file of pw2bgw.x (note that this only works for insulators) and to normalize the weights of k-points to one in the input file of pw.x.

Version 5.0 and later:

It is recommended to run a pw.x "bands" calculation with "K_POINTS crystal"
and a list of k-points produced by kgrid.x.

You can also run a pw.x "nscf" calculation instead of "bands", but in this case pw.x may generate more k-points than provided in the input file of pw.x. If this is the case for your calculation you will get errors in BerkeleyGW.

----

bgw2pw.x

Converts BerkeleyGW WFN and RHO files to the format of pw.x.

This can be useful, for example, if you generate the plane waves
on top of the valence bands and want to diagonalize them in pw.x.
Look at the documentation for SAPO code in BerkeleyGW for more information.
Another possible use is to convert between different versions of pw.x.

bgw2pw.x reads common parameters from file prefix.save/data-file.xml and
writes files prefix.save/charge-density.dat (charge density in R-space),
prefix.save/gvectors.dat (G-vectors for charge density and potential),
prefix.save/K$n/eigenval.xml (eigenvalues and occupations for nth k-point),
prefix.save/K$n/evc.dat (wavefunctions in G-space for nth k-point), and
prefix.save/K$n/gkvectors.dat (G-vectors for nth k-point).

You must have prefix.save/K$n/eigenval.xml files present, or an error will occur,
even though their contents will not be used and will be overwritten.
The best is to run a pw.x calculation and use its prefix.save, e.g. from scf
and then get unoccupied bands from bgw2pw.x.

bgw2pw.x doesn't create restart files, so you cannot use restart_mode = 'restart'
for a subsequent pw.x run. Instead, use startingwfc = 'file'. Make sure
wf_collect = .true. and there are no prefix.wfc* files present. Also, the pw.x run
that generated the prefix.save directory originally must have wf_collect = .true.
also, for the appropriate links to the K$n files to be present.

bgw2pw.x doesn't modify file prefix.save/data-file.xml so make changes to this
file manually. For example, you will need to change the NUMBER_OF_BANDS and
NUMBER_OF_PROCESSORS tags (as well as per pool and per image) to make sure these
match the number of bands from the WFN file, as well as the number of processors
you will use in a subsequent run.

The format of the input file for bgw2pw.x is described
in files MeanField/ESPRESSO/version-4.3.2/bgw2pw.inp and
MeanField/ESPRESSO/version-5.0/INPUT_bgw2pw.html (copy of espresso-5.0.3/PP/Doc/INPUT_bgw2pw.html).

Notes on running pw.x

Sometimes pw.x crashes when trying to generate a LARGE number of unoccupied
states needed for GW calculations. The error messages may refer to "Cholesky
decomposition" or "diagonalization". If you run into this problem try one of
the following workarounds, or a combination of them:

1) Increase ecutwfc. Iterative diagonalization becomes inefficient and
unstable with increasing the ratio of the number of states to the size
of the Hamiltonian. Increasing ecutwfc will decrease this ratio
at the cost of computation time.

2) Split the calculation over k-points. If one k-point fails, it won't
affect the other k-points. You can merge the final wavefunction file
using MeanField/Utilities/wfnmerge.x after running pw2bgw.x for each
k-point.

3) Start with random instead of randomized atomic wavefunctions. For
this use the following parameter in the input file of pw.x:

```
    startingwfc = 'random'
```

4) Split the diagonalization into several steps alternating between
different diagonalization schemes. For example, use the following
parameters in the input files for consequent runs of pw.x:

```
    1st run:
        conv_thr = 1.0d-2
        diagonalization = 'david'
    2nd run:
        conv_thr = 1.0d-4
        diagonalization = 'cg'
        startingwfc = 'file'

    3rd run:
        conv_thr = 1.0d-6
        diagonalization = 'david'
        startingwfc = 'file'

    etc.

5) Run pw.x with "-ndiag 1". Often serial diagonalization is more
stable than the parallel one, and the time and memory overhead
is bearable in most cases.

6) Compile pw.x without "-D__SCALAPACK" in DFLAGS in make.sys
for using a custom distributed-memory diagonalization instead
of ScaLAPACK. This is useful to check if the problem
is caused by improper installation of ScaLAPACK.
Note that you will have to explicitly specify ndiag,
e.g. "mpirun -np 24 pw.x -ndiag 16 -in prefix.in > prefix.out",
because pw.x only sets ndiag automatically in case of ScaLAPACK.

Finally, it is recommended to always use the following parameters
in the input files for pw.x:

    wf_collect = .true.
    diago_full_acc = .true.

See documentation on Quantum ESPRESSO for information on these
parameters.

BEWARE: Sometimes wavefunctions may lose orthogonality during
iterative diagonalization. Neither Quantum ESPRESSO nor BerkeleyGW
checks for orthogonality of wavefunctions. This may cause erroneous
behavior like diverging head of eps0mat and possibly many other things.
```

MeanField/ESPRESSO/kgrid.inp

```
5 5 5                    ! numbers of k-points along b1,b2,b3
0.5 0.5 0.5              ! k-grid offset (0.0 unshifted, 0.5 shifted by half a grid step)
0.0 0.0 0.001            ! a small q-shift (0.0 unshifted, 0.001 shifted by one 1000th of b3)

0.0 0.5 0.5              ! lattice vectors in Cartesian coordinates (x,y,z)
0.5 0.0 0.5              ! in units of the lattice parameter
0.5 0.5 0.0              !
2                        ! number of atoms in the unit cell
1 -0.125 -0.125 -0.125 ! atomic species and positions in Cartesian coordinates (x,y,z)
1  0.125  0.125  0.125 ! in units of the lattice parameter
0 0 0                    ! size of FFT grid
.false.                  ! turn off time-reversal symmetry for BerkeleyGW
.false.                  ! OPTIONAL: k-points in the log file are in Cartesian coordinates

# Above: typical values for Si. Below: general description.

#  nk1 nk2 nk3   !   numbers of k-points in crystal coordinates (b1,b2,b3)
#  dk1 dk2 dk3   !   k-grid offset (0.0 unshifted, 1.0 shifted by bj/nkj)
#  dq1 dq2 dq3   !   a small q-shift (0.0 unshifted, 1.0 shifted by bj)

#  a1x a1y a1z   !   lattice vectors in Cartesian coordinates (x,y,z)
#  a2x a2y a2z   !   in arbitrary units (bohr, angstrom, lattice parameter)
#  a3x a3y a3z   !
#  n             !   number of atoms in the unit cell
#   s1 x1 y1 z1    !   atomic species and positions in Cartesian coordinates (x,y,z)
#   ...........    !   in the same units as the lattice vectors
#   sn xn yn zn    !
#   nr1 nr2 nr3    !   size of FFT grid
#   trs            !   if set to .true., use time-reversal symmetry (do not use this for
BerkeleyGW)
#   Cartesian      !   OPTIONAL: set to .true. for k-points in Cartesian coordinates (only in
the log file)
```

MeanField/ESPRESSO/version-5.0/INPUT_pw2bgw.html

Input File Description

Program: pw2bgw.x / PWscf / Quantum Espresso

TABLE OF CONTENTS

INTRODUCTION

```
Converts the output files produced by pw.x to the input files for
BerkeleyGW.

You cannot use USPP, PAW, or spinors in a pw.x run for BerkeleyGW.

You cannot use "K_POINTS gamma" in a pw.x run for BerkeleyGW.
Use "K_POINTS { tpiba | automatic | crystal }" even for the
Gamma-point calculation.

It is recommended to run a pw.x "bands" calculation with "K_POINTS
crystal"
and a list of k-points produced by kgrid.x, which is a part of
BerkeleyGW
package (see BerkeleyGW documentation for details).

You can also run a pw.x "nscf" calculation instead of "bands", but in
this
case pw.x may generate more k-points than provided in the input file of
pw.x.
If this is the case for your calculation you will get errors in
BerkeleyGW.

Examples showing how to run BerkeleyGW on top of Quantum ESPRESSO
including
```

```
the input files for pw.x and pw2bgw.x are distributed together with the
BerkeleyGW package.

Structure of the input data:
============================

   &INPUT_PW2BGW
     ...
   /
```

Namelist: INPUT_PW2BGW

prefix STRING

Status: MANDATORY

```
prefix of files saved by program pw.x
```

[Back to Top]

outdir STRING

Default: './'

```
the scratch directory where the massive data-files are written
```

[Back to Top]

real_or_complex INTEGER

Default: 2

```
1 | 2
1 for real flavor of BerkeleyGW (for systems with inversion symmetry and
time-reversal symmetry) or 2 for complex flavor of BerkeleyGW (for systems
without inversion symmetry and time-reversal symmetry)
```

[Back to Top]

symm_type STRING

Default: 'cubic'

```
'cubic' | 'hexagonal'
type of crystal system, 'cubic' for space groups 1 ... 142 and 195 ... 230
and 'hexagonal' for space groups 143 ... 194. Only used if ibrav = 0 in a
pw.x run. Written to BerkeleyGW WFN, RHO, VXC and VKB files but no longer
used (except in SAPO code in BerkeleyGW). You can use the default value for
all systems. Don't set to different values in different files for the same
system or you will get errors in BerkeleyGW.
```

[Back to Top]

wfng_flag LOGICAL

Default: .FALSE.

```
write wavefunctions in G-space to BerkeleyGW WFN file
```

[Back to Top]

wfng_file STRING

Default: 'WFN'

```
name of BerkeleyGW WFN output file. Not used if wfng_flag = .FALSE.
```

[Back to Top]

wfng_kgrid LOGICAL

Default: .FALSE.

```
overwrite k-grid parameters in BerkeleyGW WFN file.
If pw.x input file contains an explicit list of k-points,
the k-grid parameters in the output of pw.x will be set to zero.
Since sigma and absorption in BerkeleyGW both need to know the
k-grid dimensions, we patch these parameters into BerkeleyGW WFN file
```

[Back to Top]

wfng_nk1 INTEGER

Default: 0

```
number of k-points along b_1 reciprocal lattice vector.
Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_nk2 INTEGER

Default: 0

```
number of k-points along b_2 reciprocal lattice vector.
Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_nk3 INTEGER

Default: 0

```
number of k-points along b_3 reciprocal lattice vector.
Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_dk1 REAL

Default: 0.0

```
k-grid offset (0.0 unshifted, 0.5 shifted by half a grid step)
along b_1 reciprocal lattice vector. Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_dk2 REAL

Default: 0.0

```
k-grid offset (0.0 unshifted, 0.5 shifted by half a grid step)
```

```
along b_2 reciprocal lattice vector. Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_dk3 REAL

Default: 0.0

```
k-grid offset (0.0 unshifted, 0.5 shifted by half a grid step)
along b_3 reciprocal lattice vector. Not used if wfng_kgrid = .FALSE.
```

[Back to Top]

wfng_occupation LOGICAL

Default: .FALSE.

```
overwrite occupations in BerkeleyGW WFN file
```

[Back to Top]

wfng_nvmin INTEGER

Default: 0

```
index of the lowest occupied band (normally = 1).
Not used if wfng_occupation = .FALSE.
```

[Back to Top]

wfng_nvmax INTEGER

Default: 0

```
index of the highest occupied band (normally = number of occupied bands).
Not used if wfng_occupation = .FALSE.
```

[Back to Top]

rhog_flag LOGICAL

Default: .FALSE.

```
write charge density in G-space to BerkeleyGW RHO file.
Only used for the GPP model in sigma code in BerkeleyGW
```

[Back to Top]

rhog_file STRING

Default: 'RHO'

```
name of BerkeleyGW RHO output file. Only used for the GPP model in sigma
code in BerkeleyGW. Not used if rhog_flag = .FALSE.
```

[Back to Top]

vxcg_flag LOGICAL

Default: .FALSE.

```
write local part of exchange-correlation potential in G-space to
BerkeleyGW VXC file. Only used in sigma code in BerkeleyGW, it is
recommended to use vxc_flag instead
```

[Back to Top]

vxcg_file STRING

Default: 'VXC'

```
name of BerkeleyGW VXC output file. Only used in sigma code in BerkeleyGW,
it is recommended to use vxc_flag instead. Not used if vxcg_flag = .FALSE.
```

[Back to Top]

vxc0_flag LOGICAL

Default: .FALSE.

```
write Vxc(G = 0) to text file. Only for testing, not required for
BerkeleyGW
```

[Back to Top]

vxc0_file STRING

Default: 'vxc0.dat'

```
name of output text file for Vxc(G = 0). Only for testing, not required for
BerkeleyGW. Not used if vxc0_flag = .FALSE.
```

[Back to Top]

vxc_flag LOGICAL

Default: .FALSE.

```
write matrix elements of exchange-correlation potential to text file.
Only used in sigma code in BerkeleyGW
```

[Back to Top]

vxc_file STRING

Default: 'vxc.dat'

```
name of output text file for Vxc matrix elements. Only used in sigma code
in BerkeleyGW. Not used if vxc_flag = .FALSE.
```

[Back to Top]

vxc_integral STRING

Default: 'g'

```
'g' | 'r'
'g' to compute matrix elements of exchange-correlation potential in G-
space.
'r' to compute matrix elements of the local part of exchange-correlation
potential in R-space. It is recommended to use 'g'. Not used if vxc_flag =
.FALSE.
```

[Back to Top]

vxc_diag_nmin INTEGER

Default: 0

```
minimum band index for diagonal Vxc matrix elements. Not used if vxc_flag =
.FALSE.
```

[Back to Top]

vxc_diag_nmax INTEGER

Default: 0

```
maximum band index for diagonal Vxc matrix elements. Not used if vxc_flag =
.FALSE.
```

[Back to Top]

vxc_offdiag_nmin INTEGER

Default: 0

```
minimum band index for off-diagonal Vxc matrix elements. Not used if
vxc_flag = .FALSE.
```

[Back to Top]

vxc_offdiag_nmax INTEGER

Default: 0

```
maximum band index for off-diagonal Vxc matrix elements. Not used if
vxc_flag = .FALSE.
```

[Back to Top]

vxc_zero_rho_core LOGICAL

Default: .FALSE.

```
set to .TRUE. to zero out NLCC or to .FALSE. to keep NLCC when computing
exchange-correlation potential. For details, see Hybertsen & Louie PRB 34,
5390 (1986), Eq. (38) and discussion afterwards
```

[Back to Top]

vscg_flag LOGICAL

Default: .FALSE.

```
write local part of self-consistent potential in G-space to
BerkeleyGW VSC file. Only used in SAPO code in BerkeleyGW
```

[Back to Top]

vscg_file STRING

Default: 'VSC'

```
name of BerkeleyGW VSC output file. Only used in SAPO code in BerkeleyGW.
Not used if vscg_flag = .FALSE.
```

[Back to Top]

vkbg_flag LOGICAL

Default: .FALSE.

```
write Kleinman-Bylander projectors in G-space to BerkeleyGW VKB file.
Only used in SAPO code in BerkeleyGW
```

[Back to Top]

vkbg_file STRING

Default: 'VKB'

```
name of BerkeleyGW VKB output file. Only used in SAPO code in BerkeleyGW.
Not used if vkbg_flag = .FALSE.
```

[Back to Top]

This file has been created by helpdoc utility.

MeanField/ESPRESSO/version-5.0/INPUT_bgw2pw.html

Input File Description

Program: bgw2pw.x / PWscf / Quantum Espresso

TABLE OF CONTENTS

INTRODUCTION

```
Converts BerkeleyGW WFN and RHO files to the format of pw.x.
This can be useful, for example, if you generate the plane waves
on top of the valence bands and want to diagonalize them in pw.x.
Look at the documentation for SAPO code in BerkeleyGW for more
information.

bgw2pw.x reads common parameters from file prefix.save/data-file.xml and
```

```
writes files prefix.save/charge-density.dat (charge density in R-space),
prefix.save/gvectors.dat (G-vectors for charge density and potential),
prefix.save/K$n/eigenval.xml (eigenvalues and occupations for nth k-
point),
prefix.save/K$n/evc.dat (wavefunctions in G-space for nth k-point), and
prefix.save/K$n/gkvectors.dat (G-vectors for nth k-point).

bgw2pw.x doesn't modify file prefix.save/data-file.xml so make changes
to this
file manually (for example, you will need to change the number of bands
if you
are using bgw2pw.x in conjunction with SAPO code in BerkeleyGW).

Structure of the input data:
============================

   &INPUT_BGW2PW
     ...
   /
```

Namelist: INPUT_BGW2PW

prefix STRING

Status: MANDATORY

```
prefix of files saved by program pw.x
```

[Back to Top]

outdir STRING

Default: './'

```
the scratch directory where the massive data-files are written
```

[Back to Top]

real_or_complex INTEGER

Default: 2

```
1 | 2
1 for real flavor of BerkeleyGW (for systems with inversion symmetry and
time-reversal symmetry) or 2 for complex flavor of BerkeleyGW (for systems
without inversion symmetry and time-reversal symmetry)
```

[Back to Top]

wfng_flag LOGICAL

Default: .FALSE.

```
read wavefunctions in G-space from BerkeleyGW WFN file
```

[Back to Top]

wfng_file STRING

Default: 'WFN'

```
name of BerkeleyGW WFN input file. Not used if wfng_flag = .FALSE.
```

[Back to Top]

wfng_nband INTEGER

Default: 0

```
number of bands to write (0 = all). Not used if wfng_flag = .FALSE.
```

[Back to Top]

rhog_flag LOGICAL

Default: .FALSE.

```
read charge density in G-space from BerkeleyGW RHO file
```

[Back to Top]

rhog_file STRING

Default: 'RHO'

```
name of BerkeleyGW RHO input file. Not used if rhog_flag = .FALSE.
```

[Back to Top]

This file has been created by helpdoc utility.

MeanField/ESPRESSO/version-4.3.2/pw2bgw.inp

```
&input_pw2bgw
   prefix = 'silicon'        ! same as in espresso
   real_or_complex = 1       ! 1 for real or 2 for complex
   wfng_flag = .true.        ! write wavefunction in G-space
   wfng_file = 'WFN'         ! wavefunction file name
   wfng_kgrid = .false.      ! overwrite k-grid in wavefunction file
   wfng_nk1 = 4              ! ( if espresso input file contains the
   wfng_nk2 = 4              !   manual list of k-points, the k-grid
   wfng_nk3 = 4              !   parameters in espresso are set to zero;
   wfng_dk1 = 0.5            !   since Sigma and absorption both need to know
   wfng_dk2 = 0.5            !   the k-grid dimensions, we patch these
   wfng_dk3 = 0.5            !   parameters into the wave-function file )
   wfng_occupation = .true.  ! overwrite occupations in wavefunction file
   wfng_nvmin = 1            ! ( set min/max valence band indices; identical to
   wfng_nvmax = 4            !   scissors operator for LDA-metal/GW-insulator )
   rhog_flag = .true.        ! write charge density in G-space
   rhog_file = 'RHO'         ! charge density file name
   vxcg_flag = .true.        ! write exchange-correlation potential in G-space
   vxcg_file = 'VXC'         ! exchange-correlation potential file name
   vxc0_flag = .true.        ! write Vxc(G=0)
   vxc0_file = 'vxc0.dat'    ! Vxc(G=0) file name
   vxc_flag = .false.        ! write matrix elements of Vxc
   vxc_file = 'vxc.dat'      ! Vxc matrix elements file name
   vxc_integral = 'g'        ! compute Vxc matrix elements in R- or G-space
   vxc_diag_nmin = 0         ! min band index for diagonal Vxc matrix elements
   vxc_diag_nmax = 0         ! max band index for diagonal Vxc matrix elements
   vxc_offdiag_nmin = 0      ! min band index for off-diagonal Vxc matrix elements
   vxc_offdiag_nmax = 0      ! max band index for off-diagonal Vxc matrix elements
   input_dft = 'sla+pz'      ! same as in espresso
   exx_flag = .false.        ! set to .true. for hybrids
   vnlg_flag = .false.       ! write Kleinman-Bylander projectors in G-space
   vnlg_file = 'KBproj'      ! Kleinman-Bylander projectors file name
   vxc_zero_rho_core = .true. ! NLCC: remove core charge when calculating Vxc
/

# above are typical values for Si. below are the defaults.
   prefix = 'prefix'
   real_or_complex = 2
   wfng_flag = .false.
   wfng_file = 'WFN'
   wfng_kgrid = .false.
   wfng_nk1 = 0
   wfng_nk2 = 0
   wfng_nk3 = 0
   wfng_dk1 = 0.0
   wfng_dk2 = 0.0
   wfng_dk3 = 0.0
   wfng_occupation = .false.
   wfng_nvmin = 0
   wfng_nvmax = 0
   rhog_flag = .false.
   rhog_file = 'RHO'
   vxcg_flag = .false.
   vxcg_file = 'VXC'
   vxc0_flag = .false.
   vxc0_file = 'vxc0.dat'
   vxc_flag = .false.
   vxc_file = 'vxc.dat'
   vxc_integral = 'g'
   vxc_diag_nmin = 0
   vxc_diag_nmax = 0
   vxc_offdiag_nmin = 0
   vxc_offdiag_nmax = 0
   input_dft = 'sla+pz'
   exx_flag = .false.
   vnlg_flag = .false.
   vnlg_file = 'VNL'
   vxc_zero_rho_core = .true.
```

MeanField/ESPRESSO/version-4.3.2/bgw2pw.inp

```
&input_bgw2pw
   prefix = 'silicon'        ! same as in espresso
   real_or_complex = 1       ! 1 for real or 2 for complex
   wfng_flag = .true.        ! read wavefunction in G-space
   wfng_file = 'WFN'         ! wavefunction file name
   wfng_nband = 0            ! number of bands to write (0 = all)
   rhog_flag = .true.        ! read charge density in G-space
   rhog_file = 'RHO'         ! charge density file name
/

# above are typical values for Si. below are the defaults.
  prefix = 'prefix'
  real_or_complex = 2
  wfng_flag = .false.
  wfng_file = 'WFN'
  wfng_nband = 0
  rhog_flag = .false.
  rhog_file = 'RHO'
```

MeanField/ESPRESSO/patch/README

```
This directory contains a patch for espresso-4.3.2 that fixes bugs
listed below. Assuming BerkeleyGW is installed in $BGWPATH/BerkeleyGW,
to apply the patch execute the following command
in the directory containing the espresso-4.3.2 directory:

% patch -p0 -i $BGWPATH/BerkeleyGW/MeanField/ESPRESSO/patch/espresso-4.3.2-patch

espresso-4.3.2 buglist:

(1) <<< THIS IS FIXED IN QUANTUM ESPRESSO 5.0 >>>
If you run espresso nscf calculation with startingwfc set to file
you may get error message saying "cannot read wfc : file not found".
This happens because subroutine verify_tmpdir in PW/input.f90 renames
data-file.xml to data-file.xml.bck and subroutines read_planewaves and
read_wavefunctions in PW/pw_restart.f90 try to read from data-file.xml
which doesn't exist. To get around this problem read from .xml.bck in
read_planewaves and read_wavefunctions if opening .xml returns error.

(2) <<< THIS IS FIXED IN QUANTUM ESPRESSO 5.0 >>>
Missing "CALL stop_pp ( )" at the end of PP/plan_avg.f90 causes it
to crash sometimes.

(3) <<< THIS IS FIXED IN QUANTUM ESPRESSO 5.0 >>>
Sometimes pw.x generates additional k-points in a "nscf" run with an explicit
list of k-points. If this is the case for your calculation, there are several
ways to go around this problem:
* Apply the patch provided with BerkeleyGW. It will prevent pw.x from
  generating additional k-points if they are provided explicitly, and
  take care of the normalization of the weights of k-points in a "bands"
  calculation.
* Do not specify the atomic positions in the input file of kgrid.x (set
  number of atoms = 0). Then pw.x will generate additional k-points which
  are the correct ones. Also set noinv = .true. in the input file of pw.x
  if time-reversal symmetry was not used in kgrid.x.
* Run a pw.x "bands" calculation instead of "nscf". In this case you have
  to explicitly specify the occupations in the input file of pw2bgw.x (note
  that this only works for insulators) and to normalize the weights of
  k-points to one in the input file of pw.x.
```

PARATEC

MeanField/PARATEC/README

```
paratecSGL is available for download through SVN repository at
https://civet.berkeley.edu/svn/CODES/paratecSGL/
(only for authorized users).

To build with support for BerkeleyGW output, in arch.mk set the
line GWWFNPATH to BerkeleyGW/library, and add -DBGW to M4OPTLIBS.

Full documentation for PARATEC 5.1:
http://www.tcm.phy.cam.ac.uk/~jry20/paratec/doc/doc.html
```

```
Literature:
* B. G. Pfrommer, J. Demmel, and H. Simon, "Unconstrained Energy Functionals
  for Electronic Structure Calculations," J. Comp. Phys. 150, 287 (1999).
* B. G. Pfrommer, M. Cote, S. G. Louie, and M. L. Cohen, "Relaxation of
  Crystals with the Quasi-Newton Method," J. Comp. Phys. 131, 233 (1997).
* Mathieu Taillefumier, Delphine Cabaret, Anne-Marie Flank, and Francesco
  Mauri, "X-ray absorption near-edge structure calculations with
  pseudopotentials: Application to the K-edge in diamond and alpha-quartz,"
  Phys. Rev. B 66, 195107 (2002).
*
http://cmsn.drupalgardens.com/sites/cmsn.drupalgardens.com/files/CMSN_Newsletter_Vol2No2.pdf

[An older version was available at http://www.nersc.gov/projects/paratec.
PARATEC 5.1 produces only the old GW wavefunction format, incompatible
with this version of BerkeleyGW. However, you can convert them to the
new format using MeanField/Utilities/convert_old_to_new.x.]

The pseudopotentials for PARATEC can be generated with
the fhi98PP program which is available for download at
http://www.fhi-berlin.mpg.de/th/fhi98md/fhi98PP/

PARATEC output for BerkeleyGW is controlled by flags
gw_shift, gwc, gwr, gwscreening and vxc_matrix_elements.
The flags can be combined with an underscore: e.g. output_flags gwr_gwscreening

gw_shift q1 q2 q3
 -- generates q-shifted grid, q-vector is in crystal coordinates
    in units of reciprocal lattice vectors (for WFNq, WFNq_fi)
    This variable does the same job as the kgrid.x utility.

output_flags gwc
 -- writes complex wavefunctions in G-space for systems without
    inversion symmetry to file WFN (for all codes)

output_flags gwr
 -- writes real wavefunctions in G-space for systems with
    inversion symmetry to file WFN (for all codes)

output_flags gwscreening
 -- writes charge density in G-space to file RHO,
    exchange-correlation potential in G-space to file VXC,
    and matrix elements of exchange-correlation potential to file
    vxc.dat (for Sigma)

vxc_matrix_elements diagmin diagmax offdiagmin offdiagmax
 -- specifies the range of bands for which to compute diagonal and
    off-diagonal matrix elements of exchange-correlation potential
    (for Sigma, in conjunction with output_flags gwscreening)

Other key input flags:

pw_job {scf, nonselfcon}
 -- Use scf for initial calculation, nonselfcon for generating
    BerkeleyGW outputs. (bandstructure does not seem to work)

energy_cutoff ecut
 -- Plane-wave cutoff for wavefunctions, in Ry.

number_kpoints
 -- set to 0  to use k_grid and reduce with symmetries
    set to -1 to use k_grid and do not reduce with symmetries
    set to any other number to read from file KPOINTS

k_grid nx ny nz
 -- 3 integers specifying Monkhorst-Pack k-grid dimensions
```

```
k_grid_shift dx dy dz
 -- Monkhorst-Pack k-grid shifts (typically 0.0 or 0.5)

number_bands nb
 -- Number of bands to use in calculation. Fraction actually useful
    or written to BerkeleyGW output determined by next variable.

eigspacefrac frac
 -- Fraction of bands to converge. Setting a higher number_bands
    and lower eigspacefrac can make the calculation more efficient
    depending on the diagonalization scheme. 0 < frac <= 1.0.

You can find the actual input files for PARATEC and BerkeleyGW in
examples/DFT, in PARATEC subdirectories for each example.

There are also bgw2para and rho2cd utilities that convert
BerkeleyGW files WFN and RHO to PARATEC format. This may be
useful, for example, if one generates the plane waves on top of the
valence and conduction bands (look into MeanField/SAPO/README for details)
and wants to diagonalize them further in PARATEC. There are no input
files; bgw2para takes as argument the wfn filename,
and it creates files WFN$n.$s and BAND needed for PARATEC.
Similarly, rho2cd requires file RHO and it creates file CD.
```

SIESTA

MeanField/SIESTA/README

```
An example siesta2bgw calculation is provided in examples/DFT/benzene.
Please see this example for usage and example siesta and denchar input
files.

The SIESTA wrapper reads lattice vectors, atomic species and
positions from SystemLabel.XV file, k-points from SystemLabel.KP
file, eigenvalues from SystemLabel.EIG file, wavefunctions in R-space
from SystemLabel{.K$k}.WF$n{.UP|.DOWN}{.REAL|.IMAG}.cube file, symmetry
group, kinetic energy cutoffs for wavefunctions and charge density,
k-grid dimensions and offsets from the input file, generates reciprocal
lattice vectors, rotation matrices, fractional translations, FFT grid and
G-vectors, performs FFT from R-space to G-space, and writes the selected
range of bands to the output WFN file. Make sure to set up the real
space grid in SIESTA/DENCHAR utility same as FFT grid. If the two grids
differ the SIESTA wrapper will return an error. The output WFN file
can be used as an auxiliary wavefunction for the SAPO code.

A full GW/BSE calculation with SIESTA wavefunctions also requires RHO and
VXC files. These files can be generated from SystemLabel.RHO{.UP|.DN}.cube,
SystemLabel.VT{.UP|.DN}.cube and SystemLabel.VH.cube files obtained with
SIESTA/GRID2CUBE utility on SIESTA real space grid. Make sure SIESTA real
space grid is equivalent to FFT grid by setting parameter MeshCutoff
in SIESTA input file same as kinetic energy cutoff for charge density.

Input is read from file siesta2bgw.inp:

&input_siesta2bgw
   systemlabel      = 'ammonia'
   wfng_output_file = 'wfng.lo'
   rhog_output_file = ''
   vxcg_output_file = ''
   ecutwfn          = 60.0
   ecutrho          = 240.0
   wfng_nk1         = 1
   wfng_nk2         = 1
   wfng_nk3         = 1
   wfng_dk1         = 0.0
   wfng_dk2         = 0.0
   wfng_dk3         = 0.0
   wfng_ref_flag    = .true.
   wfng_ref_kpoint  = 1
   wfng_ref_spin    = 1
   wfng_ref_band    = 4
   wfng_ref_energy  = -5.87
   wfng_band_flag   = .true.
   wfng_band_min    = 5
   wfng_band_max    = 28
   wfng_energy_flag = .false.
   wfng_energy_min  = 0.0
   wfng_energy_max  = 0.0
   wfng_gamma_real  = .true.
/
```

Here, lattice vectors, k-points, eigenvalues and wavefunctions in R-space are read from ammonia.XV, ammonia.KP, ammonia.EIG and ammonia.WF$n.cube files, respectively. The kinetic-energy cutoffs for wavefunction and charge density are set to 60 and 240 Ry, respectively. The k-grid is set to the Gamma-point. The SIESTA eigenvalues are shifted in energy so that the HOMO state (1st k-point, 1st spin, 4th band) appears at -5.87 eV. The real parts of resonant SIESTA wavefunctions (from 5th to 28th band) are read from Gaussian Cube files and the imaginary parts are set to zero (wfng_gamma_real), the wavefunctions are brought from R-space to G-space and written to wfng.lo file. The SIESTA charge density and exchange-correlation potential files are not generated.

MeanField/SIESTA/patch/README

This directory contains a patch for denchar-1.4 and grid2cube-1.1.1 in siesta-3.1 that adds support for non-orthogonal cells. The modified versions generate Gaussian Cube files on real-space grid. To apply the patch execute the following command in the directory where you have siesta-3.1:

```
% patch -p0 -i siesta-3.1-patch
```

Use the following keywords in siesta input file:

```
WaveFuncKPointsScale ReciprocalLatticeVectors
%block WaveFuncKPoints
0.000  0.000  0.000
%endblock WaveFuncKPoints
WriteDenchar T
SaveRho T
SaveElectrostaticPotential T
SaveTotalPotential T
MeshCutoff 240.0 Ry
```

Then run denchar for wavefunctions and grid2cube for charge density and potentials. In denchar input file, use keywords LatticeConstant and LatticeVectors same as in siesta input file. Keywords Denchar.{Min|Max}{X|Y|Z} will be ignored. In grid2cube input file, 3rd and 4th lines (shift of the origin of coordinates and nskip) will be ignored.

There is also irrbz.py that extracts irreducible wedge

```
from siesta output files prefix.KP and prefix.EIG.
This is needed because siesta reduces k-points by
time-reversal symmetry only.
```

MeanField/SIESTA/siesta2bgw.inp

```
 &input_siesta2bgw
    systemlabel      = 'prefix'  ! SystemLabel used in SIESTA calculation
    wfng_output_file = 'wfng.lo' ! file to write generated wavefunctions to
    rhog_output_file = 'rhog.lo' ! file to write rho(G) (blank = not written)
    vxcg_output_file = 'vxcg.lo' ! file to write Vxc(G) (blank = not written)
    ecutwfn          = 0.0       ! kinetic energy cutoff (Ry) for wavefunctions
    ecutrho          = 0.0       ! kinetic energy cutoff (Ry) for rho and V
    wfng_nk1         = 0         ! number of k-points along b1
    wfng_nk2         = 0         ! number of k-points along b2
    wfng_nk3         = 0         ! number of k-points along b3
    wfng_dk1         = 0.0       ! k-grid offset along b1 in units of b1/nk1
    wfng_dk2         = 0.0       ! k-grid offset along b2 in units of b2/nk2
    wfng_dk3         = 0.0       ! k-grid offset along b3 in units of b3/nk3
    wfng_ref_flag    = .false.   ! shift eigenvalues to match reference state
    wfng_ref_kpoint  = 0         ! reference state k-point index
    wfng_ref_spin    = 0         ! reference state spin index
    wfng_ref_band    = 0         ! reference state band index
    wfng_ref_energy  = 0.0       ! reference state eigenvalue (in eV)
    wfng_band_flag   = .false.   ! choose a range of bands (0 = all)
    wfng_band_min    = 0         ! minimum band number
    wfng_band_max    = 0         ! maximum band number
    wfng_energy_flag = .false.   ! choose an energy window
    wfng_energy_min  = 0.0       ! minimum energy (in eV)
    wfng_energy_max  = 0.0       ! maximum energy (in eV)
    wfng_gamma_real  = .false.   ! only real wavefunctions at the Gamma point
 /
```

Octopus

MeanField/Octopus/README

```
Octopus is a real-space DFT and TDDFT code available under the GPL from
http://www.tddft.org/programs/octopus.
Support for BerkeleyGW output is implemented in Octopus since version 4.1.0.
For more information, see the Octopus wiki:
http://www.tddft.org/programs/octopus/wiki/index.php/BerkeleyGW.
```

EPM

MeanField/EPM/README

```
==========================================================================

"The empirical pseudopotential method never lost a battle to experiment."
-- Marvin Cohen, 2008

==========================================================================
```

EPM stands for Empirical Pseudopotential Method. It is the plane-wave
part of TBPW-1.1 modified to generate input files for BerkeleyGW.
Numerous other corrections and improvements have also been made
by Georgy Samsonidze and David Strubbe.
TBPW-1.1 is available for download from http://www.mcc.uiuc.edu/software/

The input variables are described in epm.inp. Most variables are common to both
epm.x and epm2bgw.x, but a few are just for one or the other.

The real wavefunctions for systems with inversion symmetry
are obtained by applying the Gram-Schmidt process adapted from
paratecSGL-1.1.3/src/para/gwreal.f90

To plot silicon band structure calculated with the EPM executable:
```
% epm.x < silicon-epm.in
% gnuplot bands.plt
```
Uses variables NumberOfLines, NumberOfDivisions, KPointsAndLabels.

To use the explicit form factors and to compute the band gap and
the effective masses (mass only correct for Si-like band structures):
```
% epm.x < silicon-epm-ff.in
```
Uses variables gap_flag, gap_file, mass_flag, mass_file.

To generate input files for BerkeleyGW executable, use epm2bgw.x.
The scripts epm2bgw_cplx.sh and epm2bgw_cplx_spin.sh are just for use
with the testsuite. They force complex, or complex and spin-polarized.

You can find the actual input files for GW/BSE calculations on top of
EPM in directory examples/EPM, as well as testsuite directories
Si-EPM, GaAs-EPM, and Si-Wire-EPM.

The k-points for the EPM input files can be generated using utility
kgrid. Please refer to file MeanField/ESPRESSO/README for details.

Virtual crystal approximation (VCA) is implemented within EPM in
the form of two pre-processing scripts, ff2vq.py and vca.py.

ff2vq.py reads EPM form factors from file form_factors.dat, fits them
to the chosen functional form of the V(q) potential, writes potential
coefficients to file v_of_q.dat, and writes new form factors computed
from V(q) to file vq2ff.dat. This procedure is described by equations
(8) and (9) and the accompanying text in Phys. Rev. B 84, 195201 (2011).

vca.py reads V(q) potential coefficients from file v_of_q.dat,
employs the virtual crystal approximation to compute hybrid form
factors, and writes them to file vca_ff.dat. The potential mixing
is controlled by identifiers host_material and doping_level hard
coded below. This procedure is described by equations (10) -- (12)
and the accompanying text in Phys. Rev. B 84, 195201 (2011).

The original form factors from file form_factors.dat or the hybrid
ones from file vca_ff.dat can be fed to epm.x and epm2bgw.x using
keywords LatticeConstant and FormFactors in the input file for epm.x
or epm2bgw.x. See example of using these keywords in silicon-epm-ff.in.

The README file of TBPW-1.1 follows below.

==

TBPW-1.1

==

Author/Affiliation

----

* Dyutiman Das, UIUC
* William Mattson, UIUC
* Nichols A. Romero, UIUC
* Richard M. Martin, UIUC

==

Author email

----

nromero@uiuc.edu

==

Description of Software

TBPW is an electronic structure code primarily intended for pedagogical purposes. It is written from the ground-up in a modular style using Fortran 90. This code is composed of two distinct parts: a tightbinding (TB) and plane wave (PW). Additionally, there is a plane wave density (PWD) code which outputs the electron density on a grid.

The main characteristics of these codes are:

* Readily provides band structure plots
* TB implemented using a rotation matrix formalism allows the use of orbitals with arbitrary angular momentum l
* PW implemented using the option of diagonalisation via direct-inversion

Send email to tbpw-subscribe@mcc.uiuc.edu to subscribe to the mailing list, tbpw@mcc.uiuc.edu

This material is based upon work supported by the NSF under Award No. 99-76550 and the DOE under Award No. DEFG-96-ER45439

==

Submitted

2003-01-31

==

Disclaimer

MeanField/EPM/epm.inp

```
# BEWARE: you cannot comment out tags with # in EPM!
# any occurrence of a tag will be found and read.

# These parameters are only for epm2bgw
real_or_complex 1        # 1 = real, 2 = complex. default is complex.
wfng_flag T              # Write wavefunction file (default F)
wfng_file WFN            # filename (default WFN)
KPointNumbers 5 5 5      # number of points in k-grid in each direction b1,b2,b3
KPointOffsets 0.5 0.5 0.5 # shift in k-grid in each direction b1,b2,b3
rhog_flag T              # Write density file (default F)
rhog_file RHO            # filename (default RHO)
vxcg_flag T              # Write xc potential file (default F)
vxcg_file VXC            # filename (default VXC)
vxc_flag T               # Write VXC matrix element file (default F)
vxc_file vxc.dat         # filename (default vxc.dat)
vxc_diag_nmin 1          # minimum band to write diagonal matrix elements (default 0)
vxc_diag_nmax 8          # maximum band to write diagonal matrix elements (default 0)
vxc_offdiag_nmin 1       # minimum band to write off-diagonal matrix elements (default 0)
vxc_offdiag_nmax 8       # maximum band to write off-diagonal matrix elements (default 0)
disable_symmetries F     # set to true to write no symmetries (default F)
# EPM does not really do a spin-polarized calculation, but can write two copies of
everything
nspin 1                  # set to 2 to write spin up and spin down (default 1)

# These parameters are common to epm and epm2bgw
# Set form factor parameters (see examples in form_factors.dat),
# or default is to use V(q) from atomPotentialMod (only for H,Si,Ga,As)
FormFactors 5.43  -0.224  0.055  0.072  0.000  0.000  0.000

# 0 is for LAPACK direct diagonalization (default), 1 for conjugate gradients
DiagonalizationSwitch 0
AbsoluteTolerance -1.0d0  # tolerance for LAPACK diagonalization (default -1 means machine
precision)
CGIterationPeriod 5       # parameter for conjugate gradients (default 3)
CGTolerance 1.0d-10       # parameter for conjugate gradients (1d-5)

InputEnergiesInEV          # Rydberg is default
EnergyCutoff 11.0          # plane-wave cutoff for the potential
```

```
NumberOfDimensions 3              # dimensionality of the system (default 3)
#LatticeConstantFormat Angstrom   # default is Bohr
LatticeConstant 10.2612           # in units given above
LatticeVectors                    # in lattice constant units
0.0 0.5 0.5
0.5 0.0 0.5
0.5 0.5 0.0

NumberOfAtoms 2
NumberOfSpecies 1
ChemicalSpeciesLabel              # index, atomic number, name
1 14 Si
AtomicCoordinatesFormat ScaledByLatticeVectors
  # ScaledByLatticeVectors (default) is in crystal coordinates
  # ScaledCartesian is in units of LatticeConstant
AtomicCoordinatesAndAtomicSpecies  # x, y, z, species index
-0.125 -0.125 -0.125 1
 0.125  0.125  0.125 1

NumberOfBands 8
NumberOfOccupiedBands 4

KPointsScale ReciprocalLatticeVectors # default; other choice is TwoPi/a
KPointsList 19          # number of k-points supplied
  0.100000000  0.100000000  0.100000000  2.0  # kx, ky, kz, weight (will be renormalized)
  0.100000000  0.100000000  0.300000000  6.0
  0.100000000  0.100000000  0.500000000  6.0
  0.100000000  0.100000000  0.700000000  6.0
  0.100000000  0.100000000  0.900000000  6.0
  0.100000000  0.300000000  0.300000000  6.0
  0.100000000  0.300000000  0.500000000 12.0
  0.100000000  0.300000000  0.700000000 12.0
  0.100000000  0.300000000  0.900000000 12.0
  0.100000000  0.500000000  0.500000000  6.0
  0.100000000  0.500000000  0.700000000 12.0
  0.100000000  0.500000000  0.900000000  6.0
  0.100000000  0.700000000  0.700000000  6.0
  0.300000000  0.300000000  0.300000000  2.0
  0.300000000  0.300000000  0.500000000  6.0
  0.300000000  0.300000000  0.700000000  6.0
  0.300000000  0.500000000  0.500000000  6.0
  0.300000000  0.500000000  0.700000000  6.0
  0.500000000  0.500000000  0.500000000  1.0

# for plotting band structures; used if KPointsList not present
NumberOfLines 4
NumberOfDivisions 20
KPointsScale ReciprocalLatticeVectors # same as above
KPointsAndLabels
0.500 0.500 0.500 L
0.000 0.000 0.000 G
0.500 0.000 0.500 X
0.625 0.250 0.625 U
0.000 0.000 0.000 G

# These parameters are only for epm. Find band gap and effective mass.
gap_flag F              # default F
gap_file bandgap.dat    # filename (default gap.dat)
# Note: effective mass only works for silicon-like band structures.
mass_flag F             # default F
mass_file effmass.dat   # filename (default mass_file)
```

ICM

MeanField/ICM/README

```
This program reads the wavefunction file in Gaussian Cube or
XCrySDen XSF format and computes the Coulomb integral within
an image charge model.

It is based on Visual/surface.cpp code, so the input parameter
file formats for the two codes are quite similar. You can find
more details in Visual/surface.inp and Visual/README.

Input is read from file icm.inp:

inputfilename C6H6.b_15.cube
inputfileformat cube
threshold 0.99
threshold_power 1
coulomb_power 1
mirrorplaneorigin
0.0 0.0 -2.0
mirrorplanenormal
0.0 0.0 1.0
mirrorplaneunit angstrom
uc F
uco
0.0 0.0 0.0
ucv
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
ucu latvec
sc T
sco
-0.5 -0.5 -0.5
scv
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
scu latvec

In the above example, the HOMO wavefunction of benzene is read from
Gaussian Cube file. The wavefunction is placed in the center of the
supercell (see the meaning of uc, uco, ucv, ucu, sc, sco, scv, scu
parameters in Visual/surface.cpp). The parts of the wavefunction
outside an isosurface that contains 99% of the charge density are
dropped (parameters threshold and threshold_power have the same
meaning as isovalue and power in Visual/surface.cpp). Parameter
coulomb_power tells the code whether the wavefunction in the Coulomb
integral needs to be squared. Set both powers to 1 if the wavefunction
file contains the squared amplitude as produced by ESPRESSO, and to 2
for the linear amplitude as in PARATEC or SIESTA. The mirror plane
is defined by parameters mirrorplaneorigin and mirrorplanenormal,
in the above example it is parallel to the xy plane crossing the
z axis at -2 Angstrom. Problems may occur with non-orthogonal unit cells;
use of cubic cells is recommended.
```

Utilities

MeanField/Utilities/README

```
-----------------------------------------------------------------
```

```
----------  MeanField Utilities  -------------------------------
-----------------------------------------------------------------

degeneracy_check.x:

  Determine which numbers of bands do not break a degenerate subspace, to avoid warnings or
  errors in Epsilon, Sigma, and BSE codes. Supply any number of binary WFN files as command-
  line arguments, and a list of numbers that are acceptable for all k-points in all files
  will be written to standard output.

mf_convert_wrapper.sh:

  Converts WFN/RHO/VXC files between binary and ASCII. This can be useful for moving files
  correctly between big- and little-endian machines, or for examining the contents of a
  file (though wfn_rho_vxc_info.x is more user-friendly). Using this wrapper script, the
  flavor (real/complex), type of file, and binary/ASCII is determined automatically.

wfn_rho_vxc_info.x:

  Prints the contents of the header of a (binary) WFN, RHO, or VXC file in a clearly labeled
  format, for human inspection.

wfnmerge.x:

  Merges many WFN files into one. It assumes that all input files have the
  same number and same ordering of G-vectors for the charge density.
  The number and name of input files is read from "wfnmerge.inp", as well
  is the kgrid and the kshift.

  FAQ:
  Q: Why would someone want to use this?
  A: For example, maybe to do a converged calculation one needs to include a
      large number of unoccupied states. There may not be the resources (either
      CPUs or wallclock) to do this all in one shot, so it may be beneficial to
      split up a calculation of kpoints (e.g., 4x4x4 MP grid) into smaller
      pieces and then add up the final wavefunctions.
  Q: What is the deal with kpoint weights?
  A: Quantum Espresso renormalizes the kpoint weights that you use in a
     calculation. If you are splitting up a MP grid into different groups of
     kpoints, and each group is normalized, the relative weights between
     kpoints in different groups is no longer correct. BerkeleyGW does not
     make use of the kpoint weights (except for in determining whether the
     Fermi level is reasonable for metallic calculations). So for most uses
     the fact that these weights are incorrect does not matter. If it matters
     to you, you can simply modify wfnmerge to read in the kpoint weights of
     your choosing.

convert_old_to_new.x:

  Convert wavefunction, density, and exchange-correlation potential files from the old
  CWFE(q)/CD95/VXC formats (used in the pre-release version of the code prior to June 2011)
  to the current WFN/RHO/VXC formats. Some additional information is required which was not
  available in the old files, so an input file convert_old_to_new.inp is required. See the
  example in this directory. Input and output files are both binary.
```

MeanField/Utilities/convert_old_to_new.inp

```
44.0 11.0
cubic
bohr
0.0000 5.1306 5.1306
5.1306 0.0000 5.1306
5.1306 5.1306 0.0000
```

```
crystal
2
-0.125 -0.125 -0.125 14
 0.125  0.125  0.125 14
2 2 2

# example input file for convert_old_to_new.x for Si
# coordinates are crystal, in units of the lattice vectors
# Note: you need high precision in the lattice vectors
# so that the volume calculated from them matches the volume
# in the input file
# The kgrid in the last line is only read if the kgrid in
# the wfn file contains a zero, since pre-r307 paratec writes
# 0 0 0 for shifted grids.

# charge-density and wave-function cutoffs (in Ry)
# cubic or hexagonal symmetry group
# units of the lattice vectors (bohr, angstrom)
# a1x a1y a1z
# a2x a2y a2z
# a3x a3y a3z
# units of the atomic positions (bohr, angstrom, crystal)
# number of atoms
# x1, y1, z1, atomic number 1
# x2, y2, z2, atomic number 2
# etc.
# kgrid_x kgrid_y kgrid_z
```

MeanField/Utilities/wfnmerge.inp

```
WFN.OUT      ! name of output WFN file
4 4 4        ! final k-grid
0.0 0.0 0.0 ! final k-shift
2            ! total number of input WFN files
8            ! total number of k-points in all input WFN files
wfn_0/WFN    ! name of 1st input WFN file
wfn_1/WFN    ! name of 2nd input WFN file
1.0          ! relative weight of 1st input WFN file
1.0          ! relative weight of 2nd input WFN file
```

MeanField/Utilities/fix_occ.inp

```
WFN_in  ! input  WFN
WFN_out ! output WFN
0       ! original number of electrons in WFN_in (0 to autodetect)
0       ! extra charge to put in
```

Epsilon

Epsilon/README

```
-----------------------------------------------------------------
----------  GW code, Epsilon  -----------------------------------
-----------------------------------------------------------------

  Version 1.0   (September, 2011)
  Version 0.5   J. Deslippe, D. Prendergast, L. Yang, F. Ribeiro, G. Samsonidze (2008)
  Version 0.2   S. Ismail-Beigi (2002)
  Version 0.1   G. M. Rignanese, E. Chang, X. Blase  (1998)
  Version 0.0   M. Hybertsen (1985)

-----------------------------------------------------------------

Description:

Epsilon is the name of the code that generates the polarizability matrix and
the inverse dielectric matrix for a bulk or nanoscale system. The main
result of the Epsilon code is the generation of epsmat which can be used
in a Sigma or BSE calculation.

-----------------------------------------------------------------

Required Input:

        epsilon.inp     Input parameters. See example in this directory.
        WFN             This is linked to the unshifted grid.
        WFNq            For non-box type truncated Coulomb interactions
                        (including no-truncation) or metallic/graphene
                        systems this is linked to the shifted grid.  If you are using
                        Coulomb truncation on a semiconductor, this is typically
                        simply linked to WFN, and q0 is set to exactly zero.

Auxiliary Files: (output files from previous runs, used as input to speed up calculation)

        chimat          Polarizability matrix. No need to recalculate matrix elements.
        chi0mat         Polarizability matrix at q=0. No need to recalculate matrix
elements.

        The files below are used if eqp_corrections is set in epsilon.inp.
        The corrected eigenvalues are used for constructing the polarizability matrix.
        eqp.dat         A list of quasiparticle energy corrections for the bands in WFN.
        eqp_q.dat       A list of quasiparticle energy corrections for the bands in WFNq.

-----------------------------------------------------------------

        epsilon.inp     Please see example epsilon.inp in this directory
                        for more complete options.

-----------------------------------------------------------------

Output Files:

        espmat                  Inverse dielectric matrix (q<>0).
        eps0mat                 Inverse dielectric matrix (q->0).
        epsilon.log             The log file containing values of chimat and epsmat.
        chi_converge.dat        Convergence chi with respect to empty orbitals.
                                Columns: number of conduction bands,
                                Re chi(G=0,G'=0,q),     extrapolated Re chi(G=0,G'=0,q),
                                Re chi(G=Max,G'=Max,q), extrapolated Re chi(G=Max,G'=Max,q)

-----------------------------------------------------------------

Tricks and hints:

1. Comments on convergence of epsilon_cutoff

There are many ways to check convergence of this parameter:

-Ideally, the inverse epsilon matrix is almost diagonal for large
G vectors. So, for small ( G , G' ) it is usually large but, for
 G or G' close to G_max, it is smaller by a factor of 100, more or
less. Check if the ratio between the greatest and lowest matrix
```

elements is large.

-Alternatively, one can check if epsinv(G_max , G'_max) is close
to 1 for G = G'. It should be about 0.999 or so. Note that 0.9 isn't
"close to 1" in this case! Also check if epsinv(G_max , G'_max) is
indeed small for G != G'.

2. Comments on the null point

The null point (Gamma point) is treated in a special manner. In input,
declare it as slightly shifted from the true Gamma point:

```
  0.0000    0.0050    0.0050   1.0   1
```

This is the only k-point with itestq=1 instead of 0!

The magnitude of this vector must be small: of order 0.01 or smaller
for less than 100 points in full Brillouin zone. If the density
of points in BZ increases, that magnitude should decrease accordingly.

3. Macroscopic dielectric constant

In GW, the screened interaction gives the formula:

```
  eps(G,G';q) = delta_GG' - [4pi e^2/(q+G)^2 ] chi(G,G';q)
```

where chi(G,G';q) is the polarizability (P in the literature).

```
  if G=0  and q=0

  then we let

   eps(0,G';q) = delta_0G' - lim(q->0) [4pi/(q)^2] chi(0,G';q)
```

always check chi(0,0;q->0) and eps^(-1)(0,0;q->0)
in epsilon.log

```
1/eps^(-1)(0,0;q->) should be the macroscopic epsilon,
                    local-field effects
1/(1-4*pi*chi(0,0;q->0)*V_c(0) should also be (roughly) the macroscopic epsilon,
                              without local-field effects
```

4. Merging epsmat files

The Epsilon code is complex and involves lots of computation. Frequently,
it is useful to calculate the epsilon matrix for a small number
of points, and later put all those pieces of matrix in a small big
epsmat file. Use epsmat_merge to do this. See below.

Notes:
-epsilon cutoff *must* be the same for all files to be merged
-The ordering of q-points must correspond to the ordering of epsmat
files: the first file has the first set of points, the second file
has the second and so on.

5. Utilities:

```
----------------------------------------------------------------------
---------  epsmat, eps0mat conversion  -------------------------------
----------------------------------------------------------------------
```

TOOLS: epsbinasc, epsascbin

USAGE: simply type name of executable.

The input file is named epsconv.inp.

```
----------------------------------------------------------------------
--------  WFN/RHO/VXC conversion -----------------------------------
----------------------------------------------------------------------

TOOL: mf_convert_wrapper.sh

USAGE:
  sh mf_convert_wrapper.sh infile outfile

Real/complex, bin/asc, file type, is automatically detected.

----------------------------------------------------------------------
---------  epsmat merging  -----------------------------------------
----------------------------------------------------------------------

TOOL: epsmat_merge

USAGE:  epsmat_merge.[real/cplx].x epsmat1 epsmat2 [more epsmat files] epsmat_out

Merges the content of binary files epsmat1 and epsmat2 into
epsmat_out, basically concatenating data.


Converters from old versions of file formats to current version are available in version
2.4.
```

Epsilon/epsilon.inp

```
# epsilon.inp

# G-vectors will be used with kinetic energies up to this cutoff (Ry)
epsilon_cutoff           8.0

# number of bands to sum over
number_bands             146

# a list of which bands are occupied (1) and which are unoccupied (0)
band_occupation          18*1 128*0

# If you have partially occupied bands (Metal) set
# number_partial_occup to the number of these bands.
# And set occupation of these bands to zero in the
# band_occupation line above.
# (this is former ncrit)
#number_partial_occup    0

# Specify the Fermi level (in eV), if you want implicit doping
# Note that value refers to energies AFTER scissor shift or eqp corrections.
#fermi_level             0.0

# The Fermi level is treated as an absolute value
# or relative to that found from the mean field (default)
#fermi_level_absolute
#fermi_level_relative

# RECOMMENDED fast FFTW truncation schemes
# The Coulomb Interaction is cutoff on the edges of
# the Wigner-Seitz Cell in the non-periodic directions
# Periodic directions are a1,a2 for slabs and a3 for wires

#cell_box_truncation
#cell_wire_truncation
#cell_slab_truncation
```

```
# Analytic, but non-Wigner-Seitz Cell Truncation

#spherical_truncation

# For Spherical Truncation, radius in Bohr
#coulomb_truncation_radius   10.00

# Frequency dependence of the inverse dielectric matrix.
# Set to 0 to compute the static inverse dielectric matrix (default).
# Set to 2 to compute the full frequency dependent inverse dielectric matrix.
#frequency_dependence 0

# Parameters for the full frequency dependent inverse dielectric matrix.
# All are in units of eV.
# Specify the initial frequency, the initial frequency increment,
# and the Lorentzian broadening.
#
# init_frequency should always be set to zero, delta_frequency to a small
# positive real number, broadening to a small positive real number close to delta_frequency
#
# The frequency cutoffs work in the following way: Up until the low_cutoff, the frequency
grid will be
# uniform and spaced by delta_frequency.  After the frequency_low_cutoff, the frequency grid
will
# begin to be more sparse with the grid spacing increasing by an amount delta_frequency_step
# after each grid point.  The frequency grid is truncated all together after the
frequency_high_cutoff.
#
#init_frequency 0.0
#delta_frequency 0.2
#delta_frequency_step 0.5
#frequency_low_cutoff 10.0
#frequency_high_cutoff 40.0
#broadening 0.2

# Logging convergence of the head & tail of polarizability matrix.
# Set to 0 for the 5 column format including the extrapolated values (default).
# Set to 1 for the 2 column format, real part only.
# Set to 2 for the 2 column format, real and imaginary parts.
#full_chi_conv_log 0

# qx qy qz 1/scale_factor is_q0
# scale_factor is for specifying values such as 1/3
# is_q0 = 1 for a small q-vector in semiconductors
# is_q0 = 2 for a small q-vector in metals
# is_q0 = 0 for non-zero q-vectors
# if present the small q-vector should be first in the list
# You can generate this list with kgrid.x: just set the shifts to zero and use
# same grid numbers as for WFN. Then replace the zero vector with q0.
number_qpoints  16
begin qpoints
  0.000000    0.000000    0.005000   1.0   1
  0.000000    0.000000    0.062500   1.0   0
  0.000000    0.000000    0.125000   1.0   0
  0.000000    0.000000    0.187500   1.0   0
  0.000000    0.000000    0.250000   1.0   0
  0.000000    0.000000    0.312500   1.0   0
  0.000000    0.000000    0.375000   1.0   0
  0.000000    0.000000    0.437500   1.0   0
  0.000000    0.000000    0.500000   1.0   0
  0.000000    0.000000    0.562500   1.0   0
  0.000000    0.000000    0.625000   1.0   0
  0.000000    0.000000    0.687500   1.0   0
  0.000000    0.000000    0.750000   1.0   0
  0.000000    0.000000    0.812500   1.0   0
  0.000000    0.000000    0.875000   1.0   0
  0.000000    0.000000    0.937500   1.0   0
end

# Scissors operator (linear fit of the quasiparticle
# energy corrections) for the bands in WFN and WFNq.
# e_cor = e_in + es + edel * (e_in - e0)
# Defaults below. evs, ev0, ecs, ec0 are in eV.
# If you have eqp.dat and eqp_q.dat files
# this information is ignored in favor of the eigenvalues
# in eqp.dat and eqp_q.dat.
#evs     0.0
#ev0     0.0
#evdel   0.0
#ecs     0.0
#ec0     0.0
#ecdel   0.0
# or
#cvfit   0.0 0.0 0.0 0.0 0.0 0.0

# Set this to use eigenvalues in eqp.dat and eqp_q.dat
# If not set, these files will be ignored.
#eqp_corrections

# Write the bare Coulomb potential V(q+G) to file
#write_vcoul

# Matrix Element Communication Method (Chi Sum Comm)
# Default is gcomm_matrix which is good if nk*nc*nv > nmtx*nfreq
# If nk*nc*nv < nfreq*nmtx (nk*nv < nfreq since nc~nmtx),
# use gcomm_elements
gcomm_matrix
#gcomm_elements

# Communication through MPI or DISK
# comm_mpi is usually much faster and preferable but if you have only
# a few CPUs you might not have enough memory to hold wavefunctions
# comm_disk results in temporary INT_* files being created and it
# may be faster for a small unit cell and a lot of k-points
# The default is comm_mpi
comm_mpi
#comm_disk

# Number of pools for distribution of valence bands
# The default is chosen to minimize memory in calculation
#number_valence_pools 1

# By default, the code computes the polarizability matrix, constructs
# the dielectric matrix, inverts it and writes the result to file epsmat.
# Use keyword skip_epsilon to compute the polarizability matrix and
# write it to file chimat. Use keyword skip_chi to read the polarizability
# matrix from file chimat, construct the dielectric matrix, invert it and
# write the result to file epsmat.
#skip_epsilon
#skip_chi

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY
# 'unfolded BZ' is from the kpoints in the WFN file
# 'full BZ' is generated from the kgrid parameters in the WFN file
# See comments in Common/checkbz.f90 for more details
# Replace unfolded BZ with full BZ
#fullbz_replace
# Write unfolded BZ and full BZ to files
#fullbz_write

# The requested number of bands cannot break degenerate subspace
# Use the following keyword to suppress this check
# Note that you must still provide one more band in
# wavefunction file in order to assess degeneracy
#degeneracy_check_override
```

Epsilon/epsconv.inp

```
#----------------------------
# Example epsconv.inp File

# number of q-points
# list of q-points in format given in epsilon.inp,
#   i.e. qx, qy, qz, divisor (q0 flag not needed, will be ignored)
# output filename
# number of input files (will be merged)
# input filenames

1
0.0 0.0 .03125000000000000000 1.0
epsmat.out
1
epsmat.in
```

Epsilon/epsomega.inp

```
eps0mat       ! epsmat file, full-frequency or static
RHO           ! RHO file for Plasmon-Pole
0.0 0.0 0.0   ! q-vector in crystal coordinates
1 0 0         ! G-vector in crystal coordinates
1 0 0         ! G`-vector in crystal coordinates
201           ! nFreq, for static epsmat. For full-freq., overwritten from file
0.5           ! dDeltaFreq in eV, in case of static epsmat. See above.
2.0           ! dBrdning in eV, in case of static epsmat. See above.
epsPP.dat     ! Plasmon-Pole epsilon, Re and Im parts
epsR.dat      ! Retarded epsilon, Re and Im parts
epsA.dat      ! Advanced epsilon, Re and Im parts
```

Epsilon/epsinvomega.inp

```
eps0mat       ! epsmat file, full-frequency or static
RHO           ! RHO file for Plasmon-Pole
0.0 0.0 0.0   ! q-vector in crystal coordinates
1 0 0         ! G-vector in crystal coordinates
1 0 0         ! G`-vector in crystal coordinates
201           ! nFreq, for static epsmat. For full-freq., overwritten from file
0.5           ! dDeltaFreq in eV, in case of static epsmat. See Above.
2.0           ! dBrdning in eV, in case of static epsmat. See Above.
0.1           ! Exciton binding energy to use to calculate the effective eps^-1 head for BSE.
epsInvPP.dat ! Plasmon-Pole epsilon inverse, Re and Im parts
epsInvR.dat  ! Retarded epsilon inverse, Re and Im parts
epsInvA.dat  ! Advanced epsilon inverse, Re and Im parts
```

Epsilon/epsmat_merge.inp

```
44 7 ! ecuts1, nqtot
```

```
  0.0000  0.0000  0.5000  1.0 ! qx qy qz div
  0.0000  0.2500  0.0000  1.0
  0.0000  0.2500  0.5000  1.0
  0.0000  0.5000  0.0000  1.0
  0.0000  0.5000  0.5000  1.0
  0.2500  0.5000  0.0000  1.0
  0.2500  0.5000  0.5000  1.0
1 ! nfiles
epsmat ! filename(i), i=1,nfiles
```

Sigma

Sigma/README

```
-----------------------------------------------------------------
----------  GW code, Sigma  -------------------------------------
-----------------------------------------------------------------

  Version 1.0   (September, 2011)

  Version 0.5   J. Deslippe, D. Prendergast, L. Yang, F. Ribeiro, G. Samsonidze (2008)
  Version 0.2   S. Ismail-Beigi (2002)
  Version 0.1   G. M. Rignanese, E. Chang, X. Blase  (1998)
  Version 0.0   M. Hybertsen (1985)

-----------------------------------------------------------------

Description:

Sigma is the second half of the GW code.  It should be run after
calculating epsmat/esp0mat in Epsilon.  It gives the quasiparticle
self-energies and dispersion relation for quasielectron and
quasihole states.  The main result is written to sigma.log file.

-----------------------------------------------------------------

Required Input:

        sigma.inp       Input parameters.  See example in this directory.
        epsmat          Inverse dielectric matrix (q<>0).  Created using Epsilon.
        eps0mat         Inverse dielectric matrix (q->0).  Created using Epsilon.
        WFN_inner       Real or complex wavefunctions. k-grid should be the same as
                        the q-grid of epsmat, though there can be a shift.
                        This is usually the reduced points from an
                        unshifted grid, i.e. PARATEC is called with no kgrid_shift.
        RHO             Charge density (for generalized plasmon-pole model).
                        Produced by PARATEC using gwscreening on same grid
                        as WFN_inner.
        vxc.dat         Matrix elements of the exchange-correlation potential,
                        from a mean-field calculation or a previous Sigma run.
                        VXC file may be used instead via keyword dont_use_vxcdat.

Additional Input:

        WFN_outer               Outer wavefunctions between which the
                                self-energy operator and exchange-correlation
                                potential are sandwiched. If absent inner
                                wavefunctions will be used instead. Note that
                                VXC should be consistent with outer wfn while
                                RHO with inner wfn. If outer wfn is generated
                                using a hybrid functional VXC contains the
                                local part of exchange-correlation potential.
```

```
                              In this case set bare_exchange_fraction in
                              sigma.inp to compensate for the non-local part,
                              or use vxc.dat instead of VXC.

        The file below will be read only if eqp_corrections is set in sigma.inp.
        eqp.dat               A list of quasiparticle energy corrections
                              for the bands in WFN_inner.
                              The corrected eigenvalues are used for
                              constructing the self-energy operator.

        The file below will be read only if eqp_outer_corrections is set in sigma.inp.
        eqp_outer.dat         A list of quasiparticle energy corrections for
                              the bands in WFN_outer.
                              The corrected eigenvalues determine
                              the energies at which the self-energy
                              operator is calculated.

Auxiliary Files:

        x.dat           Matrix elements of the bare exchange, generated by a previous
                        Sigma run to speed up subsequent calculations.
                        Used only if use_xdat keyword is present.
        VXC             Exchange-correlation potential (whose matrix elements
                        are subtracted from DFT eigenvalues). Produced by
                        mean-field code, on same grid as WFN_inner.
                        Used only if vxc.dat is not present or keyword dont_use_vxcdat is
set.


-----------------------------------------------------------------

        sigma.inp       Please see example sigma.inp in this directory
                        for more complete options.

-----------------------------------------------------------------

Output Files:

        sigma.log       The log file containing quasiparticle energy values
                        for desired states. For a full-frequency calculation
                        only the value for Sigma calculated at energy closest
                        to the outer wavefunction eigenvalue is shown. The file
                        spectrum.dat contains the full Sigma(E) spectra.

        sigma_hp.log    High-precision version of sigma.log

        ch_converge.dat Convergence of Sigma_ch with respect to empty orbitals.

        spectrum.dat    The real and imaginary parts of Sigma as a function of
                        energy. The energy grid is specified in sigma.inp. This
                        file is only output in full-frequency calculations.
                        Some general notes:

                        * IM(SIGMA) was calculated using the same broadening
                        as RE(SIGMA), while IM(SIGMA2) was calculated without
                        any broadening. IM(SIGMA2) has the right properties
                        (non-positive and vanishing at the Fermi level), but it
                        is currently only implemented for systems with inversion
                        symmetry. If IM(SIGMA) and IM(SIGMA2) are too far apart
                        (>0.5eV), or if IM(SIGMA) is positive, rerun your
                        calculation with a finer frequency sampling.

                        * Ew is not the same as the frequencies specified in
                        sigma.inp. Ew is the absolute off-shell quasiparticle
                        energy, and it is *not* measured wrt the Fermi energy.
                        The (physically meaningful) on-shell quasiparticle
```

```
                         energy EQP is the solution of Dyson's equation:
                         Ew = E_LDA - Vxc + Re[Sig(Ew)].
```

A Note About the Wings of Epsilon (for Semiconductors Only):

The wings of Chi have terms of the following form:

< vk | e^(i(G+q)r) | ck+q >< ck+q | e^(-iqr) | vk > (1)

The matrix element on the right is < u_ck+q | u_vk > where u is the periodic part
of the Bloch function. From k.p perturbation theory, this matrix element is proportional
to q. The matrix element on the left with a non-zero G is typically roughly
a constant as a function of q for small q (q being a small addition to G).

Thus for a general G-vector, Chi_wing(G,q) \propto q. This directly leads
to the wings of the screened untruncated Coulomb interaction being proportional
to 1/q. Note that this function changes sign as q -> -q. Thus, when treating the
q=0 point, we set the value of the wing to zero (the average of its value in the
mini-Brillouin zone (mBZ).

For G-vectors on high-symmetry lines, some of the matrix elements on the left of (1)
will be zero for q=0, and therefore proportional to q. For such cases,
Chi_wing(G,q) \propto q^2, and the wings of the screened Coulomb interaction
are constant as a function of q. However, setting the q->0 wings to zero still
gives us, at worst, linear convergence to the final answer with increased k-point
sampling, because the q->0 point represents an increasingly smaller area in the
BZ. Thus, we still zero out the q->0 wings, as discussed in
A Baldereschi and E Tosatti, Phys. Rev. B 17, 4710 (1978).

In the future, it may be worthwhile to have the user calculate chi / epsilon at
two q-points (a third q-point at q=0 is known) in order to compute the linear
and quadratic coefficients of each chi_wing(G,q) so that all the correct analytic
limits can be taken. This requires a lot of messy code and more work for the user
for only marginally better convergence with respect to k-points (the wings tend
to make a small difference, and this procedure would matter for only a small set
of the G-vectors).

It is important, as always, for the user to converge their calculation
with respect to both the coarse k-point grid used in sigma and kernel as well
as with the fine k-point grid in absorption.

-JRD+MJ

Tricks and hints:

1. Comments on convergence of scc (screened_coulomb_cutoff)
   and bcc (bare_coulomb_cutoff)

The parameters scc and bcc are independent to each other and are
related to the number or terms taken into account in the
plane-wave expansion of Sigma_(SEX + COH) and V_xc, respectively.

Since Sigma_(SEX) is related to epsilon, scc should be equal to
or less than the epsilon_cutoff used to compute chi0 and epsilon.

If you are to use large values for scc and bcc (say 50 Ry or so),
a good suggestion is run the sigma code twice: first with large
bcc and small scc, just to get x.dat and vxc.dat; then with large
scc and any bcc, using the x.dat and vxc.dat files generated before.
The second run is usually much faster than a run with large scc
and bcc.

```
2. Metals

If you are doing a metal you should report the shift used in sigma.inp
whether using truncation or not.

3. Off-diagonal

If you are doing off-diagonal matrix elements of Sigma use the utility
offdiag in this directory to diagonalize the Sigma matrix.

4. Linear extrapolation for eqp1

If |eqp0 - ecor| > finite_difference_spacing linear extrapolation
for eqp1 may be inaccurate. You should test the validity of eqp1
by rerunning calculation with self-energy evaluated at the eqp0
energies. For that, use the eqp_outer.dat file created
with eqp.py script and point WFN_outer to WFN_inner (if you were
not already using WFN_outer).
```

Sigma/sigma.inp

```
# sigma.inp

# G-vectors will be used with kinetic energies up to this cutoff (Ry)
# screened_coulomb_cutoff must be <= epsilon_cutoff used for Epsilon run
screened_coulomb_cutoff  8.0
bare_coulomb_cutoff      60.0

# number of bands to sum over
number_bands             146

# a list of which bands are occupied (1) and which are unoccupied (0)
band_occupation          18*1 128*0
# Note that what is above is Fortran shorthand for 18 1's and 128 0's.
# i.e. "4*1 4*0" is equivalent to "1 1 1 1 0 0 0 0"; either may be used here.

# If you have partially occupied bands (Metal) set
# number_partial_occup to the number of these bands.
# Set occupation of these bands to zero in the
# band_occupation line above.
# This is for bands that cross the Fermi energy, so
# that they are occupied at some k-points and not at
# others, rather than for thermal partial occupations.
# (this is former ncrit)
#number_partial_occup    0

# Specify the Fermi level (in eV), if you want implicit doping
# Note that value refers to energies AFTER scissor shift or eqp corrections.
#fermi_level             0.0

# The Fermi level is treated as an absolute value
# or relative to that found from the mean field (default)
#fermi_level_absolute
#fermi_level_relative

# The matrix elements of Sigma
# < psi_n | Sigma(E_l) | psi_m >
# E_l is irrelevant for frequency_dependence -1 and 0

# The diagonal matrix elements of Sigma (n .eq. m .eq. l)
#number_diag   ndiag
#begin diag
#   n_1
#   n_2
#   ...
#   n_ndiag
#end
# or specify a range of bands spanned by n
band_index_min   15
band_index_max   22

# The off-diagonal matrix elements of Sigma
# band_index_min .le. n .le. band_index_max
# band_index_min .le. m .le. band_index_max
# band_index_min .le. l .le. band_index_max
# NOTE: the offdiag utility only works with the sigma_matrix syntax, below.
#number_offdiag   noffdiag
#begin offdiag
#   n_1   m_1   l_1
#   n_2   m_2   l_2
#   ...
#   n_noffdiag   m_noffdiag   l_noffdiag
#end
# or select a specific value of l and let n and m vary
# in the range from band_index_min to band_index_max
# Set l to 0 to skip the off-diagonal calculation (default)
# If l = -1 then l_i is set to n_i (i = 1 ... noffdiag)
# i.e. each row is computed at different eigenvalue
# If l = -2 then l_i is set to m_i (i = 1 ... noffdiag)
# i.e. each column is computed at different eigenvalue
# For l > 0, all elements are computed at eigenvalue of band l.
# Set t to 0 for the full matrix (default)
# or to -1/+1 for the lower/upper triangle
#sigma_matrix   l   t

# The range of spin indices for which Sigma is calculated
# The default is the first spin component
#spin_index_min   1
#spin_index_max   1

# What screening is present? (default = semiconductor)
screening_semiconductor
#screening_metal
#screening_graphene
#  this is for a system with linear DOS at the Fermi level

# RECOMMENDED fast FFTW truncation schemes
# The Coulomb Interaction is cutoff on the edges of
# the Wigner-Seitz Cell in the non-periodic directions
# Periodic directions are a1,a2 for slabs and a3 for wires

#cell_box_truncation
#cell_wire_truncation
#cell_slab_truncation

# Analytic, but non Wigner-Seitz cell, truncation

#spherical_truncation

# For Spherical Truncation, radius in Bohr
#coulomb_truncation_radius   10.00

# Cutoff energy for averaging the Coulomb Interaction
# in the mini-Brillouin Zones around the Gamma-point
# without Truncation or for Cell Wire or Cell Slab Truncation.
# The value is in Rydbergs, the default is 10^{-12}
#cell_average_cutoff 1.0d-12

# Frequency dependence of the inverse dielectric matrix.
# Set to -1 for the Hartree-Fock approximation.
# Set to 0 for the static COHSEX approximation.
# Set to 1 for the Generalized Plasmon Pole model (default).
# Set to 2 for the full frequency dependence.
#frequency_dependence 1

# For Full Frequency Calculations.
# The number_frequency_eval is the number of frequency points
# to evaluate Sigma(omega) at.  The init_frequency_eval (eV)
# is the first frequency and delta_frequency_eval (eV) is the
# frequency step. The variable init_frequency_eval should be set
# relative to the Fermi energy, i.e., 0.0 means the Fermi energy.
# These are NOT the values used in Epsilon for the evaluation of chi
# and epsilon - those are automatically read in from epsmat file.
# You can specify that the integrals are done on a finer grid than
# the epsmat frequency grid. To do this specify interpolation n
# (where n-1 additional points will be added for each point on
# the epsmat frequency grid and a reduction of the broadening
# by a factor n will be performed).
#init_frequency_eval 0.0
#delta_frequency_eval 0.1
#number_frequency_eval 501
#interpolation 2

# For Generalized Plasmon Pole.
# The matrix element of the self-energy operator is
# expanded to first order in the energy around Ecor.
# Finite difference form for numerical derivative of Sigma.
# none     = -2 : dSigma/dE = 0 [skip the expansion]
# backward = -1 : dSigma/dE = (Sigma(Ecor) - Sigma(Ecor-dE)) / dE
# central  =  0 : dSigma/dE = (Sigma(Ecor+dE) - Sigma(Ecor-dE)) / (2*dE)
# forward  =  1 : dSigma/dE = (Sigma(Ecor+dE) - Sigma(Ecor)) / dE
# default  =  2 : forward for diagonal and none for off-diagonal
#finite_difference_form   2
# dE is finite difference spacing given in eV
# The default value is 1.0
#finite_difference_spacing   1.0

# For Hartree-Fock, no epsmat/eps0mat files are needed.
# Instead provide a list of q-points and the grid size.
# The list of q-points should not be reduced with time
# reversal symmetry - because BerkeleyGW never uses time
# reversal symmetry to unfold the q/k-points. Instead,
# inversion symmetry does the job in the real version of
# the code.
# qx qy qz 1/scale_factor is_q0
# scale_factor is for specifying values such as 1/3
# You can generate this list with kgrid.x: just set the shifts to zero and use
# same grid numbers as for WFN_inner. Then replace the zero vector with q0.
#number_qpoints  1
#begin qpoints
#  0.0  0.0  0.0  1.0  1
#end
#qgrid  1  1  1

# Summing SX and CH over G and G' vectors.
# Set to 1 for half sum (default) or 2 for full sum.
# Using 1 assumes W(G,G') = W(G',G) which should be
# satisfied in most situations. The only
# time you need to use 2 is if you are using a different
# Coulomb interaction (truncation scheme) between Epsilon
# and Sigma.  This is not really recommended anyway.
# If you are using no truncation, doing ggpsum = 1 is
# not exactly correct: W(G,G') will not be symmetric because
# we used an averaged V(0) in Sigma and a Non-Averaged V(q0)
# in Epsilon. This leads to errors on the order of 1 meV or less
# in examples.
#ggpsum 1

# Add remainder from tail of epsilon for full frequency.
#use_epsilon_remainder

# Logging CH convergence.
# Set to 0 for the valence and conduction bands (default).
# Set to 1 for all bands, real part only.
# Set to 2 for all bands, real and imaginary parts.
# Note that CH in ch_converge.dat should be compared
# against unsymmetrized values in the standard output.
# This is different from sigma.log for degenerate states.
#full_ch_conv_log 0

# Use precalculated matrix elements of bare exchange from x.dat.
# The default is not to use them.
#use_xdat

# Do not use precalculated matrix elements of exchange-correlation from vxc.dat.
# The default is to use them.
#dont_use_vxcdat

# Fraction of bare exchange.
# Set to 1.0 if you use the exchange-correlation matrix elements
# read from file vxc.dat. Set to 1.0 for local density functional,
# 0.0 for HF, 0.75 for PBE0, 0.80 for B3LYP if you use the local
# part of the exchange-correlation potential read from file VXC.
# For functionals such as HSE whose nonlocal part is not some
# fraction of bare exchange, use vxc.dat and not this option.
# This is set to 1.0 by default.
#bare_exchange_fraction   1.0

# Broadening for the energy denominator in CH and SX within GPP.
# If it is less than this value, the sum is better conditioned than
# either CH or SX directly, and will be assigned to SX while CH = 0.
# This is given in eV, the default value is 0.5
#gpp_broadening   0.5
# Cutoff for the poles in SX within GPP.
# Divergent contributions that are supposed to sum to zero are removed.
# This is dimensionless, the default value is 4.0
#gpp_sexcutoff   4.0

# kx ky kz 1/scale_factor
# scale_factor is for specifying values such as 1/3
number_kpoints   1
begin kpoints
  0.0000  0.0000  0.0000  1.0
end

# Scissors operator (linear fit of the quasiparticle
# energy corrections) for the bands in WFN_inner
# e_cor = e_in + es + edel * (e_in - e0)
# Defaults below. evs, ev0, ecs, ec0 are in eV
#evs     0.0
#ev0     0.0
#evdel   0.0
#ecs     0.0
#ec0     0.0
#ecdel   0.0
# or
#cvfit   0.0 0.0 0.0 0.0 0.0 0.0

# Scissors operator (linear fit of the quasiparticle
# energy corrections) for the bands in WFN_outer
# e_cor = e_in + es + edel * (e_in - e0)
# Defaults below. evs_outer, ev0_outer, ecs_outer, ec0_outer are in eV
#evs_outer     0.0
#ev0_outer     0.0
#evdel_outer   0.0
#ecs_outer     0.0
#ec0_outer     0.0
#ecdel_outer   0.0
# or
#cvfit_outer   0.0 0.0 0.0 0.0 0.0 0.0

# Set this to use eigenvalues in eqp.dat
# If not set, this file will be ignored.
#eqp_corrections
# Set this to use eigenvalues in eqp_outer.dat
# If not set, this file will be ignored.
#eqp_outer_corrections

# The average potential on the faces of the unit cell
# in the non-periodic directions for the bands in WFN_inner
# This is used to correct for the vacuum level
# The default is zero, avgpot is in eV
#avgpot   0.0

# The average potential on the faces of the unit cell
# in the non-periodic directions for the bands in WFN_outer
# This is used to correct for the vacuum level
# The default is zero, avgpot_outer is in eV
#avgpot_outer   0.0

# Write the bare Coulomb potential V(q+G) to file
#write_vcoul

# Communication through MPI or DISK
# comm_mpi is usually much faster and preferable but if you have only
# a few CPUs you might not have enough memory to hold wavefunctions
# comm_disk results in temporary INT_* files being created and it
# may be faster for a small unit cell and a lot of k-points
# The default is comm_mpi.
comm_mpi
#comm_disk

# Number of pools for parallel sigma calculations
# The default is chosen to minimize memory in calculation
#number_sigma_pools 1

# Threshold for considering bands degenerate, for purpose of
# band-averaging and setting offdiagonals to zero by symmetry. (eV)
tol_degeneracy 1e-6

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY
# 'unfolded BZ' is from the kpoints in the WFN_inner file
# 'full BZ' is generated from the kgrid parameters in the WFN_inner file
# See comments in Common/checkbz.f90 for more details
# Replace unfolded BZ with full BZ
#fullbz_replace
# Write unfolded BZ and full BZ to files
#fullbz_write

# The requested number of bands cannot break degenerate subspace
# Use the following keyword to suppress this check
# Note that you must still provide one more band in
# wavefunction file in order to assess degeneracy
#degeneracy_check_override

# The sum over q-points runs over the full Brillouin zone.
# For diagonal matrix elements between non-degenerate bands
# and for spherically symmetric Coulomb potential (no truncation
# or spherical truncation), the sum over q-points runs over
# the irreducible wedge folded with the symmetries of
# a subgroup of the k-point. The latter is the default.
# In both cases, WFN_inner should have the reduced k-points
# from an unshifted grid, i.e. same as q-points in Epsilon.
# With no_symmetries_q_grid, any calculation can be done;
# use_symmetries_q_grid is faster but only diagonal matrix elements
# of non-degenerate or band-averaged states can be done.
#no_symmetries_q_grid
#use_symmetries_q_grid

# Off-diagonal elements are zero if the two states belong to
# different irreducible representations. As a simple proxy,
# we use the size of the degenerate subspaces of the two states:
# if the sizes are different, the irreps are different, and the
# matrix element is set to zero without calculation.
# Turn off this behavior for testing by setting flag below.
# Using WFN_outer effectively sets no_symmetries_offdiagonals.
#no_symmetries_offdiagonals

# Rotation of the k-points may bring G-vectors outside of the sphere.
# Use the following keywords to specify whether to die if some of
# the G-vectors fall outside of the sphere. The default is to die.
# Set to die in case screened_coulomb_cutoff = epsilon_cutoff.
# Set to ignore in case screened_coulomb_cutoff < epsilon_cutoff.
#die_outside_sphere
#ignore_outside_sphere

# Dealing with the convergence of the CH term.
# Set to 0 to compute a partial sum over empty bands.
# Set to 1 to compute the exact static CH.
# In case of exact_static_ch = 0 and frequency_dependence = 0 (COHSEX),
# columns ch, sig, eqp0, eqp1 contain the exact static CH,
# columns ch', sig', eqp0', eqp1' contain the partial sum static CH,
# and ch_converge.dat contains the static limit of the partial sum.
# For exact_static_ch = 1 and frequency_dependence = 0 (COHSEX),
# columns ch', sig', eqp0', eqp1' are not printed and
# file ch_converge.dat is not written.
# Default is 0 for frequency_dependence = 1 and 2;
# 1 for frequency_dependence = 0;
# has no effect for frequency_dependence  = -1.
#
# It is important to note that the exact static CH answer
# depends not only on the screened Coulomb cutoff but also on the bare Coulomb cutoff
# because G-G' for G's within the screened Coulomb cutoff can be outside the screened
# Coulomb cutoff sphere. And, therefore, the bare Coulomb cutoff sphere is used.
#exact_static_ch 0
```

Sigma/sig2wan.inp

```
sigma_hp.log   ! Sigma output file to read k-points, eigenvalues and symmetries from
1            ! spin component to read from sigma_hp.log file
1            ! set to 0 or 1 to read eqp0 or eqp1 from sigma_hp.log file
prefix.nnkp  ! Wannier90 input file to read k-points from
prefix.eig   ! file where the output of sig2wan is written
```

BSE

BSE/README_kernel

```
-----------------------------------------------------------------
----------  BSE code, Kernel  -----------------------------------
-----------------------------------------------------------------

  Version 1.0   (July, 2011) J. Deslippe, M. Jain, D. A. Strubbe, G. Samsonidze.

  Version 0.5   J. Deslippe, D. Prendergast, L. Yang, F. Ribeiro, G. Samsonidze (2008)
  Version 0.2   C. Spataru, S. Ismail-Beigi (2004)
  Version 0.1   M. L. Tiago, E. Chang, G. M. Rignanese (1999)

-----------------------------------------------------------------

Description:

This code constructs the direct and exchange Kernel matrix on
the coarse grid.  This is done essentially by computing Eqs 34,
35 and 42-46 of Rohlfing and Louie.

-----------------------------------------------------------------

Required Input:

        kernel.inp      Input parameters.  See detailed explanation below.
        WFN_co          Wavefunctions on coarse grid. Recommended: use an unshifted
                        grid of the same size as the q-grid in epsmat.
                        Shift will increase number of q-vectors needed in epsmat.

        epsmat          Inverse dielectric matrix (q<>0).  Created using Epsilon.
                        Must contain the all q=k-k' generated from WFN_co, including
                        with symmetry if use_symmetries_coarse_grid is set.
        eps0mat         Inverse dielectric matrix (q->0).  Created using Epsilon.

                        Note Kernel does not require quasiparticle eigenvalues. It
                        may be run in parallel with Sigma.

-----------------------------------------------------------------

        kernel.inp      Please see example kernel.inp in this directory
                        for more complete options.

-----------------------------------------------------------------

Output Files:

        bsedmat         Direct kernel matrix elements on unshifted coarse grid.
        bsexmat         Exchange kernel matrix elements on unshifted coarse grid.

-----------------------------------------------------------------

A Note About the Wings of Epsilon (for Semiconductors Only):

The wings of Chi have terms of the following form:

< vk | e^(i(G+q)r) | ck+q >< ck+q | e^(-iqr) | vk > (1)

The matrix element on the right is < u_ck+q | u_vk > where u is the periodic part
of the Bloch function. From k.p perturbation theory, this matrix element is proportional
to q. The matrix element on the left with a non-zero G is typically roughly
a constant as a function of q for small q (q being a small addition to G).

Thus for a general G-vector, Chi_wing(G,q) \propto q. This directly leads
to the wings of the screened untruncated Coulomb interaction being proportional
to 1/q. Note that this function changes sign as q -> -q. Thus, when treating the
```

```
q=0 point, we set the value of the wing to zero (the average of its value in the
mini-Brillouin zone (mBZ).

For G-vectors on high-symmetry lines, some of the matrix elements on the left of (1)
will be zero for q=0, and therefore proportional to q. For such cases,
Chi_wing(G,q) \propto q^2, and the wings of the screened Coulomb interaction
are constant as a function of q. However, setting the q->0 wings to zero still
gives us, at worst, linear convergence to the final answer with increased k-point
sampling, because the q->0 point represents an increasingly smaller area in the
BZ. Thus, we still zero out the q->0 wings, as discussed in
A Baldereschi and E Tosatti, Phys. Rev. B 17, 4710 (1978).

In the future, it may be worthwhile to have the user calculate chi / epsilon at
two q-points (a third q-point at q=0 is known) in order to compute the linear
and quadratic coefficients of each chi_wing(G,q) so that all the correct analytic
limits can be taken. This requires a lot of messy code and more work for the user
for only marginally better convergence with respect to k-points (the wings tend
to make a small difference, and this procedure would matter for only a small set
of the G-vectors).

It is important, as always, for the user to converge their calculation
with respect to both the coarse k-point grid used in sigma and kernel as well
as with the fine k-point grid in absorption.

-JRD+MJ

------------------------------------------------------------------

Tricks and hints:

1. To optimize distribution of work among PEs, do the following:

Choose processors to divide:

nk^2 (if npes < nk^2)
nk^2*nc^2 (if npes < nk^2*nc^2)
nk^2*nc^2*nv^2 (if npes < nk^2*nc^2*nv^2)

2. All input and output files (except kernel.inp) are in
binary format.

3. The interaction matrices are calculated in full! i.e. not only the
upper triangle. In diag, currently only the upper triangle is used.

4. The Brillouin zone is built using a Wigner-Seitz construction,
this way the head matrix elements are easily calculated.

5. The dielectric matrix is by default stored in memory but may be stored
on disk if the comm_disk options is specified.

6. The parameters scc (screened_coulomb_cutoff) and bcc
(bare_coulomb_cutoff) are the same as scc and bcc used in Simga.
The parameters scc and bcc should be equal to or less than the
epsilon_cutoff and wavefunction_cutoff used in Epsilon and DFT,
respectively. See Sigma/README for more details.

Converters from old versions of file formats to current version are available in version
2.4.
```

BSE/kernel.inp

```
# kernel.inp

# specify the number of valence bands (counting down from HOMO)
```

```
number_val_bands 3

# conduction bands (counting up from LUMO)
number_cond_bands 3

# screened coulomb cutoff (Ry) - must be specified as non-zero
# ecute in the code
screened_coulomb_cutoff 8.0

# bare coulomb cutoff (Ry) - if not specified set to ecute + max(bdot)
# ecutg in the code
bare_coulomb_cutoff 60.0

# Specify the Fermi level (in eV), if you want implicit doping
# Note that value refers to energies AFTER scissor shift or eqp corrections.
#fermi_level 0.0

# The Fermi level is treated as an absolute value
# or relative to that found from the mean field (default)
#fermi_level_absolute
#fermi_level_relative

# What screening is present? (default = semiconductor)
screening_semiconductor
#screening_metal
#screening_graphene
#  this is for a system with linear DOS at the Fermi level

# RECOMMENDED fast FFTW truncation schemes
# The Coulomb Interaction is cutoff on the edges of
# the Wigner-Seitz Cell in the non-periodic directions
# Periodic directions are a1,a2 for slabs and a3 for wires

#cell_box_truncation
#cell_wire_truncation
#cell_slab_truncation

# Analytic, but non Wigner-Seitz cell, truncation

#spherical_truncation

# For Spherical Truncation, radius in Bohr
#coulomb_truncation_radius   10.00

# Flag for coarse grid of k-points in the BZ.
# Should we unfold using symmetries? either
# no_symmetries_coarse_grid (default) or
# use_symmetries_coarse_grid. If you calculated
# epsmat on a reduced q grid, you should use
# symmetries here!
#use_symmetries_coarse_grid
no_symmetries_coarse_grid

# Related to number of processors.  You can break a large job up into
# many smaller jobs this way, by calculating a fraction of the kernel.
# The default is imin=1 and imax=nprocs
#partial_blocks imin imax

# Write the bare Coulomb potential V(q+G) to file
#write_vcoul

# Communication through MPI or DISK
# comm_mpi is usually much faster and preferable but if you have only
# a few CPUs you might not have enough memory to hold wavefunctions
# comm_disk results in temporary INT_* files being created and it
# may be faster for a small unit cell and a lot of k-points
# The default is comm_mpi
# *** CURRENTLY, comm_disk IS ONLY IMPLEMENTED FOR eps0mat/epsmat ***
comm_mpi
#comm_disk

# Low communication
# The default behavior of the code is to distribute the dielectric matrix
# among the processors. While this minimizes memory usage, it also
# increases the communication. By using the low_comm flag, each processor
# will store the whole dielectric matrix. It is advisable to use this flag
# whenever each PE has enough memory to hold the whole epsmat file.
#low_comm

# Low Memory option
# Calculate matrix elements separately for each k,c,v,k',c',v' pair
#low_memory

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY
# 'unfolded BZ' is from the kpoints in the WFN file
# 'full BZ' is generated from the kgrid parameters in the WFN file
# See comments in Common/checkbz.f90 for more details
# Replace unfolded BZ with full BZ
#fullbz_replace
# Write unfolded BZ and full BZ to files
#fullbz_write

# Rotation of the k-points may bring G-vectors outside of the sphere.
# Use the following keywords to specify whether to die if some of
# the G-vectors fall outside of the sphere. The default is to ignore.
# Set to die in case screened_coulomb_cutoff = bare_coulomb_cutoff.
# Set to ignore in case screened_coulomb_cutoff < bare_coulomb_cutoff.
#die_outside_sphere
#ignore_outside_sphere

# Flag to read k-points from the 'kpoints' file.
# The default is to read k-points from the wfn file.
#read_kpoints
```

BSE/README_absorption

```
-------------------------------------------------------------------
----------  BSE code, Absorption  ---------------------------------
-------------------------------------------------------------------

  Version 1.0   (July, 2011) J. Deslippe, M. Jain, D. A. Strubbe, G. Samsonidze

  Version 0.5   J. Deslippe, D. Prendergast, L. Yang, F. Ribeiro, G. Samsonidze (2008)
  Version 0.2   M. L. Tiago, C. Spataru, S. Ismail-Beigi (2004)

-------------------------------------------------------------------

Description:

This code is the second half of the BSE code.  It interpolates the coarse grid
electron-hole kernel onto the fine grid and then diagonalized the BSE equation.
The output is the electron-hole eigenfunctions and eigenvalues as well the
absorption spectra.

-------------------------------------------------------------------

Required Input:

        absorption.inp  Input parameters.  See detailed explanation below.
        WFN_fi          Wavefunctions in unshifted fine grid
                        (conduction and valence for momentum operator,
                        conduction for velocity operator).
        WFNq_fi         Wavefunctions in shifted fine grid
                        (not needed for momentum operator,
                        valence for velocity operator).
        WFN_co          Wavefunctions on unshifted coarse grid.
                        Must be the same as used for Kernel.
        eps0mat         Must be same as used in Kernel.
        epsmat          Must be same as used in Kernel.
        bsedmat         BSE matrix elements in coarse grid, direct part. This
                        should be generated with Kernel code using same WFN_co.
        bsexmat         BSE exchange matrix elements.  This should be generated
                        with Kernel code using same WFN_co.

Additional Input:

        eqp.dat         A list of quasiparticle energy corrections for the bands in WFN_fi.
                        Used if eqp_corrections is set in absorption.inp.
        eqp_q.dat       A list of quasiparticle energy corrections for the bands in WFNq_fi.
                        Used if eqp_corrections is set in absorption.inp.
        eqp_co.dat      A list of quasiparticle energy corrections for the bands in WFN_co.
                        Used if eqp_co_corrections is set in absorption.inp.

        kpoints         A list of k-points in unshifted fine grid. EXPERIMENTAL.
                        If absent k-points from WFN_fi file are used.
        kpoints_co      A list of k-points in unshifted coarse grid. EXPERIMENTAL.
                        If absent k-points from WFN_co file are used.

Auxiliary Files: (output files from previous runs - used as input to speed up calculation)

        dtmat           Transformation matrices, dcc/dvv use for interpolation
                        between coarse and fine grid.  This file must be consistent
                        with your bsedmat and bsexmat files and corresponding
                        coarse and fine wavefunctions.
        vmtxel          Optical matrix elements (velocity or momentum) between
                        single particle states.
        epsdiag.dat     Diagonal elements of dielectric matrix on the q-grid.
                        Must be consistent with epsmat and eps0mat.
        eigenvalues.dat Contains electron-hole eigenvalues and transition matrix elements.


-----------------------------------------------------------------

        absorption.inp  Please see example absorption.inp in this directory
                        for more complete options.

-----------------------------------------------------------------

Output Files:

        eigenvalues.dat         Has eigenvalues/transition matrix elements of e-h states,
                                eigenvalues in eV, mtxels in atomic units.
        eigenvectors            Has the excitonic wavefunctions in Bloch space:
                                A_svck
        absorption_eh.dat       Dielectric function and density of excitonic states.
                                Four Columns
                                energy (in eV) | epsilon_2 | epsilon_1 | DOS
                                DOS is normalized (\int (DOS) d(omega) = 1)
        absorption_noeh.dat     Non-interacting dielectric function and joint density
                                of states.  Four columns:
                                energy (in eV) | epsilon_2 | epsilon_1 | JDOS
                                JDOS is normalized (\int (JDOS) d(omega) = 1)
        dvmat_norm.dat          The norms of the dvv overlap matrices between the valence
                                band k on the fine grid and the closest k-point on the
coarse grid
```

```
        dcmat_norm.dat          The norms of the dcc overlap matrices between the conduction
                                band k on the fine grid and the closest k-point on the
coarse grid
        eqp.dat, eqp_q.dat      Quasiparticle corrections for WFN_fi and WFNq_fi interpolate
from
                                the coarse grid if eqp_co_corrections is used.
        bandstructure.dat       Same as eqp.dat and eqp_q.dat but in a format suitable for
plotting
                                as a bandstructure.

-----------------------------------------------------------------

Tricks and hints:

1. To optimize distribution of work among PEs, choose the number of
PEs so that Nk*Nc*Nv in the fine grid is a multiple of the
number of PEs. The parallelization is first done over over k-points.

2. The Brillouin zone is built using a Wigner-Seitz construction,
this way the head matrix elements are easily calculated.

3. Check if the transformation matrices have norm close to 1! They
are usually normalized (look at the end of intwfn.f90). The norms are in
the files dvmat_norm.dat and dcmat_norm.dat.

4. Unfolding of irreducible BZ: if you want to skip the unfolding and
use the set of k-points in WFN_fi as a sampling of the whole BZ, specify
the no_symmetries_* options in absorption.inp.

5. The "number_eigenvalues" keyword: using this keyword tells the code
to store only the first nn eigenvalues/eigenvectors. So far, this option
is implemented only with the SCALAPACK diagonalization routines.

6.  Analyzing eigenvectors.  We have a tool called summarize_eigenvectors to read in
and analyze eigenvectors for a group of exciton states.  For specific states
specified, it sums the total contribution from each k-point.
Please see the example input summarize_eigenvectors.inp.

Converters from old versions of file formats to current version are available in version
2.4.
```

BSE/absorption.inp

```
# absorption.inp
# OPTIONS FOR BOTH ABSORPTION AND INTEQP

# Number of occupied bands on fine (interpolated) k-point grid
number_val_bands_fine 3
# Number of occupied bands on coarse (input) k-point grid
number_val_bands_coarse 3

# Number of unoccupied bands on fine (interpolated) k-point grid
number_cond_bands_fine 3
# Number of unoccupied bands on coarse (input) k-point grid
number_cond_bands_coarse 3

# In metallic systems, PARATEC often outputs incorrect occupation
# levels in wavefunctions.  Use this to override these values.
# lowest_occupied_band should be 1 unless you have some very
# exotic situation.
#lowest_occupied_band vmin
#highest_occupied_band vmax

# Sizes of k-point meshes for input and interpolation. The
# number of k-points in fullbz can be found in kernel output
# if symmetries are being used.
coarse_grid_points 8

# Specify the Fermi level (in eV), if you want implicit doping
# Note that value refers to energies AFTER scissor shift or eqp corrections.
# If the Fermi level is moved, then you must use both or neither
# of eqp_corrections and eqp_co_corrections, or no interpolation.
#fermi_level 0.0

# The Fermi level is treated as an absolute value
# or relative to that found from the mean field (default)
#fermi_level_absolute
#fermi_level_relative

# Scissors operator (linear fit of the quasiparticle
# energy corrections) for the bands in WFN_fi and WFNq_fi.
# e_cor = e_in + es + edel * (e_in - e0)
# Defaults below. evs, ev0, ecs, ec0 are in eV
#evs     0.0
#ev0     0.0
#evdel   0.0
#ecs     0.0
#ec0     0.0
#ecdel   0.0
# or
#cvfit   0.0 0.0 0.0 0.0 0.0 0.0

# These flags define whether to use symmetries to unfold
# the Brillouin zone or not in files WFN_fi (unshifted fine
# grid), WFNq_fi (shifted fine grid), WFN_co (coarse grid).
# Warning, the default is always not to unfold!

# If your unshifted fine grid is reduced (most cases),
# you will want to use symmetries here!
#no_symmetries_fine_grid
#use_symmetries_fine_grid

# If your shifted fine grid is reduced (most cases),
# you will want to use symmetries here! Note, that
# if you use symmetries in the unshifted grid you will
# need to generate the shifted grid using the same
# procedure as generating the shifted grid for an
# epsilon calculation. i.e. use gw_shift in paratec
# and a non-zero small q-shift in kgrid.inp
#no_symmetries_shifted_grid
#use_symmetries_shifted_grid

# If you calculated the coarse grid on a reduced q-grid,
# you should use symmetries here!
#no_symmetries_coarse_grid
#use_symmetries_coarse_grid

# Regular grid used to calculate qpt_averages.
# Default is the kgrid in the WFN_fi file.
# It matters only if you want to perform minicell averages.
#regular_grid n1 n2 n3

# Communication through MPI or DISK
# comm_mpi is usually much faster and preferable but if you have only
# a few CPUs you might not have enough memory to hold wavefunctions
# comm_disk results in temporary INT_* files being created and it
# may be faster for a small unit cell and a lot of k-points.
# The default is comm_mpi
# *** Code often crashes in genwf_co with comm_disk when run on ***
# *** a large number of processors because all instances are trying to ***
# *** read coarse grid wavefunctions simultaneously from a single INT file ***
comm_mpi
#comm_disk

# Set this to use eigenvalues in eqp.dat and eqp_q.dat
# If not set, these files will be ignored.
#eqp_corrections

# Set this to use eigenvalues in eqp_co.dat
# These quasiparticle corrections will be interpolated to
# the shifted and unshifted fine grids and written to eqp.dat,
# eqp_q.dat, and bandstructure.dat. If not set, this file will be ignored.
#eqp_co_corrections

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY (ABSORPTION AND INTEQP)
# 'unfolded BZ' is from the kpoints in the WFN file
# 'full BZ' is generated from the kgrid parameters in the WFN file
# See comments in Common/checkbz.f90 for more details
# Replace unfolded BZ with full BZ
#fullbz_replace
# Write unfolded BZ and full BZ to files
#fullbz_write

# Flag to read k-points from the 'kpoints' file.
# The default is to read k-points from the wfn file.
#read_kpoints

# OPTIONS ONLY FOR ABSORPTION BELOW

# Whether to do a diagonalization (with ScaLAPACK or LAPACK) or
# an iterative solution (via Haydock Recursion).
# diagonalization is default

diagonalization
#haydock

# What screening is present? (default = semiconductor)
screening_semiconductor
#screening_metal
#screening_graphene
#  this is for a system with linear DOS at the Fermi level

# Compute only the lowest neig eigenvalues/eigenvectors.
# The default is to calculate all nv*nc*nk of them.
# Only for diagonalization - is not needed for haydock
#number_eigenvalues neig

# If using Haydock specify the number of iterations
#number_iterations 500

# RECOMMENDED fast FFTW truncation schemes
# The Coulomb Interaction is cutoff on the edges of
# the Wigner-Seitz Cell in the non-periodic directions
# Periodic directions are a1,a2 for slabs and a3 for wires

#cell_box_truncation
#cell_wire_truncation
#cell_slab_truncation

# Analytic, but non Wigner-Seitz cell, truncation

#spherical_truncation

# For Spherical Truncation
#coulomb_truncation_radius   10.00
```

```
# Cutoff energy for averaging the Coulomb Interaction
# in the Mini Brillouin Zones around the Gamma-point
# without Truncation or for Cell Wire or Cell Slab Truncation.
# The value is in Rydbergs, the default is 10^{-12}
#cell_average_cutoff 1.0d-12

# How to calculate optical transition probabilities.
# You must select one of the below two options!
use_velocity
#use_momentum
# For Haydock there is also a third option to calculate
# Joint density of states
#use_dos

# This is needed if you selected velocity operator above.
# This should match the shift in WFNq_fi.
q_shift s1 s2 s3

# This is needed if you selected momentum operator above.
# This vector will be normalized in the code.
#polarization p1 p2 p3

# If we have already calculated the optical matrix elements,
# do we read them from file to save time?
#read_vmtxel

# Read 'eps2_moments' generated during previous run (Haydock only)
#read_eps2_moments

# Reduce cost of calculation if we've already computed
# the eigensolutions.  This reads the eigenvalues and
# transition matrix elemenents from the file 'eigenvalues.dat'
#read_eigenvalues

# If we only want the noninteracting spectra (i.e. no electron-
# hole interaction).  Only WFN_fi and WFNq_fi are needed as inputs.
# With the 'read_eigenvalues' flag, both absorption_eh.dat and
# absorption_noeh.dat will be created.
#noeh_only

# If already calculated, read the interpolation matrices dvv
# and dcc from file dtmat.
#read_dtmat

# Numerical broadening width and type for generating absorption spectrum.
# Haydock only does Lorentzian. epsilon_1 is always Lorentzian.
# The width is controlled by energy_resolution.
energy_resolution 0.1
# epsilon_2 with diagonalization is Gaussian by default. You can also choose Lorentzian.
#lorentzian_broadening
gaussian_broadening

# The Gaussian and Voigt options are only available for epsilon_2 with diagonalization.
# Voigt becomes Lorentzian at sigma -> 0 and Gaussian at gamma = 0.
# Note that the parametrization of Voigt function diverges at sigma = 0.
#voigt_broadening
#energy_resolution_sigma 0.1
#energy_resolution_gamma 0.01

# This specifies what kind of epsilon we read.
# Default is to read epsmat and eps0mat, but if
# this flag is present we read file epsdiag.dat instead.
# epsdiag.dat contains only the diagonal part of eps(0)mat
# which is all that is needed by absorption. It is always created
# by an absorption run for use in future runs, to save time reading.
#read_epsdiag
```

```
# Determines output of eigenvectors (eigenvectors)
# eig = 0 : do not write eigenvectors (default)
# eig < 0 : write all eigenvectors
# eig > 0 : write eig eigenvectors
#write_eigenvectors eig

# Flag the type of kernel to calculate:
# spin_triplet: direct kernel only (no exchange).
#  WARNING: triplet transition matrix elements ignore spin overlap,
#  otherwise would be exactly zero!
# spin_singlet (default): direct + exchange. appropriate for spin-polarized too.
# local_fields: local-fields + RPA (exchange only)
# Note that triplet kernel only applies to a spin-unpolarized system;
# for a spin-polarized system, the solutions naturally include both singlets and triplets
# (or other multiplicities for a magnetic system).
#spin_triplet
#spin_singlet
#local_fields

# Write the bare Coulomb potential V(q+G) to file
#write_vcoul

# If your two grids are the same, use this flag to skip interpolation and
# reduce the time/memory you need. DO NOT use this if you have a different
# fine grid from the coarse grid, as for velocity operator.
# Then you should use eqp_corrections with eqp.dat (and eqp_q.dat).
#skip_interpolation

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY (ABSORPTION ONLY)

# The requested number of bands cannot break degenerate subspace
# Use the following keyword to suppress this check
# Note that you must still provide one more band in
# wavefunction file in order to assess degeneracy
#degeneracy_check_override

# Average the head of W in addition to the head of V using a model
# for the inverse dielectric matrix. This is done for insulators for
# all truncation schemes. The W average is limited only to the first
# minibz even if cell_average_cutoff != 0. Currently it is always done
# regardless whether average_w is included in the input file or not.
#average_w

# Multiply kernel by arbitrary factor. Default is 1.0 of course.
kernel_scaling 1.0
```

BSE/inteqp.inp

```
# inteqp.inp
# inteqp.inp uses a subset of the parameters from absorption.inp
# Please see absorption.inp for examples.
```

BSE/README_inteqp

```
-------------------------------------------------------------------
----------  BSE code, IntEqp  -------------------------------------
-------------------------------------------------------------------

  Version 1.0   (July, 2011) J. Deslippe, D. A. Strubbe
```

```
------------------------------------------------------------------

Description:

This code takes the eqp_co.dat file from Sigma and
interpolates it from the coarse to the fine grid using wavefunction
projections (to resolve band crossings) and linear interpolation
in the Brillouin zone.

------------------------------------------------------------------

Required Input:

        inteqp.inp      Input parameters. See example absorption.inp in this directory
                        for complete options (noting which sections apply to inteqp).
        WFN_fi          Wavefunctions in unshifted fine grid.
        WFN_co          Wavefunctions on coarse grid.
        eqp_co.dat      A list of quasiparticle energy corrections
                        for the bands in WFN_co.

Optional Input:

        WFNq_fi         Wavefunctions in shifted fine grid
                        (used for valence bands if use_velocity is selected)

Auxiliary Files: (output files from previous runs - used as input to speed up calculation)

        dtmat           Transformation matrices, dcc/dvv use for interpolation
                        between coarse and fine grid.  This file must be consistent
                        with your bsedmat and bsexmat files and corresponding
                        coarse and fine wavefunctions.

------------------------------------------------------------------

Output Files:

        bandstructure.dat       The GW bandstructure on the fine grid.
        eqp.dat                 Quasiparticle energy corrections for the bands in WFN_fi.
        eqp_q.dat               Quasiparticle energy corrections for the bands in WFNq_fi
(if use_velocity).
        dvmat_norm.dat          The norms of the dvv overlap matrices between the valence
                                band k on the fine grid and the closest k-point on the
coarse grid
        dcmat_norm.dat          The norms of the dcc overlap matrices between the conduction
                                band k on the fine grid and the closest k-point on the
coarse grid
```

BSE/summarize_eigenvectors.inp

```
20              ! Number of eigenvectors in file.  Set to zero to use default ns*nv*nc*nk
1.5 2.1         ! Emin Emax (eV). Energy window to print information about all states in
3               ! Number of specific states to print A(k). Output files will be exciton_01
... exciton_99
1.56783332      ! Energy values of the states to print A(k)
1.67345203
1.96328918
```

PlotXct

PlotXct/README

```
-----------------------------------------------------------------
----------  PlotXct code  ---------------------------------------
-----------------------------------------------------------------

  Version 1.0   (July, 2011)

  Version 0.5   F. Ribeiro (2008)
  Version 0.2   M. L. Tiago (2002-2007)

-----------------------------------------------------------------

Description:

PlotXct plots the exciton wavefunction in real space based on
output of BSE/absorption (with full diagonalization).

-----------------------------------------------------------------

Tricks and hints:

1. Copy/edit plotxct.inp.

   You'll need:
     . index of the exciton state you want to plot
     . q-shift
     . hole position in lattice coordinates
     . unit-cell lattice vectors
     . supercell dimensions (or k-grid)


2. Run plotxct.x

   The ASCII file 'xct.[state].a3Dr' will be created.

   'a3Dr' means that the file is in ASCII (as opposed to binary: 'b3Dr') and
   contains 3D data in 'r' real space.

   The header of this file contains information on
     . state index
     . state energy in eV
     . hole position in atomic units (a.u.)
     . supercell lattice vectors in a.u.
     . number of discretization points
   followed by three columns of data corresponding to the complex values of the
   electronic part of the excitonic wavefunction, and its magnitude squared.

   The values are written with very low precision due to file-size concerns.


3. Convert to a format readable by the plotting utility of your choice.

   Use the 'volume.py' utility:

   Usage: volume.py imfn imff ivfn ivff ovfn ovff phas cplx [hole]

   imfn = input matter file name
   imff = input matter file format
          (mat|paratec|vasp|espresso|siesta|tbpw|xyz|xsf)
   ivfn = input volumetric file name
   ivff = input volumetric file format (a3dr)
   ovfn = output volumetric file name
   ovff = output volumetric file format (cube|xsf)
   phas = remove wavefunction phase (true|false)
   cplx = complex wavefunction (re|im|abs|abs2)
   hole = plot hole (true|false)

   You can specify whether you want a .xsf (XCrysDen) or .cube (Gaussian Cube)
   file format. You can remove or keep the arbitrary phase of the wavefunction
   by setting parameter "phas" to true or false. You can plot the re, im, abs,
   or abs2 part of the wavefunction.

   You must specify a path to the DFT input file (example: "../01-scf/input")
   in a paratec or espresso directory so that the script can obtain the atomic
   positions from the file.

   You can choose whether to display or not the position of the hole by setting
   parameter "hole" to true or false.

   ****************
   Example:

     % ~/BerkeleyGW/Visual/volume.py ../01-scf/input paratec xct.001.a3Dr a3dr \
                    xct.001.abs2.xsf xsf false abs2 false
     % xcrysden --xsf xct.001.abs2.xsf

   ****************
```

PlotXct/plotxct.inp

```
#
# Index of state to be plotted, as it appears in eigenvectors
#
plot_state 20

#
#  q-shift used in the calculation of valence bands (WFNq_fi file)
#
q_shift   0.00000  0.00000  0.00000

#
# Size of supercell
#
supercell_size 1 1 60

#
# coordinates of hole, in units of lattice vectors
# (usually, hole in the center of the supercell)
#
hole_position    0.8025    0.3487    30.00
# corresponds to  21.121073141660   22.792380000000   0.005915241051  in real space + 50*
(0,0,az)
# hole 2 a.u. on x dir from an atom at 5.96137532  0.0 from center of tube
# original coord:     19.121073141660   22.792380000000    0.005915241051
#
# lattice vectors in atomic units, as used in paratec
# NOTE: no scaling is assumed, actual cell volume must match with
# determinant of the matrix below.
#
begin lattice_vectors
    26.318373457000   15.194920000000    0.000000000000
     0.000000000000   30.389840000000    0.000000000000
     0.000000000000    0.000000000000    7.964209699861
end

# input option 'restrict_kpoints' reduces the sum over k-points
# above to a sum over the specified number that give most of the contribution
# to the norm of eigenvectors. This is handy if there are many k-points
# but only a few of them give sizable contribution.
#restrict_kpoints 5

# EXPERIMENTAL FEATURES FOR TESTING PURPOSES ONLY
# 'unfolded BZ' is from the kpoints in the WFN file
# 'full BZ' is generated from the kgrid parameters in the WFN file
# See comments in Common/checkbz.f90 for more details
# Replace unfolded BZ with full BZ
#fullbz_replace
# Write unfolded BZ and full BZ to files
#fullbz_write
```

Visual

Visual/README

```
-------------------------------------------------------------------
----------  Visualization Tools  ----------------------------------
-------------------------------------------------------------------

  Version 2.4   (May, 2009)

  Version 2.3   G. Samsonidze (October, 2008)

-------------------------------------------------------------------

Description:

This directory contains a set of visualization tools designed to
simplify your work with the DFT codes and the BerkeleyGW package.
Here you will find the following scripts and codes:

1. Surface is a C++ code for generating an isosurface of a volumetric
scalar field (such as the wave function, charge density, or local
potential). The scalar field is read from Gaussian Cube or XCrySDen
XSF file, the surface triangulation is performed using marching cubes
or marching tetrahedra algorithm, and the isosurface is written in
the POV-Ray scripting language. The final image is rendered using
the ray-tracing program POV-Ray.

Running the code requires a fairly complicated input parameter file,
described with an example in surface.inp.

2. Matter is a python library for manipulating atomic structures with
periodic boundary conditions. It can translate and rotate the atomic
coordinates, generate supercells, 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 viewer.
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.

Examples:

1. Benzene Charge Density

This example demonstrates how to plot isosurfaces.
Go to directory ./benzene and run ESPRESSO:

```
$ sh link
Submit script. Suggested ncpu = 36, walltime = 0:30:00
```

This will produce the Gaussian Cube file rho.cube containing the charge density of the benzene molecule. For your convenience, the compressed file rho.cube.tgz is included in the ./benzene directory.

Let us generate an isosurface that contains 90% of the charge density using the marching cubes algorithm with the smooth triangles and render it in POV-Ray:

```
$ ../surface.x rho_uc.inp
$ povray -A0.3 +W640 +H480 +I rho_uc.pov
```

Examine the rho_uc.gif file to find that the pieces of the benzene molecule are placed in the corners of the unit cell. To assemble the pieces together we define a supercell in the rho_sc.inp file. This is done by setting the parameter sc to T and by placing the supercell origin (sco) at position (-0.5 -0.5 -0.5) in crystal coordinates (scu which stands for supercell units is set to latvec). The lattice vectors of the supercell are not changed (scv or supercell vectors in the parameter file). Let us render a new image:

```
$ ../surface.x rho_sc.inp
$ povray -A0.3 +W640 +H480 +I rho_sc.pov
```

Now the charge density comes up nice and centered in the middle of the supercell.

2. Nanotube Exciton Wavefunction

In this example we will plot an isosurface of the exciton wavefunction. Go to directory ./swcnt_8-0 and run PlotXct:

```
$ sh link
Submit script. Suggested ncpu = 32, walltime = 0:30:00
```

This will produce the a3Dr file xct.020.a3Dr containing a wavefunction of the 20th exciton in the (8,0) nanotube.
Convert the a3Dr file to Gaussian Cube:

```
$ ../volume.py ../../examples/DFT/swcnt_8-0/ESPRESSO/1-scf/in espresso xct.020.a3Dr a3dr
xct020.cube cube false abs2 true
```

This will produce the Gaussian Cube file xct020.cube that
contains the squared absolute value of the wavefunction
(cplx = abs2) without the sign (phas = false) and the
position of the hole (hole = true). Now generate an
isosurface that contains 90% of the charge denisty
and render it in POV-Ray:

```
$ ../surface.x xct020.inp
$ povray -A0.3 +W2560 +H1920 +I xct020.pov
```

The resulting image xct020.gif is included in the ./swcnt_8-0
directory.

3. Rock-Salt Lattice

Here you will learn how to make large supercells of bulk crystals.
Go to directory ./rocksalt where you will find the nacl.mat file.
Render it in POV-Ray:

```
$ ../convert.py nacl.mat mat nacl.pov povray
$ povray -A0.3 +W640 +H480 +I nacl.pov
```

The unit cell contains only two atoms. Let us make a supercell that
contains a fractional number of unit cells. The supercell is described
by the sc1 file which defines the origin of the supercell, the lattice
vectors of the supercell, and the units in which these quantities
are given. These are equivalent to sco, scv, and scu in the surface
parameter file in the first example. The last line in the sc1 file
corresponds to sct (supercell translation) in the surface parameter
file. We will get back to it later on. Let us render the supercell:

```
$ ../convert.py nacl.mat mat nacl_sc1.pov povray supercell sc1
$ povray -A0.3 +W640 +H480 +I nacl_sc1.pov
```

Note the broken bonds on the faces of the supercell.
That is because the supercell contains a fractional number
of the unit cells, so the translational symmetry is broken
which results in the Na-Na and Cl-Cl bonds. We disable the
translational symmetry of the supercell by setting sct to
false (the last line in the sc2 file). This enforces the
translational symmetry of the underlying unit cell structure.
Render the new supercell in POV-Ray:

```
$ ../convert.py nacl.mat mat nacl_sc2.pov povray supercell sc2
$ povray -A0.3 +W640 +H480 +I nacl_sc2.pov
```

Now the bonds on the faces of the supercell come out right.

4. Organic Molecule Synthesis

In this example you will assemble the bipolar molecule, bithiophene
naphthalene diimide, from the donor and acceptor units, bithiophene
and naphthalene diimide. Go to directory ./btnd and open files
nd.xyz and bt.xyz in any molecular viewer, for example Jmol.
Identify the atoms you want to link together, these are the two
carbon atoms, # 15 in nd.xyz and # 8 in bt.xyz. You need to remove
the two hydrogen atoms, # 21 in nd.xyz and # 14 in bt.xyz, and place
# 15 and # 8 at the distance of 1.49 Angstrom from each other along
the 15-21-14-8 line. This is done by invoking the following command:

```
$ ../link.py nd.xyz xyz bt.xyz xyz btnd.xyz xyz 15 21 8 14 0.0 degree 1.49 angstrom
```

You can also rotate bt.xyz around the 15-21-14-8 axis by an arbitrary
angle, although the rotation angle is set to zero in the above command.

5. Estimate the number of unoccupied states for Epsilon and Sigma

Go to directory ../examples/DFT/benzene/0-gsphere/ and run the following:

```
$ gsphere.py g_eps.in g_eps.out
```

Examine input file g_eps.in. It contains the lattice vectors in bohr, the cutoff energy in Ry (epsilon_cutoff from ../5-gw/epsilon.inp or screened_coulomb_cutoff from ../5-gw/sigma.inp), '0 0 0' for the FFT grid to be determined automatically, and 'true' to sort the G-vectors by their kinetic energies. Examine output file g_eps.out. Look for the number of G-vectors, ng. You will find 'ng = 2637' meaning that you will need at least that many unoccupied states. Of course the number of unoccupied states is a convergence parameter, but gsphere.py gives you an idea in which range of values should you check the convergence.

Now run the following:

```
$ gsphere.py g_rho.in g_rho.out
```

Input file g_rho.in is similar to g_eps.in but it has a different cutoff energy which is 4 times ecutwfc from ../ESPRESSO/1-scf/in. This is the charge density cutoff or ecutrho in espresso documentation. Note that the last line in g_rho.in is set to 'false'. This is because using built-in Python sort function on that many G-vectors (up to ecutrho) will freeze your python interpreter. Examine output file g_rho.out. You should see the following:

```
 grid = ( 128 72 140 ) -- minimal
        ( 128 72 140 ) -- factors 2, 3, 5, 7, 1*11, 1*13
        ( 128 72 144 ) -- factors 2, 3, 5
```

The third line is the size of FFT grid in espresso calculation. (Well, not necessarily, because espresso algorithm starting with version 5 is smarter than what is implemented in gsphere.py.) The z-component is equal to 144 meaning that for an optimal load-balancing you should run espresso on a number of cores which is a divisor of 144 (that is, 144, 72, 36, 18, etc. cores).

Notes:

Matter library may lack support for some advanced features of PARATEC, VASP, ESPRESSO and SIESTA formats. For example, LatticeParameters and ZMatrix are not implemented for SIESTA. This can be added to functions paratec_read, paratec_write, vasp_read, vasp_write, espresso_read, espresso_write, siesta_read and siesta_write in file matter.py.

Also note that all the additional information not related to the crystal structure (k-points, energy cutoffs, etc.) will be lost and replaced with the default parameters when converting between PARATEC, VASP, ESPRESSO and SIESTA formats.

Visual/surface.inp

parameter file	example (HOMO of benzene)
header text	header C6H6_band_15
inputfilename file	inputfilename C6H6.b_15.cube
inputfileformat cube|xsf	inputfileformat cube
outputfilename file	outputfilename C6H6.b_15.pov
outputfileformat povray	outputfileformat povray
isovalue (0.0,1.0)	isovalue 0.9

```
   sign positive|negative|both    |   sign both
   power 0|1|2                    |   power 1
   algorithm cube|tetrahedron     |   algorithm cube
   smooth T|F                     |   smooth T
   box T|F                        |   box F
   basis T|F                      |   basis F
   uc T|F                         |   uc F
   uco                            |   uco
   ucox ucoy ucoz                 |   0.0 0.0 0.0
   ucv                            |   ucv
   ucv1x ucv1y ucv1z              |   1.0 0.0 0.0
   ucv2x ucv2y ucv2z              |   0.0 1.0 0.0
   ucv3x ucv3y ucv3z              |   0.0 0.0 1.0
   ucu bohr|angstrom|latvec       |   ucu latvec
   sc T|F                         |   sc T
   sco                            |   sco
   scox scoy scoz                 |   -0.5 -0.5 -0.5
   scv                            |   scv
   scv1x scv1y scv1z              |   1.0 0.0 0.0
   scv2x scv2y scv2z              |   0.0 1.0 0.0
   scv3x scv3y scv3z              |   0.0 0.0 1.0
   scu bohr|angstrom|latvec       |   scu latvec
   sct T|F                        |   sct T

-------------------------------------------------------------------------------

   the isosurface is defined by isovalue * max |psi| for power = 0 and by
   integral_v |psi|^power = isovalue * integral |psi|^power for power > 0
   where psi is the scalar field and v is the volume inside the isosurface

   note that espresso already outputs |psi|^2, setting power = 2 yields
   |psi|^4 in this case, to produce |psi|^2 isosurface set power = 1

   algorithm stands for marching cubes and marching tetrahedra
   smooth defines whether to average normals for smooth rendering

   box specifies whether to generate supercell box in povray script
   basis specifies whether to generate coordinate axes in povray script

   sfo = scalar field origin    sfv = scalar field vectors
   uco = unit cell origin    ucv = unit cell vectors    ucu = unit cell units
   sco = supercell origin    scv = supercell vectors    scu = supercell units
   sct = supercell translational symmetry

   sfo/sfv are read from volumetric file

   if uc = T then uco/ucv are read from parameter file
           else uco/ucv are set to sfo/sfv
   if sc = T then sco/scv are read from parameter file
           else sco/scv are set to uco/ucv
   if uc = T and ucu = latvec then uco/ucv are scaled by sfv
   if sc = T and scu = latvec then sco/scv are scaled by ucv

   if sct = T then scv is used for translational symmetry
            else ucv is used for translational symmetry

   in most cases, set uc = F (uc = T is only needed if volumetric data
   do not span the whole unit cell and the supercell is being used, so
   the correct unit cell must be defined)

   use sc = T to construct the supercell spanning several unit cells or
   to assemble the unit cell around the origin of the coordinate system
   as in benzene example above (sco = (-0.5 -0.5 -0.5) in latvec units)

   the supercell may span a fractional number of unit cells, in this case
   set sct = F to produce the correct bonds at the faces of the supercell
```

Visual/gsphere.inp

```
      25.917768986   0.000000000   0.000000000
       0.000000000  14.496126116   0.000000000
       0.000000000   0.000000000  28.284379996
240.0
0 0 0
false

# example input file for gsphere.py, for benzene

# a1x a1y a1z [= lattice vectors (bohr)]
# a2x a2y a2z
# a3x a3y a3z
# energy cutoff (Ry)
# FFT_grid_x FFT_grid_y FFT_grid_z (set to zero to determine automatically)
# whether to sort G-vectors by kinetic energy (true) or leave unsorted (false)
```

Developers

doxygen/README

```
-----------------------------------------------------------------
----------  BerkeleyGW, Doxygen Documentation -------------------
-----------------------------------------------------------------

Description
-----------

 This directory automatically generates documentation for BerkeleyGW
using Doxygen. This is aimed towards developers, and end-users might
as well just ignore this directory.

 To create the documentation, simply type `make`. You`ll need doxygen and
graphviz installed. Then open doxygen/html/index.html in a web browser.


Obtaining Doxygen and Graphviz
------------------------------

 Doxygen and graphviz are available at:

 http://www.doxygen.org/
 http://www.graphviz.org/

 Alternatively, if you have a Debian-based distribution, such as
Ubuntu, you can ask your administrator to install the following packages:

 apt-get install doxygen graphviz

or on MacOS with MacPorts (www.macports.org):

 port install doxygen graphviz
```

Common/README

```
-----------------------------------------------------------------
```

```
----------  BerkeleyGW, Common  ---------------------------------
-----------------------------------------------------------------

  Version 2.2   (July, 2008)
  J. Deslippe, G. Samsonidze, L. Yang, F. Ribeiro

-----------------------------------------------------------------

Description:

This directory contains common files shared by different components of
the BerkeleyGW package. The most important files are described below:

1. typedefs.f90

This file contains the derived data types that are used throughout
the code.

2. nrtype.f90

This file contains some general constants and the definitions of
single- and double-precision real and complex types following the
convention of the Numerical Recipes book. The definitions of DP
and DPC are used throughout the code.

It also contains the fundamental physical constants. The units used
throughout the code are Rydberg atomic units with a few exceptions,
such as eV in Sigma and Hartree atomic units in EPM. The difference
between Hartree and Rydberg atomic units is sketched below.

Hartree atomic units:
hbar=1  e=1        m=1    a0=hbar^2/(m*e^2)=1  E=hbar^2/(2*m*a0^2)=0.5

Rydberg atomic units:
hbar=1  e=sqrt(2)  m=0.5  a0=hbar^2/(m*e^2)=1  E=hbar^2/(2*m*a0^2)=1

3. f_defs.h

This file contains preprocessor definitions for the built-in Fortran
functions. This helps to avoid accidental conversion from double to
single precision.
```

MeanField/spglib-1.0.9/README

```
* Imported spglib-1.0.9 from http://spglib.sourceforge.net. This is a BSD-licensed C library
for symmetries by Atsushi Togo. --DAS
* Added src/spglib_f.c which is not actually part of spglib 1.0.9 although it appears to
have been forgotten by accident since it is mentioned in the documentation and was present
in earlier version. I obtained it by e-mailing the author. --DAS
* Removed test directory which we do not need and contains a large number of files. --DAS
* Removed examples, python, etc., blank files, and automake/libtool-related files. --DAS
```

---

test/scalapack/README

```
Routines extracted from BerkeleyGW to test the proper
functioning of the ScaLAPACK library. By Georgy Samsonidze.
You can use the same arch.mk as for BerkeleyGW.
In the file "matrix.inp" enter the desired matrix dimension
and the scaling factor, the matrix will be set to identity
times scaling. Run the test via the appropriate script
in this directory.
```

~~~
                                                          _u*
                                                         {S@Q'
                                                        ,AbQ@U'``
                                                        xhS8q@o
                                                       :DYPVU8h.
                                                       "O&u$ZGh*``
                                                       =T?l%iza*'`
                                                       'ACloTF=``
                                                  ``````'/Dp6L%'.``
                                            ```.'.'``.'!^;";!!~'````
                                         `..!:::~-'.'...^^~^-,.''``'`
                   `                  ``.!:}<\:~'.-~'.'.''.'''```````.
                     !~   `   ``   ``''.-_;:^'``.''`..!~^__}?lcIvJ&,=`
     `.````````` ,uL)@##@6H#d` `.''.'^,_:_~'``'^:!.'-^,::{)=3$V8O$L9Hl
    '?~'....``,/M@F@a.`zG<;M@@@u^-^,:(<_!.~}+$s6}~_:_:;}>=ij$$313j40`
    %x^,.',.{@HH@_ ` ``    .=;QM7>:;?*}^-<7nE@Hmi="*??*="="c26hkdLli`
    Oi:___^!OMT}~--'..''~-!'.'*B@NP;*:!->1GXBH#HDTY$$uvc,1cLOL$x%/_)`
   `G,\*?{J#@b7"{!'.',..'.,!!_~'_Mq'',!;,TPP@O`.<Fq89aVAPwOorcx31"{{.
   ^l "%iI#MqGhRDLuvY@ESZ6OYOewuOM#K?'~>3O&Jg/    lR@@@WK@Rnl"l,,l>:'
      "17ONM#@HH@WKqEMNH@EE@@@WRqKMMo!{)i$=%q:      `?$9#N@qde+;_--..~
     `,Cjj$yZTS8J%=*=C%=""%1Ij"><?t=\*=l$5zFq`         -V@8Tv*,-^{?__`
     .rjJ$OZA9m#EGZuzco$uOYwwV4LuLu4fDefRRRS:            }OaOJYuvYR@6`
     *7j$6sAaQ@NMHkQERRQRqQR@HNH@QbVwOJ$V@8`               !q#@H@RP*`
   '!^(o6yUd@NM@w{`}c58mEWgQba$l70v7"*?iOR0.               '766\!`
  ':'_i$fR@H#A{`                 >6,ii%+LUs!
  '",$em@@@@R`                    +$%%l=3fL*
  ->/=T$7SQXUl`                   `2t=<(=ZT?
  .',{L?>hkhAAv!                   }T1+l,PY*
 `^'.-"cvnDffnpu                    rwvc=n9C/`
`,,^,,,?vaQqk0w&:                   .r?;~;5Tt"_
!:-,-;"IT6qbfys0G`                  `*:~.'~1Ii"~
}$,ruTj"?*eq8FeLr1'                 `"_-'.-!1Gi(!`
`=GV%}:/tZPRba4Zci'                  cl\/<)??TY1?(.
  'I7+l"?/?+````                     :ay5Tr/''_L&(--`
   ^z$1=*?=?`                         ._>2O">>(*r3i%:
     .!:_.`                               {C=\,''%{'
                                          ;=<)<\?<`
                                           .-:~,'`
~~~