Introduction to *GW*/BSE in GPAW: Monolayer MoS_2 as an example

Min-Ye Zhang

College of Chemistry and Molecular Engineering
Peking University

stevezhang@pku.edu.cn

@dpmodeling-abacus, August 6, 2021

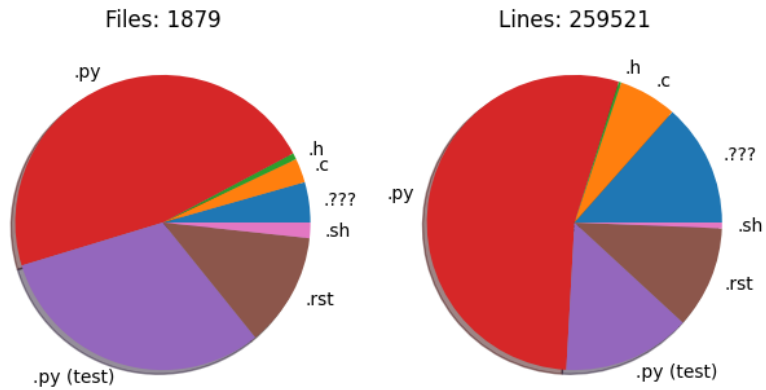
Outline

1. Introduction
   - DFT calculations in GPAW
   - Band-gap problem
2. *GW*/BSE in GPAW
   - A glance at theory
   - Workflow in GPAW
3. Example: Gap engineering of ml-MoS_2
   - Background
   - Model
   - Calculations

Outline

What is GPAW?

- Open-source DFT code with projector augmented-wave (not Global Pandemic App Watch)
- Developed by Thygesen group in Technical University of Denmark, v1.0.0 released on 2016-03-17
- Basis: plane-wave, LCAO with NAO and real-space grids
- Written in Python and C, based on ASE, LIBXC, NumPy and SciPy
- Available by pip, conda, Linux repos, Docker image ...



https://wiki.fysik.dtu.dk/gpaw/
J. Enkovaara et al., J. Phys.: Condens. Matter **22**, 253202 (2010)
Line counts from talk by Jens Jørgen Mortensen in GPAW 2021

Quick example of GPAW with plane-wave

Ground state of silicon with PBE: `Si_gs.py`

```python
from ase.build import bulk
from gpaw import GPAW, FermiDirac
from gpaw.wavefunctions.pw import PW

atoms = bulk('Si', 'diamond', a=5.43)

c = GPAW(mode=PW(400), xc='PBE', eigensolver='dav',
         kpts={'size': (8, 8, 8), 'gamma': True},
         occupations=FermiDirac(0.01), txt='Si_gs.txt')
atoms.calc = c

atoms.get_potential_energy()
c.write ('Si_gs.gpw')
```

Running serially and parallelly

Serial call

```
$ time python Si_gs.py
real  0m9.746
...
```

Parallel

```
$ time mpirun -np 4 gpaw python Si_gs.py
real  0m4.425s
...
```

- automatically handle parallel I/O
- use parallelization over **k**/spin/band
- support MPI/OpenMP and ScaLAPACK

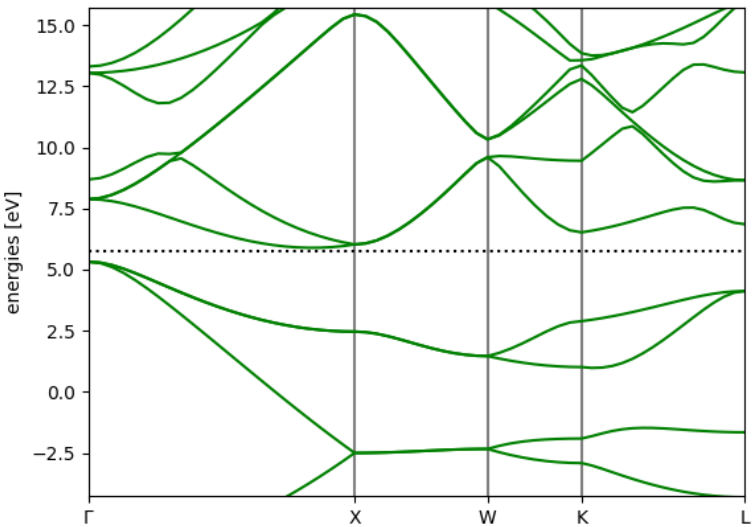
Test on 8-core i7-4790@3.60GHz, GPAW version 21.1.0

Band structure

```
from gpaw import GPAW

c = GPAW('Si_gs.gpw', nbands=16,
         fixdensity=True,
         symmetry='off',
         kpts={'path': 'GXWKL',
               'npoints': 60},
         convergence={'bands': 8})
c.get_potential_energy()
c.write ('Si_bs.gpw')

bs = c.band_structure()
bs.plot(emax=16.0, emin=-4.0,
        filename="Si_bs.png")
```



Si_bs.png

Compute band structure properties manually

Retrieve eigenvalues on the irreducible **k** points

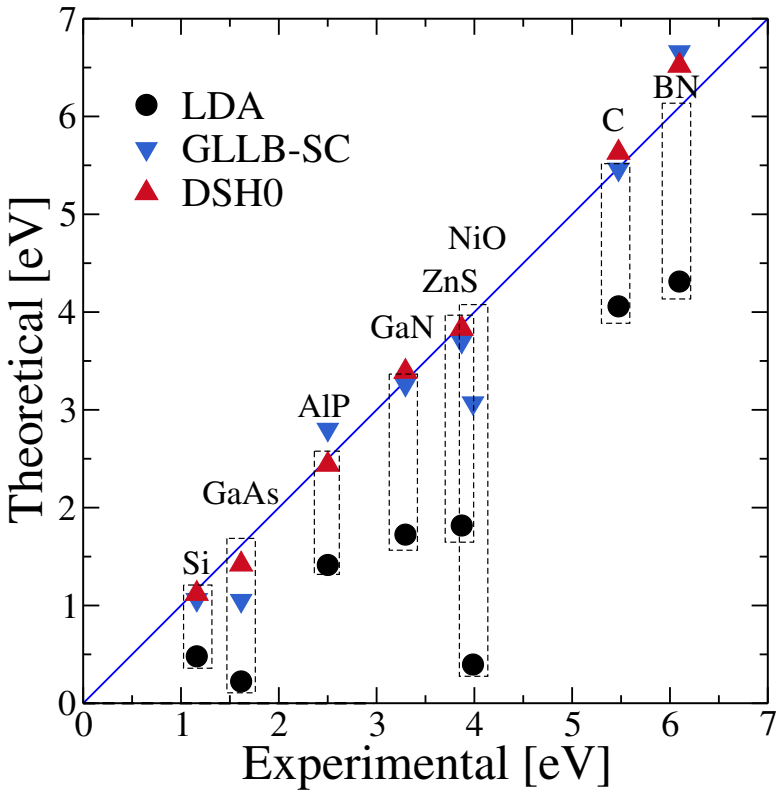
```
c = GPAW('Si_gs.gpw', txt=None)

eigens = np.array([[c.get_eigenvalues(ik, isp)
                    for ik, _ in enumerate(c.get_ibz_k_points())]
                   for isp in range(c.get_number_of_spins())])

ivb = int(np.rint(c.get_number_of_electrons()))//2 - 1

vbs = eigens[0, :, ivb]
cbs = eigens[0, :, ivb+1]
indir_gap = min(cbs) - max(vbs)
dir_gap = min(cbs - vbs)
```

Band-gap problem in LDA/GGA



Underestimation stems from the so-called derivative discontinuity

$$E_{\mathrm{g}}^{\mathrm{fund}} = \mathrm{IP} - \mathrm{EA}$$
$$E_{\mathrm{g}}^{\mathrm{KS}} = \epsilon_{\mathrm{CBM}}^{\mathrm{KS}} - \epsilon_{\mathrm{VBM}}^{\mathrm{KS}}$$
$$E_{\mathrm{g}}^{\mathrm{fund}} = E_{\mathrm{g}}^{\mathrm{KS}} + \Delta_{\mathrm{xc}}$$

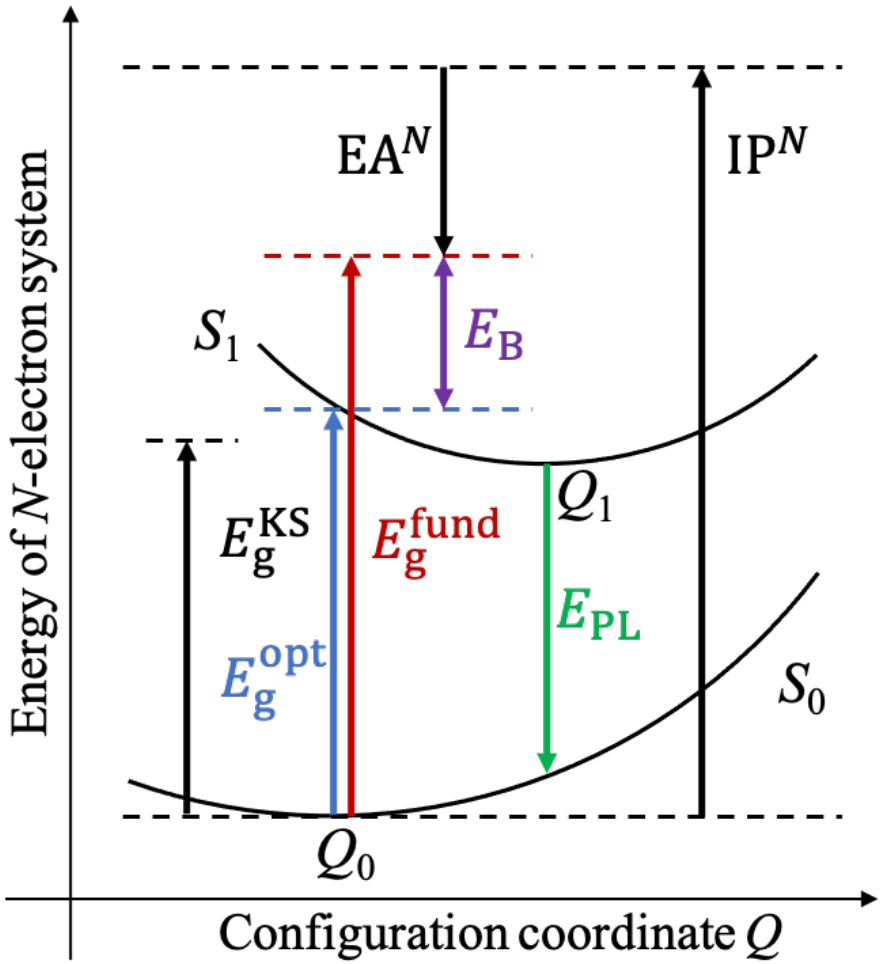
$\Delta_{\mathrm{xc}} \neq 0$ for exact KS potential

Solutions within DFT framework

- meta-GGA, e.g. mBJ, GLLB-SC
- Hybrid functional, e.g. HSE06, doubly screened hybrid (DSH)

J. P. Perdew et al., Phys. Rev. Lett. **49**, 1691–1694 (1982)
F. Tran, S. Ehsan, and P. Blaha, Phys. Rev. Materials **2**, 023802 (2018)
Z.-H. Cui et al., J. Phys. Chem. Lett. **9**, 2338–2345 (2018)

Definition of band gaps

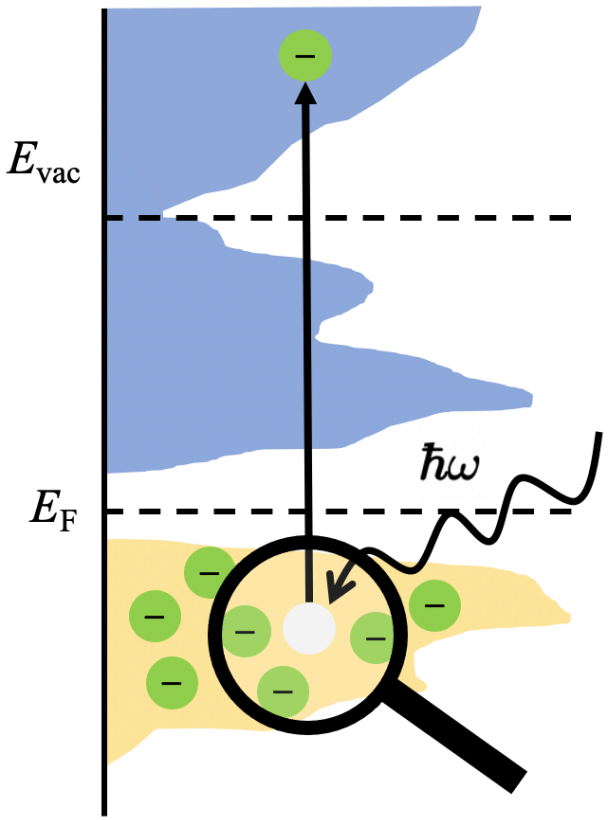


Property	Measurement
IP / EA, $E_\mathrm{g}^\mathrm{fund}$	PES / IPS
$E_\mathrm{g}^\mathrm{opt}$	DRS, ellipsometry
E_PL	photoluminescence

Picture adapted from X. Blase et al., J. Phys. Chem. Lett. **11**, 7371–7382 (2020)

Outline

Green's function in many-body perturbation theory



Time-ordered Green's function ($1 = \mathbf{r}_1 t_1$)

$$\mathrm{i}G\left(1,2\right) = \langle\Psi_N|\mathrm{T}\left\{\hat{\psi}_{\mathrm{H}}\left(1\right)\hat{\psi}^{\dagger}_{\mathrm{H}}\left(2\right)\right\}|\Psi_N\rangle$$

G can be formally solved in frequency domain

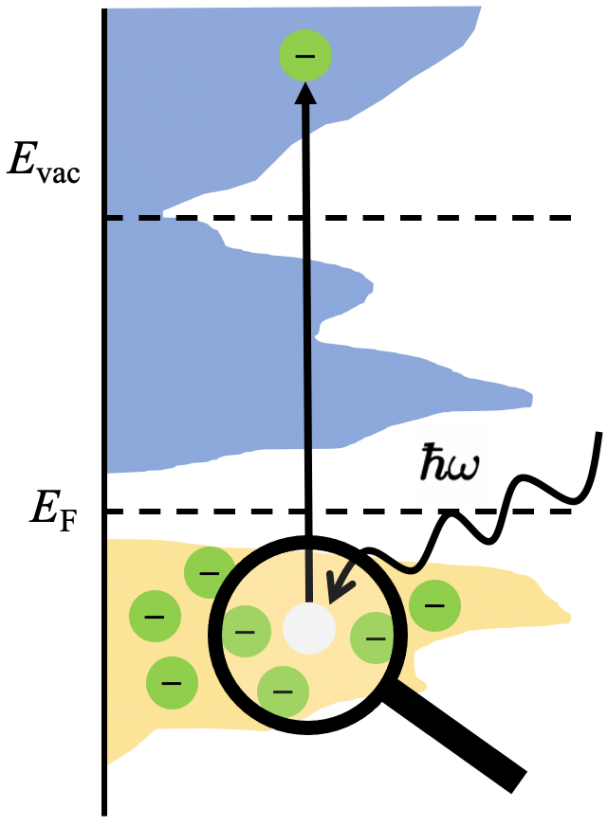
$$G(\mathbf{r},\mathbf{r}',\omega) = \sum_n \frac{\varphi_n(\mathbf{r},\omega)\varphi_n^*(\mathbf{r}',\omega)}{\omega - \epsilon_n(\omega)}$$

$$\left[h_0(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r})\right]\varphi_n(\mathbf{r},\omega) + \int \mathrm{d}\mathbf{r}'\Sigma(\mathbf{r},\mathbf{r}',\omega)\varphi_n(\mathbf{r}',\omega) = \epsilon_n(\omega)\varphi_n(\mathbf{r},\omega)$$

Σ: self-energy

F. Aryasetiawan and O. Gunnarsson, Rep. Prog. Phys. **61**, 237 (1998)

Green's function in many-body perturbation theory



Time-ordered Green's function ($1 = \mathbf{r}_1 t_1$)

$$\mathrm{i}G(1,2) = \langle \Psi_N | \mathrm{T}\left\{ \hat{\psi}_{\mathrm{H}}(1)\, \hat{\psi}^{\dagger}_{\mathrm{H}}(2) \right\} | \Psi_N \rangle$$

G can be formally solved in frequency domain

$$G(\mathbf{r}, \mathbf{r}', \omega) = \sum_n \frac{\varphi_n(\mathbf{r}, \omega)\varphi_n^*(\mathbf{r}', \omega)}{\omega - \epsilon_n(\omega)}$$

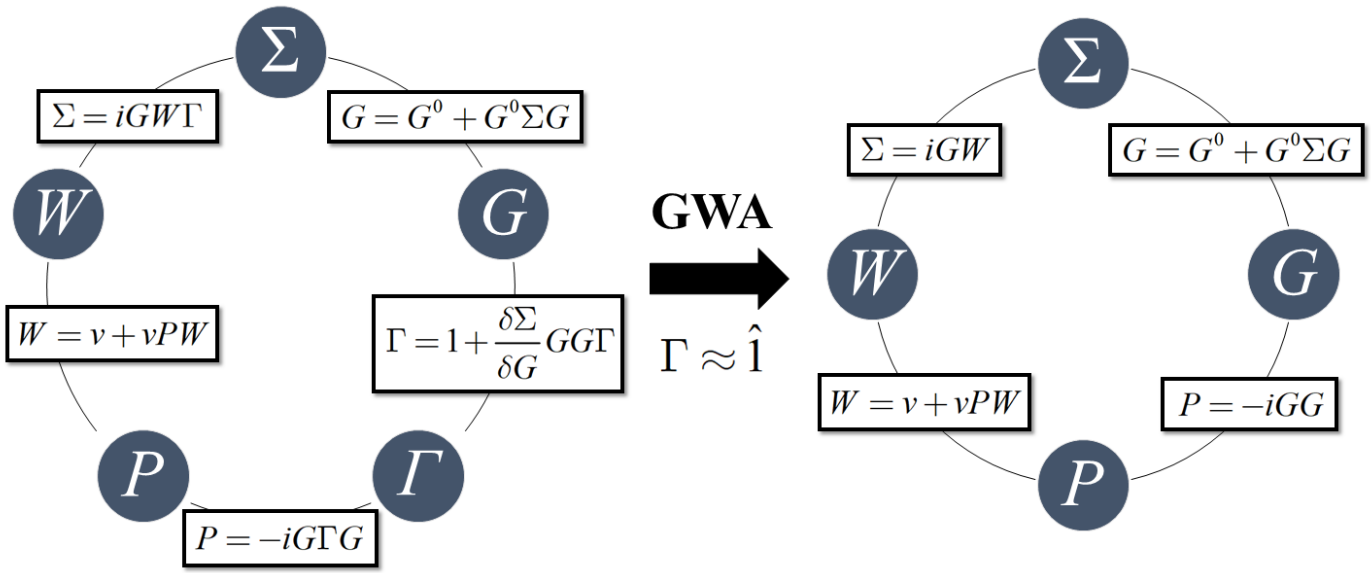
$$[h_0(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r})]\, \varphi_n^{\mathrm{QP}}(\mathbf{r}) + \int \mathrm{d}\mathbf{r}' \Sigma(\mathbf{r}, \mathbf{r}', \epsilon^{\mathrm{QP}}) \varphi_n^{\mathrm{QP}}(\mathbf{r}')$$

(Σ small) $\qquad = \epsilon_n^{\mathrm{QP}} \varphi_n^{\mathrm{QP}}(\mathbf{r})$

Σ: self-energy $\qquad \epsilon^{\mathrm{QP}}$: quasi-particle (QP) energy

F. Aryasetiawan and O. Gunnarsson, Rep. Prog. Phys. **61**, 237 (1998)

Exact solution for Σ: Hedin's equations



Hedin's pentagon with

- v (W): bare (screened) Coulomb interaction
- polarization function P and vertex function Γ

L. Hedin, Phys. Rev. **139**, A796–A823 (1965)
F. Aryasetiawan and O. Gunnarsson, Rep. Prog. Phys. **61**, 237 (1998)

G_0W_0/GW_0 approximation for calculating Σ

$$\Sigma(\mathbf{r},\mathbf{r}',\omega) = \frac{\mathrm{i}}{2\pi}\int_{-\infty}^{\infty} \mathrm{d}\omega' \mathrm{e}^{\mathrm{i}\omega'\delta} G_0(\mathbf{r},\mathbf{r}',\omega+\omega') W_0(\mathbf{r}',\mathbf{r},\omega')$$

$$G_0\left(\mathbf{r},\mathbf{r}',\omega\right) = \sum_{n\mathbf{k}} \frac{\psi_{n\mathbf{k}}\left(\mathbf{r}\right)\psi^*_{n\mathbf{k}}\left(\mathbf{r}'\right)}{\omega-\epsilon^{\mathrm{KS}}_{n\mathbf{k}}-\mathrm{i}\eta\mathrm{sgn}\left(\mu-\epsilon^{\mathrm{KS}}_{n\mathbf{k}}\right)}$$

$$W_0(\mathbf{r},\mathbf{r}',\omega) = \int \mathrm{d}\mathbf{r}''\, \varepsilon_0^{-1}(\mathbf{r},\mathbf{r}'',\omega) v(\mathbf{r}'',\mathbf{r}')$$

$$\boldsymbol{\varepsilon}_0(\omega) = \mathbf{1} - \mathbf{v}\mathbf{P}_0(\omega) = \mathbf{1} - \mathbf{v}\boldsymbol{\chi}_0(\omega)$$

$$\boldsymbol{\chi}_0(\omega) = -\frac{\mathrm{i}}{2\pi}\int_{-\infty}^{\infty} \mathrm{d}\omega' \mathrm{e}^{\mathrm{i}\omega'\delta} \mathbf{G}_0(\omega+\omega')\mathbf{G}_0(\omega')$$

$$\mathbf{G}^{-1} = \mathbf{G}_0^{-1} - \Sigma$$

M. S. Hybertsen and S. G. Louie, Phys. Rev. B **34**, 5390–5413 (1986), M. Shishkin and G. Kresse, Phys. Rev. B **75**, 235102 (2007)

GW Implementation (recp. space and freq. domain)

$$M^i_{nm}(\mathbf{k},\mathbf{q}) = \sqrt{N_k} \int_V \mathrm{d}\mathbf{r}\, \left[\chi^{\mathbf{q}}_i(\mathbf{r})\psi_{m\mathbf{k}-\mathbf{q}}(\mathbf{r})\right]^* \psi_{n\mathbf{k}}(\mathbf{r})$$

$$P_{ij}(\mathbf{q},\omega) = \frac{1}{\Omega}\sum_{\mathbf{k}}\sum_{nm} F_{nm}(\mathbf{k},\mathbf{q},\omega) M^i_{nm}(\mathbf{k},\mathbf{q}) \left[M^j_{nm}(\mathbf{k},\mathbf{q})\right]^*$$

$$F_{nm}(\mathbf{k},\mathbf{q},\omega) = 2f_{n\mathbf{k}}\left(1-f_{m\mathbf{k}-\mathbf{q}}\right)\left\{\frac{1}{\omega-\epsilon_{m\mathbf{k}-\mathbf{q}}+\epsilon_{n\mathbf{k}}+\mathrm{i}\eta} - \frac{1}{\omega+\epsilon_{m\mathbf{k}-\mathbf{q}}-\epsilon_{n\mathbf{k}}-\mathrm{i}\eta}\right\}$$

$$\Sigma^{\mathrm{x}}_{n\mathbf{k}} = -\frac{1}{N_k}\sum_{\mathbf{q}}\sum_i v_i(\mathbf{q}) \sum_m f_{m\mathbf{k}-\mathbf{q}} \left|M^i_{mn}(\mathbf{k},\mathbf{q})\right|^2 \text{ (exact exchange, EXX)}$$

$$\Sigma^{\mathrm{c}}_{n\mathbf{k}} = \frac{\mathrm{i}}{2\pi N_k}\int_{-\infty}^{\infty}\mathrm{d}\omega' \sum_m \sum_{\mathbf{q}} \frac{X_{nm}(\mathbf{k},\mathbf{q},\omega')}{\omega+\omega'-\epsilon_{m\mathbf{k}-\mathbf{q}}+\mathrm{i}\eta\mathrm{sgn}\left(\epsilon_{m\mathbf{k}-\mathbf{q}}-\mu\right)}$$

$$X_{nm}(\mathbf{k},\mathbf{q},\omega) = \sum_{ij}\left[M^i_{nm}(\mathbf{k},\mathbf{q})\right]^*\left[\varepsilon_{ij}(\mathbf{q},\omega)-1\right]M^j_{nm}(\mathbf{k},\mathbf{q})$$

- Auxiliary basis $\chi^{\mathbf{q}}_i(\mathbf{r})$: plane-wave, NAO, product basis, ...
- $\sum nm$: core states, high-lying states, basis-set completeness ...
- Frequency integration $\int \mathrm{d}\omega$: plasmon-pole model, full frequency, ...

(Static) Bethe-Salpeter equation

Similar to the Casida equations in TD-DFT

$$\begin{pmatrix} R & C \\ -C^* & -R^* \end{pmatrix} \begin{pmatrix} X^m \\ Y^m \end{pmatrix} = \Omega_m \begin{pmatrix} X^m \\ Y^m \end{pmatrix}$$

with

$$R_{ia,jb} = \left(\varepsilon_a^{\mathrm{QP}} - \varepsilon_i^{\mathrm{QP}}\right) \delta_{ij}\delta_{ab} + 2v_{ia,jb} - W_{ij,ab}$$

$$C_{ia,jb} = 2v_{ia,bj} - W_{ib,aj}$$

$$v_{ia,jb} = (ia \mid jb) = \langle ij|ab\rangle = \int \mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}'\, \psi_i^*(\mathbf{r})\psi_j^*(\mathbf{r}')v(\mathbf{r},\mathbf{r}')\psi_a(\mathbf{r})\psi_b(\mathbf{r}')$$

$$W_{ia,jb} = \int \mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}'\, \psi_i^*(\mathbf{r})\psi_j^*(\mathbf{r}')W(\mathbf{r},\mathbf{r}',\omega=0)\psi_a(\mathbf{r})\psi_b(\mathbf{r}')$$

Solve BSE to obtain the neutral excitation energy Ω_m and electron-hole eigen-state

$$\Psi_m^{\mathrm{eh}}(\mathbf{r}_\mathrm{e},\mathbf{r}_h) = \sum_{ia} \left[X_{ia}^m \psi_i(\mathbf{r}_\mathrm{h})\,\psi_a(\mathbf{r}_\mathrm{e}) + Y_{ia}^m \psi_i(\mathbf{r}_\mathrm{e})\,\psi_a(\mathbf{r}_\mathrm{h})\right]$$

X. Blase et al., J. Phys. Chem. Lett. **11**, 7371–7382 (2020)

Why *GW*/BSE in GPAW

- open-source and well-documented
- powered by Python ecosystem
- good performance due to efficient parallelization and native C code for computation-extensive parts
- various functionalities including vertex correction, Coulomb truncation and interpolation schemes

F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **88**, 245309 (2013), F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **87**, 235132 (2013), T. Olsen et al., npj Comput. Mater. **5**, 1–23 (2019), F. A. Rasmussen et al., Phys. Rev. B **94**, 155406 (2016), P. S. Schmidt, C. E. Patrick, and K. S. Thygesen, Phys. Rev. B **96**, 205206 (2017), K. S. Thygesen, 2D Mater. **4**, 022004 (2017)

Workflow for G_0W_0 / GW_0

GW functionality is provided in GPAW since 2014.

The typical workflow follows 3 steps:

Step 1 SCF for converged density

Step 2 non-SCF calculation for occupied and unoccupied states (practically on a smaller **k**-point mesh)

Step 3 *GW* calculations

F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **87**, 235132 (2013)

Workflow for G_0W_0/GW_0

GW functionality is provided in GPAW since 2014.

Step 1: SCF

```
from ase.build import bulk
from gpaw import GPAW, FermiDirac
from gpaw.wavefunctions.pw import PW

atoms = bulk('C', 'diamond', a=3.567)
c = GPAW(mode=PW(400), xc='PBE',
         kpts={'size': (4, 4, 4), 'gamma': True},
         convergence={'energy': 1e-8},
         occupations=FermiDirac(0.001),
         txt='C_gs.txt')
atoms.calc = c
atoms.get_potential_energy()
c.write ('C_gs.gpw')
```

Workflow for G_0W_0/GW_0

Step 2: get all bands (here on the same **k** grids as SCF)

```
c.diagonalize_full_hamiltonian()
c.write('C_gs_full.gpw', mode='all')
```

Note that the full GPAW file can be huge, depending on the PW cut-off and **k**-point mesh

Workflow for G_0W_0/GW_0

Step 3: GW

```
from gpaw.response.g0w0 import G0W0

gw = G0W0(calc='C_gs_full.gpw', bands=(1, 7),
          nbands=144, ecut=100,
          filename='C_g0w0-k4-ecut100')
result = gw.calculate()
```

Run serially and the following files are generated:

```
C_g0w0-k4-ecut100.exx.npy       C_g0w0-k4-ecut100.exx.txt
C_g0w0-k4-ecut100_results.pckl  C_g0w0-k4-ecut100.txt
C_g0w0-k4-ecut100.vxc.npy       C_g0w0-k4-ecut100.w.txt
```

Glimpse of *GW* output files

`filename.exx.txt`: details of EXX calculation

```
...
Distributing spins, k-points and bands (1 x 64 x 4) over 1 process
Number of blocks: 1
Using Wigner-Seitz truncated coulomb interaction.
Inner radius for 4x4x4 Wigner-Seitz cell: 4.119 Ang
Range-separation parameter: 1.214 Ang^-1
FFT size for calculating truncated Coulomb: 60x60x60
EXX eigenvalue contributions in eV:
[[[-20.239 -20.239 -20.239 -10.109 -10.109 -10.108]
  [-22.192 -20.633 -20.633  -9.845  -7.755  -9.301]
  ...
  [-22.737 -20.861 -20.861  -8.851  -8.851 -11.025]]]
```

Glimpse of *GW* output files

`filename.w.txt`: details of response function (from `Chi0`)

```
...
Point group included. Time reversal included. Disabled non symmorphic
↪  symmetries. Found 8 allowed symmetries. 18 groups of equivalent kpoints.
↪  71.875% reduction.

  ( 1  0  0)  ( 1  0 -1)  ( 0  1  0)  ( 0  1 -1)  ( 0  1 -1)  ( 0  1  0)
  ( 0  1  0)  ( 0  1 -1)  ( 1  0  0)  ( 1  0 -1)  ( 0  1  0)  ( 0  1 -1)
  ( 0  0  1)  ( 0  0 -1)  ( 0  0  1)  ( 0  0 -1)  (-1  1  0)  (-1  1  0)

  ( 1  0 -1)  ( 1  0  0)
  ( 1  0  0)  ( 1  0 -1)
  ( 1 -1  0)  ( 1 -1  0)

Integrating response function.
Integral kind: spectral function
Distributing domain [18, 1] over 1 process
Number of blocks: 1
1.3598370552062988s |------------------------------------| Time: 1.349s
...
```

repeated for 8 ($N_{\mathbf{k}}$ in IBZ) times

Glimpse of *GW* output files

`filename_results.pckl`: binary file (pickle) for saving data

```
>>> import pickle
>>> data = pickle.load(open('C_g0w0-k4-ecut100_results.pckl', 'rb'))
>>> list(data.keys())
['f', 'eps', 'vxc', 'exx', 'sigma', 'dsigma', 'Z', 'qp', 'iqp']
# occupancies at Gamma
>>> data['f'][0, 0]
array([1., 1., 1., 0., 0., 0.])
# KS gap at Gamma
>>> data['eps'][0, 0, 3] - data['eps'][0, 0, 2]
5.550967239920101 5
# QP gap at Gamma
>>> data['qp'][0, 0, 3] - data['qp'][0, 0, 2]
7.335502210222952
```

QP band structures

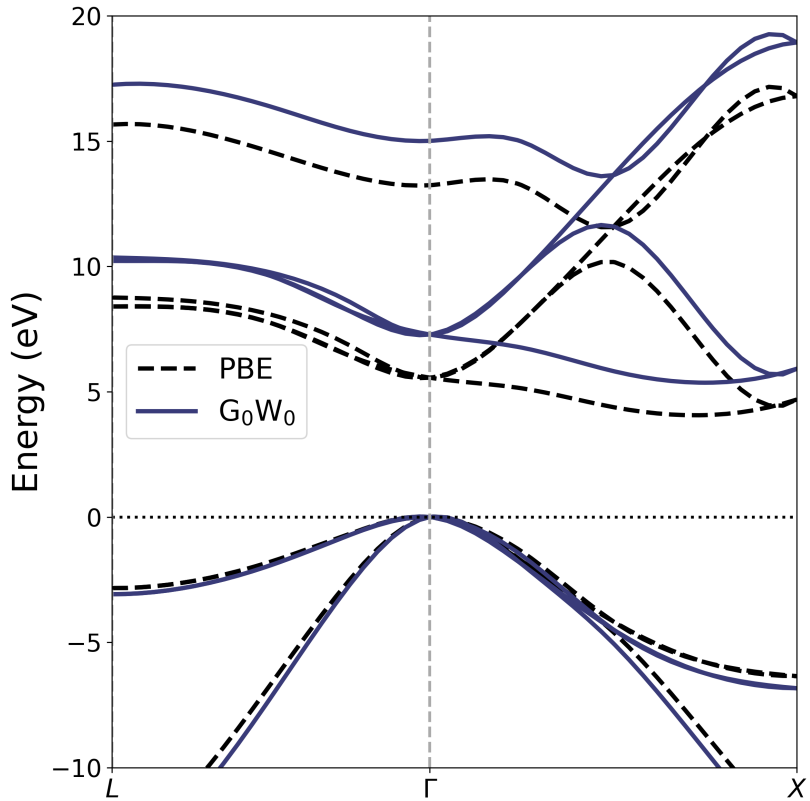
Use `GWBands` object to obtain QP band structure along a particular path by interpolating the results of a regular **k**-mesh

```
from gpaw.response.gw_bands import GWBands

L = np.array([0.5, 0.5, 0.5])
X = np.array([0.5, 0.5, 0.0])
G = np.array([0.0, 0.0, 0.0])
kpoints = np.array([L, G, X])

gwbands = GWBands(calc=gpwscf,
                  gw_file=pckl,
                  kpoints=kpoints,
                  bandrange=(0,8))

gw = gwbands.get_gw_bands(nk_Int=50,
                          interpolate=True)
```



`C_band.png`

Workflow for BSE based on QP energies

```
import numpy as np
from gpaw.response.bse import BSE

fn = 'Si_g0w0-k8-ecut100_results.pckl'
qp = pickle.load(open(fn, 'rb'))['qp']
# initialize BSE object
bse = BSE('Si_gs.gpw', ecut=20,
          gw_skn=qp[:, :, :], # should conform vb and cb
          valence_bands=[1, 2, 3],
          conduction_bands=[4, 5, 6],
          nbands=48, txt='Si_bse.txt')

# start real work
bse.get_dielectric_function(q_c=[0.0, 0.0, 0.0], eta=0.05,
                            write_eig="eig-Gamma.dat",
                            w_w=np.linspace(0, 10, 10001),
                            filename="dielec-Gamma.csv")
```

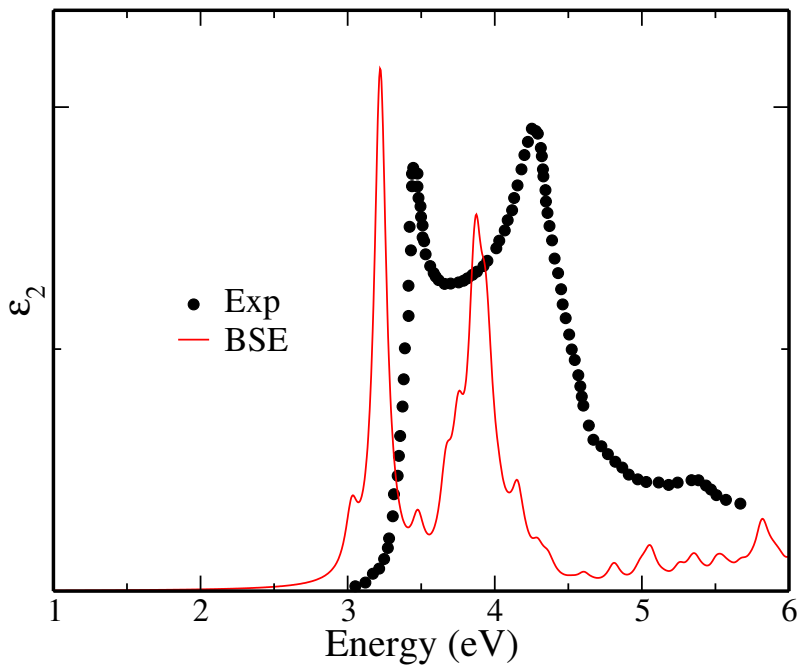
Instead of QP energies from *GW*, one can also use KS energies modified by scissors operator, especially when the **k**-mesh is rather dense.

Glimpse of BSE output

`eig-Gamma.dat`

```
# BSE eigenvalues in eV
       0     3.024679 0.0000005562442000
       1     3.024683 0.0000013704094051
       2     3.024692 0.0000631292840484
       3     3.026919 0.0000048426899972
...
       7     3.027818 0.0007571072157299
       8     3.033229 0.0000001653798213
       9     3.221671 3.5624202893152370
      10     3.221758 0.0018907765523448
      11     3.221800 1.6440997878927592
...
     163     3.874447 1.2703810427209152
...
```

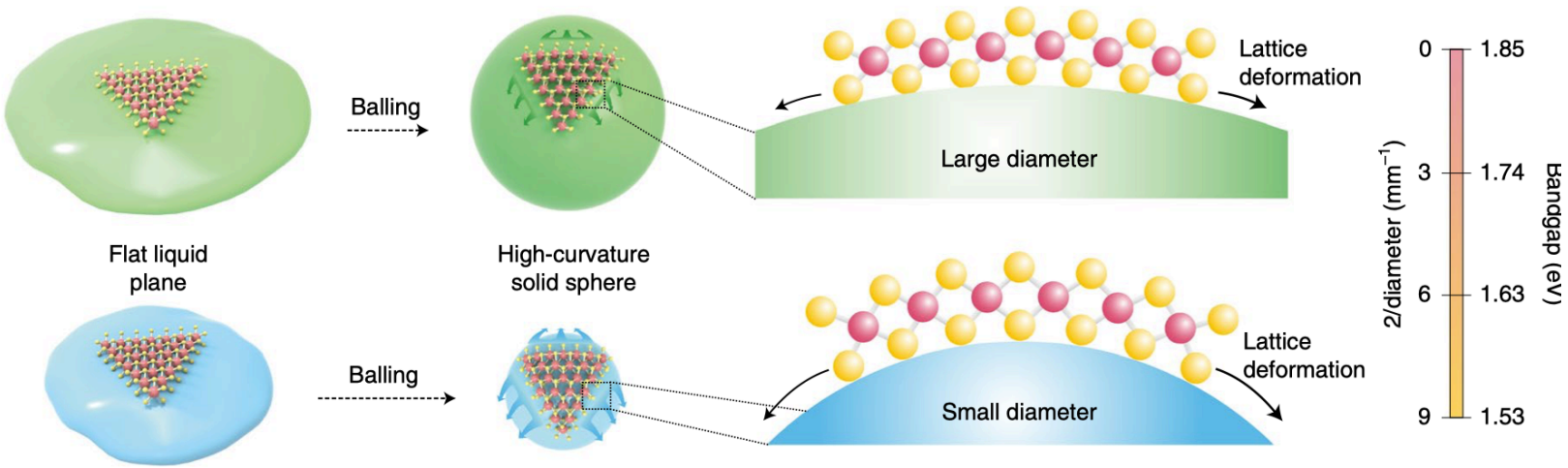
Plot with `dielec-Gamma.csv`



Outline

Background

Efficient gap tuning of mono-layer MoS_2 by sphere diameter engineering



Argument to be verified: biaxial or isotropic strain is more efficient to tune the band gap of monolayer MoS_2 than uniaxial strain.

M. Zeng et al., Nature Mater. **19**, 528–533 (2020)

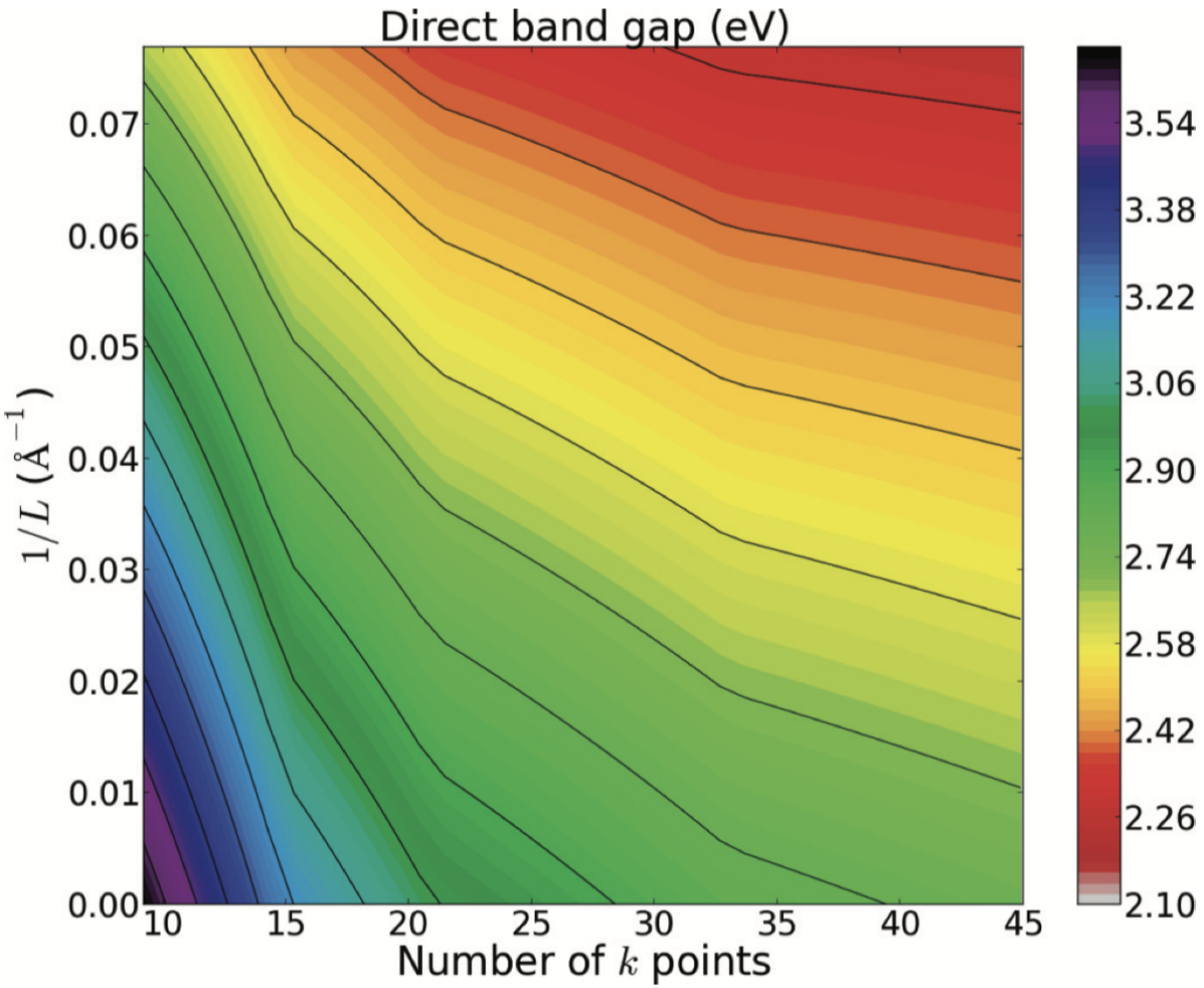
Modelling of gap engineering by strains

- strain: ml-MoS_2 with scaled lattice constant (internally relaxed)
- band gap: measured by photoluminescence (PL) spectra in experiment

  (ignore Stokes shift) $\rightarrow$ optical absorption spectra

  (ignore phonon-assisted excitation) $\rightarrow$ GW/BSE without moment transfer

Error cancellation in QP energy of 2D system



F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **88**, 245309 (2013)

Solutions in GPAW

The convergence can be improved by

- 2D truncation of Coulomb interaction for L
- analytic correction from averaging $\varepsilon(\mathbf{q} \to 0)$ around Γ for $\mathbf{k}$-mesh

```
strain = 'bi_0p'
diagfile = f'MoS2_{strain}_fulldiag.gpw'
domega0, omega2 = 0.01, 10
fn = f'MoS2_g0w0_{strain}'

gw = G0W0(calc=diagfile, method="G0W0",
          filename=fn, bands=(8,18),# vb at index 12
          ecut=200, ecut_extrapolation=True,
          truncation='2D', q0_correction=True,
          domega0=domega0, omega2=omega2)
gw.calculate()
```

Cut-off of repsonse function and frequency integration should also be checked (against our interested property: first absorption peak)

S. Ismail-Beigi, Phys. Rev. B **73**, 233103 (2006), F. A. Rasmussen et al., Phys. Rev. B **94**, 155406 (2016)
Scripts written with GPAW v1.5.2

QP band with SOC considered

For ML-MoS_2, spin-orbit coupling is essential to get qualitatively correct absorption spectra

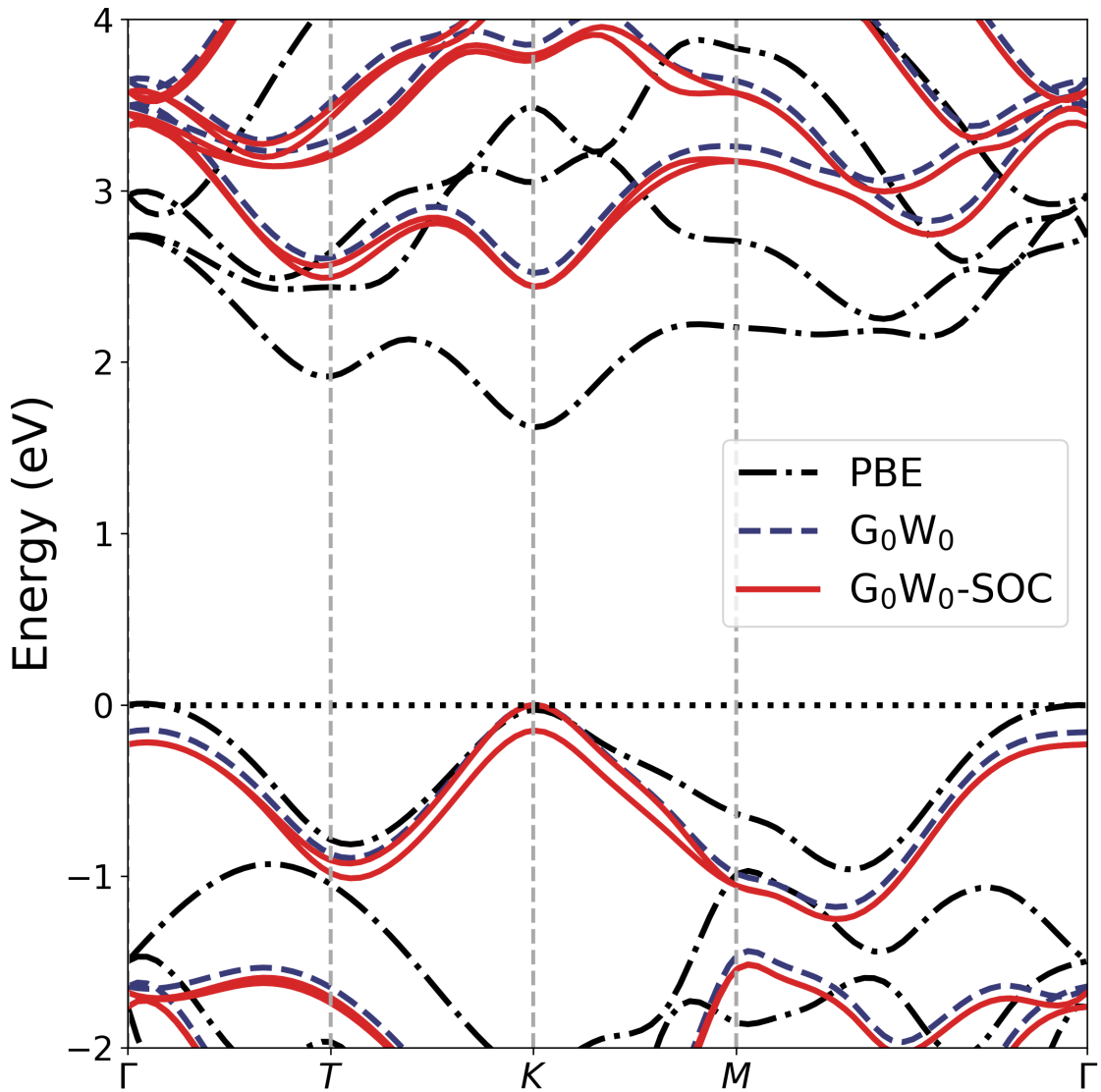
```
from gpaw.response.gw_bands import GWBands

K = np.array([1/3, 1/3, 0.0])
M = np.array([0.0, 0.5, 0.0])
G = np.array([0.0, 0.0, 0.0])
kpoints = np.array([G, K/2, K, M, G])

GW = GWBands(calcfile=gsfile,
             gwpckl=f'{filename}.pckl',
             kpoints=kpoints, bandrange=(8,17))
soc = GW.get_gw_bands(SO=True, dft=False,
                      interpolate=True, vac=False)
```

Scripts written with GPAW v1.5.2

QP band with SOC considered



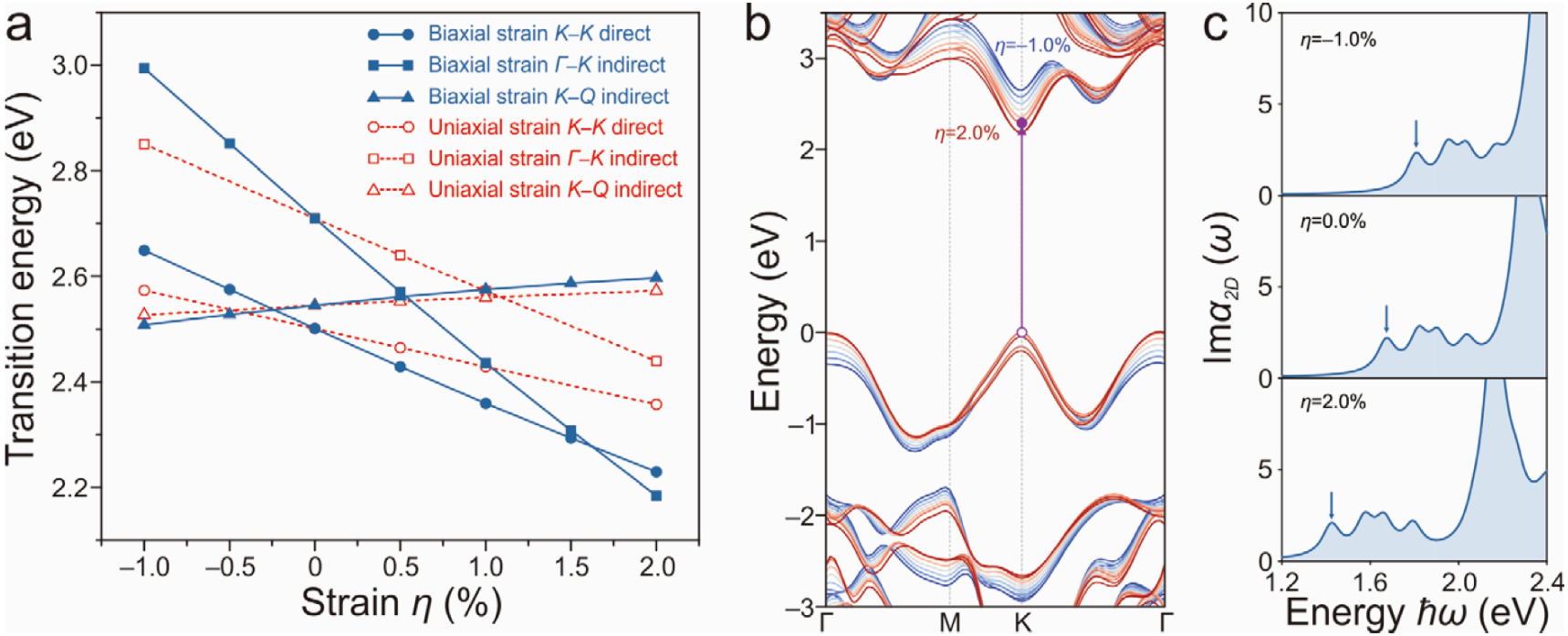
BSE on top of *GW*-SOC

Modification to the original BSE object

```
bse = BSE(gsfile, txt='Gamma_bse.txt'
          spinors=True, gw_skn_soc=gw_skn_soc,
          truncation='2D',
          valence_bands=[11, 12],
          conduction_bands=[13, 14],
          nbands=nbands_bse, ecut=ecut)
```

- shape of `gw_skn_soc` conforms that of eigenvalues in ground-state calculations, which is different from `gw_skn`
- parameter `gw_skn_soc` in fact was not working properly in v1.5.2
- in v21.1.0, `gw_skn_soc` is removed and SOC is handled by `spinors`

Results



M. Zeng et al., Nature Mater. **19**, 528–533 (2020)

Thank you for listening!

References

[1] F. Aryasetiawan and O. Gunnarsson, Rep. Prog. Phys. **61**, 237 (1998).

[2] X. Blase et al., J. Phys. Chem. Lett. **11**, 7371–7382 (2020).

[3] Z.-H. Cui et al., J. Phys. Chem. Lett. **9**, 2338–2345 (2018).

[4] J. Enkovaara et al., J. Phys.: Condens. Matter **22**, 253202 (2010).

[5] L. Hedin, Phys. Rev. **139**, A796–A823 (1965).

[6] F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **88**, 245309 (2013).

[7] F. Hüser, T. Olsen, and K. S. Thygesen, Phys. Rev. B **87**, 235132 (2013).

[8] M. S. Hybertsen and S. G. Louie, Phys. Rev. B **34**, 5390–5413 (1986).

[9] S. Ismail-Beigi, Phys. Rev. B **73**, 233103 (2006).

[10] T. Olsen et al., npj Comput. Mater. **5**, 1–23 (2019).

[11] J. P. Perdew et al., Phys. Rev. Lett. **49**, 1691–1694 (1982).

[12] F. A. Rasmussen et al., Phys. Rev. B **94**, 155406 (2016).

[13] P. S. Schmidt, C. E. Patrick, and K. S. Thygesen, Phys. Rev. B **96**, 205206 (2017).

[14] M. Shishkin and G. Kresse, Phys. Rev. B **75**, 235102 (2007).

[15] K. S. Thygesen, 2D Mater. **4**, 022004 (2017).

[16] F. Tran, S. Ehsan, and P. Blaha, Phys. Rev. Materials **2**, 023802 (2018).

[17] M. Zeng et al., Nature Mater. **19**, 528–533 (2020).