Practicum, GW package

Mark van Schilfgaarde, King's College London

The current implementation of the GW code:

- requires information about eigenfunctions supplied by a driver, lmfgwd. lmfgwd is built from the LDA package.
- Comes in a separate package which consist of a family of binaries an shell scripts to run them. Each binary reads a single integer from stdin which tells it what branch to run.
- generates response functions, and self-energies Σ, QP levels.
- Quasiparticle calculations come in two flavors:
  - 1-shot calculations where QP shifts relative to LDA (or some starting point) are calculated from the GW approximation
  - Quasiparticle self-consistent calculations (QSGW). A nonlocal potential is generated, which replaces the LDA exchange correlation potential. This theory is (in principle) independent of the LDA.

Shell Scripts

There are three main shell scripts:

lmgw : the main driver for a single cycle to calculate response functions or the self-energy GW

lmgwsc : a script for QSGW calculations. It runs lmgw until self-consistency is reached (more details next slide)

lmgw1-shot : a script for running 1-shot GW calculations from some starting H_0 (usually $H_0=H_{\mathrm{LDA}}$)

Each script requires the extension naming the ctrl file lmf uses.

Additionally each script has optional switches, e.g.

lmgw --version prints out the version number of the script.

Use lmgw --help to see the available options.

Here is typical invocation (we will describe the switches later):

lmgwsc --code2 --maxit=0 –sym --metal coo2

Input file ctrl.coo2

QSGW cycle

QSGW starts from some noninteracting H_0 and builds a self-energy from it through the GW cycle. Initially $H_0=H_{\mathrm{LDA}}$ or $H_{\mathrm{LDA}}+U$. The cycle is:

0. An LDA or LDA+U calculation is performed to make some H_0.
1. Information needed for a GW cycle (e.g. eigenvalues, eigenfunctions and projections into augmentation spheres) is generated by lmfgwd.
2. The self-energy Σ is calculated through the GW cycle
3. A "frozen" or quasiparticlized self-energy Σ' is made and V_{xc}^{LDA} subtracted. $\Sigma'-V_{xc}^{\mathrm{LDA}}$ is stored on disk.
4. lmf reads $\Sigma'-V_{xc}^{\mathrm{LDA}}$ as an external potential added to H_{LDA}. lmf makes the density self-consistent in the presence of this external potential.
5. Steps 1-4 are repeated until Σ' stops changing. At self-consistency $H_0=H_{\mathrm{LDA}}+\Sigma'-V_{xc}^{\mathrm{LDA}}$ -- independent of the LDA.

Where codes reside

These codes are made in the build of the LDA package:

lmfa	LDA free atom code
lmf	LDA band code
lmfgwd	Driver for GW codes
lmf2gw_0	file format translator for original GW codes
lmf2gw_2	file format translator for Sep12 version

Install these codes into some directory in your path, e.g. bin, together with the shell scripts lmgw, lmgwsc, lmgw1-shot.

These executables : hbasfp0 hvccfp0 hx0fp0 hsfp0 hef hqpe qg4gw heftet rdata4gw_v2 hx0fp0_sc hsfp0_sc hqpe_sc

and some others are built with the GW package and go into subdirectory code2, e.g. bin/code2. (They will be explained soon)

If you compile the original (pre Sep12) package put those executables into bin/code0

Modification of the ctrl file for QSGW

lmf is a stand-alone LDA code, but can add $\Sigma'-V_{xc}^{\mathrm{LDA}}$ generated by the GW code to complete the QP part of the QSGW cycle. Its primary input file is ctrl.ext (also rst.ext)

$\Sigma'-V_{xc}^{\mathrm{LDA}}$, if present, can be read from file sigm.ext.

To tell lmf or lmfgwd to read sigm from the ctrl file, set RDSIG=12 by adding a token to the HAM category:

```
HAM      RDSIG=12 SIGP[MODE=4 EMAX=2]
```

At the same time, you need to set parameters in SIGP, as shown above. The reason for this is a complicated story, but it has to do with mitigating difficulties in interpolating Σ' to $\mathbf{k}$ points other than the ones for which it is generated. See doc/gw.html .

Help with lmfgwd

The LDA codes have two primary input streams:
(1) the input file ctrl.ext and (2) command-line switches.
To see what command-line switches lmfgwd looks for, do:

```
%  lmfgwd --help
 usage:  lmfgwd [--OPTION] [-var-assign] [ext]

 --h
 --help          Print this message, and quit
 --input         List categories, tokens, and data program expects, and quit
 --show          Print control file after parsing by preprocessor,
                 and echo input data as read from the control file
```

...

```
lmfgwd-specific options: (see doc/Command-line-options.html for documentation)
--gwcode=#     #=0 original code;  1 spex code;  2 code from Sep12
```

Most of the time we use --gwcode=2. (Note: the driver to the Julich spex code has been made but it is not finished...)

GW input file

The GW codes themselves need one input file you must supply in addition to the several files lmfgwd will generate.

File GWinput contains input specific to the GW codes, e.g. information about the product basis and cutoffs.

lmfgwd does not read GWinput but it needs some information from the GW codes (e.g. k-point data) to know what to make.

The connection between the GW codes and lmfgwd involves a rather elaborate handshaking between the two (see later).

GWinput is a complicated, unfriendly file. Rather than make it from scratch, it is best to let lmfgwd make a template you can modify. lmfgwd accepts a single integer from stdin to tell it what mode to run. Use -1 to make a template:

```
echo -1 | lmfgwd ...
```

GWinput template for CoO2

The following will create a template GWinput for CoO_2:

```
cp gwd/test/coo2.sep12/* .
echo -1 | lmfgwd --gwcode=2 coo2
```

It will create a working GWinput file. Most of the default values are reasonable, but you need to check the entire file.

Many of the values lmfgwd puts into GWinput can be set in the ctrl file. To see, what values you can set, do

```
lmfgwd –input
```

Look for tokens GW_xxx, e.g.

```
GW_GCUTB          opt    r8       1,  1          default = 2.7
  G-vector cutoff for basis envelope functions
GW_GCUTX          opt    r8       1,  1          default = 2.2
  G-vector cutoff for interstitial part of response function
```

This pair of tokens sets the values of QpGcut_psi and QpGcut_cou in GWinput (shown in the next slides).

GWinput template for CoO2

The box shows the beginning of the General section of GWinput.
Lines beginning with `!' are not read.

```
!Verbose    0   ! 0-->default; 100-->debug
!Q0P_Choice 0   ! 0-->along plat(default);  1-->along x,y,z
!CoreOrth  off  ! off --> Do not orthogonalize core to valence (default)
                ! on  --> Orthogonalize cores to valence (may give strange core functions!)
!multitet 2 2 2 ! tetrahedra divided into micro-tetrahedra
!EXonly    .15  ! for exchange-only calculations
EIBZmode off    ! turn off to suppress use of symmetrization in calculating polarization
!TimeReversal off ! when time-reversal symmetry is broken
KeepEigen  on   ! keep eigenfunctions in memory
KeepPPOVL  on   ! keep PPOVL in memory
!Chi_RegQbz on  ! Offset Gamma point mesh for chi.  on => no offset
BZmesh     1    ! Offset Gamma point mesh for Sigma=iGW
WgtQ0P     0.01 ! Weight used when BZmesh is 2
```

Use EIBZmode off for now (on makes better use of symmetry, but it is not working properly yet)

Set KeepEigen off and KeepPPOVL off to save memory

Chi_RegQbz, BZmesh, WgtQ0P generate an offset k mesh --- useful for anisotropic systems. See Eq. 53 in PRB76, 165106

GWinput template for CoO2, cotd

The box below shows the remainder of the General section.

Most tags are explained in the manual GW-man033_ver1.pdf (a bit outdated now) or in the slides below.

```
n1n2n3  4  4  4 ! for GW BZ mesh
QpGcut_psi 2.7  !|q+G| cutoff for eigenfunction
QpGcut_cou 2.2  !|q+G| cutoff for coulomb int.
unit_2pioa off  ! off --> units of 2 preceding Gcut are a.u.; on--> units are 2*pi/ala
alpha_OffG 1    !(a.u.) parameter in the auxiliary function in the offset-Gamma
!nband_chi0 999 !nband cutoff for chi0  (Optional)
!emax_chi0 999. !emax  cutoff for chi0, Ry  (Optional)
!nband_sigm 999 !nband cutoff for Sigma (Optional)
emax_sigm 2     !Energy cutoff for Sigma, Ry (Optional)
dw       0.01   !mesh spacing along Real axis (Hartree)
omg_c    0.04   !Used in S. Faleev's real-axis mode
iSigMode 3      !QSGW mode switch (QSGW only)
niw      6      !# freq. on Im axis; used for integration to make Sigma_c
delta    -1e-4  !delta-function broadening for calc. x0, a.u.. delta<0 => tetrahedron
deltaw   0.02   !width in finite diff for sigma energy derivative, a.u.
esmr     3e-3   !Broadening in the poles of G(LDA) (hsfp0)
                !Change esmr for metals: see DOSACC* --- especially around Ef
GaussSmear on   !on  => broadening of poles in G(LDA) by Gaussian
                !off => broadening of poles by a rectangle
!mixbeta   .25  !mixing of input, output sigma for self-consistency
```

GWinput template for CoO2 General section

```
n1n2n3  4  4  4 ! for GW BZ mesh
QpGcut_psi 2.7  !|q+G| cutoff for eigenfunction
```

Specifies k mesh, analog of BZ_NKABC in the LDA codes.
Make as small as you can for accuracy you need:
Note: computer time scales as $(n1n2n3)^2$

```
n1n2n3  4  4  4 ! for GW BZ mesh
QpGcut_psi 2.7  !|q+G| cutoff for eigenfunction
QpGcut_cou 2.2  !|q+G| cutoff for coulomb int.
```

G cutoffs respectively for the interstitial part of the eigenfunctions and eigenfunction products (product basis).

Computational time scales as $(QpGcut\_cou)^3$. It scales only linearly in QpGcut_psi (I think).
QpGcut_cou and QpGcut_psi should scale inversely with the atom sizes:
QpGcut_cou=2.2 is ok for materials w/out 2nd row elements.
For an oxide like COO_2 or CdO it should be bigger, about 2.8

GWinput template for CoO2 General section

```
!nband_sigm 999 !nband cutoff for Sigma (Optional)
emax_sigm 2     !Energy cutoff for Sigma, Ry (Optional)
dw       0.01   !mesh spacing along Real axis (Hartree)
omg_c    0.04   !Used in S. Faleev's real-axis mode
```

Use nband_sigm to truncate the number of unoccupied states when calculating polarization or Σ. Normally you include all unoccupied states.

Use emax_sigm to truncate the energy window over which Σ is made. Used by QSGW only. It is necessary because including Σ at high energy causes problems for the k-point interpolation of $\Sigma' - V_{xc}^{\mathrm{LDA}}$ in the lmf code.

How the k-point interpolation is handled is complicated. It is described in some detail in Section II G in PRB 76, 165106. See also gw.html .

GWinput template for CoO2 General section

```
!nband_sigm 999 !nband cutoff for Sigma (Optional)
emax_sigm 2     !Energy cutoff for Sigma, Ry (Optional)
dw        0.01  !mesh spacing along Real axis (Hartree)
omg_c     0.04  !Used in S. Faleev's real-axis mode
```

dw and omg_c define the energy mesh used for real-axis integration of the polarizability. For low frequencies the energy mesh spacing is uniform; At around omg_c the spacing between each successive mesh point increases.
See around Eq. 39 in PRB 76, 165106; or
see routine optics/dosmsh.f for a definition of the mesh:

$$\omega_i = \mathrm{dw} \times (i-1) + \frac{\mathrm{dw}^2}{2 \times \mathrm{omg\_c}} (i-1)^2$$

dw=.01, omg_c=.04 is a bit conservative:
dw=.02, omg_c=.02 seem to work well most of the time.
You want to make dw=.02, omg_c=.02 as large (small) as you can without sacrificing accuracy.

GWinput template for CoO2 General section

```
iSigMode 3      !QSGW mode switch (QSGW only)
niw      6      !# freq. on Im axis; used for integration to make Sigma_c
```

iSigMode defines the QSGW "norm."
We almost always use iSigMode=3 which corresponds to `mode A' in PRB 76, 165106, Eq. (10).
iSigMode=1 corresponds to 'mode B', Eq. (11).
iSigMode=5 does eigenvalue-only self-consistency.

```
iSigMode 3      !QSGW mode switch (QSGW only)
niw      6      !# freq. on Im axis; used for integration to make Sigma_c
delta    -1e-4  !delta-function broadening for calc. x0, a.u.. delta<0 => tetrahedron
```

niw specifies the energy mesh for the (residual part) of the imaginary axis integration for Σ^c (the correlation part of the self-energy). It is described in detail around Eq. 56 in PRB 76, 165106. niw=6 seems to work well.

GWinput template for CoO2 General section

```
niw       6       !# freq. on Im axis; used for integration to make Sigma_c
delta     -1e-4   !delta-function broadening for calc. x0, a.u.. delta<0 => tetrahedron
deltaw    0.02    !width in finite diff for sigma energy derivative, a.u.
esmr      3e-3    !Broadening in the poles of G(LDA) (hsfp0)
```

esmr defines the width of the gaussian broadening of the poles of G when calculating Σ. For metals this quantity needs to be monitored. Sometimes (in Fe, for example) 0.003 causes problems; 0.01 is needed in that case.

For a detailed discussion, see Section 13 (around p45) in the GW reference manual GW-man033_ver1.pdf in the doc directory.

GWinput template for CoO2 General section

```
esmr      3e-3    !Broadening in the poles of G(LDA) (hsfp0)
                  !Change esmr for metals: see DOSACC* --- especially around Ef
GaussSmear on     !on  => broadening of poles in G(LDA) by Gaussian
                  !off => broadening of poles by a rectangle
!mixbeta    .25   !mixing of input, output sigma for self-consistency
```

esmr There is an option to smear the poles with a rectangle rather than a gaussian. This is rarely used.

```
esmr      3e-3    !Broadening in the poles of G(LDA) (hsfp0)
                  !Change esmr for metals: see DOSACC* --- especially around Ef
GaussSmear on     !on  => broadening of poles in G(LDA) by Gaussian
                  !off => broadening of poles by a rectangle
!mixbeta    .25   !mixing of input, output sigma for self-consistency
```

In convergence to self-consistency it is sometimes necessary to mix some some input with output self-energies to control convergence. It is the analog of the mixing beta in the LDA code.

GWinput template for CoO2 Product Basis section

```
<PRODUCT_BASIS>   ! Pr
  tolerance = minimum
   1.00000E-03
  lcutmx(atom) = l-cut
  4  4  4
  atom   l   nnvv   nnc
     1    0    2    3
     1    1    3    1
     1    2    3    0
     1    3    2    0
     1    4    2    0
     2    0    2    1
     2    1    2    0
     2    2    2    0
     2    3    2    0
     2    4    2    0
     3    0    2    1
```

Table should not be touched

For product basis inside augmentation spheres. Initially all possible products of designated partial waves are made. This is a complete product basis. The overlap between product functions is diagonalized and functions with eigenvalues below tolerance are discarded, to reduce the basis size.

Tolerance=1e-3 is usually safe;

Tolerance=1e-4 is very conservative.

Tolerance can be l-dependent, e.g. use

1E-4 1E-3 1E-2 1E-2 1E-2

There is one lcutmx for each site. lcutmx can be 3 for elements without d states, should be 6 for f

GWinput template for CoO2 Product Basis II

atom	l	n	occ	unocc	:Valence(
1	0	1	1	1	! 4S_p *
1	0	2	0	0	! 4S_d
1	1	1	1	1	! 4P_p
1	1	2	0	0	! 4P_d
1	1	3	1	1	! 3P_l
1	2	1	1	1	! 3D_p
1	2	2	0	0	! 3D_d
1	2	3	0	1	! 4D_l
1	3	1	0	1	! 4F_p
1	3	2	0	0	! 4F_d
1	4	1	0	0	! 5g_p
1	4	2	0	0	! 5g_d
2	0	1	1	1	! 2S_p *
2	0	2	0	0	! 2S_d
2	1	1	1	1	! 2P_p
2	1	2	0	0	! 2P_d
2	2	1	1	1	! 3D_p
2	2	2	0	0	! 3D_d
2	3	1	0	1	! 4F_p
2	3	2	0	0	! 4F_d
2	4	1	0	0	! 5g_p
2	4	2	0	0	! 5g_d
3	0	1	1	1	! 2S_p *

This table specifies which partial waves to use to assemble product basis. Each wave ϕ, ϕ' (phidot), or ϕ_z (local orbital) can enter into:
(1) the left side of the product pair
(2) the right side, or both.
'occ' specifies the lhs, 'unocc' specifies the rhs.

Usually we don't need to include phidots. Where local orbitals are used: if ϕ_z is deep, e.g. describing the Ga 3d state, include it in the 'occ' column; if it is high (above Ef) include it in the 'unocc' column.

GWinput template for CoO2 Product Basis III

atom	l	n	occ	unocc	ForX0	ForSxc	:CoreSta
1	0	1	0	0	0	0	! 1S *
1	0	2	0	0	0	0	! 2S
1	0	3	1	0	1	1	! 3S
1	1	1	0	0	0	0	! 2P
2	0	1	0	0	0	0	! 1S *
3	0	1	0	0	0	0	! 1S *

This table specifies which core levels to include in the product basis. There is only a ϕ for core levels.

Sometimes the highest-lying core of a particular l channel needs to be included, e.g. the Co 3s level in this case. Its inclusion in this case has a small effect. Run lmfa to see how deep the core levels are.

The last two columns set whether level should be included in the screening (ForX0) or in calculating Σ (ForSxc). Caution: experience has shown that ForX0 and ForSxc switches must be used with caution. If levels are deep they have no effect. If they are too shallow, the slight nonorthogonality with valence states causes difficulties.

GWinput template for CoO2 qpts specification

This section is for 1-shot GW only.

```
<QPNT>   ! QPNT block exactly as in file QPNT
 --- Specify qp and band indices at which to evaluate Sigma

*** Sigma at all q -->1; to specify q -->0.  Second arg : up only -->1, ot
  0  0
*** no. states and list of band indices to make Sigma and QP energies
 18
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
*** q-points (must belong to mesh of points in BZ).
 3
  1     0.0000000000000000     0.0000000000000000     0.0000000000000000
  2     0.1111111111111111     0.0000000000000000     0.0209533699514427
  3     0.2222222222222222     0.0000000000000000     0.0419067399028853
  4     0.3333333333333334     0.0000000000000000     0.0628601098543280
```

You can specify which k-points in the irreducible Brillouin zone you want to calculate QP levels for and also which QP levels. In this case all of the first 18 levels are calculated.

This section used when calculating response functions: you specify which **q** for which to calc. eps.

```
QforEPSIBZ off
<QforEPS>
0d0 0d0 0.015d0
</QforEPS>
```

Some test cases

The standard distribution has several tests.
To see what they are, do
gwd/test/test.gwd --list

Most of the checks just test the driver lmfgwd. (job 1)
There are three tests which invoke the GW codes:

```
  ... The following apply to jobs 2,3,4.  These tests require that the GW package be ins
  si2       tests operation of GW code for an insulator, (jobs 2 and 4)
  fe        tests 1-shot and 1 iteration of QSGW for a metal (jobs 2, 4, 5)
  coo2      0th iteration of QSGW for a difficult material (jobs 2 and 4, --code2 only)

jobs:  1: tests driver lmfgwd (main function of this script)
       2: one-shot calculations
       3: RPA total energy (not working)
       4: QSGW calculation
       5: tests generation of the energy-dependent self-energy
          Job 4 must be completed before running this test
```

Job script on henry cluster for CH3-NH3-PbIr3

The next slides show the QSGW cycle for a calculation of a newly discovered solar cell material, CH_3-NH_3-$PbIr_3$

```
#!/bin/tcsh
# Number of processors to be allocated by the queueing system:
#$ -pe orte  12
# queue to submit to
#$ -q all.q
# Put batch output files the local directory
#$ -cwd
# Export environmental variables to the nodes
#$ -V
# This environment variable must be set for nodes to communicate
setenv OMPI_MCA_btl_tcp_if_exclude lo,virbr0
# Start execution in this directory
cd /home/ms4/work/nh3ch3-pbi3

# Execution line
lmgwsc --wt --openmp=12 --code2 --sym -maxit=10 --insul=25 --getsigp --tol=2e-5 --save= basp.float1.sep12 nh3ch3
```

The critical steps have been parallelized with OMP: henry has 12 processors on a node so using 12 processors is natural. This is a moderately large cell for QSGW. CPU time scales as N^4 and memory requirements as N^3.

Job on henry cluster for CH3-NH3-PbI3

lmgwsc calls lmgw to do the actual GW calculation

```
lmgwsc: starting job Sun Oct 13 18:22:46 BST 2013, running version fpgw sep12 lmf LM 7.10 FP 7.10
    lmgwsc : starting iteration 0 of 10
    lmgwsc : invoking /home/ms4/bin/lmgw  --scrho --sc:sym --sc --wt --openmp=12 --code2 --getsigp --lmv6=no --insul=25 nh3ch3
    lmgw : extracting SIGP_EMAX ... /home/ms4/bin/lmf --show --quit=show --no-iactive   -vles=11 nh3ch3 | grep SIGP_EMAX | tail
found emax = 2.5 ... use emaxs = 3
          copying file GWinput to GWinput~
          writing file GWinput created from GWinput~, changing ESIGCUT line to 3
    lmgw : removing multitet ... nothing to change
    lmgw : removing EXonly ... nothing to change
    lmgw : extracting GW_NKABC= ... from ctrl file
          set nkabc =  (`/home/ms4/bin/lmfgwd --gwcode=2 --show --quit=show --no-iactive   -vles=11 nh3ch3 | grep GW_NKABC | ta
          extracted raw GW_NKABC as :   1, -- 3
          using for nkabc: 3 3 3
          copying file GWinput to GWinput~
          writing file GWinput created from GWinput~, changing n1n2n3 line to 3  3  3
          rm -f mixm.nh3ch3
    lmgw  18:22:48 : invoking         /home/ms4/bin/lmf --no-iactive   -vles=11 nh3ch3 >llmf
        setenv OMP_NUM_THREADS 12
    lmgw  18:24:27 : invoking echo 0 |/home/ms4/bin/lmfgwd --gwcode=2 --no-iactive   -vles=11 nh3ch3 >llmfgw00
    lmgw  18:24:29 : invoking echo 1 |/home/ms4/bin/code2/qg4gw >lqg4gw
    lmgw  18:24:29 : invoking echo 1 |/home/ms4/bin/lmfgwd --gwcode=2 --no-iactive   -vles=11 nh3ch3 >llmfgw01 ... 15.4m (0.3h)
          rm -f v_xc evec
          ln -s vxc.nh3ch3 v_xc
          ln -s evec.nh3ch3 evec
          ln -s evec.nh3ch3 evec0
 OK! lmfgwd mode=1
```

The initial stages of the script synchronize some tags in GWinput with corresponding tags in the ctrl file. This only takes place if switch --getsigp is used

Job on henry cluster for CH3-NH3-PbIr3

The density is made self-consistent for *fixed* $\Sigma' - V_{xc}^{\mathrm{LDA}}$.
Initially there is no self-energy, so the starting density will be the self-consistent LDA density

```
          copying file GWinput to GWinput~
          writing file GWinput created from GWinput~, changing n1n2n3 line to 3  3  3
          rm -f mixm.nh3ch3
   lmgw  18:22:48 : invoking         /home/ms4/bin/lmf --no-iactive   -vles=11 nh3ch3 >llmf
       setenv OMP_NUM_THREADS 12
   lmgw  18:24:27 : invoking echo 0 |/home/ms4/bin/lmfgwd --gwcode=2 --no-iactive   -vles=11 nh3ch3 >llmfgw00
   lmgw  18:24:29 : invoking echo 1 |/home/ms4/bin/code2/qg4gw >lqg4gw
   lmgw  18:24:29 : invoking echo 1 |/home/ms4/bin/lmfgwd --gwcode=2 --no-iactive   -vles=11 nh3ch3 >llmfgw01 ... 15.4m (0.3h)
          rm -f v_xc evec
          ln -s vxc.nh3ch3 v_xc
          ln -s evec.nh3ch3 evec
          ln -s evec.nh3ch3 evec0
OK! lmfgwd mode=1
```

Next follows the handshaking between lmfgwd and the GW codes.

Each executable writes to a different output file. The outputs for lmfgwd are llmfgw00 and llmfgw01.
If the code completes successfully it returns 'OK! ...' to stderr. If it does not the script proceeds anyway (a bug).

Job on henry cluster for CH3-NH3-PbIr3

Next codes translate the output of lmfgwd into internal formats specific to the GW codes.

```
lmgw  18:39:55 : invoking echo nh3ch3|/home/ms4/bin/lmf2gw_2 >llmf2gw
lmgw  18:40:02 : invoking echo 0 |/home/ms4/bin/code2/rdata4gw_v2 >lrdata4gw
lmgw  18:40:43 : invoking echo 1|/home/ms4/bin/code2/heftet   >leftet
lmgw  18:40:43 : invoking echo 1|/home/ms4/bin/code2/hchknw >lchknw
```

Next the Fermi level is determined using the same tetrahedron integrator as used by the GW codes

```
lmgw  18:39:55 : invoking echo nh3ch3|/home/ms4/bin/lmf2gw_2 >llmf2gw
lmgw  18:40:02 : invoking echo 0 |/home/ms4/bin/code2/rdata4gw_v2 >lrdata4gw
lmgw  18:40:43 : invoking echo 1|/home/ms4/bin/code2/heftet   >leftet
lmgw  18:40:43 : invoking echo 1|/home/ms4/bin/code2/hchknw >lchknw
```

Job on henry cluster for CH3-NH3-PbIr3

Now the Hartree potential, exchange potential, screening, and correlation self-energy are calculated for the core

```
lmgw  18:40:44 : invoking echo 3|/home/ms4/bin/code2/hbasfp0 >lbasC
lmgw  18:40:49 : invoking echo 0|/home/ms4/bin/code2/hvccfp0 >lvccC ... 50.8m
lmgw  19:31:38 : invoking echo 3|/home/ms4/bin/code2/hsfp0_sc_om >lsxC ... 32.
```

These quantities are evaluated as matrix elements of the product basis, so the product basis is generated first.

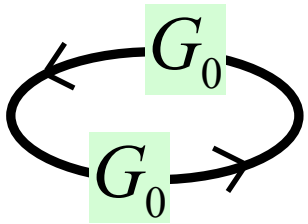
The same quantities are made for the valence electrons:

```
lmgw  20:03:57 : invoking echo 0|/home/ms4/bin/code2/hbasfp0 >lbas
lmgw  20:04:00 : invoking echo 0|/home/ms4/bin/code2/hvccfp0 >lvcc ... 37.
lmgw  20:41:30 : invoking echo 1|/home/ms4/bin/code2/hsfp0_sc_om >lsx ... 
```

Job on henry cluster for CH3-NH3-PbIr3

Next comes an expensive step: matrix elements of the polarization in the RPA

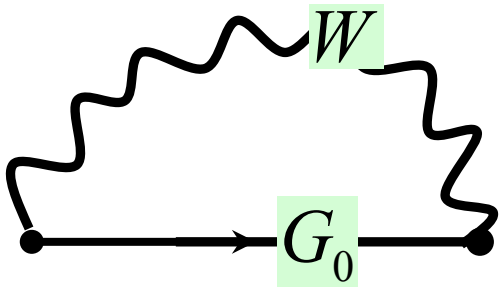
$$\Pi = -i\, G_0 \times G_0$$




```
invoking echo 11|/home/ms4/bin/code2/hx0fp0_sc_om >lx0 ... 98.5m
```

Now we can calculate matrix elements of the self-energy $\Sigma(\mathbf{r},\mathbf{r}',\mathbf{q},\omega)$:

$$\Sigma = i\, G_0 W$$




```
invoking echo 2|/home/ms4/bin/code2/hsfp0_sc_om  >lsc ... 719.9m
```

For a one-shot GW calculation, calculating Π is the most costly step. For a QSGW calculation, obtaining Σ is usually the most expensive, as it is in this case.

Job on henry cluster for CH3-NH3-PbIr3

Next the static $\Sigma' - V_{xc}^{\mathrm{LDA}}$ is made and written to file sigm.

```
invoking echo 25|/home/ms4/bin/code2/hqpe_sc >lqpe ... 0.1m
```

A soft link is made so the lmf code can read it:

```
ln -s -f sigm sigm.nh3ch3
```

sigm is symmetrized with the available symmetry operations

```
/home/ms4/bin/lmf --symsig --no-iactive --wsig --mixsig=1   -vles=11 nh3ch3 >llmf-sym
```

I created a file 'maxit' with contents 0 while this job was running. lmgwsc detected the file and reset the maximum iterations count to 0 and stops without further iterations:

```
lmgwsc : found file maxit ... resetting maxit to 0
lmgwsc : completed iteration 0 of 0  more=F Mon Oct 14 10:40:16 BST 2013 elapsed wall time 977.5m
```

Job on henry cluster for CH3-NH3-PbIr3

Finally come some cleanup steps. These steps are not necessary but are designed to package the key files neatly.

```
savegwfiles --job="/home/ms4/bin/lmgwsc --wt ..." --mkdir nh3ch3 basp.float1.sep12
gw-extract-prodbas-and-time-from-output > basp.float1.sep12/timings
savegw : mkdir -p basp.float1.sep12
savegw: the following files were copied to basp.float1.sep12
job version llmfgw00 GWinput switches-for-lm ctrl.preprocessed.nh3ch3 save.nh3ch3 QP
```

Script savegwfiles will run if you invoke lmgwsc with --save= dir-name. It copies essential files to dir-name.

Script gw-extract-prodbas-and-time-from-output prints information about the size of the product basis, the eigenfunction basis, and timings.

Invoking lmgwsc

lmgwsc was invoked as follows for this job:

```
lmgwsc --wt --openmp=12 --code2 --sym -maxit=10 --insul=25 \
       --getsigp --tol=2e-5 --save= basp.float1.sep12 nh3ch3
```

--wt tells lmgwsc to print out wall times at each step.
--openmp=12 runs lmgwsc with OPENMP, 12 processors
--code2 uses the new code (starting from Sep12)
--sym tells lmgwsc to symmetrize the generated sigma
-maxit=10 tells lmgwsc to do at most 10 QSGW cycles
--insul=25 an insulator with 25 occupied states
--getsigp extract selected data (eg n1n2n3) from the ctrl file and replace data in GWinput
--tol=2e-5 run until the RMS change in sigma < tol
--save=fn is explained on the previous slide.
nh3ch3 the ctrl file is named ctrl.nh3ch3 .

Passing command line arguments to lmf

It is convenient for lmgwsc to pass command line arguments to lmf. This is done with either of the following options:

1. Any variable –vxx=# declared on the lmgwsc command-line will be passed as a command line argument when lmgwsc runs lmf or lmfgwd .
2. The contents of file switches-for-lm will automatically be included as command-line arguments to lmf.

For example, suppose switches-for-lm contains

```
-vabc=11 --rhopos
```

When lmgwsc is invoked as follows it calls lmf as shown:

```
lmgwsc --wt  --insul=4 --tol=2e-5 --maxit=2 -vxx=3 si2
...
/home/ms4/bin/lmf --no-iactive  -vxx=3 abc=11 --rhopos si2 >llmf
```

A simple example

The following is a quickly executing demo+check of the GW codes

gwd/test/test.gwd si2

Job 2 shows the results of a 1-shot calculation.

Job 4 shows the results of a QSGW calculation.

The following test is much slower but it demonstrates a QSGW for Fe, a spin polarized metal.

gwd/test/test.gwd fe 4 5

Job 4 does one iteration of a QSGW calculation. If you have a new installation you should run this check to ensure your code is working properly.

Job 5 illustrates how to make the spectral functions, from which you can make the interaction density-of-states, and also simulate photoemission experiments.

Manipulating the QS*GW* self-energy file

There is a 'sigma' editor that enables you to manipulate the QS*GW* self energy in various ways. See doc/gw.html . Test

gwd/test/test.gwd si

cp gwd/test/si/{sigma,syml}.si .

lmf si -vsig=12 --rs=1,2 --rsig:ascii -vnit=5 --iactiv -vmetal=5

lmf si -vsig=12 --rs=1,2 --rsig:ascii -vnit=5 --iactiv --band:fn=syml

uses the results of a prior QS*GW* calculation to generate the energy bands of Si.

It tells you how to run a variety of other tests, e.g. uses of the sigma editor.