OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Building and Running LSMS

Markus Eisenbach

February 7, 2022

ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Getting LSMS

Obtaining the LSMS source code:
`https://github.com/mstsuite/lsms`

```
mkdir lsms
cd lsms
git clone https://github.com/mstsuite/lsms.git .
cd ..
```

Dependencies:

- Fortran and C++
- Cmake
- HDF5
- BLAS and LAPACK
- MPI

Lua and libxc are included or can be linked if already installed

Building LSMS

Create a build directory for Cmake outside of the source tree:

```
mkdir lsms-build
cd lsms-build
```

There are `Cmake` toolchain file examples available in `lsms/toolchain`. Also see `README.md` for more explanations.

```
cmake \
-DCMAKE_TOOLCHAIN_FILE=../lsms/toolchain/macos-gfortran.cmake \
../lsms/root
```

This will crate the Makefile.

Building LSMS

Now `lsms` can be built using

```
make
```

This will generate the lsms executable:

```
lsms-build/lsms/bin/lsms
```

and the Wang-Landau LSMS executable as

```
lsms-build/lsms/bin/wl-lsms
```

Running LSMS

Examples for running `lsms` can be found in "`lsms/Test`"

We will need two types of input files:

- The main input file that describes the calculation parameters and atom and crystal configuration. The default name for this input file is '`i_lsms`'.
- Starting potentials for each atom.

If the binaries are found in `$LSMS_PATH`, we can run `lsms` using MPI as

```
mpirun -np ⟨number of MPI ranks⟩ $LSMS_PATH/lsms
i_lsms
```

where ⟨*number of MPI ranks*⟩ has to be equal or less than the number of atoms in the system

Input File

The input file (default name: `i_lsms`) is a script in the LUA language. (`lua.org`)
`lsms` will read the input values from LUA variables or use default values when appropriate.
Everything after `--` is considered a comment.

Input File
Iron Cobalt

```
systemid="FeCo"
system_title = "Iron-Cobalt test for LSMS 3"
```

`systemid` is used to construct potential filenames.
`system_title` is an arbitrary text to describe the system.

```
pot_in_type=1
pot_out_type=0
```

`pot_in_type` and texttt`pot_out_type` are the file formats for the potential files. (0: HDF5 binary file, one file for the whole system; 1: Text file, same format as in MST2, one file per atom type; -1: don't write an output file / try to generate the starting potentials from scratch)

Input File
Iron Cobalt

```
num_atoms=2
nspin=3
nscf=10
```

`num_atoms`: number of atoms in the simulation cell
`nspin`: how to treat magnetism (1: no spin polarization; 2: collinear spins; 3: non-collinear magnetism; most general)
`nscf`: maximum number of selfconsistent iterations

Input File
Iron Cobalt

```
mixing = { {quantity = "potential", algorithm = "broyden",
mixing_parameter = 0.05} }
numberOfMixQuantities = 0
for k,v in pairs(mixing) do
  numberOfMixQuantities = numberOfMixQuantities + 1
end
```

set how mixing of quantities during selfconsistency iterations it so be performed.
the last four lines are boilerplate to count the number of quantities to be mixed.

Input File
Iron Cobalt

```
energyContour = {npts=31, grid=2, ebot=-0.3, etop=0.0,
eitop=0.825, eibot=0.0025}
```

Contour for integration of the Green's function.
`npts`: number of grid points.
`ebot`: bottom of the contour. (`etop`=0 stipulates that the top is the Fermi energy)
`eitop` and `eibot`: imaginary part at the top and end points of the contour.

```
site_default = {lmax=3, rLIZ=13.5, rsteps={96.9, 97.9,
98.9, 99.9} }
```

`lmax`: angular momentum / expansion cutoff.
`rLIZ`: radius of the Local Interaction Zone.

Input File
Iron Cobalt

```
a = 5.218
bravais = {}
bravais[1]={a,0,0}
bravais[2]={0,a,0}
bravais[3]={0,0,a}
```

Define lattice constant *a* as an arbitrary variable names, not defined in `lsms`. The units are Bohr radii (as in MST2).
Set up the Supercell lattice.

```
site = {}
for i =1,num_atoms do site[i]={} end
```

Boilerplate to prepare the input of the atomic sites.

Input File

Iron Cobalt

```
site[1].pos={0,0,0}
site[1].evec={0,0,1}
site[1].pot_in_idx=0
site[1].atom="Fe"
site[1].Z=26
site[1].Zc=10
site[1].Zs=8
site[1].Zv=8
```

```
site[2].pos={0.5*a,0.5*a,0.5*a}
site[2].evec={0,0,1}
site[2].pot_in_idx=1
site[2].atom="Co"
site[2].Z=27
site[2].Zc=10
site[2].Zs=8
site[2].Zv=9
```

Main definition of the site occupations:

`pos`: the position of the atom in units of Bohr radii.

`evec`: direction of the spin quantization axis

`pot_in_idx`: potential input file (defaults to atom index-1)

`atom, Z, Zc, Zs, Zv`: atomic element name and number, number of core, semicore and valence electrons. `Z=Zc+Zs+Zv`

Input File
Iron Cobalt

```
- - set site defaults
for i =1,num_atoms do
  for k,v in pairs(site_default) do
    if(site[i][k]==nil) then site[i][k]=v end
  end
end
```

Final boilerplate to copy values defined in `site_default` into the atomic sites that have not defined them.

Potential Files

Iron Cobalt

`pot_in_type` determines format of the input potentials to use:

-1: Generate new starting potential. No potential file required (experimental!)

0: A single HDF5 file for all potentials: `v_`⟨*systemid*⟩ [*e.g.* `v_FeCo`] (mainly used to restart calculations)

1: Text format potential file. One for each atom type index: `v_`⟨*systemid*⟩`.`⟨*idx*⟩ [*e.g.* `v_FeCo.0`, `v_FeCo.1`]

`pot_out_type` determines format of the output potentials:

-1: Do not write any output potential.

0: A single HDF5 file for all potentials: `w_`⟨*systemid*⟩ [*e.g.* `w_FeCo`] (mainly used to restart calculations)

1: Text format potential file. One for each atom: `w_`⟨*systemid*⟩`.`⟨*idx*⟩ [*e.g.* `w_FeCo.0`, `w_FeCo.1`] For large systems this will generate many files!

Restarting Calculations

`lsms` will generate:

- output potential `w_*`
- restart input file `i_lsms.restart`

To restart:

```
cp w_FeCo v_FeCo
mpirun -np ⟨number of MPI ranks⟩ $LSMS_PATH/lsms
i_lsms.restart
```

Output Files
`stdout`

```
LSMS_3: Program started
Using 1 MPI processes
Reading input file 'i_lsms'
Loaded input file!
System information:
====================
Number of atoms : 2
Number of atomic types : 2
Performing Muffin-Tin (MT) calculation
...
```

Output Files

`stdout`

```
...
Band Energy = 8.561411927312962 Ry
Fermi Energy = 0.703165574806340 Ry
Total Energy = -5323.290350746572585 Ry
timeScfLoop[rank==0] = 1506.861875 sec
    number of iteration:10
timeScfLoop/iteration = 150.686188 sec
timeCalcChemPot[rank==0]/iteration = 0.000013 sec
timeCalcPotentialsAndMixing[rank==0]/iteration          =
0.005190 sec
timeBuildLIZandCommList[rank==0]: 0.000926 sec
FOM Scale = 361684992.000000
Energy Contour Points = 32
FOM = 2.40025e+06/sec
FOM * energyContourPoints = = 7.68081e+07/sec
```

Output Files

`k.out`

Convergence of the calculation:

```
 0 -5322.587937171199  0.752965  1.961606  0.0240507202
 1 -5323.217618607828  0.754935  2.713605  0.0185740399
 2 -5323.270518830965  0.754390  2.605249  0.0036581916
...
35 -5323.291125864639  0.702599  2.775391  0.0015410203
36 -5323.293215570445  0.701733  2.756462  0.0004177423
37 -5323.290350746573  0.703166  2.788327  0.0006690433
```

The `k.out` file has 5 columns:

1: iteration number
2: total energy (in Ry units)
3: Fermi energy
4: magnetic moment on atom 1 (in Bohr magnetons)
5: RMS error (convergence parameter)

Output Files

`info_evec_out`

```
-5323.2903507465725 8.5614119273129 0.7031655748063
26  0  0.0000  0.0000  0.0000  25.825567  2.788327  0.0000
0.0000 1.0000 ...
27  1  2.6995  2.6995  2.6995  27.174433  1.672012  0.0000
0.0000 1.0000 ...
```

First line: Total Energy, Band Energy, Fermi Energy

Then, one line for each atom site with 18 columns:

- 1: Atomic number
- 2: Site index (starting at 0)
- 3–5: site coordines (x y z)
- 6: Charge at site
- 7: Magnetic moment at site
- 8–10: Direction of magnetic moment
- 11–18: More information for magnetic calculations *e.g.* torque, desired moment direction *etc.*

Density of States

First perform a selfconsistent calculation as described before using the Gaussian complex integration contour.
Copy the restart file and converged potential

```
cp i_lsms.restart i_lsms.dos
cp w_⟨systemid⟩ v_⟨systemid⟩
```

Edit `i_lsms.dos` for dos calculations. Replace

```
lsmsMode="main"
```

with

```
lsmsMode="dos"
```

Density of States

In `i_lsms.dos` change the energy contour by setting

```
energyContour.grid=3
energyContour.npts= number of energy points
energyContour.ebot= start of DOS
energyContour.etop= end of DOS
energyContour.eibot=imaginary part
```

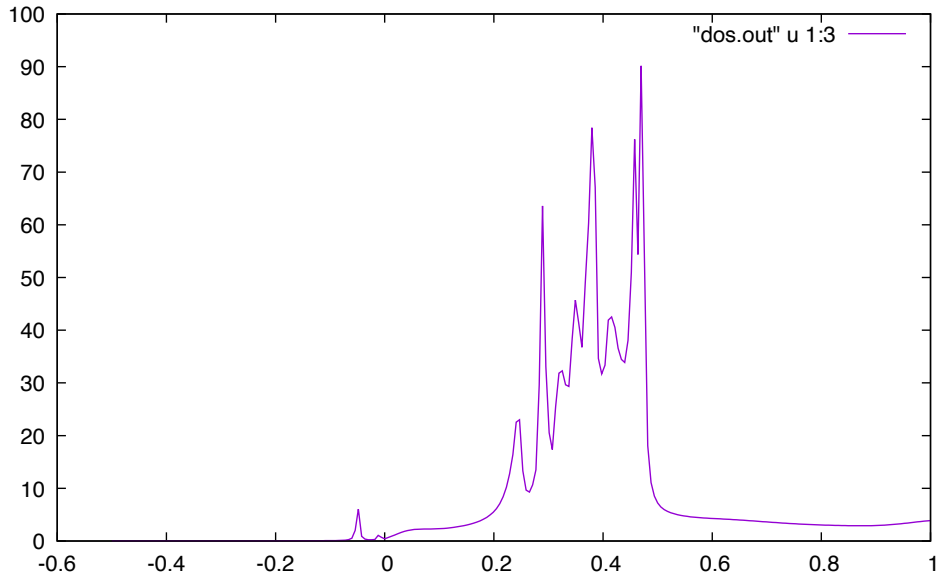
lsms will generate a file `dos.out` with three columns.

1: Real part of the energy
2: Imaginary part of the energy
3: Density of States

DOS output

`dos.out`

```
-0.500000 0.0025 0.0157603
-0.493976 0.0025 0.015958
...
0.993976 0.0025 3.81616
1.000000 0.0025 3.86047
```



Matsubara

Similarly to the DOS calculation we can write the Green's function at the Matsubara frequencies:

```
lsmsMode = "gf_out"
temperature = Temperature in Kelvin
```

And set the Matsubara energy contour that will generate points of the form

$$E_n = E_F + \pi i k_B T(2n + 1)$$

```
energyContour.grid=4
energyContour.npts= number of points
```

LSMS will generate files `greens_function_`⟨*atom index*⟩`.out`

Green's Function Output

The `greens_function_*.out` contains the local Green's function in blocks for each energy

```
Energy #index (complex energy)
0 0 0 -0.58697 -0.796096
...
0 15 15 -0.530649 -0.00268214
```

with the five columns in the energy block:

1: Spin index
2: $L = (l, m)$ index
3: $L' = (l', m')$ index
4: Real part of $G_{LL'}$
5: Imaginary part of $G_{LL'}$

Acknowledgements

- Basic Energy Science, Office of Science, US Department of Energy
- Advanced Scientific Computing Research, Office of Science, US Department of Energy
- Laboratory-Directed Research and Development (LDRD) Oak Ridge National Laboratory
- Oak Ridge Leadership Computing Facility (OLCF), Oak Ridge National Laboratory
- This research used resources of the Oak Ridge Leadership Computing Facility, which is supported by the Office of Science of the U. S. Department of Energy under Contract No. DE-AC05-00OR22725