

Notes on the implementation and test of unbalanced charge density initialization in CASTEP

Vincent Cocula and Emily A. Carter*
*Department of Chemistry and Biochemistry, Box 951569,
University of California, Los Angeles, California 90095-1569*
(Dated: June 30, 2003)

I. PRELIMINARY NOTES

- All the coding added and modified has been labeled by my name: `frax`. If a block was added, it was sandwiched by:

```
! frax: begin alteration
    bla bla bla
! frax: end alteration
```

- The code was successfully compiled on an Intel Linux cluster, using Intel Fortran Compiler v7.0.
- The code was not tested on other architectures.

II. MOTIVATION

In CASTEP, when one wants to do magnetic materials, one has to provide the net spin polarization of the entire model to be considered. This results in a situation where the initial charge and spin densities are evenly distributed over the atoms constituting the current model. The idea of this work is to implement a scheme where the user would have the freedom to set the atomic spin-polarizations when the charge densities have to be initialized. It could then be possible to set models where each atom present has a different initial spin-density, allowing among many other things, the possibility to do real antiferromagnetic phases in CASTEP.

III. IMPLEMENTATION

For the implementation of the unbalanced charge density initialization in CASTEP, we had to alter both the way the symmetry of the `current_model` was managed, and the charge density initialization routine. Almost all changes were made within the `CELL MODULE` and the `DENSITY MODULE`. All others were left unchanged - except a small modification in the `castep.f90` file, and in the `PARAMETERS MODULE`.

A. New input required

The only extra input required sits in the `seed.cell` file. If one wants a specific atom to have an initial spin polarization, it should be specified as an extra argument in the `POSITIONS_*` blocks. The new flag associated is `MAGMOM`, specifying the atomic spin-polarizations, $\frac{(N^{\uparrow} - N^{\downarrow})}{N_{tot}}$. For example, a Cr atom that would be set to have 4 up and 2 down electrons would have a polarization of $(4-2)/6=0.33$.

Here's an example of new `seed.cell` file, for an antiferromagnetic run of bcc Fe:

```
%BLOCK LATTICE_CART
  3.80    0.00    0.00
  0.00    3.80    0.00
```

*Correspondence to: eac@chem.ucla.edu; Visit: <http://www.chem.ucla.edu/carter>

```

    0.00    0.00    3.80
%ENDBLOCK LATTICE_CART

%BLOCK POSITIONS_FRAC
  Fe 0.00    0.00    0.00
  Fe 0.50    0.50    0.00 MAGMOM= 0.5
  Fe 0.50    0.00    0.50 MAGMOM : -0.2
  Fe 0.00    0.50    0.50 MAGMOM 0.5
%ENDBLOCK POSITIONS_FRAC

%BLOCK SPECIES_POT
  Fe fe.uspcc
%ENDBLOCK SPECIES_POT

KPOINTS_MP_GRID 10 10 10

SYMMETRY_GENERATE

```

As seen on this example, the format of the new input is very flexible: separation characters, tabulations, etc, are ignored. If `MAGMOM` was found at least one time in the list of atoms, the code will consider that an unbalanced charge density initialization must be performed. In this case, if no `MAGMOM` key has been found for a specific atom, it is automatically set to zero.

If we found one valid `MAGMOM` entry, a global logical variable (`atom_init_magmom`) is set to `TRUE`. This global variable will be used to check throughout the program if we have unbalanced initial densities or not.

B. Modifications in the CELL MODULE

Most of the modifications affect the `CELL MODULE`. This is the module where the new input is read, sorted and stored, and the symmetry operations checked/modified. Several new variables have been added:

- A new attribute in the `unit_cell` type: `initial_magnetic_moment`.
`real(kind=dp), dimension(:, :), pointer:: initial_magnetic_moment`
 Its dimensions are (number of atoms per species)x(number of species). It will be used to store the input values of the initial magnetic moment per atom.
- A new global variable: `atom_init_magmom`.
`logical, public:: atom_init_magmom`
 It is by default set to `.FALSE.`, and becomes `.TRUE.` only if the keyword `key_atom_magmom` is found. It is the flag that will be used throughout the program to determine whether we have unbalanced charge density initialization or not.

1. Modified subroutines

- subroutine `cell_allocate`
 Needed to allocate the `cell%initial_magnetic_moment` array and zero its content.
- subroutine `cell_deallocate`
 Needed to deallocate the `cell%initial_magnetic_moment` array once done.
- subroutine `cell_nullify`
 Needed to nullify the new pointers
- subroutine `cell_read`
 This is the subroutine that reads in the `seed.cell` file and sets a couple of relevant quantities.
 - We set the `atom_init_magmom` logical to `false` as default.
 - When the block lines are read from either `POSITIONS_FRAC` or `POSITIONS_ABS`, we try to read the optional `MAGMOM` value using a new subroutine (`cell_read_line_real`).
 - If the subroutine found a valid `MAGMOM` entry for block line being read, `atom_init_magmom` is set to `true` and

an unbalanced charge density initialization must be performed.

- The read `MAGMOM` values are then sorted and ordered accordingly when one sorts the atoms by species.
- If the user specified symmetry and kpoints informations, those need to be tested against the new magnetic species. The kpoints list(s) is (are) unfolded using user-provided symmetry operations, then those operations are reduced if a certain symmetry operation violates the initial magnetic moments set for the atoms. The kpoints list(s) is (are) then reduced using the newly reduced symmetry operations. This usually results in less symmetry operations and consequently more kpoints.
- We copy the new attribute `cell%initial_magnetic_moment` to all nodes using `comms_gcopy`.
- We copy the new variable `atom_init_magmom` to all nodes using `coms_gcopy`.

- subroutine `cell_reread`

This subroutine re-reads the kpoints informations given in the `seed.cell` file for `OPTICS`, `BS`, and `PHONON`. If we have unbalanced charge density we need to make sure those kpoints set respect the reduced symmetry operations. Therefor, we need to read again the provided symmetry operations, unfold those kpoints sets, reduce symmetry operations, and then reduce kpoints sets.

- subroutine `cell_setup_keywords`

Needed to include the new keyword `ikey_atom_magmom`

- subroutine `cell_copy`

Needed to include the new `initial_magnetic_moment` array when one copies the cell from one model to another.

- subroutine `cell_dump_cell`

Needed to save the cell attributes in file.

- subroutine `cell_restore_cell`

Needed to read back the cell attributes from file.

- subroutine `cell_generate_symmetry`

In the case we generate our symmetry operations (K290), and if `atom_init_magmom = .TRUE.`, we need to check whether the symmetry operations comply with the magnetic species. There is a call to the new subroutine `cell_find_related_magnetic_atoms`, right before the call to `cell_find_related_atoms`. Obviously.

- subroutine `cell_output`

The modifications output the new informations about the atoms if `MAGMOM` was found: initial spin polarizations, as well as number of corresponding up and down electrons.

2. Added subroutines

- subroutine `cell_find_related_magnetic_atoms`

This new subroutine is basically a copy of the `cell_find_related_atoms`, except that it returns in output, not the list of symmetry related atoms, but a new set of symmetry operations, adapted for the magnetic species. It takes each symmetry operations and check if the atoms found to be symmetry-related are actually of the same magnetic specie (.e.g. have same value of `cell%initial_magnetic_moment`). If not, the symmetry operation is bad and removed. As output is the new set of symmetry operations, along with the new number of symmetry operations.

- subroutine `cell_read_line_real`

This new subroutine is used to read the `MAGMOM` flags. The calling sequence is:

```
call cell_read_line_real (line, keyword, value, found)
```

The subroutine reads in the character string `line` to look for the character sequence `keyword`. If it is found, it then reads the corresponding `value` and set the logical variable `found` to `true`. The subroutine is quite general enough so that it could be used for other purposes. This specific one requires `value` to be `REAL`. But similar copies of the subroutine could read other types of data as well.

C. Modifications in the DENSITY MODULE

This is where the density initialization is done. We initialize using approximate atomic densities.

1. Modified subroutines

- subroutine `density_initialise_cell`

The subroutine is now subdivided in 2 parts, almost equivalent. Before anything is done, we test whether we have unbalanced charge density or not (through `atom_init_magmom`). If not, the old way of initializing is executed. If we do have `atom_init_magmom = .TRUE.` then we go on with the new way...

The trick is to initialize the charge density atom per atom, and not specie per specie. The old way does:

```
loop over species
  loop over ions in specie
    compute structure factor
  end loop over ions
  compute total charge density for this specie
end loop over species
```

The new way does a decoupling of the atoms, computing the structure factor for each ion, and then computing its initial charge density (along with its initial magnetic moment given in input). All the charges are summed up as we go along the ions:

```
loop over species
  loop over ions in specie
    compute structure factor
    compute atomic charge density for this ion
    compute spin density for this ion
    add atomic densities to total cell density
  end loop over ions
end loop over species
```

It is the sign of the initial magnetic moment that determines whether the corresponding atom should have a majority of up or down spin.

The subroutine now also includes tests to check if the input files are consistent one another:

- We first test if we are actually doing a spin-polarized calculation. If not, the presence of the `ATOM_MAGMOM` block makes no sense to begin with. The user is warned and we abort using `io_abort`.
- Once the charge density has been initialized, we check if the charge density generated using the provided atomic initial magnetizations make sense with the spin information in the `seed.param` file. We integrate the total charge and compare with the total number of electrons in the system, then we compute how many down-spin electrons we have and compare to the total cell spin-polarization given (through the keyword `spin`). If something is inconsistent, we warn the user, and exit using `io_abort`.

2. Added subroutines

None.

D. Modifications in the `PARAMETERS` module.

When we have unbalanced charge density initialization, we need to make sure we are using the density mixing scheme. An additional test is there required to make sure that `metals_method = 'DM'`. If not, an error is generated and we exit.

E. Modifications in `castep.f90`.

In order for the test in the `PARAMETERS` module to be possible, we need to pass the logical `atom_init_magmom` as argument to `parameters_read`.

IV. TESTS

In this section, we will give a short description of the various calculations conducted to debug and or test the new method. We will also give a more extensive study of the phase diagram of Fe, which presents an interesting case study for our method.

A. Debugging

All the calculations below are of a fcc Fe cell. This should not matter for the debugging purpose, but this particular system was chosen because it allows us to play with a little so that we can monitor how the system reacts. Unless explicitly mentioned, the input files are:

```
%BLOCK LATTICE_CART
  3.8000000000    0.0000000000    0.0000000000
  0.0000000000    3.8000000000    0.0000000000
  0.0000000000    0.0000000000    3.8000000000
%ENDBLOCK LATTICE_CART

%BLOCK POSITIONS_FRAC
  Fe 0.0000000000    0.0000000000    0.0000000000
  Fe 0.5000000000    0.5000000000    0.0000000000
  Fe 0.0000000000    0.5000000000    0.5000000000
  Fe 0.5000000000    0.0000000000    0.5000000000
%ENDBLOCK POSITIONS_FRAC

%BLOCK SPECIES_POT
  Fe fe.uspcc
%ENDBLOCK SPECIES_POT

KPOINTS_MP_GRID 8 8 8

SYMMETRY_GENERATE

for the seed.cell file, and the seed.param file is:

cut_off_energy    500.00
xc_functional      PBE
nbands            50
metals_method      DM
elec_energy_tol    0.0001
spin_polarized     TRUE
spin              16
```

The `fe.uspcc` potential is an USPP including core correction.

- First test: stupid but crucial, do we get the same answer if we use the original code and my modified code in the absence of the `ATOM_MAGMOM` keyword. We should.
 $E = -3463.6719$ for the original code
 $E = -3463.6719$ for my modified code
 total energies are fine.
- Second test: total spin of the cell is 16 (from `seed.param` file). Which means that each atom has 6 up electrons and 2 down electrons. If we set the initial magnetic moment to be $(6-2)/8 = 0.5$ for each atom, we should get the same initial density and answer as for the original code.
 This is in fact true.
- Third test: if we now set unbalanced charge density, but still do a ferromagnetic calculation and let the system relax, we should get same total energy and magnetic moment per atom than with the original code. The only difference is that in the former, some of the 48 symmetry operations of the fcc unit cell are going to be shut

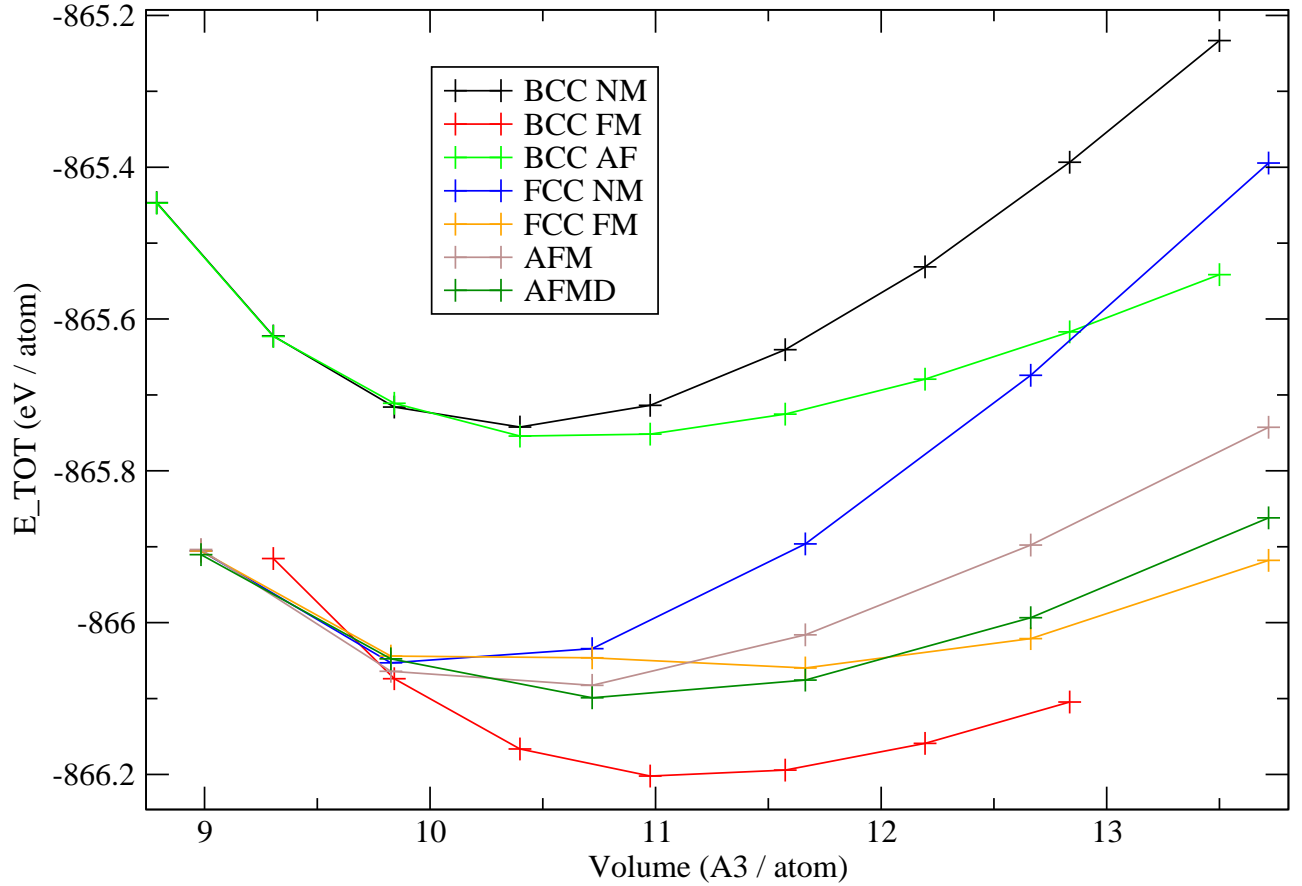


FIG. 1: Phase diagram of Fe

down. Two atoms were set to have 0.5 and two others to have 0.2 as their initial magnetization.

This is true also, we recover exactly the same total energy and magnetic moment per atom (which is now equal on all four atoms) than for the original code. The convergence was a little more troublesome. The charge mixing amplitude had to be reduced from 0.8 to 0.4. No surprises, since the number of symmetry operations was modified from the 48 usual ones for an fcc cell to only 16.

B. Real case study: phase diagram of Fe

We will now draw the phase diagram of Fe (no hcp phases included though). It is a well studied subject, and all the work here refers to Herper et al. PRB 60,p.3839(1999) - Ab initio full potential study of the structural and magnetic phase stability of iron. Fe presents also a very interesting and complex phase diagram, with competition between the different magnetic phases, and an important magnetovolume effect. We will be of course more particularly interested in the antiferromagnetic (AF) phases, and Fe has 3 of them, which offers a good and strong test of our new code. It has a bcc AF phase, simply noted AF, a fcc AF phase, where spins alternate by layers, and an AFMD phase with spin "domains". the AFM and AFMD phases are equivalent to the ones from Herper's paper.

FIGURE 1 shows the phase diagram of Fe. The theory used is PBE, although the USPP used is an LDA one. For the purpose of this study, it should be okay.

As we followed the geometrical description of Herper, the AFM cell is tetragonal containing 2 atoms, while the AFMD is tetragonal containing 4 atoms. Nevertheless, the AFM is equivalent to a fcc lattice with atomic spin orientations alternating layer by layer. We in fact conducted both sets of calculations, and saw that both models give exactly the same results, although the AFM phase was usually easier to converge. This was also another good test to show that our method works properly.

In terms of convergence issues, nothing has been problematic, for some AF calculations, the charge mixing amplitude was reduced from 0.8 to 0.5 or 0.2 and better convergence was achieved. The use of a higher smearing width was also

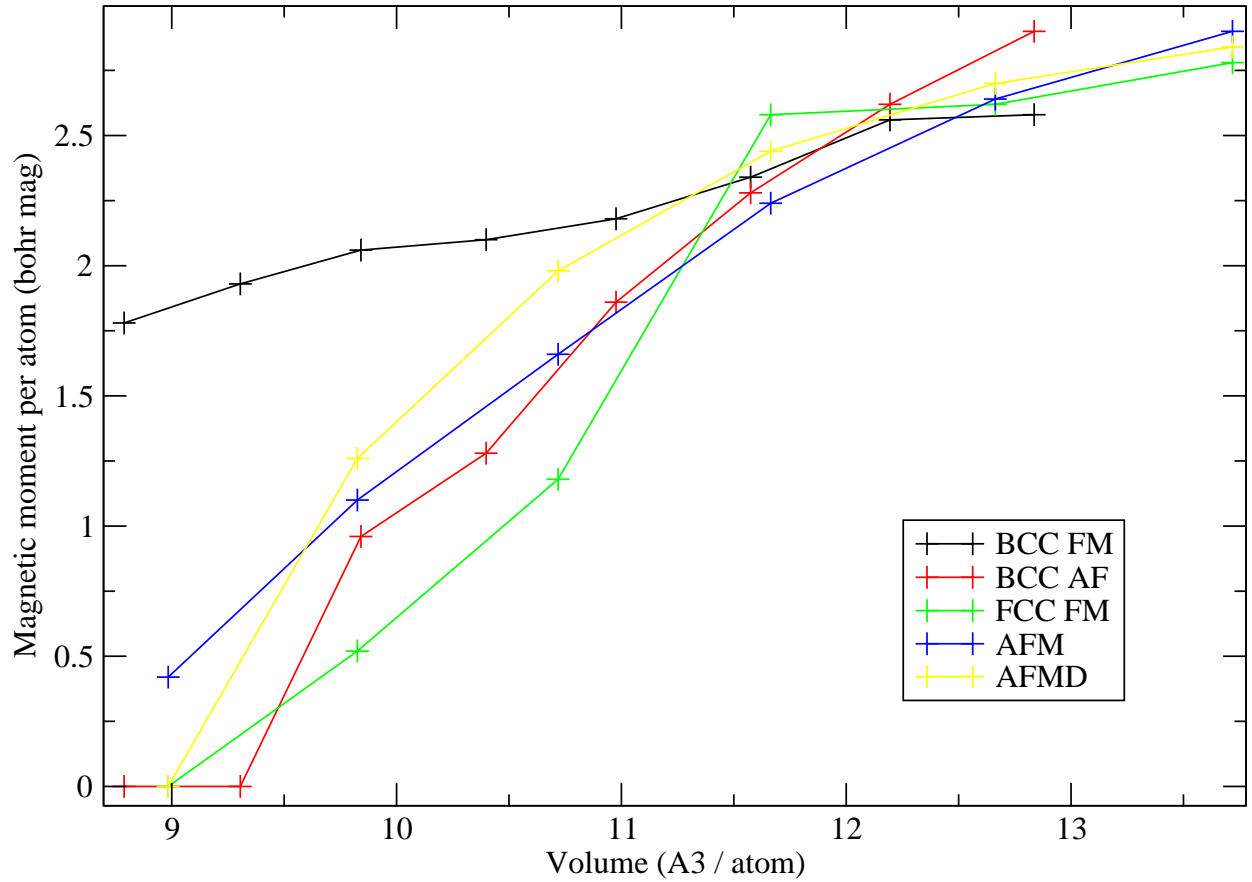


FIG. 2: Fe: dependence of the magnetic moment per atom on the cell volume for several magnetic phases.

making it easier (from 300K to about 1000K).

Concerning symmetry issues, the symmetry of the bcc cells was not needed to be changed as the K290 routine seem to give only the non-symmorphic space group. For fcc phases, as well as AFM and AFMD, symmetry was reduced and the number of k-points increased.

FIGURE 2 shows the evolution of the magnetic moment per atom when the volume increases. In general, for both the magnetic moment, and the phase ordering, our results show good agreement with the all-electron results of Herper. We see that the ground state is correctly predicted to be the bcc FM one. Other bcc phases show the branching between the NM and AF phases, which occurs before the minimum of the potential energy of the NM phase. This is also similar to all-electron results. For the fcc phases (or related phases, like the AFM and AFMD), we again recover the correct physics and relative stabilities. Branching between the NM and FM fcc phases, and very close in energy are the two AF phases: AFM and AFMD, with AFMD slightly more stable. This is all very similar once again with Herper's FLAPW results.

C. Yet another real case study: bulk Cr

Chromium is another good example of antiferromagnetic material, and we will this time use both LDA and PBE levels of theory for the study. Bulk Cr is cubic centered, with an experimental lattice parameter of 2.88 Å and its magnetic ground state is a spin-density wave (SDW) with a period of about 20 to 21 lattice parameters. We do not here study the SDW (which would consist of non-collinear spins in a big rod), but the simpler true AF bcc phase: two atoms in the cell, opposite spin polarization. The case of Cr is a very challenging one for theoretical work, and regarding the DFT, it is unclear whether the LSD or GC approximation is more suitable for its study. LSDA predicts a nonmagnetic (NM) ground state (which is wrong), but a reasonable magnetic moment for its antiferromagnetic (AF) phase. Gradient corrected functionals correctly predicts the ground state to be AF, but the magnetic moment per atom is ridiculously overestimated (apparently a lot more than the well-known tendency of

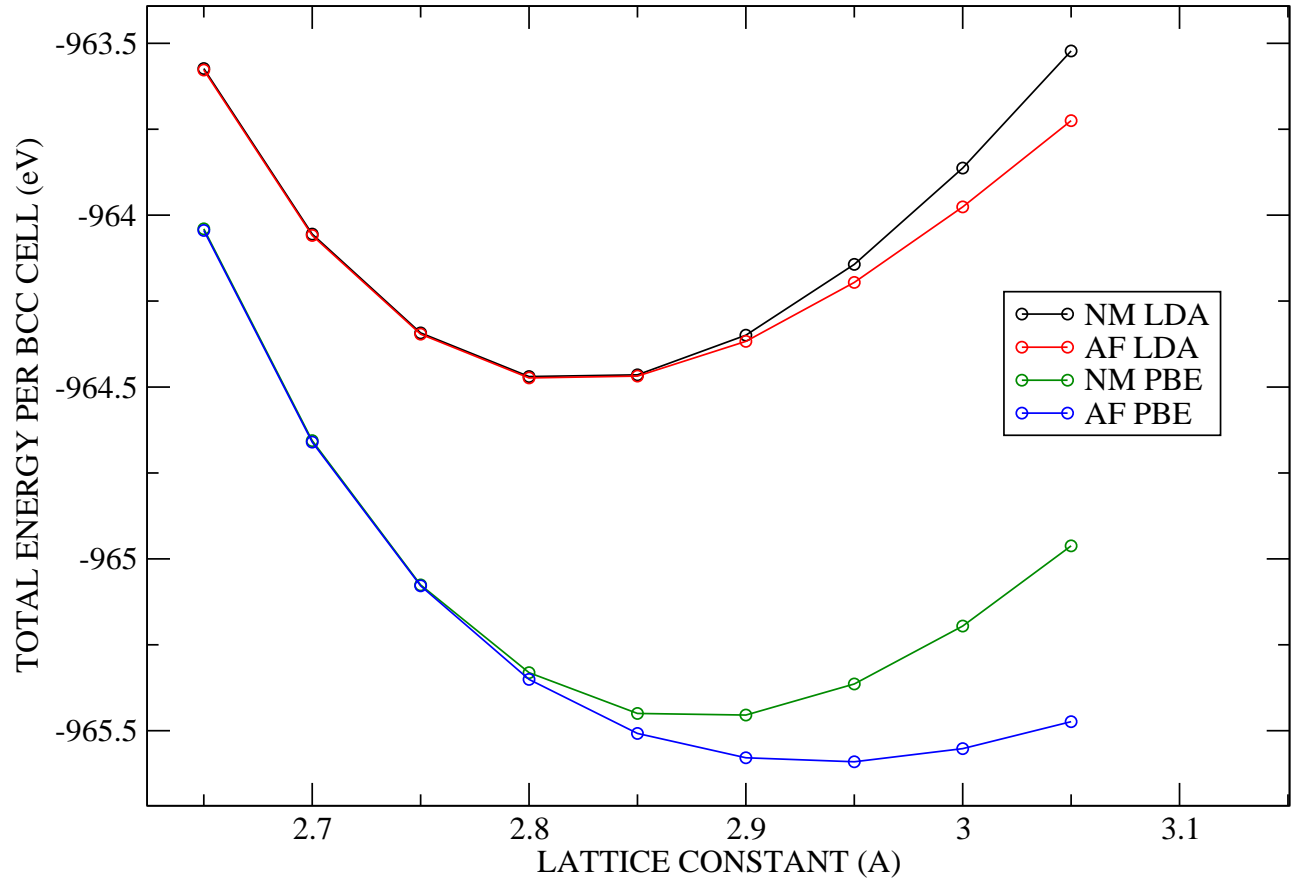


FIG. 3: LDA and PBE DFT NM and AF phases of bulk bcc Cr.

GGA's to overestimate magnetization).

For this study, we used both LDA and GGA levels, using an ultrasoft pseudopotential including non-linear correction (Cr.00.uspcc). Once again, this PsP has been generated using LDA, so inherent inconsistencies will show up for the PBE calculations, but we'll forget it for this study. A good paper about the system is the one of S. Cottenier et al., "What density-functional theory can tell us about the spin-density wave in Cr", J. Phys.: Condens. Matter, p.3275(2002).

FIGURE 3 shows the branching between the NM and AF phases at both LDA and GGA levels. As seen in all-electrons calculations, LDA predicts a NM ground state, while when the GGA is employed, the branching between both phase occurs a lot sooner, correctly predicting an AF ground state. Lattice constants are in fairly good agreement with AE results, although generally slightly overestimated. FIGURE 4 shows the evolution of the magnetic moment per atom as we increase the cell volume. Again, the trend is definitely the same as the one seen in AE results.

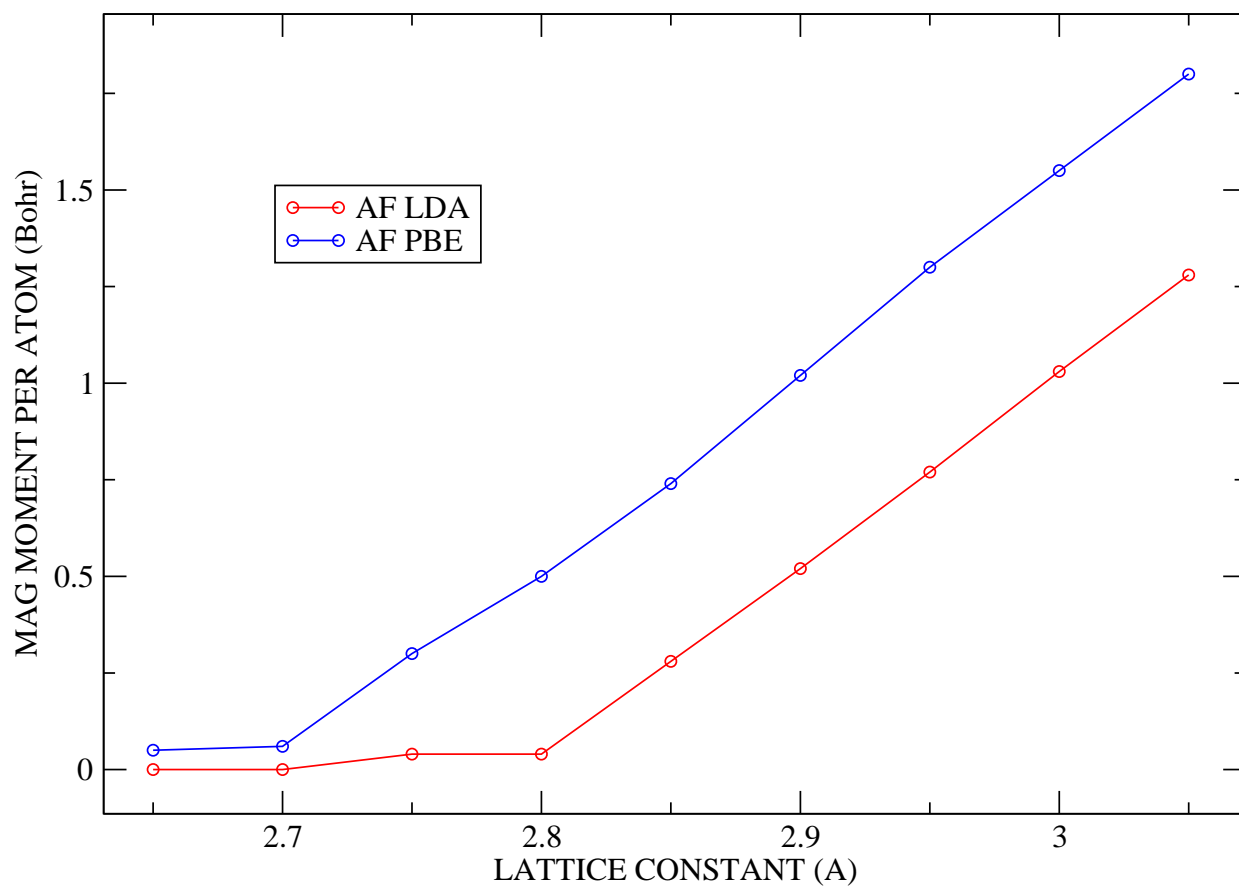


FIG. 4: Phase diagram of Fe