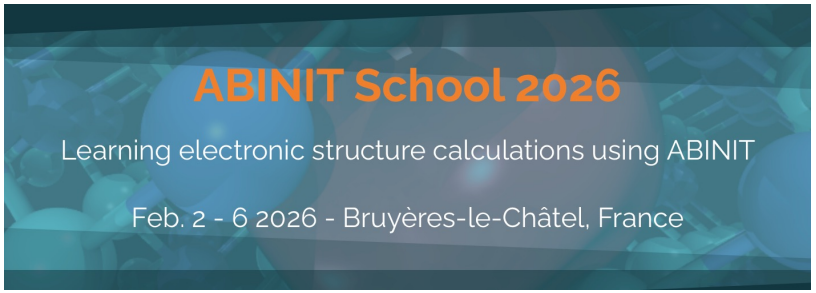
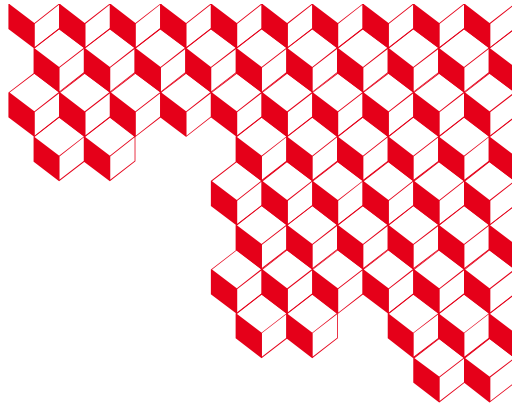




Tuning ABINIT

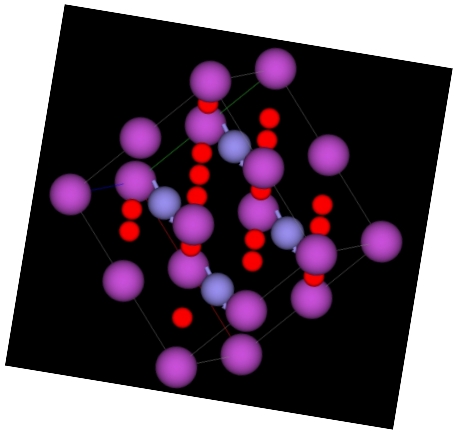
Hands-on session

Aluminium melting temperature

A first application of Ab Initio Molecular Dynamics (AIMD)

- **Objectives**
  - By following this step-by-step tutorial, you should experiment how to change the default settings of ABINIT to make it converge faster and better.

- **Test case: $BiFeO_3$**
  - For this *hands-on* session, the working system is a 20-atom cell of Perovskite $BiFeO_3$ in its (theoretical) cubic phase. Two oxygen vacancies were introduced in it and the cell was distorted;
    The final number of atoms is 18.
  - Introducing defects in the cell makes it more difficult to converge with ABINIT.



Preliminary information

- All the necessary files can be found in the **ABISCHOOL/handson_tuning/** directory.
- To run the examples in parallel with e.g. X MPI processes and Y openMP threads, possibly GPUs, use the attached `job.sub` file and edit it as follows:

```
#MSUB -E '--reservation=Formation-Abinit-gpu-20260205'
#MSUB --n=X          # Number of MPI processes
#MSUB --c=Y          # Number of openMP threads
ccc_msub job.sub
```

- The standard output of the application will be redirected to `abinit.log`.
- Don’t hesitate to activate the use of GPUs (remember : 1GPU = 1MPI process) For that you must use `gh200-bxi` partition.

First attempt

Initial convergence with all keywords set to default

- Run ABINIT with the initial **`ttuning.abi`** file.
- To perform the calculations in an acceptable time, it is preferable to use about 200 processors (50MPI x 4 threads is a good choice).
  Note that the automatic parallelization is activated: autoparal is on.
- Use the standard procedure to submit the calculation on the supercomputer (ccc_msub). You can use the attached `job.sub` submission file :

```
ccc_msub job.sub
```

- The run stops after 20 iterations without having really converged the electronic self-consistent cycle.

Now, let's tune the SCF cycle

Can we make it converge or reduce the number of iterations

- Now, you can play with various input parameters in order to evaluate their respective influence on the convergence rate.
- Before modifying a parameter in the input file, don't hesitate to read ABINIT documentation. See f.i. topics *SCFControl* (docs.abinit.org/topics/SCFControl) or *SCFAlgo* (docs.abinit.org/topics/SCFAlgorithms).
- Parameters you can play with:
  - The history size of the mixing scheme, **npulayit**.
  - The parameters of the dielectric matrix used to precondition the density residual: **diemix**, **diemac**.
  - The maximal number of iterations of the (iterative) diagonalization scheme: **nline**.
  - The number of non-self-consistent iterations, (number of restarts of the iterative algorithm): **nnsclo**.
  - The mixing scheme **iscf** (7=on the potential, 17=on the density).

What is the best compromise?

Do you succeed in making the code converge in less than 20 iterations?

In principle, with optimal settings, ABINIT should reach convergence for this system in less than 20 iterations...

Can we do more?

Is it possible to further improve this result?

- Let's try to play with the **number of bands.** Let's make an additional experiment.

  Increase slightly (10-20%) the **nband** parameter (setting `nband=125`)
  and try this new setting.

- Let's try to change the iterative diagonalization algorithm.
  We have used *LOBPCG* algorithm (minimization).
  Let's try *Chebysdhev filtering* algorithm (subspace filtering).
  Set **`wfoptalg`**`=111` and run ABINIT again.

Do you improve the time to solution?

What about parallelism?

So far, we have not changed the parameters dedicated to parallelism.

- Let's try now to run the code using different workload distributions over processors. Modifying `ttuning.abi` and `job.sub` files, try other way to allocate the computer resources.
- Parameters you can play with:
  - The number of MPI processes dedicated to k-points (try to maximize it), **npkpt**.
  - The number of MPI processes dedicated to bands (try to maximize it), **npband**.
  - If you use LOBPCG algorithm, the number (and size) of "blocks": **nblock_lobpcg**.
  - The number of OpenMP THREADS per MPI process: **OMP_NUM_THREADS** environment variable
- And now, on a modern supercomputer — GPUs!
  Use the keyword **gpu_option** to enable GPU support. The *Inti* cluster has 4 GPUs per node, so in our case, you should launch 4 MPI processes, each linked to one GPU. *Do you observe a speedup?*
  Speedup is quite significant, isn't it? With such acceleration, you can afford to be more demanding with other parameters. For instance, you can ask the diagonalization algorithm to converge more tightly by increasing **nline** (for example, `nline = 10`).

Note: with the GPUs, you must impose « #MSUB –c 72 »

Want to prioritize speed over precision?

Is it possible to further improve this result?

- Decrease the value of the input parameter **accuracy**.
  Try to set **accuracy=3**, then **accuracy=2** and finally **accuracy=1**.
- You should notice a speed-up of the calculation... and a change in the significant digits (look for instance at the value of the energy `ETOT`).

Is this loss of precision acceptable?

- To evaluate this, you could perform a structural relaxation of the simulation cell.
  Add the following line in the input file and run ABINIT:

```
geoopt « lbfgs » ntime 20
```

Does the relaxation run in the same way with accuracy=1 or accuracy=4?