# FYS-4096 Computational Physics: Exercise 8

Ilkka Kylänpää

Week 11

**General:**

- You should have access to `ex8_help` on teacher's repo.
- There you will find, e.g., files `hartree_1d.py`, and `machines.py` that will most likely be useful in this exercise.
- You also need a personal username and password to access the CSC supercomputing facilities. You should find these at Moodle from your feedbacks of grading related to CSC. If not, contact the teacher.
- For general future use the Quantum Espresso (pw.x) input file description might become useful.
- General information on Puhti at CSC might also be of use.
- **Deadline always before noon on Monday the following week!** That is, for these problems before noon on Monday March 22.

**Problem 1: 1D harmonic quantum dot with $N$ interacting electrons (revisited)**

- Extend your Hartree code to also include the possibility of Hartree-Fock solution with proper commenting.
- Compare the Hartree and Hartree-Fock energetics and electron densities for $N = 6$, $S = 0$ case.
- Plot your comparison and include the energetics in the figure.

**Problem 2: Quantum Espresso installation to CSC (with QMCPACK patch)**

- Open a Linux terminal.
- Write `ssh username@puhti.csc.fi` and press enter (as username you need to use your own csc student username).
- It asks for a password, which you should know already at this point.
- Once logged in write `csc-workspaces` and press enter. This should show available work-spaces. The `home` is for general files and setups, `/projappl/...` is for installing applications etc., `/scratch/...` is for maintaining and running projects.
- Go to your projappl directory: `cd /projappl/project_...`
- Create in it a folder with your username, i.e., `mkdir student...`
- Change the folder rights: `chmod g-rwx student...`
- Then go to the folder you just created.

- Next clone QMCPACK by writing
  `git clone https://github.com/QMCPACK/qmcpack.git`
- Go to external codes and there to Quantum Espresso, i.e.,
  `cd qmcpack/external_codes/quantum_espresso`
- There write `./download_and_patch_qe6.7.0.sh` (then you need to wait a bit)
- Move the qe-6.7.0 folder to same directory as the `qmcpack` folder, i.e., `mv qe-6.7.0 ../../../.`
- Go in to the moved folder qe-6.7.0 that now is located at `/projappl/project_.../student...`
- Load HDF5 modules as `module load hdf5` and see list of your currently loaded modules with `module list`, which should show something like

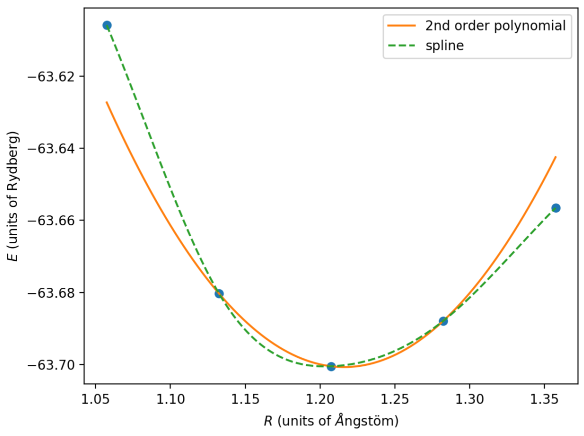
  ```
  Currently Loaded Modules:
    1) intel/19.0.4     3) intel-mkl/2019.0.4   5) hdf5/1.10.4
    2) hpcx-mpi/2.4.0   4) StdEnv
  ```

- More information on modules at `https://docs.csc.fi/computing/modules/`
- Write `module show hdf5` in order to see the path for your hdf5 include and lib folders (the path given on the next bullet point should be present in the list).
- Run the command
  `./configure --with-hdf5=/appl/spack/install-tree/intel-19.0.4/hdf5-1.10.4-6yrbnf`
- The previous command creates a configuration file `make.inc`.
- Modify the line in `make.inc` by adding “-g -fopenmp -pthread” after the equal sign on the line where it reads “LDFLAGS = “. (Look for LDFLAGS in the file.)
- Now, compile Quantum Espresso using the command `make all`. This will take some time so wait or fine tune problem 1 while the compilation is progressing.
- After compilation has finished go to the `bin` folder by writing `cd bin`. Check that you find at least `pw.x` and `pw2qmcpack.x` in the folder.
- Write `pwd` and press enter. This gives you the path to the bin folder. Copy the path.
- Add the copied path to your `.bashrc` in your home directory, e.g., by using a text editor (emacs, nano, etc.).
  - First go to your home directory by writing `cd` and press enter.
  - Open the file `.bashrc`.
  - Add the path as `export PATH="copy_your_path_here:$PATH"`
- After this add also the path at `/projappl/project_.../student.../qmcpack/nexus/bin` to your PATH in .bashrc file.
- And add path `/projappl/project_.../student.../qmcpack/nexus/lib` to your PYTHONPATH in your .bashrc, e.g.,
  `echo 'export PYTHONPATH="copy_your_path_here:$PYTHONPATH"' >> .bashrc`
- Copy the `machines.py` from teacher’s help files to Nexus library folder. (This replaces the one there already exists.)

- Write `source .bashrc` in order to activate the changes. (For some reason you might need to logout and log back in.)
- Next, go to your work directory, i.e, `cd /scratch/project...`. Create your own forlder `mkdir student...` and modify the rights of your folder `chmod g-rwx student...`
- **Include your CSC `.bashrc` file as `dot_bashrc.txt` into your git return.**

**Problem 3: Potential Energy Surface of $O_2$**

- Calculation of the potential energy surface of $O_2$ near the equilibrium distance using Quantum Espresso you built in Problem 2.
- Consider only a few points (e.g., 5 points) relatively close to the experimental equilibrium value 1.2074Å. Use for example `linspace(d-0.15,d+0.15,5)` grid, in which $d$ is the experimental equilibrium value.
- The runs should be performed within a single workflow using Nexus!
- The converged energies can be found in the output file on the line starting with symbol "! ", e.g., as `grep '!   ' scf.out`.
- The $O_2$ help files will most likely be helpful.
- Plot the calculated points together with a spline fit over the whole range, as well as a parabolic fit close to the minimum (using numpy's polyfit). You should get something like



- **Include your workflow file as `problem3.py` into your git return along with figures.**
- **How and where! Read this carefully through before beginning to do.**:
  - Go to puhti.csc.fi, i.e, `ssh username@puhti.csc.fi`
  - **Go to your work directory, i.e,** `cd /scratch/project.../student...`
  - Once you are in your work directory make a folder for this problem.
  - In addition to modules loaded in problem 2, load python-env, i.e., `module load python-env`.
  - Look into the help files, copy the $O_2$ related files into your new folder, and test with command `./o2_flow.py`. (If for some reason the file would not be executable this way, make it such by "chmod +x o2_flow.py".)

  - This should generate the input files and demonstrate what would have been done (due to the flag `generate_only=1`), but it really only went through the current workflow without submitting any jobs.
  - Test also the command `./o2_flow.py --status_only`, which shows you the status of the current workflow.
  - Once you are truly confident that the workflow is working as it should (by testing in the generate only mode), then you can make that flag to zero. Remember also to delete (or rename) the current "runs" folder, since otherwise Nexus assumes the project is already completed.
  - Notice that you can make a list of "scf" objects, and the list can be passed to the `run_project` function. The path is the only thing you need to change for the different scf objects.
  - (Actually objects only need to created, so what arguments you are passing to the run_project function should not matter.)
  - Also notice that once "./o2_flow.py" has submitted the jobs it keeps on analyzing the progress. You can stop this by "Ctrl-c". Next time you execute "./o2_flow.py" it will continue the progress sweep. That is, you do not need to keep it running all the time.
  - You can see if you have any jobs in queue or running by command
    `squeue -u your_csc_username`
  - You can cancel a job, e.g., by using the number value of your jobs "JOBID" as `scancel job_id`

**Problem 4: Converging energy cut-off and k-grid for diamond**

- **This also needs to be performed at CSC in your own work folder.**
- Make properly named folder for it into your work directory at /scratch/...
- Modify the diamond.py file in order to get it working (see the o2_wflow.py for details). You also need the machine_configs.py from problem 3.

a The energy cut-off in plane wave codes describes the amount of basis functions used in the calculations. Larger value indicates more accuracy, but slower calculations. In general, we want to get accurate enough results with feasible amount of computer time. (Notice that the softer the pseudopotential the smaller energy cut-off is needed.)

  - In Quantum Espresso the energy cut-off is given as `ecutwfc`. Make a single workflow that considers the convergence with respect to the energy cut-off. Use, for example, values 100, 150, 200, and 250 (Rydbergs), and k-grid of (2,2,2).

b The k-grid in solid state calculations describes the repetition of the simulation cell. That is, the larger the k-grid the closer is the thermodynamic limit. However, again, larger k-grid implies longer and more demanding calculations. To this end, by considering the k-grid convergence you can balance between accuracy and computer time.

  - In Quantum Espresso (using the Nexus workflow tool) the k-grid is specified by `kgrid` variable. Make a single workflow that considers the convergence with respect to the k-grid. Use, for example, values (1,1,1), (2,2,2), and (3,3,3).

- Make plots on how the total energy converges as a function of the energy cut-off and as a function of the k-grid.
- Do you think you have reached convergence or would you need to increase the energy cut-off and/or size of k-grid?
- Visualize the crystal structure using for example XCrySDen or VESTA, and make a png-figure of it.
- Notice that in addition to the "path" you now need to modify either the kgrid or ecutwfc in the scf object.
- You might need to increase the time, i.e., "minutes", for the largest k-grid. Also, you can test increasing the "cores", but do not exceed 10 cores, since we are using test accounts.

- Include the diamond.py, and the figures to your git repo. Not all the output files.

**Returning your exercise**

1. Make a new folder “exercise8” to your existing Computational Physics repo.
2. In folder “exercise8” write a comma separated list of problems you have solved, e.g., 1,2,3, into “solvedProblems.txt”.
3. Make sure all required files are under version control and push them to GitLab.
4. Push your commits to GitLab before noon on Monday March 22:
   ```
   git push --all && git push --tags
   ```