Parallelism on images, the string method

String method for the computation of minimum energy paths, in parallel.

This tutorial aims at showing how to perform a calculation of a minimum energy path (MEP) using the string method.

You will learn how to run the string method on a parallel architecture and what are the main input variables that govern convergence and numerical efficiency of the parallelism on “images”.

You are supposed to know already some basics of parallelism in ABINIT, explained in the tutorial A first introduction to ABINIT in parallel (../basepar), and ground state with plane waves (../paral_gspw).

Important

All the necessary input files to run the examples can be found in the *~abinit/tests/* directory where *~abinit* is the absolute path of the abinit top-level directory.

To execute the tutorials, you are supposed to create a working directory (`Work*`) and copy there the input files and the *files* file of the lesson.

The *files* file ending with *_x* (e.g. *tbase1_x.files*) **must be edited** every time you start to use a new input file. You will discover more about the *files* file in section 1.1 (../../guide/abinit#intro) of the help file.

To make things easier, we suggest to define some handy environment variables by executing the following lines in the terminal:

```
export ABI_HOME=Replace_with_the_absolute_path_to_the_abinit_top_level_dir

export ABI_TESTS=$ABI_HOME/tests/

export ABI_TUTORIAL=$ABI_TESTS/tutorial/           # Files for base1-2-3-4, GW ...
export ABI_TUTORESPFN=$ABI_TESTS/tutorespfn/       # Files specific to DFPT tutorials.
export ABI_TUTOPARAL=$ABI_TESTS/tutoparal/         # Tutorials about parallel version
export ABI_TUTOPLUGS=$ABI_TESTS/tutoplugs/         # Examples using external libraries.
export ABI_PSPDIR=$ABI_TESTS/Psps_for_tests/       # Pseudos used in examples.

export PATH=$ABI_HOME/src/98_main/:$PATH
```

The examples in this tutorial will use these shell variables so that one can easily copy and paste the code snippets into the terminal (**remember to set ABI_HOME first!**)

The last line adds the directory containing the executables to your PATH so that one can invoke the code by simply typing *abinit* in the terminal instead of providing the absolute path.

Finally, to run the examples in parallel with e.g. 2 MPI processes, use *mpirun* (*mpiexec*) and the syntax:

```
mpirun -n 2 abinit < files_file > log 2> err
```

The standard output of the application is redirected to `log` while `err` collects the standard error (runtime error messages, if any, are written here).

This tutorial should take about 1.5 hour and requires to have at least a 200 CPU cores parallel computer.

1 Summary of the String Method

The string method *[Weinan2002] (../../theory/bibliography#weinan2002)* is an algorithm that allows the computation of a Minimum Energy Path (MEP) between an initial (*i*) and a final (*f*) configuration. It is inspired from the Nudge Elastic Band (NEB) method. A chain of configurations joining (*i*) to (*f*) is progressively driven to the MEP using an iterative procedure in which each iteration consists of two steps:

1. **Evolution step**: the images are moved following the atomic forces.
2. **Reparametrization step**: the images are equally redistributed along the string.

The algorithm presently implemented in ABINIT is the so-called “simplified string method” *[Weinan2007] (../../theory/bibliography#weinan2007)*. It has been designed for the sampling of smooth energy landscapes.

Before continuing you might work in a different subdirectory as for the other tutorials. Why not work_paral_string?

Important

In what follows, the names of files are mentioned as if you were in this subdirectory. All the input files can be found in the *$ABI_TUTOPARAL/Input* directory. You can compare your results with reference output files located in *$ABI_TUTOPARAL/Refs*.

In the following, when “run ABINIT over *nn* CPU cores” appears, you have to use a specific command line according to the operating system and architecture of the computer you are using. This can be for instance: mpirun -n *nn* abinit < abinit.files or the use of a specific submission file.

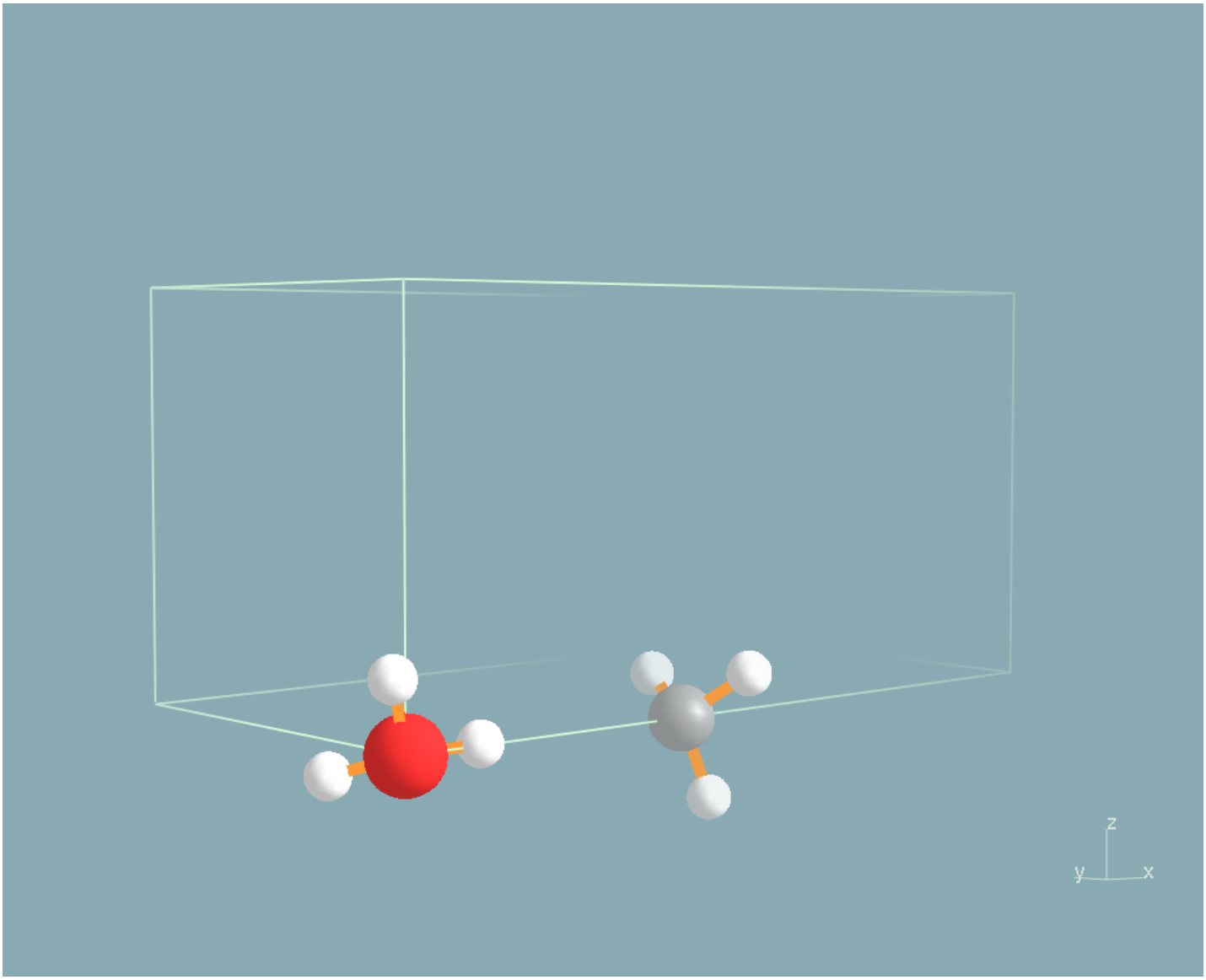
2 Computation of the initial and final configurations

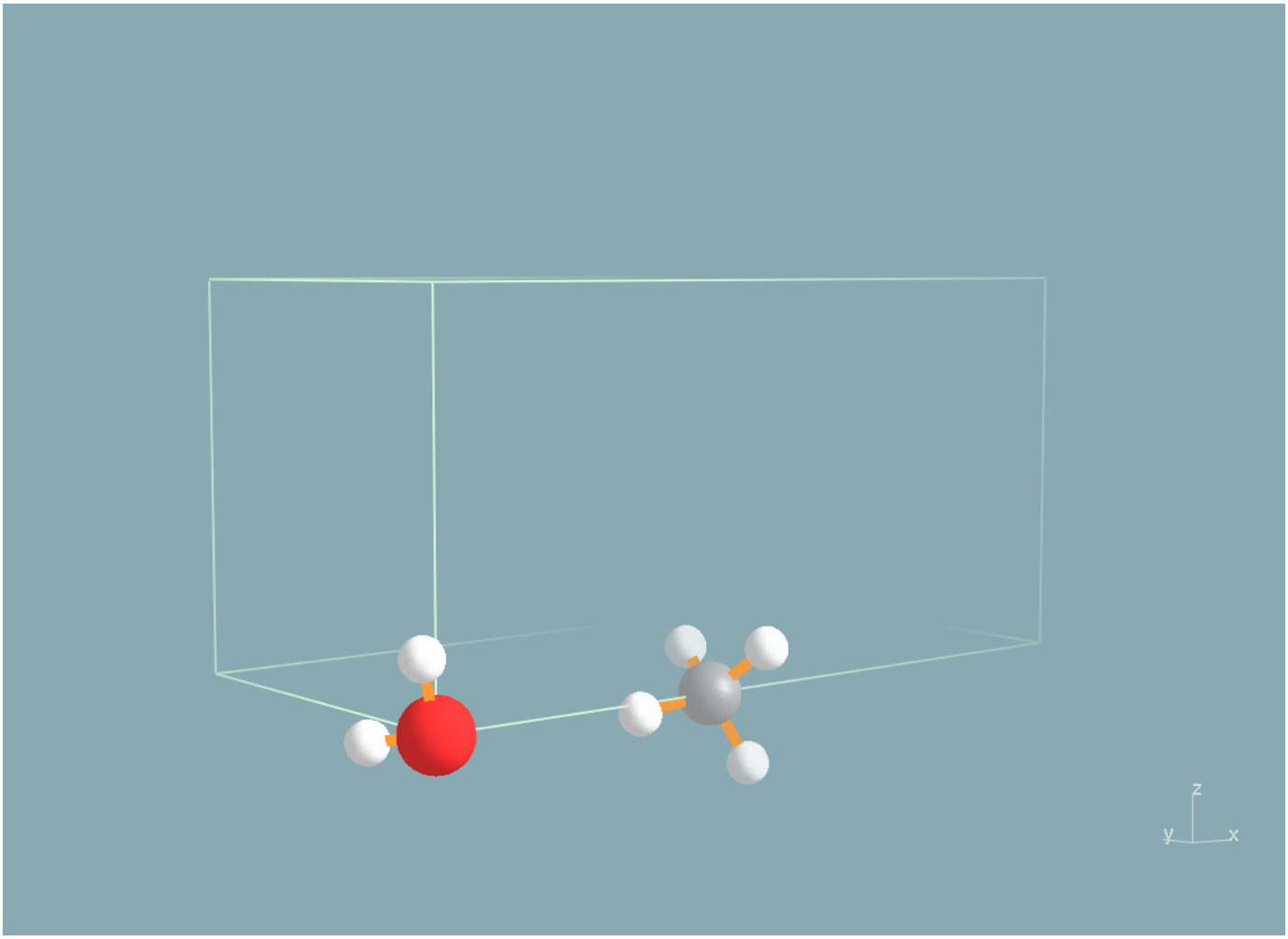
We propose to compute the energy barrier for transferring a proton from an hydronium ion (H_3O^+) onto a NH_3 molecule:

$$\mathbf{H_3O^+ + NH_3 \rightarrow H_2O + NH_4^+}$$

Starting from an hydronium ion and an ammoniac molecule, we obtain as final state a water molecule and an ammonium ion NH_4^+. In such a process, the MEP and the barrier are dependent on the distance between the hydronium ion and the NH_3 molecule. Thus we choose to fix the O atom of H_3O^+ and the N atom of NH_3 at a given distance from each other (4.0 Å). The calculation is performed using a LDA exchange-correlation functional.

You can visualize the initial and final states of the reaction below (H atoms are in white, the O atom is in red and the N atom in grey).


z
y
x



Before using the string method, it is necessary to optimize the initial and final points. The input files *tstring_01.in* and *tstring_02.in* contain respectively two geometries close to the initial and final states of the system. You have first to optimize properly these initial and final configurations, using for instance the Broyden algorithm implemented in ABINIT.

View tests/tutoparal/Input/tstring_01.in

View tests/tutoparal/Input/tstring_02.in

Open the *tstring_01.in* file and look at it carefully. The unit cell is defined at the end. Note that the keywords `natfix (../../variables/rlx#natfix)` and `iatfix (../../variables/rlx#iatfix)` are used to keep fixed the positions of the O and N atoms. The cell is tetragonal and its size is larger along x so that the periodic images of the system are separated by 4.0 Å of vacuum in the three directions. The keyword `charge (../../variables/gstate#charge)` is used to remove an electron of the system and thus obtain a protonated molecule (neutrality is recovered by adding a uniform compensating charge background).

The exchange-correlation functional uses the external library *libxc*. You have to compile ABINIT using the *libxc* plugin (if not, simply replace `ixc (../../variables/basic#ixc)` = -001009 by `ixc (../../variables/basic#ixc)` 7). This input file has to be run in parallel using 20 CPU cores. You might use the *tstring.files* file. Edit it and adapt it with the appropriate file names.

Then run the calculation in parallel over 20 CPU cores, first for the initial configuration (*tstring_01.in*), and then for the final one (*tstring_02.in*). You should obtain the following positions:

1) for the initial configuration:

```
xangst      0.0000000000E+00  0.0000000000E+00  0.0000000000E+00
           -3.7593832509E-01 -2.8581911534E-01  8.7109635973E-01
           -3.8439081179E-01  8.6764073738E-01 -2.8530130333E-01
            4.0000000000E+00  0.0000000000E+00  0.0000000000E+00
            4.3461703447E+00 -9.9808458269E-02 -9.5466143436E-01
            4.3190273240E+00 -7.8675247603E-01  5.6699786920E-01
            4.3411410402E+00  8.7383785043E-01  4.0224838603E-01
            1.0280313162E+00  2.2598784215E-02  1.5561763093E-02
```

2) for the final configuration:

```
xangst      0.0000000000E+00  0.0000000000E+00  0.0000000000E+00
           -3.0400286349E-01 -1.9039526061E-01  9.0873550186E-01
           -3.2251946581E-01  9.0284480687E-01 -1.8824324581E-01
            4.0000000000E+00  0.0000000000E+00  0.0000000000E+00
            4.4876385468E+00 -1.4925704575E-01 -8.9716581956E-01
            4.2142401901E+00 -7.8694929117E-01  6.3097154506E-01
            4.3498225718E+00  8.7106686509E-01  4.2709343135E-01
            2.9570301511E+00  5.5992672027E-02 -1.3560839453E-01
```

3 Related keywords

Once you have properly optimized the initial and final states of the process, you can turn to the computation of the MEP. Let us first have a look at the related keywords.

`imgmov (../../variables/rlx#imgmov)`

Selects an algorithm using replicas of the unit cell. For the string method, choose 2.

nimage (../../variables/rlx#nimage)

gives the number of replicas of the unit cell including the initial and final ones.

dynimage (../../variables/rlx#dynimage)(nimage (../../variables/rlx#nimage))

arrays of flags specifying if the image evolves or not (0: does not evolve; 1: evolves).

ntimimage (../../variables/rlx#ntimimage)

gives the maximum number of iterations (for the relaxation of the string).

tolimg (../../variables/rlx#tolimg)

convergence criterion (in Hartree) on the total energy (averaged over the `nimage (../../variables/rlx#nimage)` images).

fxcartfactor (../../variables/rlx#fxcartfactor)

"time step" (in $Bohr^2$/Hartree) for the evolution step of the string method. For the time being (ABINITv6.10), only steepest-descent algorithm is implemented.

npimage (../../variables/paral#npimage)

gives the number of processors among which the work load over the image level is shared. Only dynamical images are considered (images for which `dynimage (../../variables/rlx#dynimage)` is 1). This input variable can be automatically set by ABINIT if the number of processors is large enough.

prtvolimg (../../variables/files#prtvolimg)

governs the printing volume in the output file (0: full output; 1: intermediate; 2: minimum output).

4 Computation of the MEP without parallelism over images

You can now start with the string method. First, for test purpose, we will not use the parallelism over images and will thus only perform one step of string method.

View tests/tutoparal/Input/tstring_03.in

Open the *tstring_03.in* file and look at it. The initial and final configurations are specified at the end through the keywords [[xangst] and `xangst_lastimg (../../variables/rlx#nimage)`. By default, ABINIT generates the intermediate images by a linear interpolation between these two configurations. In this first calculation, we will sample the MEP with 12 points (2 are fixed and correspond to the initial and final states, 10 are evolving). `nimage (../../variables/rlx#nimage)` is thus set to 12. The keyword `npimage (../../variables/paral#npimage)` is set to 1 (no parallelism over images) and `ntimimage (../../variables/rlx#ntimimage)` is set to 1 (only one time step).

You might use the *tstring.files* file. Edit it and adapt it with the appropriate file names. Since the parallelism over the images is not used, this calculation has to be run over 20 CPU cores.

5 Computation of the MEP using parallelism over images

Now you can perform the complete computation of the MEP using the parallelism over the images.

View tests/tutoparal/Input/tstring_04.in

Open the *tstring_04.in* file. The keyword `npimage (../../variables/paral#npimage)` has been set to 10, and `ntimimage (../../variables/rlx#ntimimage)` has been increased to 50. This calculation has thus to be run over 200 CPU cores. Note that the output file is very big, so that no reference file is provided in the ABINIT package.

The convergence of the string method algorithm is controlled by `tolimg (../../variables/rlx#tolimg)`, which has been set to 0.0001 Ha. In order to obtain a more lisible output file, you can decrease the printing volume and set `prtvolimg (../../variables/files#prtvolimg)` to 2. Here again, you might use the *tstring.files*. Edit it and adapt it with the appropriate file names. Then run ABINIT over 200 CPU cores.

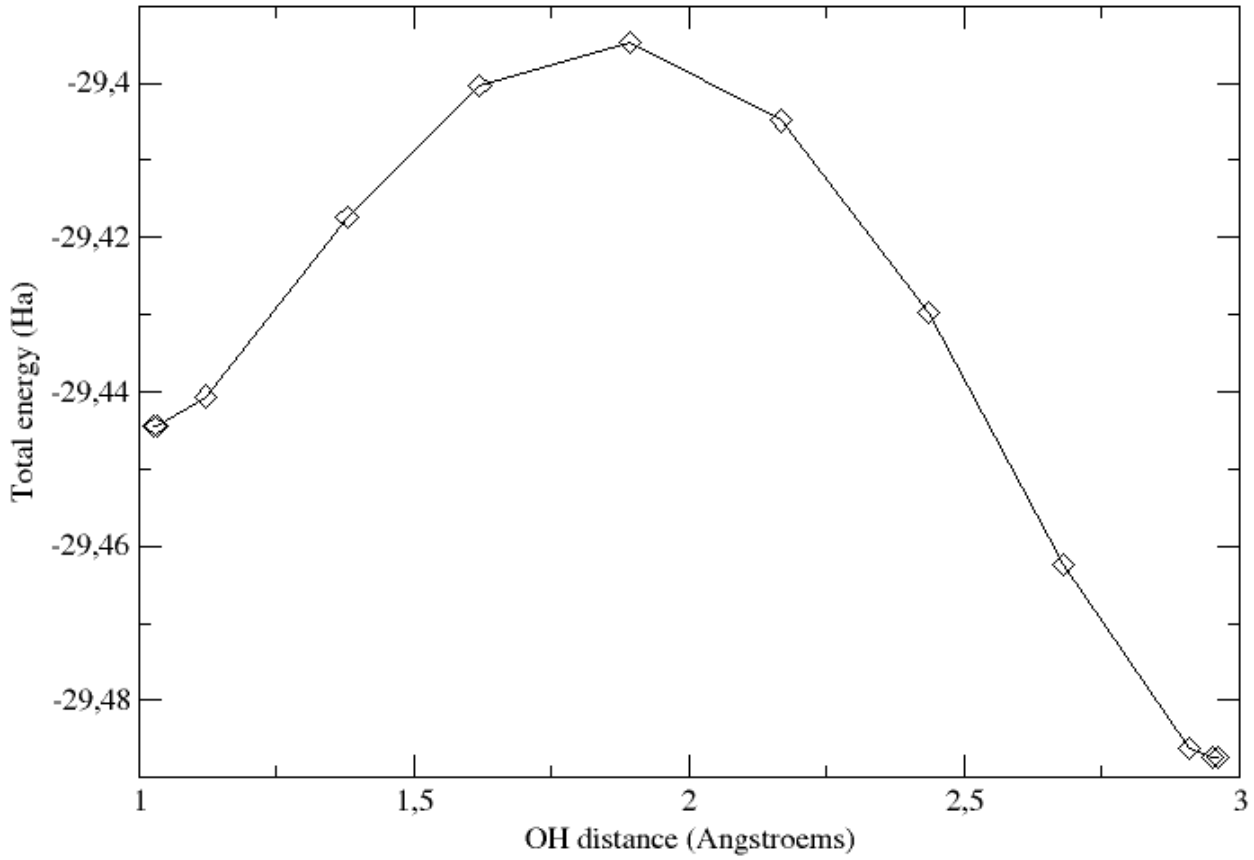
When the calculation is completed, ABINIT provides you with 12 configurations that sample the Minimum Energy Path between the initial (*i*) and final (*f*) states. Plotting the total energy of these configurations with respect to a reaction coordinate that join (*i*) to (*f*) gives you the energy barrier that separates (*i*) from (*f*). In our case, a natural reaction coordinate can be the distance between the hopping proton and the O atom of H_2O (d_{OH}), or equivalently the

distance between the proton and the N atom (d_{HN}). The graph below shows the total energy as a function of the OH distance along the MEP. It indicates that the barrier for crossing from H_2O to NH_3 is ~ 1.36 eV. The 6^{th} image gives an approximate geometry of the transition state. Note that in the initial state, the OH distance is significantly stretched, due to the presence of the NH_3 molecule.

Note that the total energy of each of the 12 replicas of the simulation cell can be found at the end of the output file in the section:

```
-outvars: echo values of variables after computation  --------
```

The total energies are printed out as: etotal_1img, etotal_2img, ..., etotal_12img. Also, you can can have a look at the atomic positions in each image: in cartesian coordinates (xcart_1img, xcart_2img, …) or in reduced coordinates (xred_1img, xred_2img, …). Similarly, the forces are printed out: fcart_1img, fcart_2img, …, fcart_12img.



Total energy as a function of OH distance for the path computed with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.0001 (which is very close to the x coordinate of the proton: first coordinate of xangst for the 8th atom in the output file).

The keyword `npimage (../../variables/paral#npimage)` can be automatically set by ABINIT. It takes the requested total number of CPU cores divided by the number of dynamical images. The remaining cores are, if possible, distributed over k-points, bands and FFT.

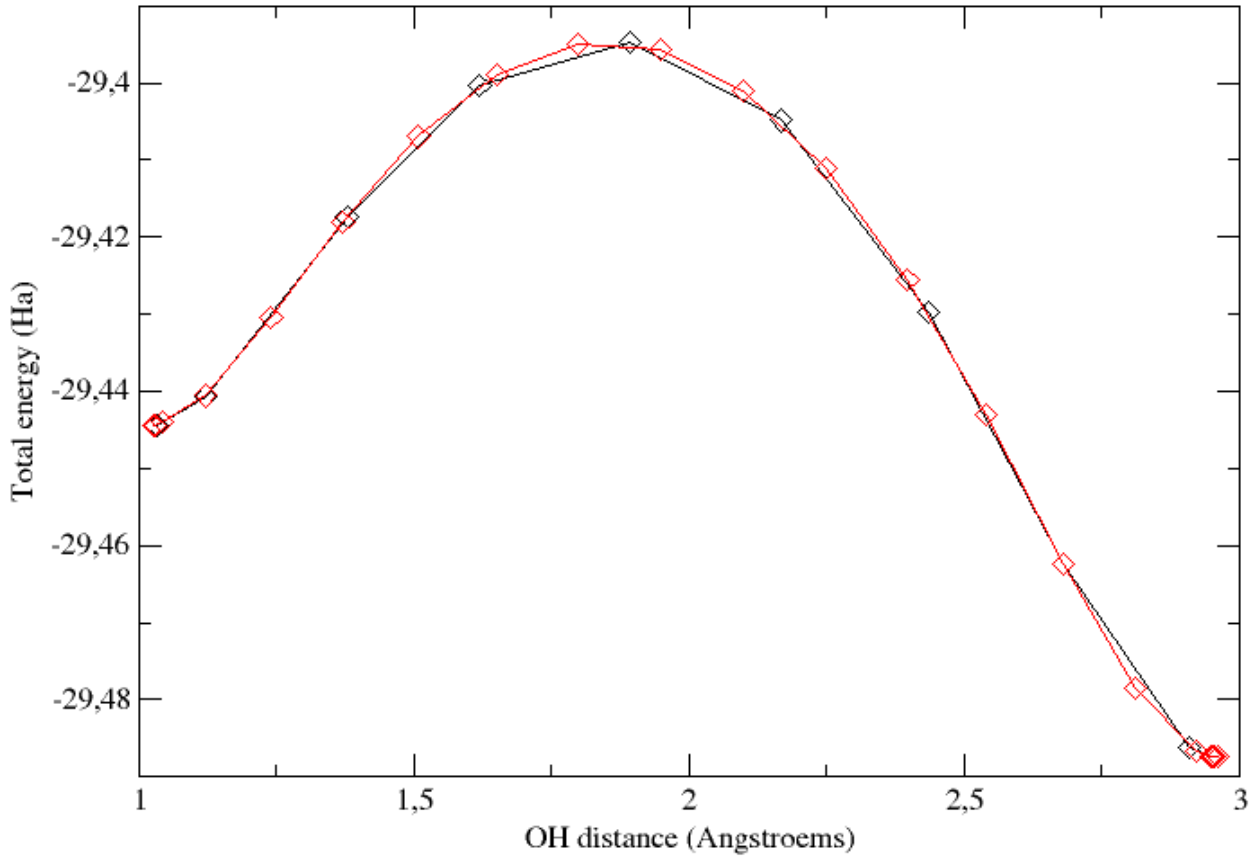
Let us test this functionality. Edit again the `tstring_04.in` file and comment the `npimage (../../variables/paral#npimage)` line. Then run the calculation again over a number of cores of your choice (less than 200). If the code stops with an error message indicating that the number of k-point, band and FFT processors is not correct, adapt the value of `npband (../../variables/paral#npband)` and `npfft (../../variables/paral#npfft)`.

Open the output file and look at the `npimage (../../variables/paral#npimage)` value ...

6 Converging the MEP

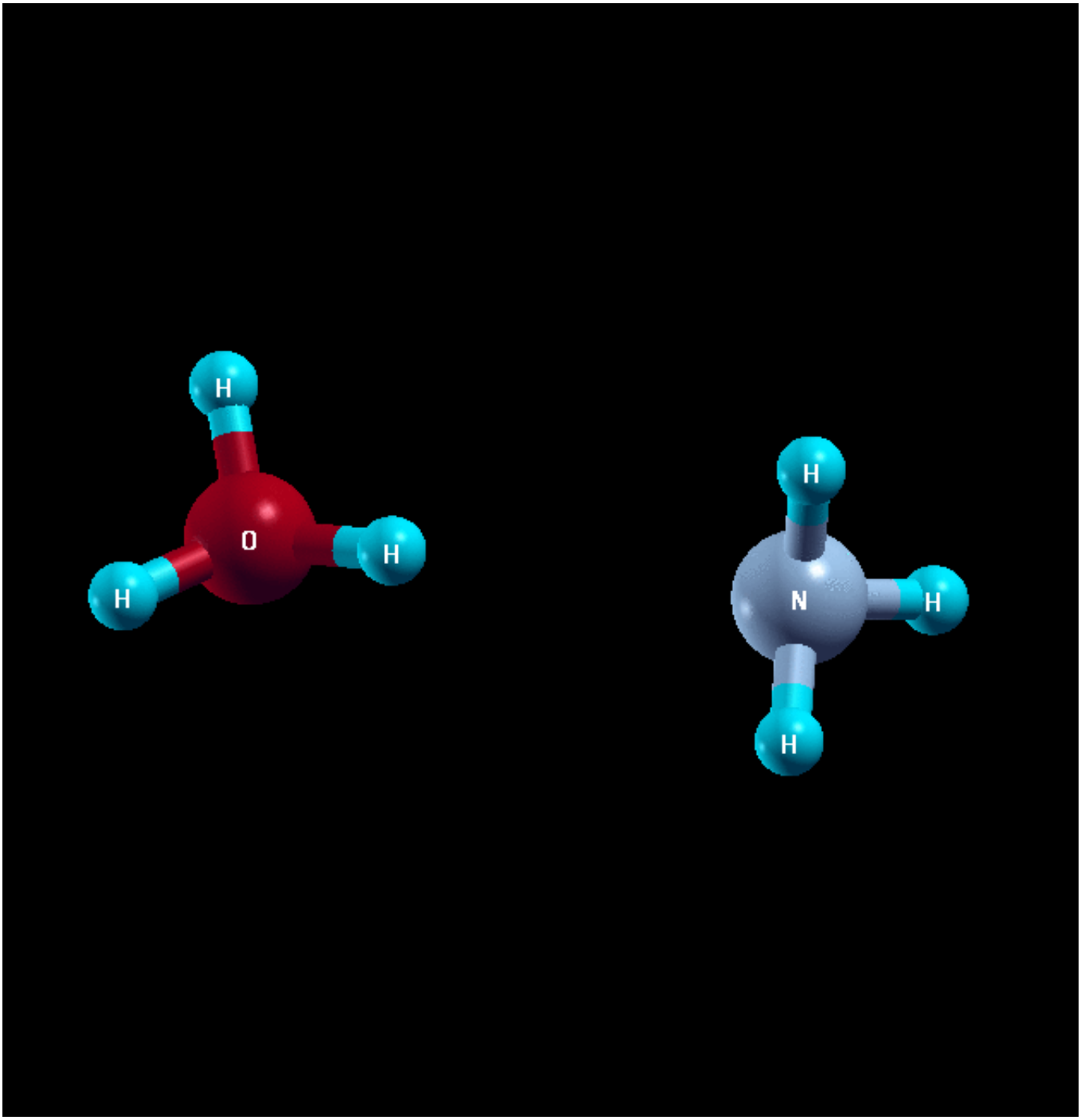
Like all physical quantities, the MEP has to be converged with respect to some numerical parameters. The two most important are the number of points along the path (`nimage (../../variables/rlx#nimage)`) and the convergence criterion (`tolimg (../../variables/rlx#tolimg)`).

1) `nimage (../../variables/rlx#nimage)`
Increase the number of images to 22 (2 fixed + 20 evolving) and recompute the MEP. The graph below superimposes the previous MEP (black curve, calculated with 12 images) and the new one obtained by using 22 images (red curve). You can see that the global profile is almost not modified as well as the energy barrier.



Total energy as a function of OH distance for the path computed with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.0001 (black curve) and the one computed with 22 images and `tolimg (../../variables/rlx#tolimg)`=0.0001 (red curve).

The following animation (animated gif file) is made by putting together the 22 images obtained at the end of this calculation, from (*i*) to (*f*) and then from (*f*) to (*i*). It allows to visualize the MEP.

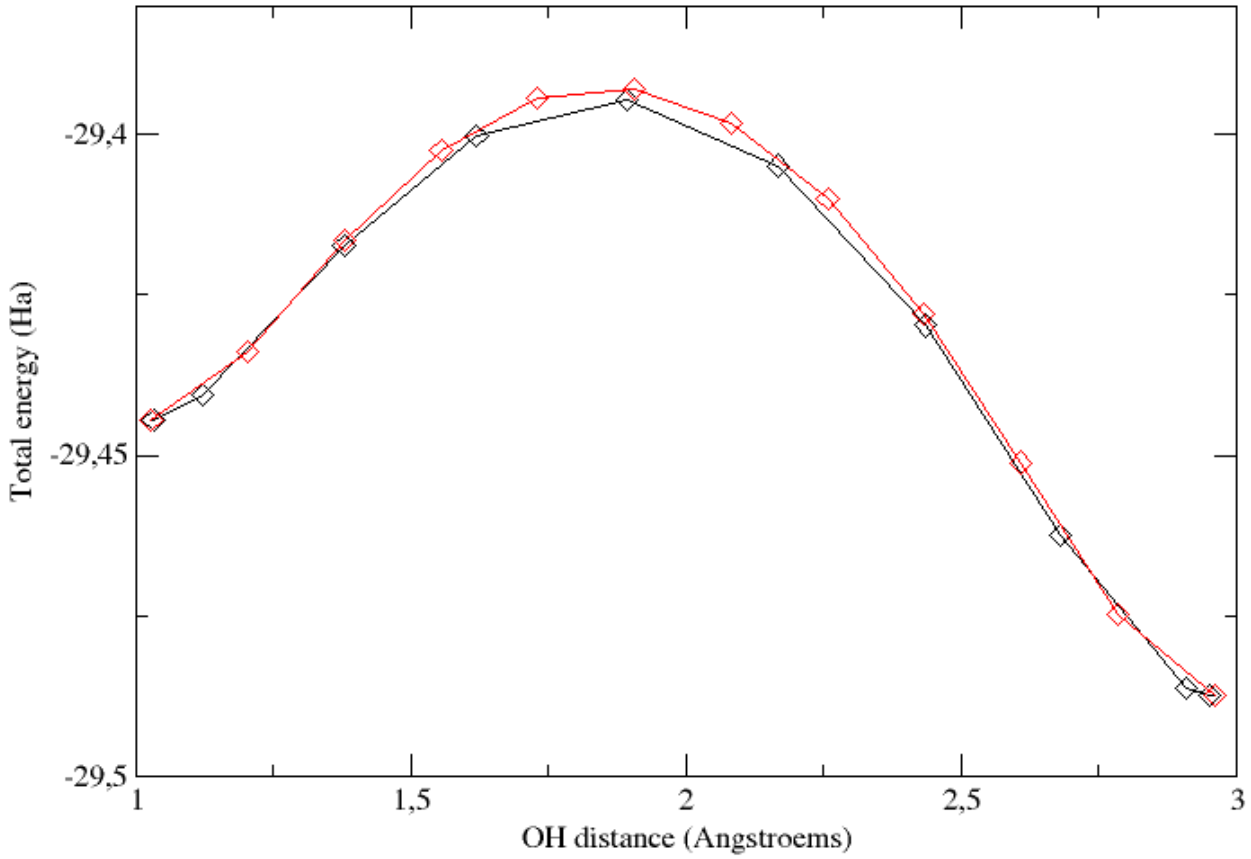


(../paral_images_assets/stringvideo.gif)

Click on the above image to visualize the animation (that should open in another window).

2) `tolimg (../../variables/rlx#tolimg)`
Come back to `nimage (../../variables/rlx#nimage)`=12. First you can increase `tolimg (../../variables/rlx#tolimg)` to 0.001 and recompute the MEP. This will be much faster than in the previous case.

Then you should decrease `tolimg (../../variables/rlx#tolimg)` to 0.00001 and recompute the MEP. To gain CPU time, you can start your calculation by using the 12 images obtained at the end of the calculation that used `tolimg (../../variables/rlx#tolimg)` = 0.0001. In your input file, these starting images will be specified by the keywords `xangst (../../variables/basic#xangst)`, `xangst_2img (../../variables/rlx#nimage)`, `xangst_3img (../../variables/rlx#nimage)` ... `xangst_12img (../../variables/rlx#nimage)`. You can copy them directly from the output file obtained at the previous section. The graph below superimposes the path obtained with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.001 (red curve) and the one with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.0001 (black curve).



Total energy as a function of OH distance for the path computed with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.0001 (black curve) and the one computed with 12 images and `tolimg (../../variables/rlx#tolimg)`=0.001 (red curve).