Manual: ART Nouveau

Normand Mousseau, Benjamin Bourassa, Simon Gelin and Mickaël Trochet
Département de physique - Université de Montréal,
C.P. 6128, Succursale Centre-Ville,
Montréal, Québec, Canada H3C 3J7.

normand.mousseau@umontreal.ca

Antoine Jay
LAAS, Toulouse

(Dated: April 22, 2020)

CONTENTS

I. INTRODUCTION

When using this code in publications, make sure to cite the following four papers:

1. G. T. Barkema et N. Mousseau, Phys. Rev. Lett. **77**, 4358 (1996).
2. R. Malek et N. Mousseau, Phys. Rev. E **62**, 7723-7728 (2000).
3. E. Cances, F. Legoll, M.-C. Marinica, K. Minoukadeh, and F. Willaime, J. Chem. Phys. **130**, 114711 (2009).
4. E. Machado-Charry, L.K. Béland, D. Caliste, L. Genovese, T. Deutsch, N. Mousseau and P. Pochet, J. Chem. Phys. **135**, 034102 (2011).

II. THE DISTRIBUTION

This distribution is provided with under the terms of the GNU General Public License, see /COPYING file or http://www.gnu.org/copyleft/gpl.txt.

It contains the code that defines ART nouveau, with all the optimisations described in Machado-Charry et al. (2011).

It can generate events (i.e. identify saddle points and minima) form a given minimum or be launched to refine a saddle point (if the estimated position of the saddle points exists).

It is provided with the Stillinger-Weber potential and a test case run on a 1000-atom box of amorphous silicon (we use a SW potential with modified parameters for the three-body strength and the overall energy scale). It can be used with any type of potential (empirical or ab-initio), provided forces are continuous and their derivative exist.

However, the code is really meant to be used with the force librairies of LAMMPS, which provides a wide range of potential to explore. To use ART nouveau with this new feature, LAMMPS must be downloaded via `http://lammps.sandia.gov` and compiled before compiling the source code of ART nouveau. A number of examples are provided with this coupling.

This short manual described the various parameters used in the method. Although, there is a fair number of them, most do not need to be modified. We recommend that you first compile and run the test examples to see how the program works.

III. COMPILING

ART nouveau is known to compile with gfortran, g95 and the Intel Fortran compiler. It requires BLAS and LAPACK libraries. It can be compiled with MPI support (in case the force-field is parallelized).

To adapt the code to your machine, identify the most appropriate Makefile in `source/MAKE` directory and modify the lines accordingly. However, to those who want to compile ART nouveau without Lammps, we refer to the subsection III A and to those who want the Lammps library, the procedure is explained at subsection III B.

A. Building without Lammps

To build ART nouveau without the lammps library type the following command : select one of the `Makefile`s without lammps in the `source/MAKE` directory of ART nouveau. This should create an executable called art. You can now pass the next subsection and go directly to the test section in subsection IV A.

B. Building on mac OS X (10.9.4) with Intel compiler and Lammps and optionnal Openmpi

To build ART nouveau on a mac, with the lammps library, users can either compile lammps without or with openmpi.

In the latter case, users need to install a version of Openmpi (1.6.5 and 1.8.1 1.10.2 tested) . On most clusters, it is possible to load modules of all these necessary software. Otherwise check the two subsection below.

1. *From webpage: openmpi, fftw and lapack with openblas, intel or gcc compilers*

a. Openmpi installation from webpage For more details, we recommend to have a look at this two webpages :
`http://www.open-mpi.org/faq/?category=building` section 15
`https://software.intel.com/en-us/articles/performance-tools-for-software-developers-building-open-mpi-with`
`10`

One can download the openmpi version that you desire on their webpage. Untar, unzip the file, (the command should be tar -xf openmpi-*.*.*.tar.gz). Once it is done, go into the folder openmpi-1.8.1 and then type the following command :
./configure –prefix= (Installation location) CC=icc CXX=icpc F77=ifort FC=ifort. We highly recommend that you create a specific and unique path for the installation of openmpi before typing the configure command because your machine may already have a version of it (for example /usr/MPI/intel/openmpi/number_version/) and interference may occur between installation. After the configuration command and if everything went well just type make all install. The two previous command produce a lot of screen output and can take about 20 mins to be done, be patient! Once the installation is done, it is still necessary to set the PATH for the bin directory and the library directory this way :
export PATH=(installation_directory)/bin:$PATH
export DYLD_LIBRARY_PATH=(installation_directory)/lib:$DYLD_LIBRARY_PATH You can add this two lines in your $\sim /.profile$ or $\sim$ /*.bashrc* to set them as a default path.

b. FFTW installation from webpage If you choose to usu FFTW (FFTW3 tested), follow the next lines. For simplicity, however, we recommand that you use the fftw routine by lammps and leave the jpeg library out (see lammps documentation for more details).

Download the FFTW version that you desire on the FFTW webpage (`http://www.fftw.org`). Untar, unzip the file. Once it's done go to the folder fftw and then type the following command :
./configure –prefix= (Intallation location). Once again we highly recommend that you create a specific installation directory (for exemple /usr/FFTW/). After the configuration type make and then make install. Now the installation is done but it is not over yet, you have to set the PATH for the bin directory and the library directory this way :
export PATH=(installation_directory)/bin:$PATH
export DYLD_LIBRARY_PATH=(installation_directory)/lib:$DYLD_LIBRARY_PATH

c. Lapack with openblas Go to this webpage : `http://www.netlib.org/lapack/#_software`
download the latest version of lapack we test it with the version 3.6.0

2. *From Homebrew: openmpi, fftw and lapack with openblas, gcc compilers on Mac os*

Please see the follow step by step process:

```
brew install gcc
export HOMEBREW_CC = gcc-5  from Homebrew
export HOMEBREW_CXX = g++-5 from Homebrew
brew install openmpi --build-from-source
brew install scalapack --build-from-source
brew install openblas --build-from-source  # Might not be necessary
```

Compile lammps library with mpi architecture (comment the MPI_INC field).
Create lammps shared library:

```
make mpi   # this is creating a executable to test lammps by itself
make mode=shlib mpi  # this is creating the actual shared library needed by ARTnouveau
```

In the Makefile.art_gnu5_lammps_mpi please set the USER_PATH to the path of your lammps src folder.
Compile ART_nouveau with the art_gnu5_lammps_mpi architecture :

```
make   art_gnu5_lammps_mpi
```

3. ART nouveau and Lammps Makefiles

For LAMMPS, make sure that you compile all the libraries and load the needed packages. The architecture that you might want to use is mac_mpi but you also need to modify it by setting the path for the mpi and fftw that you have just created. For ART nouveau, the architecture `maclammps_mpi` is already setup. However, you will need to verify the `bin`, `include` and `lib` directory for these softwares : `openmpi`, `fftw`, `mkl` and `Lammps`. One more warning: when building on Mac OSX using intel compiler, one will probably need to source the file `mklvars.sh` with the option intel64 (especially if the compilation succeeds but not the execution of the code).

C. Using with LAMMPS

ART nouveau can be compiled with the LAMMPS library, giving it access to a wide range of efficiently programmed forcefields. ART nouveau includes a modified version of the fortran interface to the library as the original one has some limitations. For this, download the latest version of LAMMPS : `http://lammps.sandia.gov`

1. Build LAMMPS as a Library

   (a) Load all the packages from LAMMPS that you need before compilation. (for example, the package manybody is needed for using tersoff and sw potential) you can find the list of all the available packages by typing make package-status in the src directory of LAMMPS or check the list on their web site (`http://lammps.sandia.gov/doc/Section_packages.html`)

   (b) Now Compile LAMMPS as a library by entering :
   `% make mode=lib foo` for a static library or
   `% make mode=shlib foo` for a dynamic (shared) library
   (where foo is the architecture makefile name, for further details check the section 2.5 of their web page, `http://lammps.sandia.gov/doc/Section_start.html#start_5`)

   (c) You should have now a file named liblammps_foo.a in the src folder. You can move it to the source code directory of ART nouveau or update the path to the library in the ART nouveau Makefile.

2. To compile ART nouveau with the lammps library, you need to give the path to the lammps directory in the `MAKE/Makefile.xx` file (selecting a makefile with `lammps` in the name): After adjusting the compiling options to your compiler, just enter, for example :
   `% make artlammps`
   this should create an executable called ARTn_exec.

Example tests with ART nouveau and the LAMMPS library are available in the Example directory, see the subsection IV B.

Note that if you used a shared library for lammps, you might need to copy the library file (.so) into the Example directory.

IV. TEST

A. Test without Lammps library

After compiling, you can run a prepared test on amorphous silicon (1000 atoms) situated in the `example/a-Si` directory or a prepared test on a box of 511 atoms of silicon with one vacancy situated in the `example/Si-vac` directory (with modified Stillinger-Weber potential - see subroutine `init_potential_SW` in `calcfo_sw.f90`).

To launch a simulation you need two files:

`bart.sh`: input file - sets the simulation.

`refconfig`: Atomic position file. We suppose a box with periodic boundary conditions running from $[-0.5, 0.5]L$ where L is the box size.

`filecounter`: this file set the counter position for relaunching a simulation (optionnal).

For this test, the variable ENERGY_CALC in the `bart.sh` file must be set with the forcefield SWA to ensure that the parameters used in the Stillinger-Weber potential correspond to the case of amorphous silicon. In the case of the diamond silicon structure, one must set ENERGY_CALC with the forcefield name SWP. Also, be sure that the executable at the end of the `bart.sh` is the good one. In this case, it should be called art. Once the program is compile you can go in the appropriate directory and launch the test with simply

```
% ./bart.sh
```

B. Test with Lammps library

After compiling ART nouveau with Lammps library, you can go in the `example` directory and run tests. In that directory. you will find two subdirectory, `a-Si-LAMMPS` which correspond to a test on 1000 atoms of amorphous silicon and `Si-vac-LAMMPS` corresponding to a box of 511 atoms of silicon with one vacancy. These examples use Stillinger-Weber potential directly from LAMMPS library. Be sure that when compiling LAMMPS as a library, the package MANYBODY is installed. To launch a simulation you need these files:

`bart.sh`: input file - sets the simulation.

`refconfig`: Atomic position file. We suppose a box with periodic boundary conditions running from $[-0.5, 0.5]L$ where L is the box size.

`in.lammps`: input file for LAMMPS.

`conf.lammps`: Atomic position file for LAMMPS (conf.sw for the tests).

`parameter.file`: parameter file (Si.sw for the tests) - specific for each atom types and potential in LAMMPS.

`library.file`: library file (required for certain potential in LAMMPS like MEAM for example).

`filecounter`: this file set the counter position for relaunching a simulation (optionnal).

For these tests, the variable ENERGY_CALC in the `bart.sh` file must be set with the keyword LAM to specify that the force calculation is made by LAMMPS. In addition, the variable INPUT_LAMMPS_FILE must be set with the appropriate name for the input LAMMPS file, in the tests, we call it `in.lammps`. Also, be sure that the executable at the end of the `bart.sh` is the good one. In this case, it should be called artlammps. Once the program is compile you can go in the appropriate directory and launch the test with simply

```
% ./bart.sh
```

V. INPUT FILES

Input coordinate files must be provided for both ARTn and LAMMPS (when needed) although the are defined by ARTn configuration file, the LAMMPS file is necessary to set up the atomic type and the box size.

The box type is defined in the ARTn box.

1. Box size and boundary conditions

Currently, ARTn allows orthorhombic as well as triclinic boxes with periodic conditions in all directions. The box size is defined in length units and should be the full box size.

For an orthorhombic box, use a header of this type in the configuration file:

```
run_id:         1009
total_energy:   -2187.7695832636687
P   21.719999999999999        21.719999999999999        21.719999999999999
     1        3.94619485        3.97430605        1.31195598
  ...
```

where the line starting with a "P" indicates a periodic othorhombic box with lengths running 0.0d0 to 21.72 in the three directions. Note the three directions can be different.

The last line indicates the atomic type as well as the positions in angstroms.

For triclinic box, the input file should be :

```
run_id:         1000
total_energy:   -4047417.1388702886
T   1025.4115364835884        1.1523760001271599       0.43554693952618401
    0.0000000000000000        993.03783130273587        1.5941286529841301
    0.0000000000000000        0.0000000000000000        991.27942692464990
     1       41.82986869       19.16022413       23.72110786
     1      100.52528331       35.93805275       41.11089115
     ...
```

Here the triclinic box is indicated by the letter "T" Three lines are needed. The triclinic parameters matrix is defined following the LAMMPS formalism. The documentation can be found on the triclinic simulation boxes LAMMPS' page.

Note that the three last off-diagonal are zero. Only the tree on the top are taken into account.

VI. OUTPUT FILES

While there is an output on the screen, it is more interest to look at the file `log.file.1` that stores a more readable output as well as `events.list` which contains the list of attempted events.

Let us look at the two files:

A. events.list

The output of the `events.list` file looks something like this:

```
  min1000             sad1001             min1001                accepted
  min1001             sad1002             min1002                rejected
  min1001             sad1003             min1003                rejected
  min1001             sad1006             min1006                accepted
  min1006             sad1007             min1007                rejected
```

It displays where to find the initial minimum, the saddle and the final configurations. It also indicates whether this event has been accepted or rejected. With this list and the configuration files, you can perform the analysis you wish to do.

B. log.file.1

The `log_file.#` provides the details of the event generation. The first part summarizes the selected parameters.

A first relaxation is attempted. If the system is already in its minimum (the total force is less than the threshold, no minimization is attempted):

```
  RELAXATION
  - Configuration stored in file :          min1000
```

A first event is then attempted, with the initial details provided:

```
  - Simulation                  :                 1
  - Attempt                     :                 1
  - Starting from minconf       :              1001
  - Reference Energy (eV)       : -3.0327393805E+03
  - Temperature                 :          0.500000

    - That atom                   :               372
```

If a local activation is selected, the last line indicates the atom around which the deformation is applied.

First, a deformation is applied to this atom and its neighbourhood and details of the steps are provided:

```
           E-Eref  m_perp  ftot    fpar     fperp    eigen     delr  npart  evalf    a1
           ( eV )               ( eV/Ang )         ( eV/Ang**2 )
 0  K=   0  0.0111  1 1    0.5343  -0.4916   0.2094   0.0000   0.051   0     2   0.00
 1  K=   1  0.2113  3 3    1.9783  -1.9355   0.4090   0.0000   0.278   4     6   0.00
 2  K=   2  0.6685  3 4    3.3559  -2.9701   1.5621   0.3693   0.545   7    28   0.00
 3  K=   3  1.3384  3 4    5.0209  -4.4871   2.2530   0.2187   0.811  12    50   0.79
 4  K=   4  2.1376  3 5    5.4512  -5.3370   1.1097   0.1854   1.063  13    73   0.81
 5  K=   5  3.1155  3 3    6.1048  -6.0313   0.9444   0.1502   1.344  19    94   0.89
 6  K=   6  4.2055  3 5    6.5059  -6.3110   1.5806   0.1334   1.639  27   117   0.97
 7  K=   7  5.3216  3 3    6.9059  -6.7877   1.2721   0.0460   1.930  30   138   0.35
 8  K=   8  6.4951  3 5    7.0132  -6.8066   1.6898  -0.1985   2.228  38   161   0.76
 9  K=   9  7.6389  3 3    7.1883  -6.8966   2.0268  -3.3823   2.536  41   182   0.52
10  L=   1  6.6452  3 3    5.4540   0.0521   5.4538  -5.8090   2.491  43   211   0.99
11  L=   2  1.5856 1013    1.6252  -0.9609   1.3107  -3.0373   1.415  31   256   0.90
12  L=   3  1.3182 1010    0.8181  -0.6905   0.4389  -1.8871   1.372  16   304   0.96
13  L=   4  1.3623  1 1    0.5371  -0.4152   0.3407  -1.0826   1.440  13   349   0.93
14  L=   5  1.4116  0 0    0.5544  -0.2409   0.4994  -3.2086   1.433  12   399   0.91
15  L=   6  1.3052  1 1    0.1017  -0.0207   0.0996  -3.4817   1.361  10   456   1.00
16  L=   7  1.2967  1 1    0.0362  -0.0008   0.0361  -3.3445   1.370  12   519   1.00
```

The phase associated with the push to leave the harmonic basin is indicated by a K in the second column. Once a Lanczos phase is started, bringing the configuration onto the saddle point, a L is indicated in the second column.

Columns, from left to right, are : (1) iteration number (total); (2) type of move - bassin (K) or lanczos (L); (3) energy difference from the ininitial minimum; (4) and (5) number of iterations for force convergence; (6) total force (absolute value); (7) and (8) parallel and perpendicular force to the deformation or the eigenvector of negative eigenvalue; (9) value of the lowest eigenvalue; (10) displacement from the initial minimum; (11) number of atoms having moved by more than 0.1 Å; (12) number of force evaluations since last successful event; (13) projection of the current eigenvector onto the previous one.

If the push leads to a first-saddle point (i.e., total force below the fixed threshold and a negative eigenvalue), then the event is considered successful (converged):

```
SADDLE 1001 CONVERGED |ret 20016 |delta energy= 1.2967 |force_(tot,par,perp)= 0.036 -0.0008  0.036
  |eigenr=    1.370 |evalf=   519 ||dell
   - Configuration stored in file :           sad1001
   - Total energy Saddle (eV)     : -3.0314426995E+03
```

The first two lines repeat the information provided above as computed at the saddle point. Eigenr is the displacement from the initial minimum and `ret 20016` is a code that indicates successful event with 16 iterations.

Since converge is achieved, the system is then pushed slightly over and the configuration is relaxed:

```
 RELAXATION
 21  M=   5  1.2295  1 1  0.3044    0.2160  0.0000  0.0000  1.475  11   525   0.00
 26  M=  10  1.1944  1 1  0.1370    0.0558  0.0000  0.0000  1.515  13   530   0.00
 31  M=  15  1.1880  1 1  0.1026    0.0389  0.0000  0.0000  1.597  16   535   0.00
 - Configuration stored in file :           min1001
 - Total energy Minimum (eV)    : -3.0315520829E+03
MINIMUM 1001 REJECTED |E(fin-ini)=  1.1873 |E(fin-sad)= -0.1094 |npart= 15 |delr= 1.593
|evalf=   537 |
```

Here the event is rejected but the saddle and minimum have been stored and a new event is launched.

In some cases, the negative eigenvalue is lost due to the structure of the energy landscape and we get:

```
SADDLE 1002  FAILED |ret 60012 |delta energy=  2.283 |force_(tot,par,perp)= 0.2676 -0.2296  0.1376
  |eigenr=    2.138 |evalf=   392 ||dell
```

The number of force evaluations is not set back to zero (so that we can get a clear evaluation of costs) but a new search is made from the same initial minimum.

VII. PARAMETERS

All parameters are defined in the bart.sh file to be found in the test subdirectory. They are divided into 7 sections with a short description and the default value indicated as comments, when appropriate.

Many parameters are self-explanatory. We review the others here.

A. ATOMS

N_atoms: Number of atoms in the simulated box.

Type_1: Atomic species for an xyz output useful for a number of graphical programs such as jmol. Add other types when needed. Note that this will also be used to define the number of types and it is therefore required when many atomic species are used.

B. ART

Event_type: Defines the type of simulation to be run.

- **NEW:** When launching simulation from a new event. Then one only needs to provide the reference configuration (REFCONFIG) and the initial file number (FILECOUNTER).
- **REFINE_SADDLE:** Use when starting from a configuration near the saddle point. This can be used with a stricted convergence criterion, for example, but also in replacement of NEBM when a reasonnable guess can be made regarding the saddle point. Stops when the saddle point has been reached.
- **REFINE_AND_RELAX:** Same as above but also provides the final minimum.
- **RESTART:** Is there a restart??

Temperature: Temperature used in the Metropolis accept/reject move. It is in the same units as the potential. Use a negative temperature to reject all moves and sample a local minimum.

Max_Number_Events: Maximum number of successful events generated.

Type_of_Event: Type of initial random deformation use in generating an event. `global` is for an initial deformation involving all atoms in the system. With `local`, an atom and its neighbours are selected and moved in a random direction, with the rest of the system allowed to react. The central atom is selected at random (default) or is specified (see `Central_Atom`) and the site of the neighbouring area is defined by `Radius_Initial_Deformation`.

In general, the later parameter is preferable at it leads more rapidly to a more diversified set of barriers. With sufficint sampling, however, both parameters lead ot the same barriers.

Radius_Initial_Deformation: Radius of the spherical region surrouding the central atom selected in the initial random displacement. It is typically set to the first or second neighbour shell.

Central_Atom: When defined, identifies the atom that is always at the center of the initial displamement for a `local` initial move.

Sym_Break_Dist: When in a perfect crystal, it is sometimes difficult to generate events, due to the symmetry of the system. This parameters breaks the symmetry by moving randomly all atoms by this distance.

Forcefield: Forcefield used. In this version, two potentials are provided, `SWP`, the Stillinger-Weber potential, and `MSW`, a modified version defined in Vink et al..

Activation_MaxIter: The maximum number of Lanczos steps during the activation. Typically something between 50 and 150, depending on the system and your patience.

Increment_Size: Overall length scale for the various displacements during the activation. Typically between 0.05 and 0.1 Å in bulk systems. In units of the positions.

Force_Threshold_Perp_Rel: Minimum force relaxation for the perpendicuar hyperplane during the Lanczos part of the activation. Applies a variable step steepest descent on this perpendicular force until this threshold or the maximum number of iterations is reached In units of the energy divided by units of positions.

C. Harmonic Well

These parameters determine the exit of the harmonic well. There are two important parameters here: `Eigenvalue_Threshold`, which fixes the eigenvalue at which we consider that the system has left the harmonic basin, and `Max_Perp_Moves_Basin`, which sets the degree of hyperpendicular relaxation.

This second parameter somewhat controls how much strain will be allowed in the system as we push in a random direction. If we relax too much in the perpendicular direction, then the system will typically deform along the softest direction. If there is not enough perpendicular relaxation, then the stored strain at the edge of the harmonic basin will be very large and it is likely that the system will fall back in the harmonic basin during the activation phase.

Initial_Step_Size: Size of the initial deformation. This is not very important. It simply needs to be large enough to generate a reliable force. In units of positions. To decrease the number of force calls, it can be set relatively large such as 1 Å or more depending on the number of atoms moved initially (as defined by `Radius_Initial_Deformation`)

Basin_Factor: Factor multiplying the reference length `Increment_Size` when pushing for leaving the harmonic basin. Typically around 2-3.

Max_Perp_Moves_Basin: Maximum number of relaxation steps in the hyperplane perpendicular to the direction of deformation. This ensures that head-on collisions do not take place and that the total energy remains under control during this phase. This needs to be adjusted to maximize the probability of successful events.

Min_Number_KSteps: Minimum number of steps along the direction of random deformation before computing the lowest eigenvalue. Ideally, this should be as big as possible, to decrease the computer costs. However, there should typically be two Lanczos steps before the system leaves the harmonic well to ensure that the eigenvalue is converged. This value related to the `Increment_Size` and the `Initial_Step_Size` in addition to the more physical features of the system.

Eigenvalue_Threshold: Threshold below which the system is reputed to have left the harmonic basin. Should be balance to maximise the number of successful events while making sure that the system does not jump basin.

Max_Iter_Basin: Maximum number of iterations in the basin. Keep it around 20 and adjust the other parameters.

When the lowest eigenvalue (λ_0) moves <u>below</u> the threshold given by `Eigenvalue_Threshold`, the system is now considered outside the harmonic basin and the push will be along the corresponding eigenvector ($\hat{e}_0$). However, to avoid changing too rapidly, the move takes place over a few steps, combining the initial random push ($\hat{r}_{rand}$) and $\hat{e}_0$:

$$\hat{r}_e = \cos(\frac{i}{n})\hat{r}_{rand} + \sin(\frac{i}{n})\hat{e}_0 \tag{1}$$

where i increases by one at each step starting from the step during which the eigenvalue becomes negative.

Smooth_Dir_Change: This defines the number of steps over which the direction is change from the initial random displacement to the eigenvector associated with the lowest eigenvalue. By default it is set to 3.

D. Lanczos

The parameters here are not very sensitive and can be left unchanged.

Lanczos_of_minimum: Check if minima are really minima (all eigenvalues positive). This is useful for <u>ab initio</u> calculation but generally left to `False` for empirical potential where minima do not pose problem.

Number_Lanczos_Vectors: Number of vectors included in the Lanczos procedure. Generally set to 15, can be used with 12, if more agressive.

delta_displ_lanczos: Displacement for the numerical derivative. Should be bigger for quantum calculations.

E. Limiting the output files

There are a lot of files generated by ARTn. It is possible to limit the output by not saving the rejected events. To do so, it suffices to define

`WRITE_REJECTED_EVENT:` to `.false.`

When set to `.false.` then `Max_Number_Events` defined in `bart.sh` is now the total of <u>accepted</u> events and only those are saved for the saddle and final configuration

VIII. CONVERGENCE

Parameters are for the convergence of the forces in the hyperpendicular plane to the direction of negative eigenvalue. The only important parameter is the exit force threshold.

Exit_Force_Threshold: Threshold at which we consider that the system is converged at the saddle point. Typically around 0.1 eV/A. This is the total force on the system.

Prefactor_Push_Over_Saddle: After reaching a saddle point, the system is pushed away from the initial minimum along the negative eigenvalue. The displacement is a percentage of the distance between the initial and saddle state. This parameter fixes the percentage (typically 15 %).

Save_Conf_Int: Save the configuration at every step. Used only for ab initio calculations.

IX. DIIS

We do not favor using DIIS without extensive experience with the method. Please contact the developers on this point. In the meanwhile, it is recommended to keep `Use_DIIS` as `False`.

X. LOCAL FORCES

By default, ART nouveau computes the force on all atoms even though events tend to be local. This is fine for small boxes, but not for large systems. In those case, it is possible to limit the force calculations used to find saddle points and new minima to the atoms involved.

The option `LOCAL_FORCE = .true.` does just that. It defines an spherical inner region around the selected atom (radius defined with `INNER_REGION_RADIUS` where atoms are allowed to move in response to the deformation. To ensure that the forces are correct in this region, an additional shell of thickness defined by `OUTER_REGION_WIDTH` is used to compute forces but in which atoms are frozen.

By default, there are additional convergence steps at the saddle point and the final minimum using the global force, before the accept/reject step. In general, this does not affect significantly the energy at the saddle or the final step. This is why an additional option is available to avoid this relaxation - `GLOBAL_CONVERGENCE`. By default, this value is `.true.`, which means a global always takes place.

To avoid building long-range strain, however, accepted conformations are always fully relaxed before being stored, irrespective of the various parameters.

```
#---------------------------- LOCAL FORCE
setenv LOCAL_FORCE                    .true.          # Use local forces (def: .false.)
setenv INNER_REGION_RADIUS            15.0     # Radius of inner region
setenv OUTER_REGION_WIDTH             6.0      # Thickness of outer region shell
setenv GLOBAL_CONVERGENCE    .false.          # Performs a global force convergence step at the saddle and
                                                                            # before accepting an event (def:
```

Note that this feature works only with the options `Type_of_Events = local` or `list_local`

A. Setting up parameters

The use of local forces is an approximation. To make it valid, the number of atoms in the selected region must be sufficient to include most the elastic deformations. It must therefore adapted to your system.

For events that are relatively local (involving 10-20 atoms, for example), we find that the inner region much count between 600 and 1000 atoms; if the events displaces more atoms, the inner region must be larger.

You can check this by looking at the log file. Before each event, the number of atoms in the inner (corresponding to the `INNER_REGION_RADIUS`) and in the outer region (the region that is rigid but is used to ensure that forces are correct on the moving atoms) are printed. For example:

```
nat_inner  :   978   nat_outer : 1300     nat_local : 2278
```

You can validate these parameters by confirming that you find the same event with global and local forces.

XI. RECONSTRUCTION PATHS OF STEEPEST DESCENT CONNECTING MINIMA AND SADDLE POINTS

It is now possible to reconstruction paths of steepest descent connecting a saddle point with a minimum, either the initial minimum or the final minimum. This is useful, for example, to understand the motion of atoms along the pathway.

As it is currently implemented, the minimisation is done very carefully and can therefore require a fairly large number of force evaluations. however, it is possible to adjust the various parameters to increase the efficiency of the minimisation.

Starting from a minimum config *ini*, ART nouveau searches for a neighboring saddle point *sad*, and then relaxes the system towards a minimum *fin*. The program defined in source/path.f90, aims at building the path of steepest descent connecting *sad* to *ini*, and that connecting *sad* to *fin*.

A. Description

The path of steepest descent is built by solving the following equation:

$$\frac{dx(s)}{ds} = -\frac{\nabla\mathcal{V}(x(s))}{\|\nabla\mathcal{V}(x(s))\|} \tag{2}$$

with $\mathcal{V}(x(s))$ the potential energy of the system at the position $x(s)$. Since the starting point is the saddle point, where $\nabla\mathcal{V}(x(s)) = 0$, it is necessary to initially push the system in an arbitrary direction. This is done along the eigenvector of negative eigenvalue.

Then, the integration scheme is:

$$x_{n+1} = x_n - \alpha_n \nabla\mathcal{V}(x_n) \tag{3}$$

At each step, α_n is initialized using $\alpha_n = \min\{ALPHA\_MAX\_CP,\ \frac{DMAX\_CP}{\|\nabla\mathcal{V}(x_n)\|_\infty}\}$. $DMAX\_CP$ is calculated as $\frac{\|\vec{r}_{ini \text{ or } fin} - \vec{r}_{sad}\|}{2000}$. The system is moved according to eq. 3 with α_n as the step length, unless:

$$\mathcal{V}(x_{n+1}) \geq \mathcal{V}(x_n) \tag{4}$$

In this latter case, the step length is decreased, $\alpha_n = \alpha_n \times ALPHA\_REDUCE\_CP$, and the equality in eq. 4 is tested again.

When $\|\nabla\mathcal{V}(x_n)\|_\infty < FTHRESHOLD\_CP$, the steepest descent is finished, and the fire minimizer is used if criteria are not met.

B. Compilation

To compile path.f90, follow these two steps:

```
$ cd $ARTDIR/source

$ make prog=path machine_architecture
```

C. Usage

To understand the usage of path, we will rely on the example located in example/a-Si-LAMMPS. First, run an ART nouveau search, and then try to connect the saddle point to the minima:

```
$ cd $ARTDIR/example/a-Si-LAMMPS

$ bart.sh new

$ path.sh -ini min1000 -sad sad1001 -fin min1001
```

While descending from the saddle towards the minima, several quantities are written on the screen, in columns:

$$step \;\; npart \;\; nalpha \;\; evalf\_nb \;\; alpha \;\; length \;\; dr\_min \;\; de\_ini \;\; fdotf \;\; finf$$

- *step*: Number of times the system has been displaced.
- *npart*: Number of particles which were displaced between current config and target minimum by more than DR_NPART_CP.
- *nalpha*: Number of times alpha was decreased.
- $evalf\_nb$: Number of force evaluations so far.
- *alpha*: Magnitude of displacement increment: step length.
- *length*: Curvilinear abscissa of the path, increasing from *ini* to *fin* and shifted when trying to connect the three states textitsad, *ini* and *fin* so that it is equal to zero at *sad*.
- *dr_min*: Euclidian distance between current position and target minimum.
- *de_ini*: Difference of potential energy between the current position $de\_ini = e_{current} - e_{ini}$.
- $fdotf$: Dot product of force vector on current config with itself.
- $finf$: Uniform norm of force vector on current config.

These quantities are also written in a file, named path_sd.ssv.

At the end of the descent, a file connectivity.ssv is also generated. It contains three columns:

- *nb_pts*: This is the number of configurations given as argument to path: *ini* and *sad* or *ini*, *sad*, and *fin*.
- *connected_ini*: Equal to 0 if the saddle point is not connected to the initial minimum, that is if the path of steepest descent started from the saddle point in the direction of the initial minimum does not converge to this latter; equal to 1 if they are connected.
- $connected\_fin$: Same as above, but for the saddle point and the final minimum.

D. Parameters

Parameters that need to be set to generate the path of steepest descent:

- Parameters used to build the path.
  - MAX_ITER_CP: Maximum number of steps for building the whole path.
  - $FTHRESHOLD\_CP$: Criterium to stop building the steepest descent: $finf < FTHRESHOLD\_CP$. It can not be lower that the uniform norm of the force field in the target minimum: *ini* or *fin*.
  - $LSTORE\_CP$: Store configs along the path or not.
  - DS_STORE_CP: Store a config every time the curvilinear abscissa has increased of DS_STORE_CP.
  - DR_NPART_CP: Used to compute $npart$, the number of particles which were displaced between current config and target minimum by more than DR_NPART_CP.
  - $INTEGRATOR\_CP$: Type of integrator used to build the path, for now only 'euler' (Euler explicit scheme) is implemented.
  - $ALPHA\_MAX\_CP$: Maximum value of $alpha$ (see above, description).
  - $ALPHA\_REDUCE\_CP$: Prefactor to reduce $alpha$ when pushing along the gradient increases energy.
- Parameters used to decide if two end point of a path *end*, obtained after the fire minimization, is equal to *ini* or *fin*.
  - $DE\_CONNECTED\_CP$: if $\|e_{end} - e_{ini \text{ or } fin}\| < DE\_CONNECTED\_CP$, test the next parameter, else the configurations are at the same position.
  - $DR\_CONNECTED\_CP$: if $\|\vec{r}_{end} - \vec{r}_{ini \text{ or } fin}\| < DR\_CONNECTED\_CP$, the configs are at the same position, else not.
- Output file names
  - $SD\_LOGFILE\_CP$: path_sd.ssv
  - $CONNECTIVITY\_FILE\_CP$: connectivity.ssv
- Parameters used by fire, to finish converging to local minima after steepest descent has finished.
  - $NORM\_CRITERIUM\_FIRE$: See above in this doc.
  - $FMAX\_CRITERIUM\_FIRE$: See above in this doc.
  - DT_MAX_FIRE: See above in this doc.
- Parameters used to compute the eigenvector of negative eigenvalue at the saddle point, and push the system.
  - $Prefactor\_Push\_Over\_Saddle$: Step length to push the system at the saddle point. It is decreased until the energy change is negative.
  - $Number\_Lanczos\_Vectors$: Number of vectors used in Lanczos procedure.
  - $delta\_disp\_Lanczos$: Displacement used to estimate the product of the Hessian with vectors.
  - $Lanczos\_collinear$: Stop iterating for the scalar product between the current and the previous eigenvectors is higher than $Lanczos\_collinear$.