Performing advanced MD simulations with ORAC

Giorgio F. Signorini
Università di Firenze (IT)

2011, march 22
Cycle “Modélisation de Macromolécules Biologiques”
Université di Cergy-Pontoise (FR)

last update: May 10, 2011

ORAC

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator
- electrostatic interactions can be handled with the Smooth Particle Mesh Ewald method

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator
- electrostatic interactions can be handled with the Smooth Particle Mesh Ewald method

In the latest release (5.1):

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator
- electrostatic interactions can be handled with the Smooth Particle Mesh Ewald method

In the latest release (5.1):

- Replica Exchange MD

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator
- electrostatic interactions can be handled with the Smooth Particle Mesh Ewald method

In the latest release (5.1):

- Replica Exchange MD
- Steered MD

ORAC rel 5.1

ORAC is a program for running classical simulations of biomolecules at the atomistic level.

- Simulations can be carried out in the NVE, NPT, NHP, and NVT thermodynamic ensembles.
- The integration of the equations of motion in any ensemble is carried out with the r-RESPA multiple time step integrator
- electrostatic interactions can be handled with the Smooth Particle Mesh Ewald method

In the latest release (5.1):

- Replica Exchange MD
- Steered MD
- Metadynamics

Home Page

Home page:

http://www.chim.unifi.it/orac

References

[1] P. Procacci, E. Paci, T. Darden, and M. Marchi. ORAC: A Molecular Dynamics Program to Simulate Complex Molecular Systems with Realistic Electrostatic Interactions. *J. Comput. Chem.*, 18:1848–1862, 1997.

[2] S. Marsili, G. F. Signorini, R. Chelli, M. Marchi, and P. Procacci. ORAC: A molecular dynamics simulation program to explore free energy surfaces in biomolecular systems at the atomistic level. *J. Comput. Chem.*, 31:1106–1116, 2010.

ORAC is an open project

- written originally in Fortran77; evolution in fortran90

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:
  - Serial Generalized-Ensemble methods

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:
    - Serial Generalized-Ensemble methods
- near-future developents:

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:
  - Serial Generalized-Ensemble methods
- near-future developents:
  - Coarse-Grained Potential [Ha-Duong, Borgis et al.]

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:
  - Serial Generalized-Ensemble methods
- near-future developents:
  - Coarse-Grained Potential [Ha-Duong, Borgis et al.]
  - Implicit solvent: Analytical Generalized Born plus NonPolar (AGBNP) model [Gallicchio et al.]

ORAC is an open project

- written originally in Fortran77; evolution in fortran90
- extensions and new features can be added to the package rather easily
- in next release:
  - Serial Generalized-Ensemble methods
- near-future developents:
  - Coarse-Grained Potential [Ha-Duong, Borgis et al.]
  - Implicit solvent: Analytical Generalized Born plus NonPolar (AGBNP) model [Gallicchio et al.]

In practice

Be Unix!

- ORAC runs on UNIX systems only.

Be Unix!

- ORAC runs on UNIX systems only.
- Prerequisites:

Be Unix!

- ORAC runs on UNIX systems only.
- Prerequisites:
  - GNU Make
  - GCC 4.3 or higher
  - other Fortran 90 compilers that are known to work:

Be Unix!

- ORAC runs on UNIX systems only.
- Prerequisites:
  - GNU Make
  - GCC 4.3 or higher
  - other Fortran 90 compilers that are known to work:
    - Intel
    - IBM xlf family
    - . . .

(see file BUILDING)

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/
```

Anatomy of a distribution


```
   orac.5_1_1.rev257/
 |-- BUILDING
 |-- COPYRIGHT_NOTICE
 |-- Makefile
 |-- README
 |-- RELEASE
 |-- doc/  # The manual
 |-- lib/  # Force Field (Parameter and Topology files)
 |-- pdb/  # Input Molecular Structures
```

Anatomy of a distribution


```
 orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
|-- tests/
```

Anatomy of a distribution


```
 orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
|-- tests/  # Tests (more on those later)
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
|-- tests/  # Tests (more on those later)
`-- tools/
```

Anatomy of a distribution


```
 orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
|-- tests/  # Tests (more on those later)
`-- tools/  # auxiliary programs
```

Anatomy of a distribution


```
  orac.5_1_1.rev257/
|-- BUILDING
|-- COPYRIGHT_NOTICE
|-- Makefile
|-- README
|-- RELEASE
|-- doc/  # The manual
|-- lib/  # Force Field (Parameter and Topology files)
|-- pdb/  # Input Molecular Structures
|-- src/  # Program Sources; also, executable after compilation
|-- tests/  # Tests (more on those later)
`-- tools/  # auxiliary programs
```

Look for README files in each directory for help!

Compiling

- to list all available compilation targets:

Compiling

- to list all available compilation targets:

```
$ cd orac-5_1_x
$ make show
```

Compiling

- to list all available compilation targets:

```
$ cd orac-5_1_x
$ make show
AVAILABLE COMPILATION TARGETS:
   default
   gfortran
   Intel
   IBM
   ...
   PARALLEL
   gfortran_PARALLEL
   Intel_PARALLEL
   IBM_PARALLEL
   ...
```

Compiling

- to list all available compilation targets:

```
$ cd orac-5_1_x
$ make show
AVAILABLE COMPILATION TARGETS:
   default
   gfortran
   Intel
   IBM
   ...
   PARALLEL
   gfortran_PARALLEL
   Intel_PARALLEL
   IBM_PARALLEL
   ...
```

- to create the executable `orac_XXXXX` in directory `src/default`:

Compiling

- to list all available compilation targets:

```
$ cd orac-5_1_x
$ make show
AVAILABLE COMPILATION TARGETS:
   default
   gfortran
   Intel
   IBM
   ...
   PARALLEL
   gfortran_PARALLEL
   Intel_PARALLEL
   IBM_PARALLEL
   ...
```

- to create the executable `orac_XXXXX` in directory `src/default`:

```
$ make default
```

Compiling

- to list all available compilation targets:

```
$ cd orac-5_1_x
$ make show
AVAILABLE COMPILATION TARGETS:
   default
   gfortran
   Intel
   IBM
   ...
   PARALLEL
   gfortran_PARALLEL
   Intel_PARALLEL
   IBM_PARALLEL
   ...
```

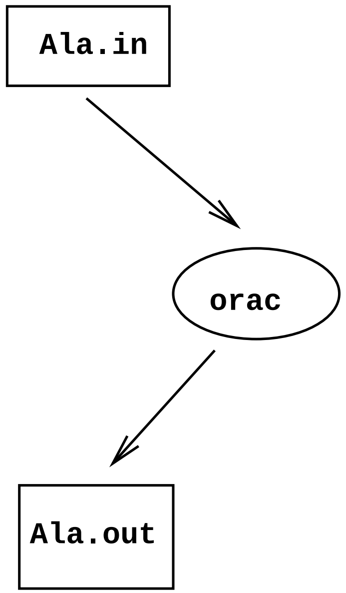
- to create the executable `orac_XXXXX` in directory `src/default`:

```
$ make default
```

- edit `src/config.H` and recompile, to change maximum array dimensions

Input and Output files

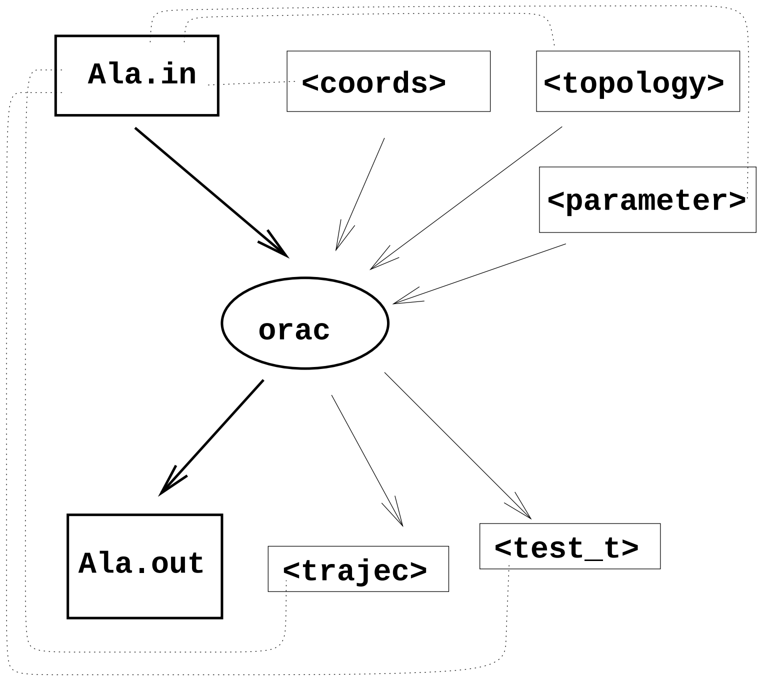
- Standard input and standard output:




```
orac < Ala.in > Ala.out
```

Input and Output files

- Auxiliary files whose names are defined in main input:




```
orac < Ala.in > Ala.out
```

The main input file

- The main input file has a block structure
  - each block begins with &KEYWORD and ends with &END

The main input file

- The main input file has a block structure
  - each block begins with &KEYWORD and ends with &END
- Block order does not matter
  All the input is read first; then the required computations are started

The main input file (2)


```
&SETUP
   CRYSTAL  62.0 62.0 62.0 90.0 90.0 90.0
   READ_PDB ../../pdb/glc.pdb
&END
```

The main input file (2)


```
&SETUP
  CRYSTAL 62.0 62.0 62.0 90.0 90.0 90.0
  READ_PDB ../../pdb/glc.pdb
&END

&PARAMETERS
  READ_TPG_ASCII ../../lib/amber03.tpg
  READ_PRM_ASCII ../../lib/amber03.prm
  JOIN SOLUTE
    glc
  END
&END
```

The main input file (2)


```
&SETUP
   CRYSTAL  62.0 62.0 62.0 90.0 90.0 90.0
   READ_PDB ../../pdb/glc.pdb
&END

&PARAMETERS
   READ_TPG_ASCII ../../lib/amber03.tpg
   READ_PRM_ASCII ../../lib/amber03.prm
   JOIN SOLUTE
     glc
   END
&END

&POTENTIAL
   CUTOFF 12.0
   STRETCHING
   UPDATE 10.0 2.0
&END

&SIMULATION
   MINIMIZE
     CG 0.00001
   END
   FREQUENCIES
     print OPEN glc.frq
   END
&END

&RUN
  PRINT 10.0
  TIME 1000.0
&END
```

The main input file (2)


```
&SETUP
   CRYSTAL 62.0 62.0 62.0 90.0 90.0 90.0
   READ_PDB ../../pdb/glc.pdb
&END

&PARAMETERS
   READ_TPG_ASCII ../../lib/amber03.tpg
   READ_PRM_ASCII ../../lib/amber03.prm
   JOIN SOLUTE
     glc
   END
&END

&POTENTIAL
   CUTOFF 12.0
   STRETCHING
   UPDATE 10.0 2.0
&END

&SIMULATION
   MINIMIZE
     CG 0.00001
   END
   FREQUENCIES
     print OPEN glc.frq
   END
&END

&RUN
  PRINT 10.0
  TIME 1000.0
&END
```

- See section “Input to ORAC” in the manual for detailed description of input

The main output file

1. a copy of the input

The main output file

1. a copy of the input
2. output from startup operations (reading files, setup box, ...)

The main output file

1. a copy of the input
2. output from startup operations (reading files, setup box, ...)
3. intermediate simulation results:

The main output file

1. a copy of the input
2. output from startup operations (reading files, setup box, ...)
3. intermediate simulation results:

```
      Tstep  =      60.0 Total =  -2889.889 TotPot = -8624.839
      Coulomb = -14817.630 Recipr = -8874.305 NonBond = -15058.432
      Ener14 =   1285.083 Bonded = 6433.593 Stretch = 2125.946
      Angle  =   1728.847 I-Tors =  88.797 P-Tors = 2490.003
      TotTemp =  328.854 Hoover =    54.3 ResTemp = 323.494
      TraTemp =  367.831

<------ Dumping Restart File ------->

Neighbor Lists Dimensions *neighbor( 36855)*



      Tstep  =      72.0 Total =  -2918.812 TotPot = -8891.027
      Coulomb = -15052.095 Recipr = -9031.840 NonBond = -15282.772
      Ener14 =   1264.188 Bonded = 6391.744 Stretch = 2014.560
      Angle  =   1780.318 I-Tors = 107.506 P-Tors = 2489.360
      TotTemp =  342.152 Hoover =   157.6 ResTemp = 333.165
      TraTemp =  407.513

Velocities have been rescaled ---->
```

Analysis of results

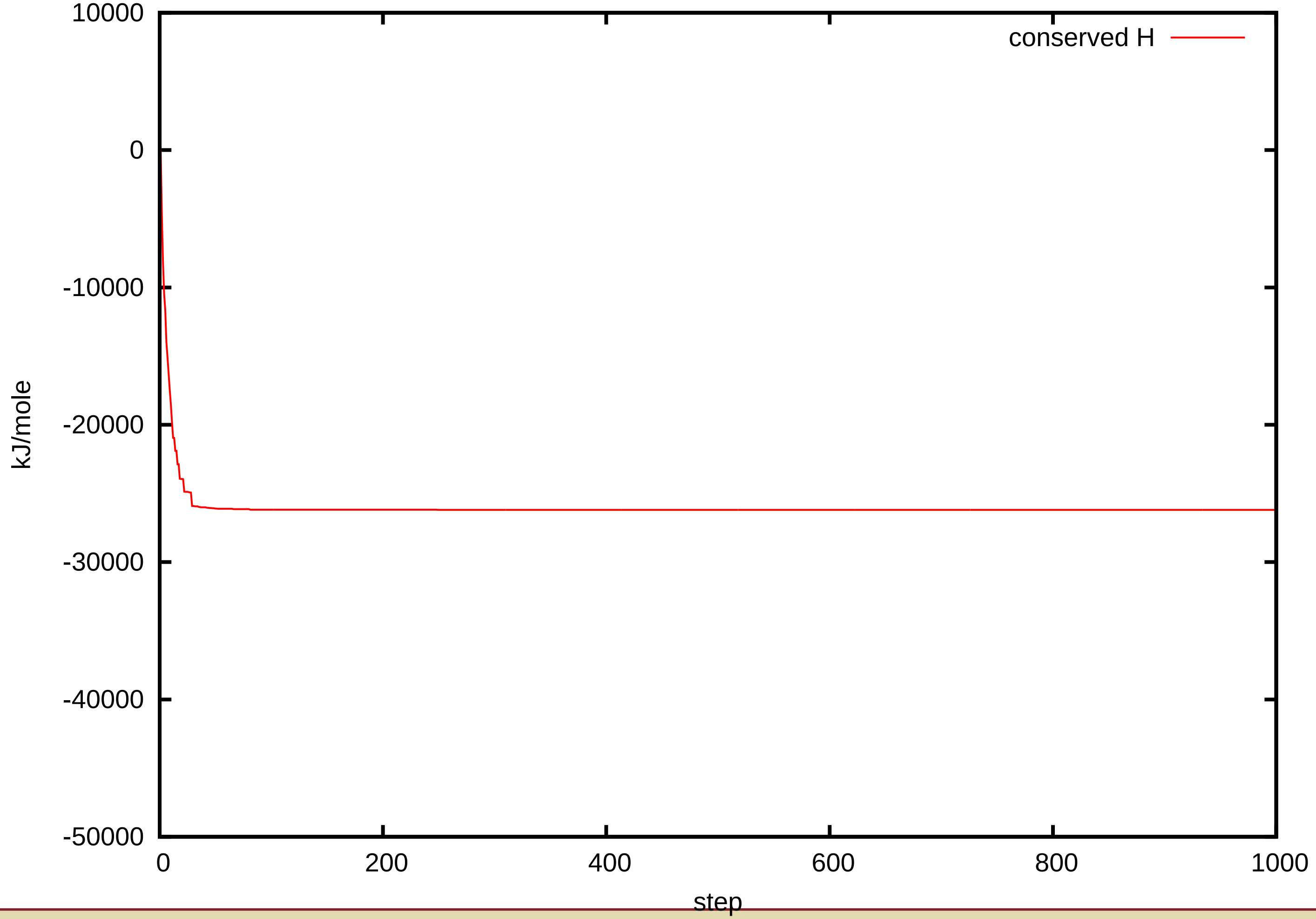
- use tools:
  - orac-post-proc
  - analysis
  - rmsd90

Examples

A basic example

NpT simulation of a small protein in water, with initial equilibration

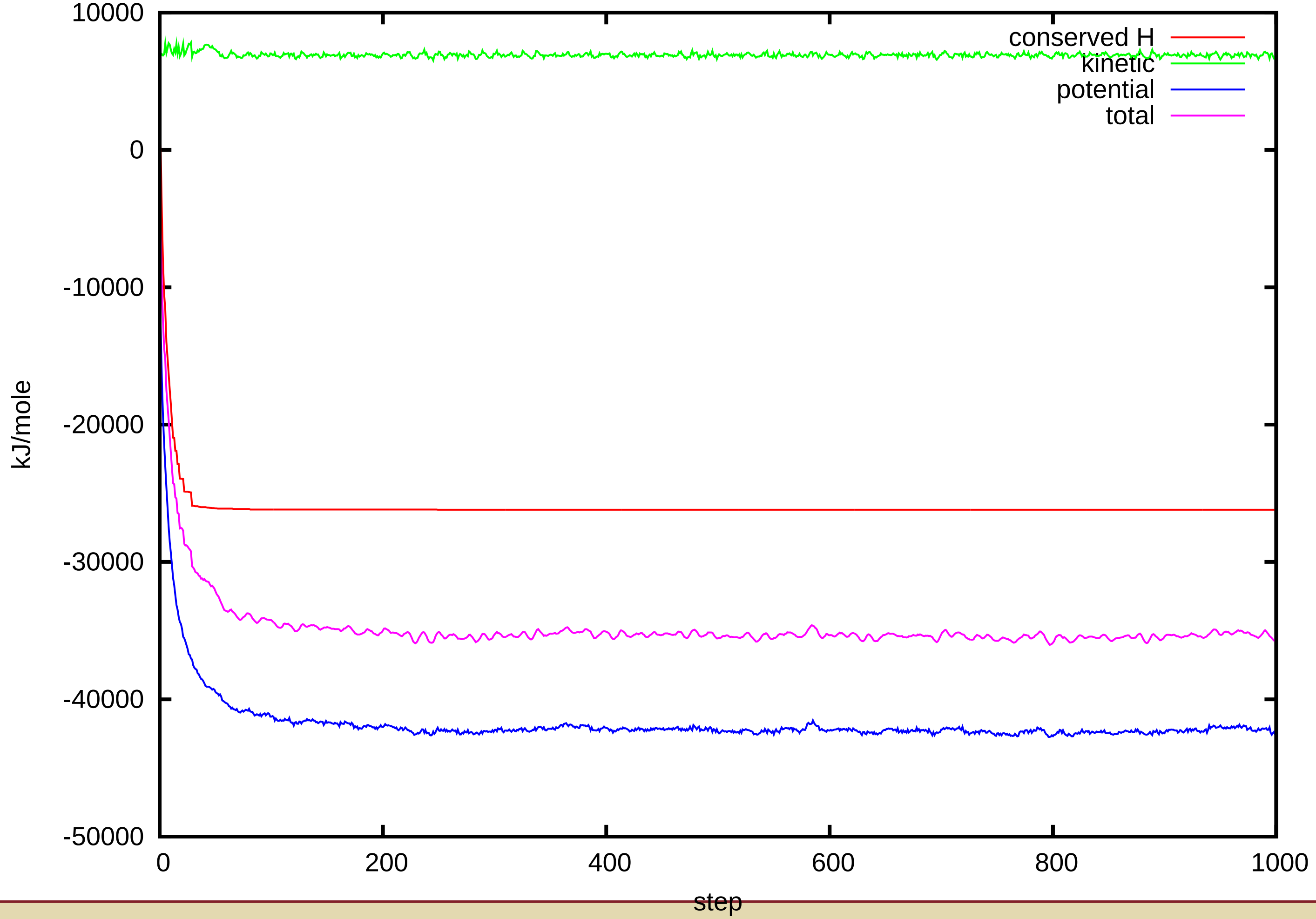
- Conservation of Hamiltonian:



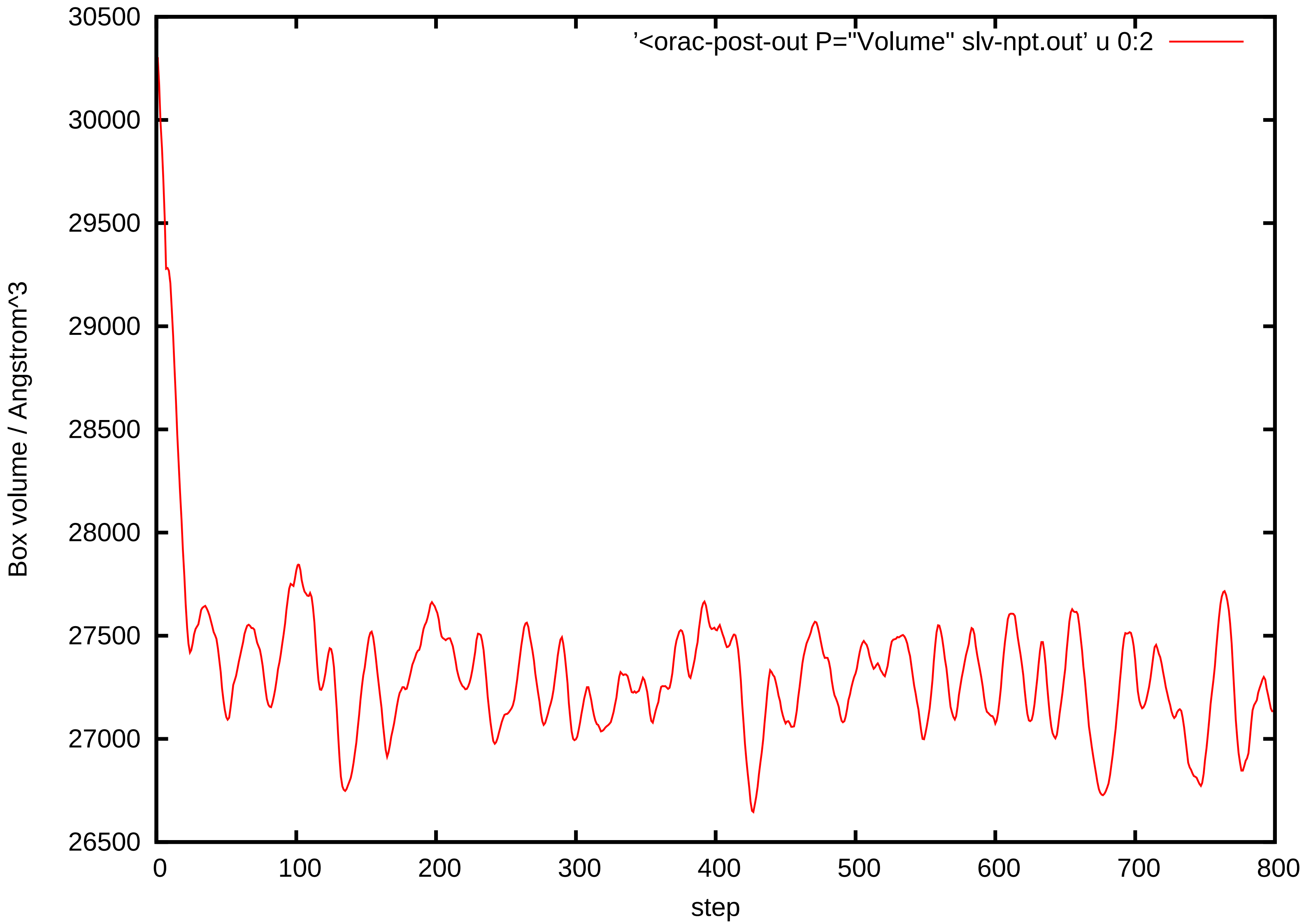
A basic example

NpT simulation of a small protein in water, with initial equilibration

- Energies of the real system:



plot selected properties from output



compute structural properties with `analysis`

`analysis` is a small, extensible auxiliary program to compute structural properties from PDB trajectory:

```
$ analysis < ana.in
```

compute structural properties with `analysis`

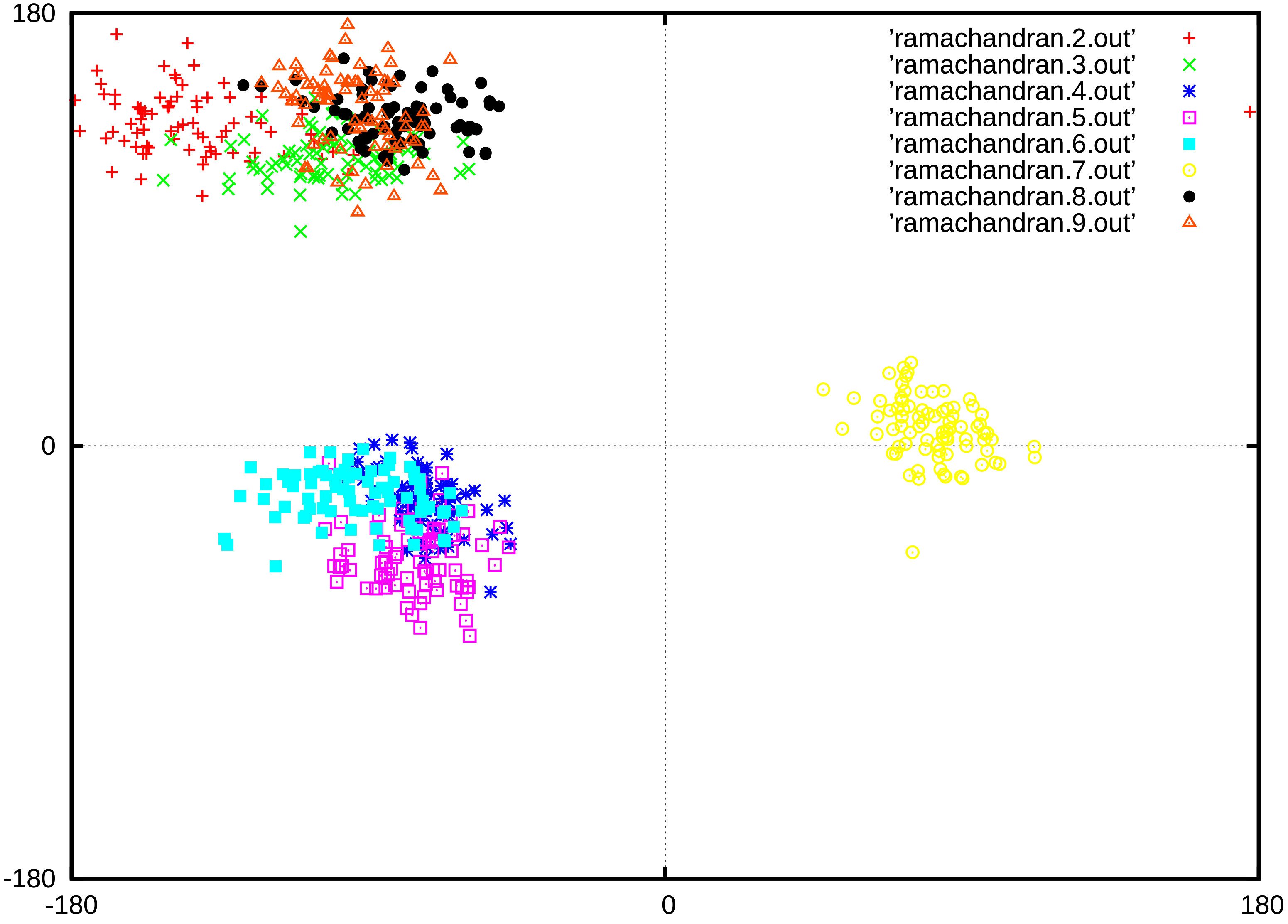
`analysis` is a small, extensible auxiliary program to compute structural properties from PDB trajectory:

```
$ analysis < ana.in
```

The file `ana.in` looks like the following:

```
   # name of the PDB file to analize:
slv.pdb
   #
& ramachandran 1
& ramachandran 2
& ramachandran 3
& ramachandran 4
& ramachandran 5
& ramachandran 6
& ramachandran 7
& ramachandran 8
& ramachandran 9
```

compute structural properties with `analysis` (2)



Tests

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD
- Metadynamics

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD
- Metadynamics

ORAC tests serve two purposes:

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD
- Metadynamics

ORAC tests serve two purposes:

- check if the function is working and produces the expected results

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD
- Metadynamics

ORAC tests serve two purposes:

- check if the function is working and produces the expected results
- provide sample input files

ORAC tests

The ORAC distribution includes a test suite for each functionality:

- basic
- Replica Exchange MD
- Steered MD
- Metadynamics

ORAC tests serve two purposes:

- check if the function is working and produces the expected results
- provide sample input files

Tests are designed to be very short! no real-world calculation!

You need to go parallel

Some of the tests run in parallel. This means you will need

You need to go parallel

Some of the tests run in parallel. This means you will need

1. to set up an MPI parallel environment, such as OpenMPI or MPICH2. For example, in ubuntu:

```
$ sudo apt-get install mpich2
```

You need to go parallel

Some of the tests run in parallel. This means you will need

1. to set up an MPI parallel environment, such as OpenMPI or MPICH2. For example, in ubuntu:

   ```
   $ sudo apt-get install mpich2
   ```

   then configure it

   (see README_PARALLEL files, and `http://www.mcs.anl.gov/research/projects/mpich2`)

You need to go parallel

Some of the tests run in parallel. This means you will need

1. to set up an MPI parallel environment, such as OpenMPI or MPICH2. For example, in ubuntu:

   ```
   $ sudo apt-get install mpich2
   ```

   then configure it

   (see README_PARALLEL files, and `http://www.mcs.anl.gov/research/projects/mpich2`)

2. to build the parallel version of the program, e.g.:

   ```
   $ cd orac-5_1_x/
   $ make PARALLEL
   ```

Is this what you expected?

- at the end of each test, you may find that the output is different from the reference data listed in the package

Is this what you expected?

- at the end of each test, you may find that the output is different from the reference data listed in the package
- this is not necessarily an error:

Is this what you expected?

- at the end of each test, you may find that the output is different from the reference data listed in the package
- this is not necessarily an error:
  - the evolution of a REM simulation, or of a steered MD trajectory, depend critically on random number generators, that may work differently in your hw/sw environment, soon driving your simulation away from the reference one.

Is this what you expected?

- at the end of each test, you may find that the output is different from the reference data listed in the package
- this is not necessarily an error:
  - the evolution of a REM simulation, or of a steered MD trajectory, depend critically on random number generators, that may work differently in your hw/sw environment, soon driving your simulation away from the reference one.
  - just check that results are *reasonable and similar* to what is expected

REM Tests

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

- N copies of the system are simulated, with potential scaling coefficients

$$c_1 = 1, c2, \ldots, c_N$$

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

- N copies of the system are simulated, with potential scaling coefficients

  $$c_1 = 1, c2, \ldots, c_N$$

  and exchanges between adjacent copies are attempted at definite intervals

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

- N copies of the system are simulated, with potential scaling coefficients

  $$c_1 = 1, c2, \ldots, c_N$$

  and exchanges between adjacent copies are attempted at definite intervals
- In what follows:

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

- N copies of the system are simulated, with potential scaling coefficients

$$c_1 = 1, c2, \ldots, c_N$$

  and exchanges between adjacent copies are attempted at definite intervals
- In what follows:
  - "replica" or "temperature" designates one value of the scaled potential

Hamiltonian REM

Hamiltonian Replica Exchange MD is implemented in ORAC

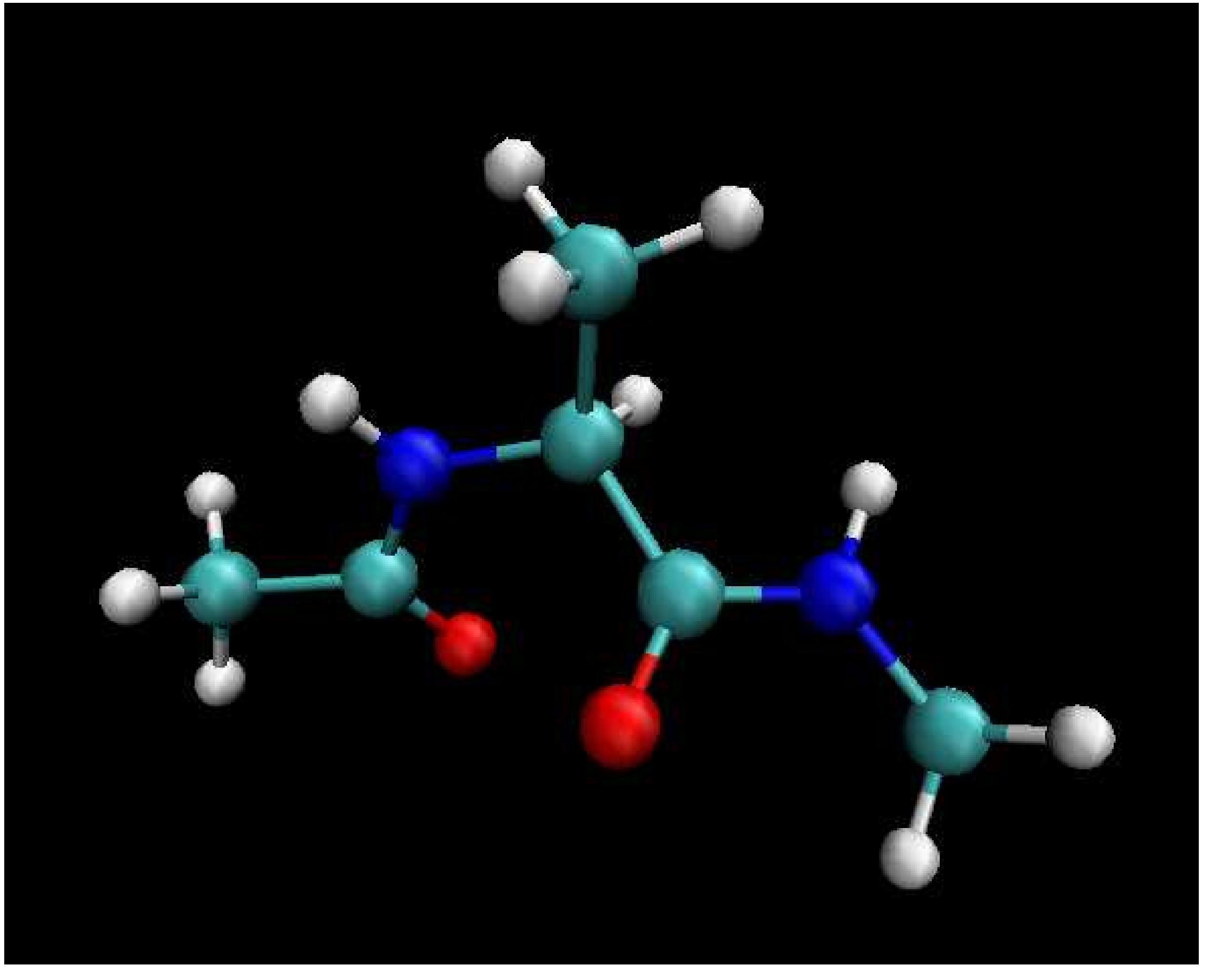
- N copies of the system are simulated, with potential scaling coefficients

$$c_1 = 1, c2, \ldots, c_N$$

  and exchanges between adjacent copies are attempted at definite intervals
- In what follows:
  - "replica" or "temperature" designates one value of the scaled potential
  - "trajectory" designates one simulation:
    "one *trajectory* explores different *temperatures*"

A simple REM example

Figure 1: Alanine dipeptide



input file `1.in`

- REM Test n.1: run a cold start of Alanine dipeptide, on 8 processors.

```
&REM
   SETUP 0.75 1
   STEP 5.
   PRINT 1000.
&END
```

- This will run a REMD simulation with replicas having a scaling coefficient going from 1 to 0.75, (equivalent to temperature going from T to $\frac{T}{0.75}$), attempting a swap every $5.fs$

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
```

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
$  # possibly edit Makefile
```

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
$  # possibly edit Makefile
$ make test1
```

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
$  # possibly edit Makefile
$ make test1
Starting test 1...
mpiexec -n 8 ../../src/PARALLEL/orac_Linux < 1.in
```

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
$  # possibly edit Makefile
$ make test1
Starting test 1...
mpiexec -n 8 ../../src/PARALLEL/orac_Linux < 1.in
```

NOTE that the number of processes is given as argument to `mpiexec`

Run the test

(Before actually running the tests, you may need some tweaking of the Makefile in that directory, e.g. setting `O_BIN_P` to a different path for the parallel executable).

```
$ cd orac-5_1_x/tests/REM_tests
$  # possibly edit Makefile
$ make test1
Starting test 1...
mpiexec -n 8 ../../src/PARALLEL/orac_Linux < 1.in
```

NOTE that the number of processes is given as argument to `mpiexec`
Each trajectory (process) outputs to a separate directory:

```
REM_tests/
|-- PAR0000/
|-- PAR0001/
|-- PAR0002/
|-- PAR0003/
|-- PAR0004/
|-- PAR0005/
|-- PAR0006/
`-- PAR0007/
```

What's in a PARxxxx

In each subdirectory one finds output from one trajectory.
A useful file is REM_DIAGNOSTICS:

```
Time(fs) Ens.Index Etot Ekin  Epot     Unscaled_Epot(1) Unscaled_Epot(2) Unscaled_Epot(3)
   0.000       1 -15798.591 3975.097 -20294.758 36.86108963 66.27519055 -20397.89417554
   9.000       1 -15799.197 3941.835 -20268.162 48.33958736 69.89173141 -20386.39351932
  18.000       1 -15799.348 3849.030 -20180.790 36.61301332 64.85898411 -20282.26240292
  27.000       1 -15799.181 3885.160 -20221.080 59.92605948 76.18079698 -20357.18730765
...
1350.000       1 -15810.839 3921.483 -20032.673 44.99945318 71.23901455 -20148.91159616
1359.000       1 -15810.624 3863.698 -19971.024 55.31566941 65.84458965 -20092.18387400
1368.000       2 -15005.928 3832.054 -19934.166 46.66017953 64.12593199 -20044.95241986
1377.000       2 -15008.400 3910.391 -20015.584 55.69435608 66.28114096 -20137.55997783
1386.000       1 -15812.089 3905.204 -20005.686 39.16567754 64.59704046 -20109.44839358
1926.000       2 -15013.292 3673.233 -19566.784 71.44607447 65.80181736 -19704.03189459
1935.000       2 -15013.742 3753.891 -19648.255 48.80251814 72.37553816 -19769.43267586
2475.000       2 -15017.244 3886.572 -19709.729 65.61279004 59.80027938 -19835.14226040
2484.000       2 -15015.418 3799.623 -19619.400 46.35157101 59.93559335 -19725.68686377
2493.000       3 -14258.520 3823.324 -19648.693 66.39847825 60.42549336 -19775.51746544
2502.000       3 -14257.609 3846.527 -19674.834 47.17524086 60.32764207 -19782.33721673
8892.000       5 -12883.864 4017.228 -18906.366 85.57769971 63.23855096 -19055.18207049
8901.000       4 -13556.068 3933.928 -18804.655 67.43223259 61.38770229 -18933.47506054
8910.000       4 -13556.754 3912.949 -18776.002 84.37418764 73.93300859 -18934.30906146
8919.000       5 -12887.471 3860.507 -18708.009 84.38999028 69.17496051 -18861.57386744
```

- column 2 is the replica (=”temperature”) index applied to that trajectory at that time

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
==== current simulation time = 900.0
```

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
==== current simulation time = 900.0
==== current simulation time = 1800.0
```

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
==== current simulation time = 900.0
==== current simulation time = 1800.0
   Exchange number:    112
   1 < = >  2 Nacc/N% 0.000
   2 < = >  3 Nacc/N% 14.286
   3 < = >  4 Nacc/N% 0.000
   4 < = >  5 Nacc/N% 37.500
   5 < = >  6 Nacc/N% 0.000
   6 < = >  7 Nacc/N% 7.143
   7 < = >  8 Nacc/N% 0.000
```

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
==== current simulation time = 900.0
==== current simulation time = 1800.0
   Exchange number:    112
   1 < = >  2 Nacc/N% 0.000
   2 < = >  3 Nacc/N% 14.286
   3 < = >  4 Nacc/N% 0.000
   4 < = >  5 Nacc/N% 37.500
   5 < = >  6 Nacc/N% 0.000
   6 < = >  7 Nacc/N% 7.143
   7 < = >  8 Nacc/N% 0.000
==== current simulation time = 2700.0
```

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
           ==== current simulation time = 900.0
           ==== current simulation time = 1800.0
              Exchange number:    112
              1 < = >  2 Nacc/N% 0.000
              2 < = >  3 Nacc/N% 14.286
              3 < = >  4 Nacc/N% 0.000
              4 < = >  5 Nacc/N% 37.500
              5 < = >  6 Nacc/N% 0.000
              6 < = >  7 Nacc/N% 7.143
              7 < = >  8 Nacc/N% 0.000
           ==== current simulation time = 2700.0
 ...
```

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

```
           ==== current simulation time = 900.0
           ==== current simulation time = 1800.0
              Exchange number:    112
              1 < = >  2 Nacc/N% 0.000
              2 < = >  3 Nacc/N% 14.286
              3 < = >  4 Nacc/N% 0.000
              4 < = >  5 Nacc/N% 37.500
              5 < = >  6 Nacc/N% 0.000
              6 < = >  7 Nacc/N% 7.143
              7 < = >  8 Nacc/N% 0.000
           ==== current simulation time = 2700.0
 ...
```

When program ends, you will probably want to check whether

- the exchanges are efficient ($\geq 20\%$)

output to terminal

Standard output reports program progression, and the exchange ratio between couples of replicas.

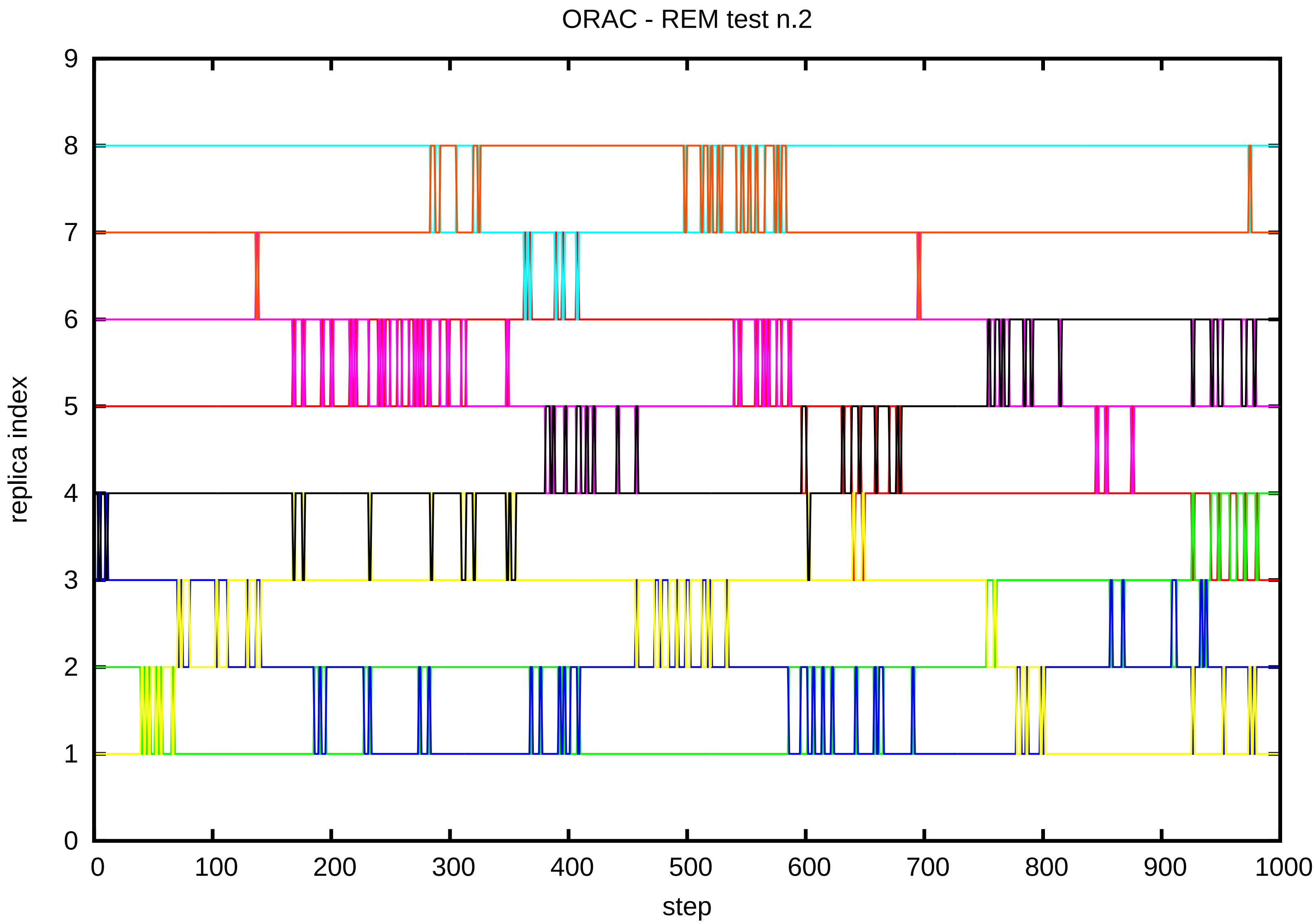
```
           ==== current simulation time = 900.0
           ==== current simulation time = 1800.0
              Exchange number:    112
              1 < = >  2 Nacc/N% 0.000
              2 < = >  3 Nacc/N% 14.286
              3 < = >  4 Nacc/N% 0.000
              4 < = >  5 Nacc/N% 37.500
              5 < = >  6 Nacc/N% 0.000
              6 < = >  7 Nacc/N% 7.143
              7 < = >  8 Nacc/N% 0.000
           ==== current simulation time = 2700.0
 ...
```

When program ends, you will probably want to check whether

- the exchanges are efficient ($\geq 20\%$)
- each trajectory explores all "temperatures"

exchange chart

ORAC REM test n.2. Exchanges between replicas



total energy

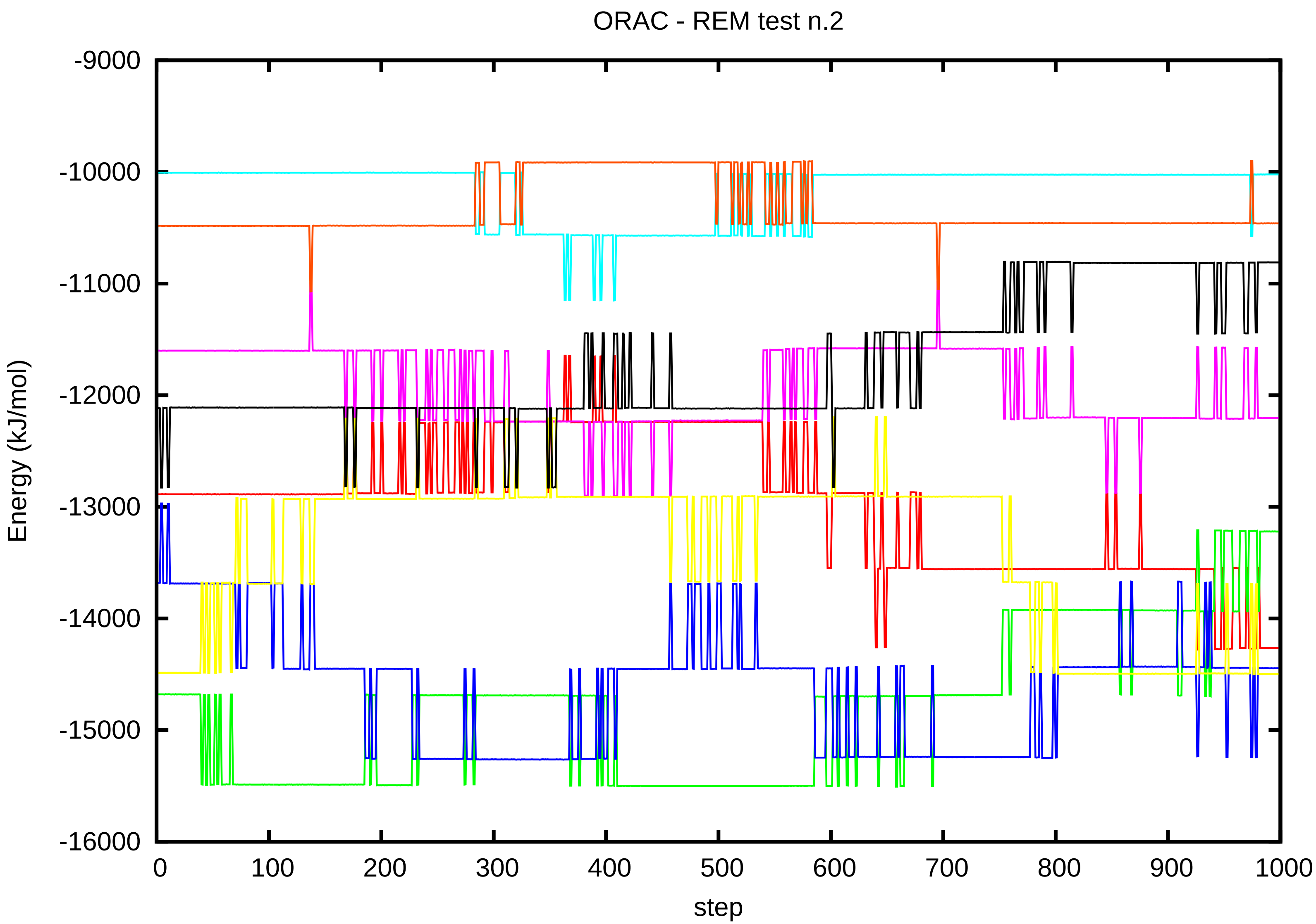


Figure 2: ORAC REM test n.2. Energy of trajectories

collecting data for the target potential

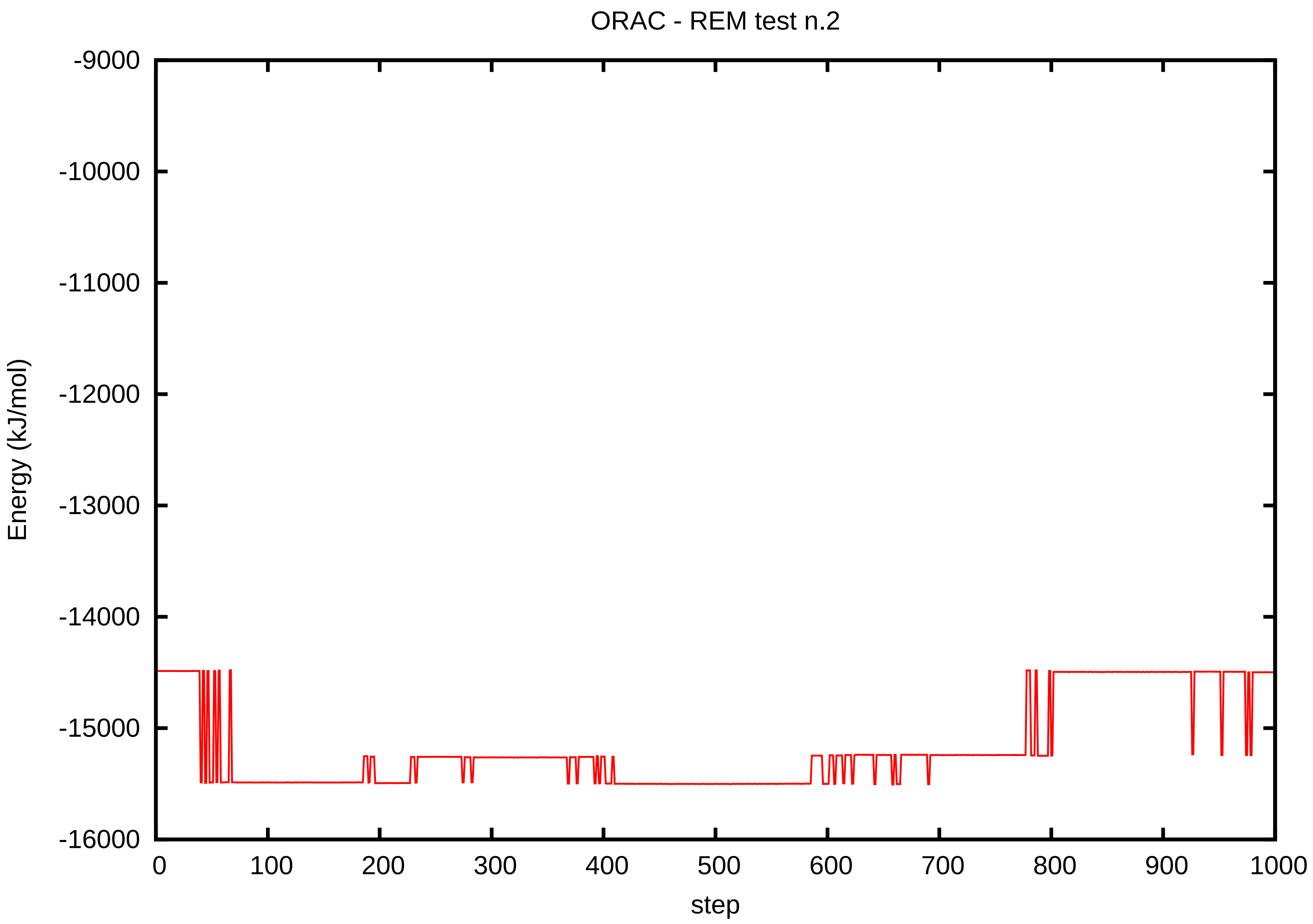


Figure 3: ORAC REM test n.2. Energy of target potential (replica 1)

how to reorder data by replica

using `order.sh`, energy and structural data can be reordered by replica:

```
$ cd orac-5_1_x/tests/REM_tests
$ make test2
```

how to reorder data by replica

using `order.sh`, energy and structural data can be reordered by replica:

```
$ cd orac-5_1_x/tests/REM_tests
$ make test2
$ ./order.sh 2.pdb REM_DIAGNOSTICS
```

how to reorder data by replica

using `order.sh`, energy and structural data can be reordered by replica:

```
$ cd orac-5_1_x/tests/REM_tests
$ make test2
$ ./order.sh 2.pdb REM_DIAGNOSTICS
$ ls
```

how to reorder data by replica

using `order.sh`, energy and structural data can be reordered by replica:

```
$ cd orac-5_1_x/tests/REM_tests
$ make test2
$ ./order.sh 2.pdb REM_DIAGNOSTICS
$ ls
2-0001.pdb # data from PAR????/2.pdb, replica=1
2-0002.pdb #                          replica=2
2-0003.pdb
2-0004.pdb
2-0005.pdb
2-0006.pdb
2-0007.pdb
2-0008.pdb
REM_DIAGNOSTIC-0001 # data from PAR????/REM_DIAGNOSTIC, replica=1
REM_DIAGNOSTIC-0002
REM_DIAGNOSTIC-0003
REM_DIAGNOSTIC-0004
REM_DIAGNOSTIC-0005
REM_DIAGNOSTIC-0006
REM_DIAGNOSTIC-0007
REM_DIAGNOSTIC-0008
$
```

REM efficiency

- The program provides a default set of scaling coefficients, with equal spacing between adjacent temperatures (this set gives optimal overlap in a harmonic oscillator collection)

REM efficiency

- The program provides a default set of scaling coefficients, with equal spacing between adjacent temperatures (this set gives optimal overlap in a harmonic oscillator collection)
- If replica exchange is not effective, you can

REM efficiency

- The program provides a default set of scaling coefficients, with equal spacing between adjacent temperatures (this set gives optimal overlap in a harmonic oscillator collection)

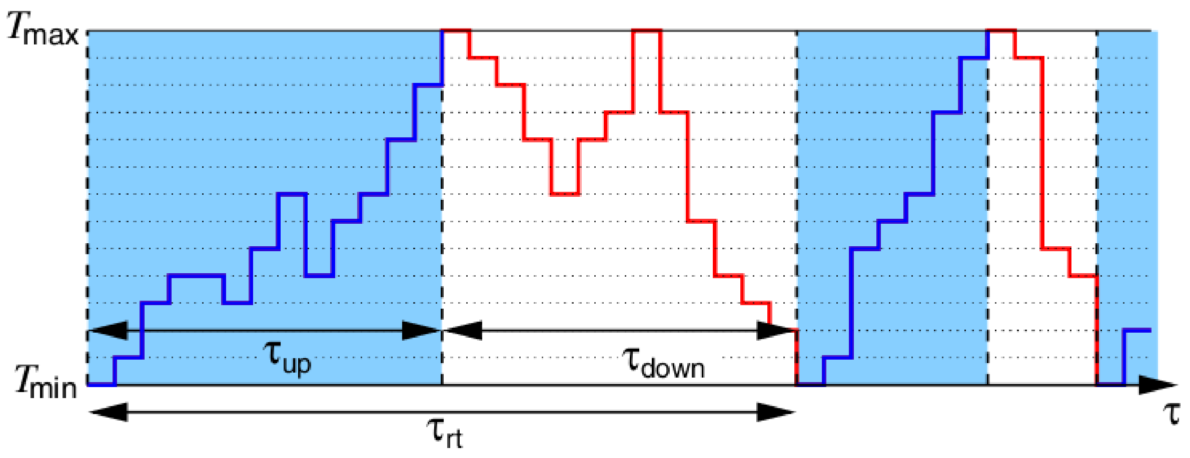
- If replica exchange is not effective, you can
  - increase the number of replicas

REM efficiency

- The program provides a default set of scaling coefficients, with equal spacing between adjacent temperatures (this set gives optimal overlap in a harmonic oscillator collection)

- If replica exchange is not effective, you can
  - increase the number of replicas
  - set the coefficients manually, decreasing the spacing between replicas that don't exchange well

monitoring trajectory flow



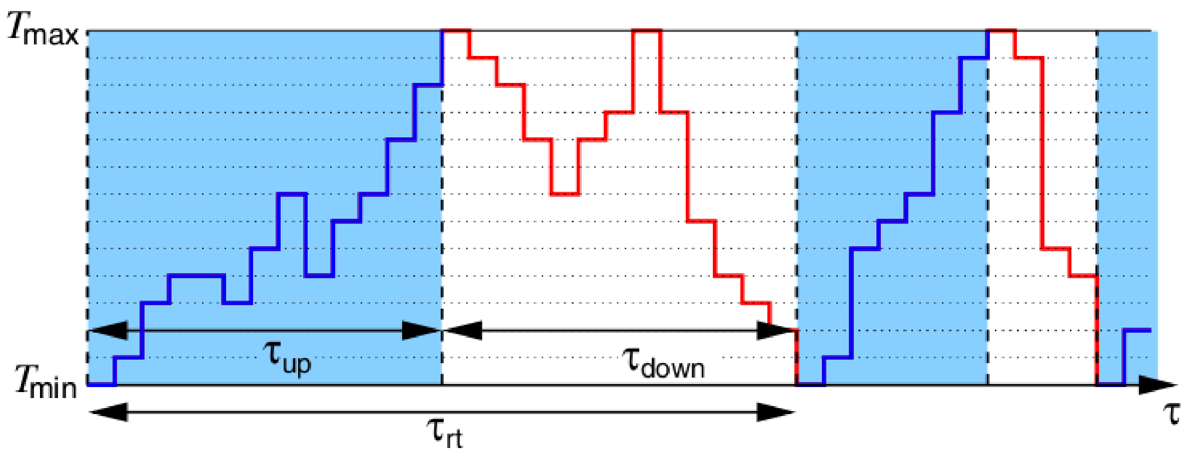
fraction of trajectories that are travelling "up" at replica i:

$$f_i^{up} = \frac{n_i^{up}}{n_i^{up} + n_i^{down}}$$

diffusivity at replica i:

$$D_i = \frac{\Delta T}{df/dT}$$

monitoring trajectory flow



fraction of trajectories that are travelling "up" at replica i:

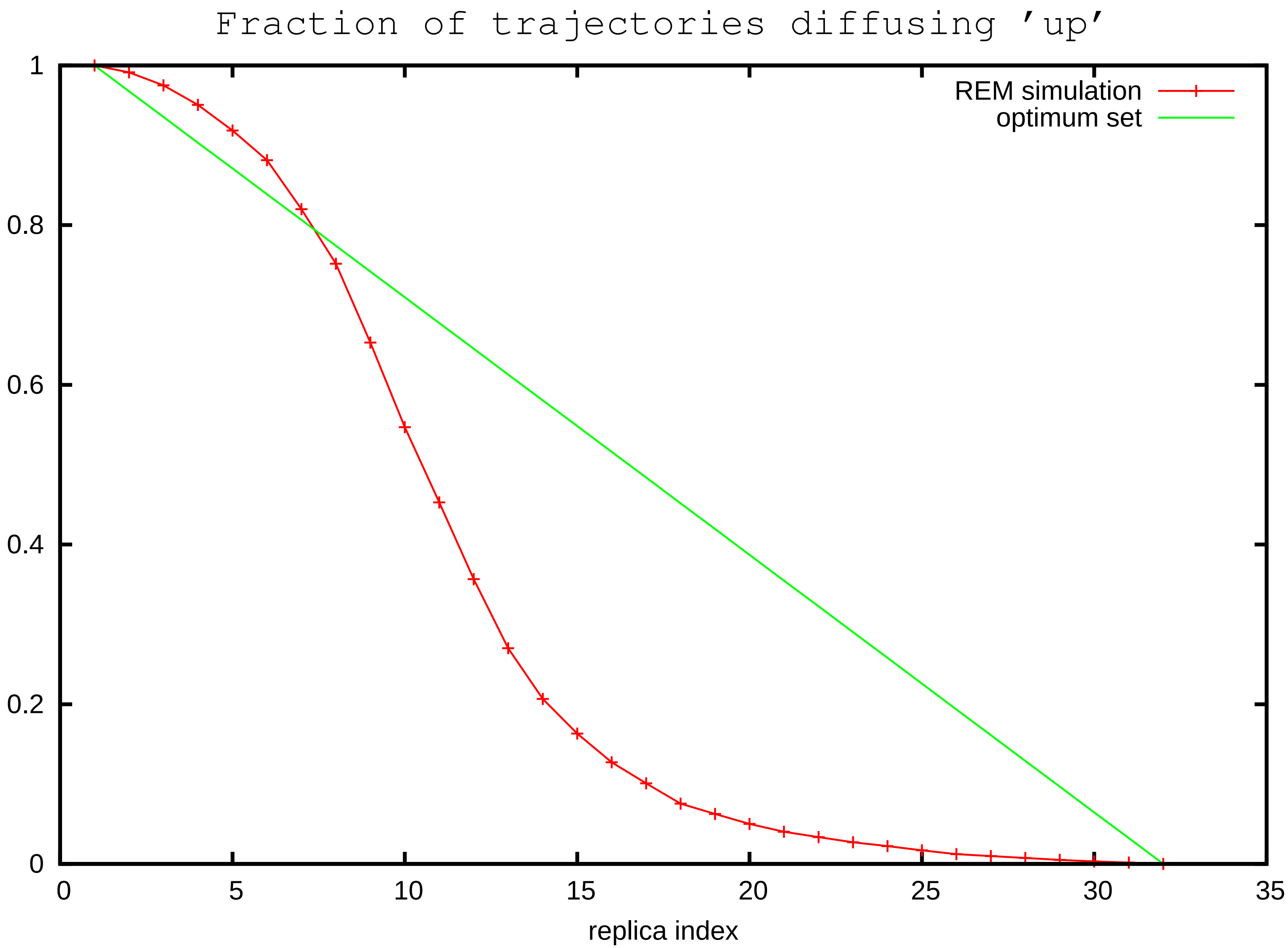
$$f_i^{up} = \frac{n_i^{up}}{n_i^{up} + n_i^{down}}$$

diffusivity at replica i:

$$D_i = \frac{\Delta T}{df/dT}$$

optimal: constant diffusivity

monitoring trajectory flow (2)



HREM with potential partitioning

- In many cases, there is little advantage for conformational sampling in heating “stiff” degrees of freedom (e.g. stretching).

HREM with potential partitioning

- In many cases, there is little advantage for conformational sampling in heating "stiff" degrees of freedom (e.g. stretching).
- The potential of these DoF can be kept fixed

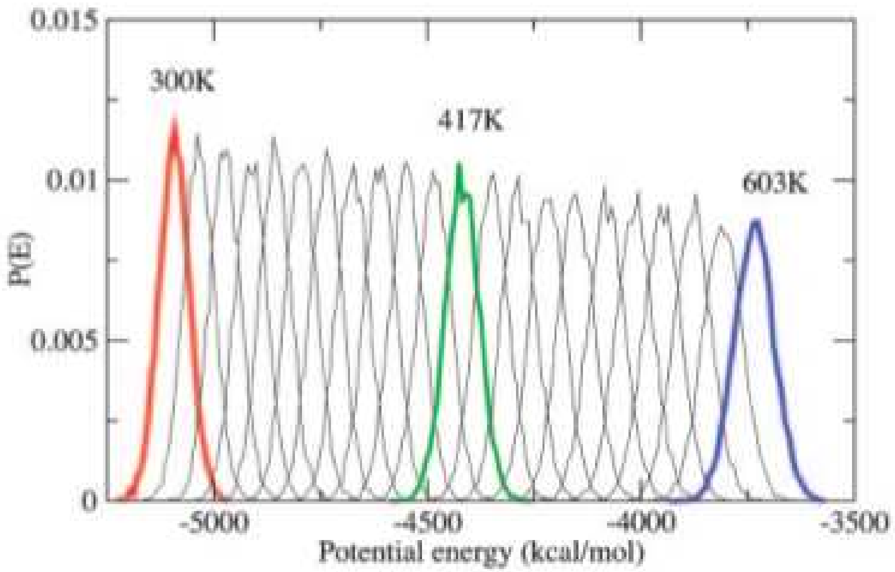
HREM with potential partitioning

- In many cases, there is little advantage for conformational sampling in heating "stiff" degrees of freedom (e.g. stretching).
- The potential of these DoF can be kept fixed
  - less energy spread $\Longrightarrow$ you can use fewer replicas / higher temperatures for the relevant degrees of freedom

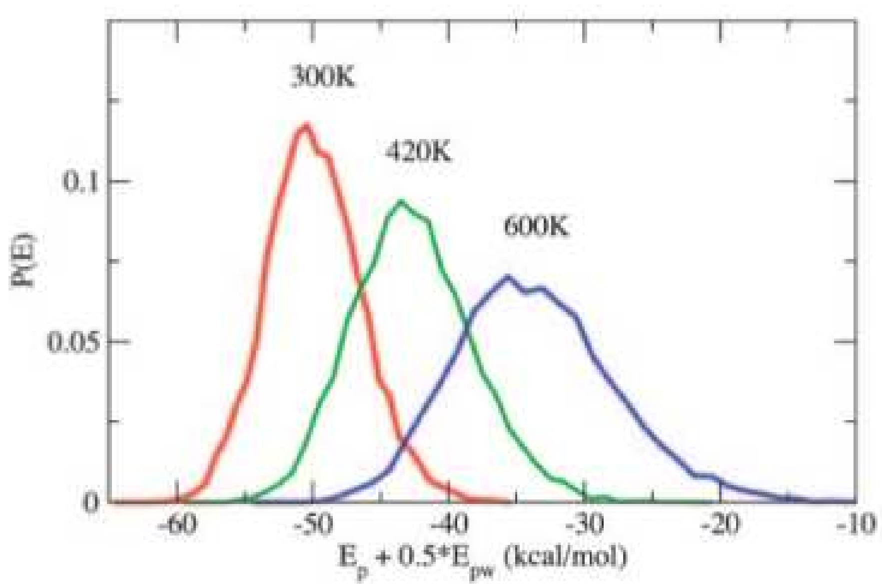
HREM with potential partitioning

- In many cases, there is little advantage for conformational sampling in heating “stiff” degrees of freedom (e.g. stretching).
- The potential of these DoF can be kept fixed
  - less energy spread $\Longrightarrow$ you can use fewer replicas / higher temperatures for the relevant degrees of freedom

normal REM



REM with pot. partitioning



HREM with potential partitioning (2)

- In ORAC, different scaling can be applied to different parts of the potential

max. scaling	interaction
c_1	stretching and bending
c_2	torsion and 1-4 interaction
c_3	non-bonded

HREM with potential partitioning (2)

- In ORAC, different scaling can be applied to different parts of the potential

max. scaling	interaction
c_1	stretching and bending
c_2	torsion and 1-4 interaction
c_3	non-bonded

- Here is how this is implemented in REM test 3:

```
&REM
#       bend+bond torsion+1-4 nonbonded
  SETUP  1.0       0.20      0.75      1
  STEP 5.
  PRINT 1000.
&END
```

Solute Tempering extension

- It is also useful to partition the system in a "solute" and a "solvent".

"solute" $\longrightarrow$ any portion of the system $\longrightarrow$ "Segment"
"solvent" $\longrightarrow$ the rest $\longrightarrow$ "Environment"

interaction	S	S-E	E-E
stretching and bending	1	1	1
torsion and 1-4	c_2	c_2	1
non-bonded	c_3 (or 1)	1 (or c_3)	1

Solute Tempering extension

- It is also useful to partition the system in a "solute" and a "solvent".

  "solute" $\longrightarrow$ any portion of the system $\longrightarrow$ "Segment"
  "solvent" $\longrightarrow$ the rest $\longrightarrow$ "Environment"

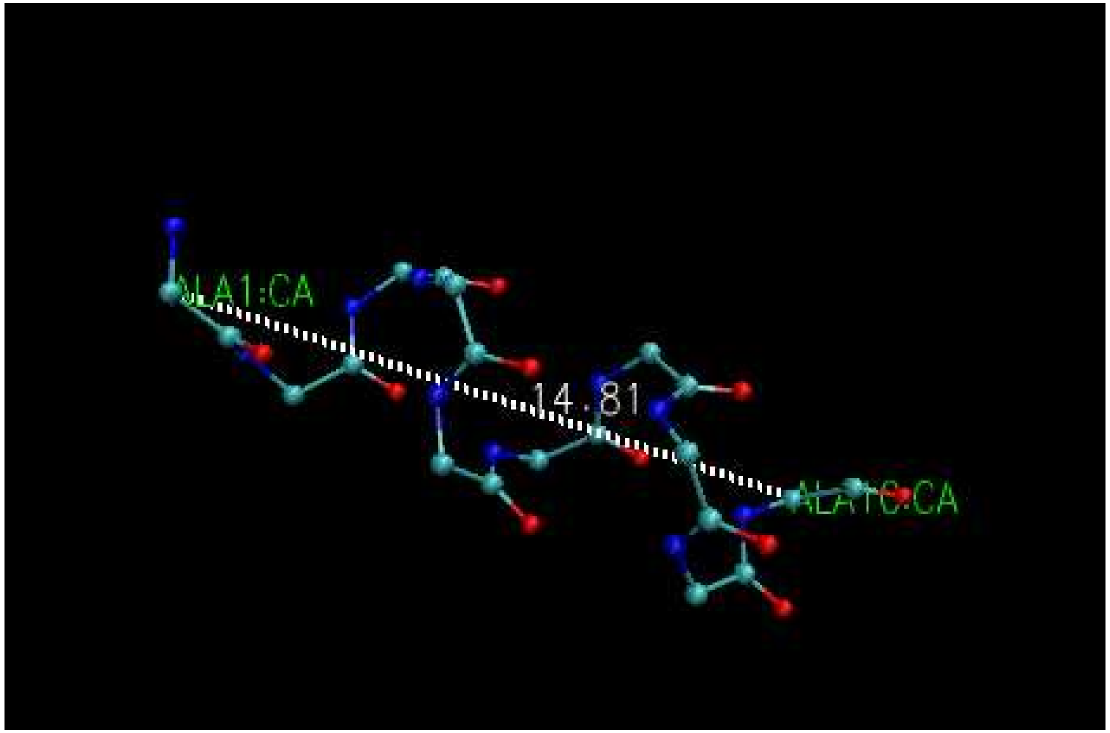
interaction	S	S-E	E-E
stretching and bending	1	1	1
torsion and 1-4	c_2	c_2	1
non-bonded	c_3 (or 1)	1 (or c_3)	1

- Here is how this is implemented in REM test 5:

```
&REM
#        bend+bond torsion +1-4 n-bonded
   SETUP  1.0        0.20       0.20      1
   SEGMENT
     define 1 22
     kind intra
   END
   ...
&END
```

Steered MD Tests

Deca-Alanine



SMD additional potential: $V' = \frac{1}{2}k\left(d - (d_0 + v \cdot t)\right)^2$

- end-to-end distance $d = |C_1^\alpha - C_{10}^\alpha|$
- state A: $d = 15.5$ (α-helix)
- state B: $d = 31.5$

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$
2. For each starting state A_i, run a SMD simulation toward state B, recording the total work W done by the steering force; do the same, now going from B to A (*)

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$
2. For each starting state A_i, run a SMD simulation toward state B, recording the total work W done by the steering force; do the same, now going from B to A (*)
3. From the distribution of works, derive ΔF using Jarzynski identity

$$e^{-\beta \Delta F} = \overline{e^{-\beta W}} \tag{1}$$

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$
2. For each starting state A_i, run a SMD simulation toward state B, recording the total work W done by the steering force; do the same, now going from B to A (*)
3. From the distribution of works, derive ΔF using Jarzynski identity

$$e^{-\beta \Delta F} = \overline{e^{-\beta W}} \tag{1}$$

or Crooks' theorem

$$P_{\mathcal{A}\to\mathcal{B}}(W)\, e^{-\beta W} = P_{\mathcal{A}\leftarrow\mathcal{B}}(-W)\, e^{-\beta \Delta F} \tag{2}$$

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$
2. For each starting state A_i, run a SMD simulation toward state B, recording the total work W done by the steering force; do the same, now going from B to A (*)
3. From the distribution of works, derive ΔF using Jarzynski identity

$$e^{-\beta\Delta F} = \overline{e^{-\beta W}} \tag{1}$$

or Crooks' theorem

$$P_{\mathcal{A}\to\mathcal{B}}(W)\, e^{-\beta W} = P_{\mathcal{A}\leftarrow\mathcal{B}}(-W)\, e^{-\beta\Delta F} \tag{2}$$

or other “bidirectional” methods that can estimate the free energy profile along the whole path $A \to B$

Flow chart

1. Prepare a canonical sample of state A, $\{A_i\}$, and of state B, $\{B_j\}$
2. For each starting state A_i, run a SMD simulation toward state B, recording the total work W done by the steering force; do the same, now going from B to A (*)
3. From the distribution of works, derive ΔF using Jarzynski identity

$$e^{-\beta \Delta F} = \overline{e^{-\beta W}} \tag{1}$$

or Crooks' theorem

$$P_{\mathcal{A}\rightarrow\mathcal{B}}\left(W\right) e^{-\beta W} = P_{\mathcal{A}\leftarrow\mathcal{B}}\left(-W\right) e^{-\beta \Delta F} \tag{2}$$

or other "bidirectional" methods that can estimate the free energy profile along the whole path $A \longrightarrow B$

(*) Step 2 can be performed in parallel

step 1: prepare canonical distributions


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pa.in
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pb.in
```

step 1: prepare canonical distributions


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pa.in
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pb.in
```

- These commands create configuration files corresponding to a canonical distribution of state A and state B in directories RESTART_A/ and RESTART_B/, respectively.

step 1: prepare canonical distributions


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pa.in
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pb.in
```

- These commands create configuration files corresponding to a canonical distribution of state A and state B in directories RESTART_A/ and RESTART_B/, respectively.
- The parallell version of ORAC is used (on 1 processor) only for compatibility with step 2

step 1: prepare canonical distributions


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pa.in
$ mpiexec -n 1 ../../src/PARALLEL/orac_Linux < 2Pb.in
```

- These commands create configuration files corresponding to a canonical distribution of state A and state B in directories RESTART_A/ and RESTART_B/, respectively.
- The parallell version of ORAC is used (on 1 processor) only for compatibility with step 2
- this is how this is implemented in test 2Pa:

```
&INOUT
  RESTART
    write 250.0 SAVE_ALL_FILES ../RESTART_A/ala10_A
  END
&END
```

step 2: run SMD in parallel


```
$ cd orac-5_1_x/tests/jarzynski_tests
```

step 2: run SMD in parallel


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 4 ../../src/PARALLEL/orac_Linux < 3a.in
$ mpiexec -n 4 ../../src/PARALLEL/orac_Linux < 3b.in
```

- These commands start 4 parallel processes in directories PAR0000 ... PAR0003 creating workfunction files WRKa.1 and WRKb.1 in each directory

step 2: run SMD in parallel


```
$ cd orac-5_1_x/tests/jarzynski_tests
$ mpiexec -n 4 ../../src/PARALLEL/orac_Linux < 3a.in
$ mpiexec -n 4 ../../src/PARALLEL/orac_Linux < 3b.in
```

- These commands start 4 parallel processes in directories PAR0000 ... PAR0003 creating workfunction files WRKa.1 and WRKb.1 in each directory

```
# The bending inolving dum atom with large mass is required
# in order to stretch 10-alanine along the z-coordinate
&POTENTIAL
  ADD_STR_BONDS 1 102 400.0 15.5 31.5
  ADD_STR_BENDS 102 1 105 600.0 180.0
  ...
&END
# Unfolding of 10-ala occurs in 10000 fs.
&RUN
  CONTROL              2
  REJECT           0.0
  STEER      0.0   9900.0
  TIME             9900.0
  MAXRUN           9900.0
&END

&INOUT
  RESTART
    rmr ../RESTART_A/ala10_A 0
  END
  PLOT STEER_ANALYTIC 300.0 OPEN WRKa.1
&END
```

step 2: (2) work paths

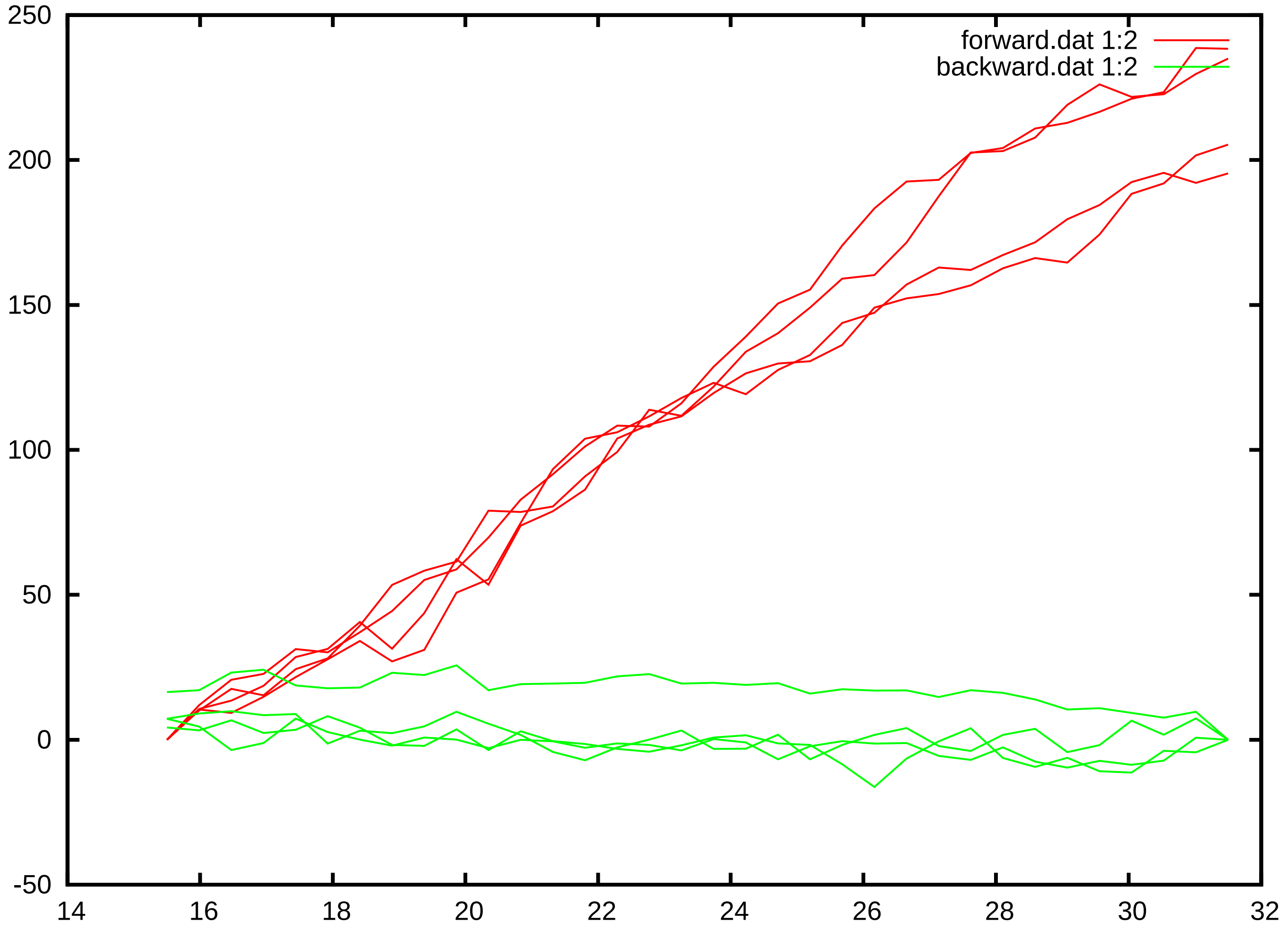


Figure 4: Jarzynski test 2P

how it looks in real SMD simulations

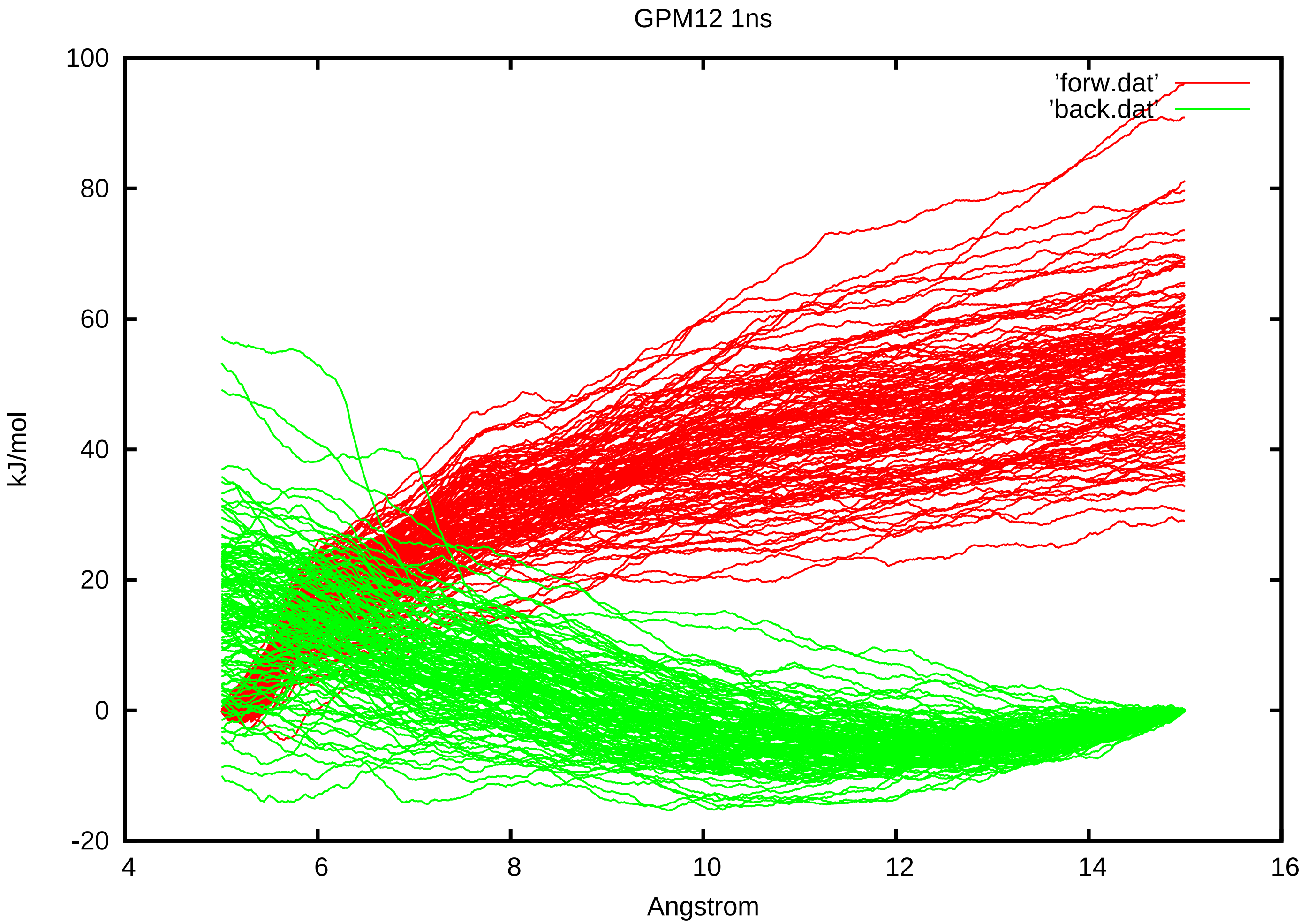


Figure 5: real-world example

step 3: estimate Free Energy Profile

- the most accurate results are obtained with bidirectional methods

step 3: estimate Free Energy Profile

- the most accurate results are obtained with bidirectional methods
- these are *experimentally* implemented in the auxiliary program `fes`, which currently uses fixed names for input files

```
$ cd orac-5_1_x/tests/jarzynski_tests
```

step 3: estimate Free Energy Profile

- the most accurate results are obtained with bidirectional methods
- these are *experimentally* implemented in the auxiliary program `fes`, which currently uses fixed names for input files

```
$ cd orac-5_1_x/tests/jarzynski_tests
$ make fes
```

step 3: estimate Free Energy Profile

- the most accurate results are obtained with bidirectional methods
- these are *experimentally* implemented in the auxiliary program `fes`, which currently uses fixed names for input files

```
$ cd orac-5_1_x/tests/jarzynski_tests
$ make fes
$ for i in ‘ls -d PAR*‘; do mv $i/WRKa.1 FORWARD/WRKa.$i; done
$ for i in ‘ls -d PAR*‘; do mv $i/WRKb.1 REVERSE/WRKb.$i; done
```

step 3: estimate Free Energy Profile

- the most accurate results are obtained with bidirectional methods
- these are *experimentally* implemented in the auxiliary program `fes`, which currently uses fixed names for input files

```
$ cd orac-5_1_x/tests/jarzynski_tests
$ make fes
$ for i in ‘ls -d PAR*‘; do mv $i/WRKa.1 FORWARD/WRKa.$i; done
$ for i in ‘ls -d PAR*‘; do mv $i/WRKb.1 REVERSE/WRKb.$i; done
$ echo 4 | ./fes
```

step 3: estimate Free Energy Profile (2)

- the last command produces the following output:

```
# position, DF_AQ_EQ8, DF_AQ_EQ9, DF_AQ_EQ16, DF_AQ_MINH, DF_AQ_PMFA, DF_AQ_J, DF_BQ_J
 15.9848500  5.3572   5.3496   5.3572  10.6992   0.0000  10.6992   6.5107
 16.4697000  5.6583   5.5398   5.6583  12.1777   1.4786  12.1777   5.4125
 16.9545500  4.5535   4.8978   4.5535  16.4513   5.7521  16.4513  -0.1450
 17.4393900  9.7112   9.7197   9.7112  24.1973  13.4981  24.1973   1.7528
 ...
 31.0151500 92.3263  93.6842  93.6842  91.7988  81.0997 195.5132  -4.3500
 31.5000000 95.5277  97.2839  97.2839  94.9066  84.2075 198.8083  -1.2422
```

step 3: estimate Free Energy Profile (2)

- the last command produces the following output:

```
# position, DF_AQ_EQ8, DF_AQ_EQ9, DF_AQ_EQ16, DF_AQ_MINH, DF_AQ_PMFA, DF_AQ_J, DF_BQ_J
 15.9848500  5.3572   5.3496   5.3572  10.6992   0.0000  10.6992   6.5107
 16.4697000  5.6583   5.5398   5.6583  12.1777   1.4786  12.1777   5.4125
 16.9545500  4.5535   4.8978   4.5535  16.4513   5.7521  16.4513  -0.1450
 17.4393900  9.7112   9.7197   9.7112  24.1973  13.4981  24.1973   1.7528
 ...
 31.0151500 92.3263  93.6842  93.6842  91.7988  81.0997 195.5132  -4.3500
 31.5000000 95.5277  97.2839  97.2839  94.9066  84.2075 198.8083  -1.2422
```

columns 2-6 report the FEP (with different methods), while the last two columns give Jarzynski estimate for the forward and backward transformation