MagiC tutorial:

Computation of solvent-mediated effective potentials between Na+ and Cl- ions in water

Alexander Lyubartsev[(1)], Alexander Mirzoev[(2)]

(1): Department of Materials and Environmental Chemistry, Stockholm University, Stockholm, Sweden
(2): School of Biological Sciences, Nanyang Technological University, Singapore

August 2017

This tutorial illustrate example of computation of effective potentials for ions in NaCl solution. This example follows these works:

A.Lyubartsev, A.Laaksonen, "*Calculation of Effective Interaction Potentials from Radial Distribution Functions: A Reverse Monte Carlo Approach*", Phys.Rev.E, 52, 3730 (1995)
A.Mirzoev, A.P.Lyubartsev, "*Effective solvent-mediated interaction potentials of Na+ and Cl- in aqueous solution: temperature dependence",* PCCP, 13, 5722 (2011)

In the example, radial distribution functions (RDF) between ions in water solutions, computed by atomistic molecular dynamics, are used to compute effective solvent-mediated potentials, which, in simulation of ions without explicit water, reproduce the same RDF as those obtained in the atomistic simulations.

The example contains 4 folders in which input and output files relevant for each step of computations of the effective potentials are collected:

1 - High-resolution (atomistic) molecular dynamics (1-HighResolutionMD)

2 - Coarse-graining of the trajectory (2-CGtraj)

3 - Computation of the radial distribution functions from the coarse-grained trajectory (3-RDF)

4 - Computations of the effective potentials (4-IMC)

Additionally, folder Input-only contains the same 4 subfolders with only input files. It can be advisable to run test study in this folder and then compare with the reference output files.

It is assumed that MagiC package (v. > 2.2) is compiled and installed.

Step 1:

Folder `1-HighResolutionMD` : High-resolution (atomistic) molecular dynamics.

This directory contains setup of molecular dynamics simulations of 20 NaCl ion pairs (modelled by Smith-Dang potentials) dissolved in 1000 water molecules (flexible SPC model) using MDynaMix software (www.fos.su.se/~sasha/mdynamix).

Files:

`md.input` : main input file for MDynaMix (12 ns for 20 ion pairs in 1024 water of which 2 ns are left for equilibration)

`H2O.mmol`
`Na+_SD.mmol`
`Cl-_SD.mmol` - structure-parameter files for water and ions

Output: trajectory files obtained by running Mdymamix:

To run (implies v. > 5.2.8):

```
-> ./md  md.input > md.output
```

or

```
-> mpirun -np xx mdp md.input > md.output
```

(with XX giving the number of available cores/processors)

The simulation generates 12 trajectory files 1 ns each with names "`nacl-1M.001`",..., "`nacl-1M.012`" in XMOL format. To save space, only ions are included into trajectory. These files (except .001 and .002 covering equilibration stage) are already included in the Example and can be used directly in step 2 without running atomistic simulations.

Step 2:

Converting atomistic trajectory to a coarse-grained trajectory

enter directory `2-CGtraj`

Executable file: `cgtraj`

Input files: `cgtraj_NaCl.inp` : main input file
`nacl-1M.0xx` from step 1: atomistic trajectory files

Run:

```
~> cgtraj cgtraj_NaCl.inp
```

Output:
- `cgtraj.NaCl.001` - Coarse-grained trajectory (= coordinates of Na+ and Cl- ions only) in .xmol format
- Coarse-grained `.CG.mmol` files for the involved molecules

Note 1. Because of simplicity of the example, and because water coordinates were already excluded from the atomistic trajectory, the coarse-grained trajectory contains the same ion coordinates as the atomistic one

Note 2. An alternative version of the cgtraj input file is also present which does not require atomistic `.mmol` files inherited from MDynaMix program. In this case the `.CG.mmol` files are generated with default masses of the CG sites equal to 1 and default charges equal to 0, these need to be set to real values manually.

Step 3:

Compute RDFs from the coarse-grained trajectory.

Input:
`rdf.inp` - main input file
`../2-CGtraj/cgtraj.NaCl` - coarse-grained trajectory created at the previous step

Output:

`NaCl_1M.rdf` - files with radial distribution functions in MagiC format
`Na+_SD.CG.mcm`
`Cl-_SD.CG.mcm` - topology files for coarse-grained molecules

Enter directory 3-RDF . Run python script which computes RDFs:

```
~> rdf-2.0.py -i rdf.inp
```

This results in file `NaCl_1M.rdf` which contains Na-Na, Na-Cl and Cl-Cl RDFs in format suitable for MagiC. Also, topology files `Na+_SD.CG.mcm` and `Cl-_SD.CG.mcm` for the next step are created

Copy these file into directory 4-IMC

4. Inverse MC - the main MagiC module.

Input:

`NaCl-rdf.rdf` - file with reference RDFs
`Na+_SD.CG.mcm`
`Cl-_SD.CG.mcm` - molecular description files for MagiC

`magic_NaCl.inp1, magic_NaCl.inp2` - input files for two steps of the inverse procedure

Run:

1 stage:

```
~> magic-gfortran  Magic_NaCl.inp1 > NaCl.out1 &
```

or (implies 8 processors or cores) :

```
~> mpirun -np 8 magic-gfortran-mpi  Magic_NaCl.inp1 > NaCl.out1 &
```

This will run 10 iterations of the inverse Monte Carlo, 11000000 steps each,
with regularization parameter 0.2, starting from zero potential. This creates,
after each iteration, a new potential file (`01.NaCl.i001.pot - 01.NaCl.i010.pot`)
and log output from each core `NaCl.01.p001 - NaCl.01.p008` (only in case of parallel execution)
The main output file `magic_NaCl.out1` contains log of the calculation.

An easy way to check convergence is to use "grep" command:

```
~>grep Devi magic_NaCl.out1
```

2 stage:

```
~> magic-grortran  magic_NaCl.in-2 > magic_NaCl.out-2 &
```

This will run a second series of 10 iterations of the inverse Monte Carlo.
Note the changes from the first series of iterations:

- the program now starts from the last potential of the previous series (InputPot parameter uncommented):

```
 InputPot = ' 01.NaCl.i010.pot',
```

- the number of MC steps is increased to 20 000 000 and for equilibration to 4 000 000

```
 MCSteps = 20000000,
 MCStepsEquil = 4000000
```

- the regularization parameter is increased to 0.5

```
  REGP = 0.5,
```

- the Base output filename is changed to

```
 Output = NaCl.02
```

in order to not overwrite results of the first series

Again, the program creates after each iteration a new potential file
(`02.NaCl.i001.pot - 02.NaCl.i010.pot`).
The main output file magic_NaCl.out-2 contains log of the calculation.

Note.

The number of MC steps is given per processor, and the quality of computations/noise will depend on the number of processor used. The output files of the example are computed at 8 processors. If you use smaller number of processor, it is advisable to increase the number of steps accordingly.

Post-analysis:

A fast convergency check:

```
 -> grep Deviation NaCl_IMC.out2

 Final Deviation from references: S:    0.012367  RDF:    0.103638
 Final Deviation from references: S:    0.006590  RDF:    0.057830
 Final Deviation from references: S:    0.003620  RDF:    0.032893
 Final Deviation from references: S:    0.002161  RDF:    0.018688
 Final Deviation from references: S:    0.001186  RDF:    0.010191
 Final Deviation from references: S:    0.000770  RDF:    0.005979
 Final Deviation from references: S:    0.000598  RDF:    0.003407
 Final Deviation from references: S:    0.000495  RDF:    0.002556
 Final Deviation from references: S:    0.000586  RDF:    0.001918
 Final Deviation from references: S:    0.000596  RDF:    0.002819
```

The final potential can be found in file `02.NaCl.i010.pot` ;
It can be also viewed in the end of the main output file `NaCl_IMC.out2` (about 800 last lines) which contains computed and reference RDFs (columns 2,3), and final potential (column 4).

The output file can be also analyzed using MagicTools:

```
~> ipython

In [1]: import MagicTools
```

Load magic output from the first and second run:

```
RDF1=MagicTools.ReadMagiC('NaCl_IMC.out1')
RDF2=MagicTools.ReadMagiC('NaCl_IMC.out2')
POT1=MagicTools.ReadMagiC('NaCl_IMC.out1',DFType='Pot')
POT2=MagicTools.ReadMagiC('NaCl_IMC.out2',DFType='Pot')
RDFref=MagicTools.ReadMagiC('NaCl_IMC.out2',iters=(1),DFType='RDFref')
```

Show convergence of RDFs during the second run:

```
MagicTools.PlotAllDFs(RDF2)
```

Plot 1th RDF from te first run, 1st and 10th RDF from the seconf run, and reference RDF:

```
MagicTools.PlotAllDFs([RDF1[0]]+[RDF2[0]]+[RDF2[9]]+[RDFref])
```

(10th RDF from the second run coincide with the reference on the plot

Show potentials obtained during the first run:

```
MagicTools.PlotAllDFs(POT1)
```

Show potentials obtained during the second run:

```
MagicTools.PlotAllDFs(POT2)
```

Show final potentials:

```
MagicTools.PlotAllDFs(POT2[9])
```

Show total potentials with addition of the Coulombic part:

```
TotalPots=MagicTools.TotalPots(POT2[9], eps=78.0, mcmfile=['Na+_SD.CG.mcm','Cl-
_SD.CG.mcm'])
MagicTools.PlotAllDFs(TotalPots)
```