Systematic Coarse-Graining for Python (scg4py) user guide

Saeed Mortezazadeh

Department of Biophysics, Faculty of Biological Sciences, Tarbiat Modares University, Tehran, Iran

April 15, 2019

1 Introduction

The molecular dynamics (MD) simulation method has become a striking tool to unveil the driving forces governing biomolecular processes. Time evolution of the interacting particles is computed based on Newton's laws of motion so that pairwise forces are the major part of interactions between particles. Nowadays, MD simulations are applied as a 'computational microscopy' tool along with experimental microscopy methods in the fields of chemical physics, materials science and the modeling of biomolecules. However, simulating of the large spatiotemporal scales are not feasible by using traditional all-atom MD simulation. Coarse-graining is a popular approach which allows us to investigate long-time dynamics of large structures by neglecting unimportant atomistic degrees of freedom. Coarse graining includes two basic strategies known as bottom-up and top-down. The bottom-up systematic coarse-graining uses atomistic simulations to derive effective potential functions for coarse-grained (CG) sites. This approach can be implemented using several methods such as the Iterative Boltzmann Inversion [7], Inverse Monte Carlo [4], force matching [3], and relative entropy [9]. There are several good literature reviews on the systematics coarse-graining methods covering various methodological issues [1, 8, 6]. The main aim of the top-down approach is to reproduce key experimental data, such as partitioning of solutes between polar and apolar solvents. Both approaches have their own advantages and limitations which can be very generally summarized that the bottom-up approach is better suitable to capture specific details of the interactions between the involved molecules whereas the top-down approach provides a force field framework that usually can be easily extended to other systems [2].

1.1 Theoretical background

Systematic coarse-graining includes two main steps: the CG molecular mapping design and computation of CG interaction potential functions. The coarse-grain mapping can be done in different ways depending on the chosen level of resolution suitable to describe the studied phenomena, and importance of specific chemical details. When a CG mapping scheme is determined, the next step is to define effective interaction potentials between the CG sites. The CG potential function usually includes bonded and non-bonded terms as well as coulombic interactions. The initial guess for each bonded and non-bonded terms can be obtained by direct Boltzmann inversion [7]:

$$\begin{aligned}
U(r,\vartheta,\varphi) &= U_{bond}(r) + U_{angle}(\vartheta) + U_{torsion}(\varphi) \\
&\quad + U_{non-bond}(r) + U_{coulomb} \\
U(r) &= -k_B T \ln \frac{\langle S(r) \rangle}{4\pi r^2} \\
U(\vartheta) &= -k_B T \ln \frac{\langle S(\vartheta) \rangle}{\sin(\vartheta)} \\
U(\varphi) &= -k_B T \ln \langle S(\varphi) \rangle
\end{aligned} \tag{1}$$

where k_B and T are the Boltzmann constant and temperature respectively. The average histograms of bonded and non-bonded distance distributions $\langle S(r) \rangle$, angle $\langle S(\vartheta) \rangle$, and torsion angle $\langle S(\varphi) \rangle$ distributions are calculated from the mapped atomistic trajectory. However, the initial potential functions (1) usually cannot reproduce the reference atomistic distribution functions. A method to improve the CG potentials is the Iterative Boltzmann Inversion (IBI) [8] which refines them during multiple iterative simulations. The tabulated potentials at the end of the nth CG simulation are modified according to the following equation:

$$U_\alpha^{n+1} = U_\alpha^n + \Delta U_\alpha^n = U_\alpha^n + a k_B T \ln \frac{\langle S_\alpha^n \rangle}{\langle S_\alpha^{ref} \rangle} \tag{2}$$

where a, $0<a<1$ is a correction factor to regulate the convergence of the distribution functions. The iterative process proceeds until an acceptable convergence is observed between the atomistic reference $\langle S_\alpha^{ref} \rangle$ and CG simulation $\langle S_\alpha^n \rangle$ distribution functions.

The basic assumption of the IBI is that all terms of the tabulated potentials are independent of each other. This assumption leads often to convergence problem in multi-component systems, which include many types of different non-bonded interactions. In the Inverse Monte Carlo method, the CG potential functions are corrected by taking into account correlations between all the interactions. Details of this method can be found in papers [5, 4]. Within IMC, the CG potentials are modified by inverting the correlation matrix based on the following equation:

$$
\begin{aligned}
\Delta U_{\alpha}^{n} &= a\sum_{\gamma} A_{\alpha\gamma}^{-1} \times (\langle S_{\alpha}^{ref}\rangle - \langle S_{\alpha}^{n}\rangle) \\
A_{\alpha\gamma} &= \frac{(\langle S_{\alpha}^{n}\rangle\langle S_{\gamma}^{n}\rangle - \langle S_{\alpha}^{n} S_{\gamma}^{n}\rangle)}{k_B T}
\end{aligned}
\tag{3}
$$

The correlation matrix ($A_{\alpha\gamma}$) is calculated between all terms of bonded and non-bonded interactions. This is usually done by Monte Carlo sampling, however, any sampling algorithm, such as constant-temperature molecular dynamics simulation, can be used provided that the phase space canonical distribution is generated. Long-time simulation is usually required to avoid errors in calculating the correlation matrix due to the inadequate sampling of the phase space. Furthermore, insufficient sampling may perturb the stability of the IMC algorithm. To overcome this problem, one can apply the Tikhonov regularization method to calculate inversion of the correlation matrix in the IMC approach [8].

$$
\Delta U_{\alpha}^{n} = a\sum_{\gamma} [(A_{\alpha\gamma}^{T} A_{\alpha\gamma} + \lambda I)^{-1} A_{\alpha\gamma}^{T}] \times (\langle S_{\alpha}^{ref}\rangle - \langle S_{\alpha}^{n}\rangle)
\tag{4}
$$

where I is an identity matrix as a regularization operator and $\lambda > 0$ is the regularization parameter. The singular value decomposition (SVD) of the matrix A allows us to guess an initial value of λ. A good approximation of parameter λ should dominate the small singular values but is smaller than larger ones [8].

The above methods allow us to obtain potential functions in order to optimize the structure of the CG system. Since sampling in the systematic coarse-graining is done by simulation in NVT mode, pressure control becomes important in many issues. In this case, a simple linear potential is added to the non-bonded potential functions [7].

$$
\begin{aligned}
\Delta U_{PressCorr}^{n}(r) &= A(1 - \frac{r}{r_{cutoff}}) \\
A &= -\mathrm{sgn}(\Delta P) 0.1 k_B T \min(1, |f \Delta P)
\end{aligned}
\tag{5}
$$

where $\Delta P = P^{n} - P^{target}$, and scaling factor f and P^{target} can be specified in the input settings file of the **scg4py.RefinePot**.

2 scg4py module

scg4py [1] is written in python language to generate CG potentials of the system of interest through systematic coarse-graining approach. This module is a collection of tools to implement all coarse-graining steps including mapping the atomistic trajectory to the CG trajectory, generating the CG topology file, calculating distribution functions and refining the CG potentials through IBI and IMC methods. In the following paper, we applied this module to study the

[1] `https://github.com/saeedMRT/scg4py`

lamellar and inverse hexagonal formation of lipid phases at different conditions. If you find this module useful, please cite our paper.

- Implicit Solvent Systematic Coarse-Graining of Dioleoylphosphatidylethanolamine Lipids: from the Inverted Hexagonal to the Bilayer Structure, Saeed Mortezazadeh, Yousef Jamali, Hossein Naderi-Manesh, and Alexander P.Lyubartsev, `https://doi.org/10.1371/journal.pone.0214673`

For using this module you need python 3 and the following prerequisite modules: numpy, scipy, matplotlib, and mdtraj for parsing the 'dcd' and 'xtc' trajectory file formats. For ease of use, you can append the path of scg4py code to PYHTONPATH in the *.bashrc* file.

```
export PYTHONPATH="/path/to/scg4py/:$PYTHONPATH"
```

This module has several programs that are described below.

2.1 scg4py.Mapping(inFile, outTop=None, readTRJ=True, coulomb=False)

This program reads an input mapping file, then translates the atomistic trajectory to the coarse-grained. the coordinates of each CG site are calculated by the center of mass of involved atoms. Also, a CG topology file and CG structures of each molecule type are generated as output. This module supports a range of trajectory file format including 'lammpstrj', 'pdb', 'gro', 'xtc', and 'dcd'.
Parameters:

- **inFile**: Input mapping file (*.map).
- **outTop**: Output CG topology file. If None, it is specified as the name of the input mapping file.
- **readTRJ**: If False, generates only CG topology file.
- **coulomb**: If True, the electric charge of each bead is calculated from the total electric charge of the atoms involved in that bead otherwise, it is considered zero.

Outputs:

- The CG topology file ([outTop].CGtop).
- The CG trajectory file.
- The log file ([inFile].log) .
- The last snapshot of the CG trajectory (*.pbd).
- The CG structure of each molecule type in different files (*.pdb).

2.2 scg4py.histCalc(inTraj, top, outTab=None, RMaxNB=20, RMaxB=10, BinNB=0.2, BinB=0.02, BinA=2, BinD=5, normalizeBonded=False)

This program calculates all radial and bonded distribution functions of the system from the CG trajectory based on the topology file.
Parameters:

- **inTraj**: Input CG trajectory path.
- **top**: The CG topology file.
- **outTab**: The name of output distribution file. The default filename is 'CGsystem'.
- **RMaxNB**: The maximum distance used in the calculation of the radial distribution functions. The default is 20 Angstrom.
- **RMaxB**: The maximum distance used in the calculation of the bond distribution functions. The default is 10 Angstrom.
- **BinNB**: The bin width used in the calculation of the radial distribution functions. The default is 0.2 Angstrom.
- **BinB**: The bin width used in the calculation of the bond distribution functions. The default is 0.02 Angstrom.
- **BinA**: The bin width used in the calculation of the angular distribution functions. The default is 2 degrees.
- **BinD**: The bin width used in the calculation of the dihedral distribution functions. The default is 5 degrees.
- **normalizeBonded**: if True, all bonded histograms are normalized to one. The default is False.

Output:

- System's distribution functions ([outTab].hist if normalizeBonded=False or [outTab].dist if normalizeBonded=True)

2.3 scg4py.histTrim(inTabs, outTab=None)

This program exerts some modifications on distribution functions by smoothing and trimming the edge of tables in order to produce the reference distribution functions.
Parameters:

- **inTabs**: Input distribution functions file(s). If a list of names is entered, you can choose the best distribution function from them.

- **outTab**: Output distribution functions file.

Output:

- Modified distribution functions file ([outTab].hist or [outTab].dist)

2.4 scg4py.hist2pot(inTab, outPot=None, initPot=True, T=300)

This program produces the initial CG potential from the reference distribution functions through direct Boltzmann inversion.
Parameters:

- **inTab**: Input reference distribution function file.
- **outPot**: Output potential file. The default is the same as the input filename.
- **initPot**: If True, the non-bonded potentials are weakened to eliminate the bad effects of the initial potentials. The default is True.
- **T**: The temperature of the system. The default is 300 Kelvin.

Output:

- Initial potential functions ([outPot].pot)

2.5 scg4py.plotTab(inTabs, legend=-1)

This program stores the figures of both the distribution function tables and potential function tables.
Parameters:

- **inTabs**: List of the input tables.
- **legend**: List of the legend names. If -1, The legends are named from number 1.

Output:

- The figure saved for each table (*.png)

2.6 scg4py.genLMPdata(inConf, top, output=None)

This program reads an input configuration and then generates the LAMMPS data file. Also, it produces a 'psf' file to view the CG structure in VMD software.
Parameters:

- **inConf**: Input structure file. All types of 'lammpstrj', 'pdb', 'gro', 'xtc', and 'dcd' file formats are supported.

- **top**: CG topology file.
- **output**: Output name of the LAMMPS data file and 'psf' file format. The default is the same as input filename.

Outputs:

- The LAMMPS data file ([output].data.in)
- The PSF file ([output].psf)

2.7 scg4py.genLMPscript(pot, top, sysName, cutoff, cutoffSkin=2, maxNB=100, maxB=100, maxA=10, binNB=0.01, binAD=0.05, script4inverse=True, gpu=False)

This program generates a typical script file for simulating on NVT mode with LAMMPS software. Also, it produces tabulated potential files by interpolation and extrapolation of the input potential file. Preparation of the potential tables for LAMMPS software was done as follows: each table is smoothed through radial basis function interpolation. Torsion angle potentials are interpolated periodically while potential tables of bond, angle, and left-hand side of the non-bonded interactions are extrapolated by a quadratic function. The right-hand side of the non-bonded potentials are extrapolated by polynomial functions so that the potential and force at the cutoff radius became zero.
Parameters:

- **pot**: Input potential file.
- **top**: CG topology file.
- **sysName**: The system name used as the base name of the output files.
- **cutoff**: The non-bonded interaction cutoff distance in Angstrom.
- **cutoffSkin**: A distance before the cutoff radius used in the extrapolation of the right-hand side of the non-bonded potentials. The default is 2 Angstrom.
- **maxNB**: The maximum force at distance 0 Angstrom used in the extrapolation of the left-hand side of the non-bonded potential tables. The default is 100 Kcal/mole-Angstrom.
- **maxB**: The maximum force at distance 0 Angstrom used in the extrapolation of the bond potential tables. The default is 100 Kcal/mole-Angstrom.
- **maxA**: The maximum force at angle 0 degrees used in the extrapolation of the angle potential tables. The default is 10 Kcal/mole.
- **binNB**: The bin width used in the interpolation and extrapolation of the bond and non-bonded potential tables. The default is 0.01 Angstrom.

- **binAD**: The bin width used in the interpolation and extrapolation of the angle and dihedral tables. The default is 0.05 degrees.
- **script4inverse**: If true, the script file will be compatible with the 'scg4py.runLMP' and 'scg4py.RefinePot' programs. Otherwise, the script file can be used in the bash terminal.
- **gpu**: If true, all the commands required to simulate on GPUs are written in the script file. Be careful that you must compile the LAMMPS software with the GPU package to use this command.

Outputs:

- The LAMMPS script file ([sysName].lmp.in)
- The non-bonded tabulated potential file ([sysName].tab_NB)
- The bond tabulated potential file ([sysName].tab_B)
- The angle tabulated potential file ([sysName].tab_A)
- The dihedral tabulated potential file ([sysName].tab_D)

2.8 scg4py.runLMP(sysName, LMP, MPI=None, nProc=None, restart=False, maxRestarting=10)

This program uses outputs of 'scg4py.genLMPscript' and 'scg4py.genLMPdata' to run the CG simulation with LAMMPS software.
Parameters:

- **sysName**: The system name used as the base name of the input files.
- **LMP**: The path of the LAMMPS executable.
- **MPI**: The path of the mpirun executable. By default, the MPI executable is not used.
- **nProc**: The number of parallel processes. The default is None.
- **restart**: If True, Simulation continues from the previous restart point. The default is False.
- **maxRestarting**: The maximum number of restarting if the simulation was interrupted. If the simulation is interrupted frequently, this indicates bad physics, e.g. too large a timestep, etc. The default is 10.

2.9 scg4py.trajCat(inTrajs, outTraj, logging=True)

This program concatenates input trajectory files in sorted order.
Parameters:

- **inTrajs**: List of input trajectory files.
- **outTraj**: Output trajectory file.
- **logging**: If true, the time and the number of snapshots of input trajectories are displayed on the screen.

2.10 scg4py.trajConv(inTraj, top, outTraj=None, begin=1, end=-1, stride=1, rmPBC=True, lastSnap=False)

This program converts the input CG trajectory in ways of cutting the trajectory, removing periodic boundary condition, and changing the file format.
Parameters:

- **inTraj**: Input trajectory file.
- **top**: CG topology file.
- **outTraj**: Output trajectory file.
- **begin**: The number of the first frame to read from the input trajectory.
- **end**: The number of the last frame to read from the input trajectory.
- **stride**: Only write every nr-th frame.
- **rmPBC**: If True, it removes the periodic boundary condition. The default is True.
- **lastSnap**: If True, it stores only last snapshot. The default is False.

2.11 scg4py.parseLMPLog(logFile, timeStep, plot=True)

This program parses the LAMMPS log file. Only 'thermo_style multi' of the log file is acceptable.
Parameters:

- **logFile**: Input LAMMPS log file.
- **timeStep**: The simulation time step in femtosecond.
- **plot**: If true, the plot of each parameter over time is saved.

Outputs:

- A figure for the total energy over time([logFile].TotEng.png)

- A figure for the kinetic energy over time([logFile].KinEng.png)
- A figure for the temperature energy over time([logFile].Temp.png)
- A figure for the potential energy over time([logFile].PotEng.png)
- A figure for the bond energy over time([logFile].E_bond.png)
- A figure for the angle energy over time([logFile].E_angle.png)
- A figure for the dihedral energy over time([logFile].E_dihed.png)
- A figure for the improper dihedral energy over time([logFile].E_impro.png)
- A figure for the Van der Waals pairwise energy over time([logFile].E_vdwl.png)
- A figure for the coulombic pairwise energy over time([logFile].E_coul.png)
- A figure for the long-range kspace energy over time([logFile].E_long.png)
- A figure for the pressure energy over time([logFile].Press.png)
- A figure for the volume energy over time([logFile].Volume.png)

Return variables:

- Step, CPU, TotEng, KinEng, Temp, PotEng, E_bond, E_angle, E_dihed, E_impro, E_vdwl, E_coul, E_long, Press, Volume

2.12 scg4py.RefinePot(setFile, LMP, MPI=None, nProc=None)

This program reads an input options file, then refines the initial potential function through IBI or IMC methods.
Parameters:

- **setFile**: Input setting file.
- **LMP**: The path of the LAMMPS executable.
- **MPI**: The path of the mpirun executable. By default, the MPI executable is not used.
- **nProc**: The number of parallel processes. The default is None.

Outputs:

- The last configuration of iteration n (i-*.[output].data.out)
- The potential fuctions of iteration n (i-*.[output].pot)
- The potential updates of iteration n (i-*.[output].dpot)
- The distribution functions of iteration n (i-*.[output].hist or i-*.[output].dist)
- The simulation log file of iteration n (i-*.[output].log)
- The simulation trajectory file of iteration n (i-*.[output].lammpstrj or .xtc)
- The log file of this program ([output].SCG.log)

2.13 Mapping file

AATRAJ The path of the input atomistic trajectory. All file formats of lammptrj, xtc, dcd, gro, and pdb are supported.

CGTRAJ The path of the output CG trajectory.

MOL The molecule name for each type of CG molecule (e.g. MOL = pep pro).

NMOL The number of each molecule type (e.g. NMOL = 20 40).

MOL:(MOL name):ATOM = N The atomistic description section of molecule(s). N is the number of atoms involved in each molecule type (e.g. MOL:pep:ATOM = 112). Information about each atom must be entered in a separate line with below template.
AtomId AtomName AtomCharge AtomMass (e.g. 1 C1 -0.302 12.011).

MOL:(MOL name):MAPPING = N The mapping description section of molecule(s). In this section, the name, type, and id of atoms involved in each bead are defined. N is the number of beads involved in each molecule type (e.g. MOL:pep:MAPPING = 4). Information about each bead must be entered in a separate line with below template.
BeadId BeadName BeadType : AtomIds (e.g. 1 C1 CH3 : 1 2 3 4).

MOL:(MOL name):BOND = N The definition section of the bonds in the molecule(s). N is the number of bond type(s) in each molecule type. Information about each bond type must be entered in a separate line with below template.
BondTypeId : BeadId1 BeadId2 [, Beadid1 Beadid2 , ...]
(e.g. 1 : 1 2, 2 3)
Enter the identical *BondTypeId* for the same bond type between the different molecule types. For example:
MOL:pep:BOND = 2
1 : 1 2, 2 3
2 : 3 4
MOL:pro:BOND = 3
3 : 1 2, 3 4, 5 6
1 : 2 3, 4 5
4 : 6 7

MOL:(MOL name):ANGLE = yes or no if yes, generates all types of angle in the molecule automatically (e.g MOL:pep:ANGLE = yes).

MOL:(MOL name):DIHEDRAL = yes or no if yes, generates all types of dihedrals in the molecule automatically (e.g MOL:pep:DIHEDRAL = no).

The ATOM, MAPPING, BOND, ANGLE, and DIHEDRAL sections should be repeated for each molecule type.

2.14 Setting file

TOP The CG topology file

Temp The temperature of the system in Kelvin.

Cutoff The non-bonded interaction cutoff distance. The default is 20 Angstrom.

cutoffSkin A distance before the cutoff radius used in the extrapolation of the right-hand side of the non-bonded potentials. The default is 2 Angstrom.

maxNB The maximum force at distance 0 Angstrom used in the extrapolation of the left-hand side of the non-bonded potential tables. The default is 100 Kcal/mole-Angstrom.

maxB The maximum force at distance 0 Angstrom used in the extrapolation of the bond potential tables. The default is 100 Kcal/mole-Angstrom.

maxA The maximum force at angle 0 degrees used in the extrapolation of the angle potential tables. The default is 10 Kcal/mole.

nEQsnaps The number of snapshots from the trajectory is discarded as the equilibration of the system in each iteration. The default is 20.

METHOD There are two options for potential refinement: IBI and IMC.

Iterate The number of iterations for potential refinement.

CorFactor It is the correction factor to regulate the convergence of the distribution functions. The default is 0.1.

Lambda It is the regularization parameter used in the Tikhonov regularization method. If $0 \leq Lambda \leq 1$, it is multiplied by the maximum singular values of the correlation matrix. If $Lambda > 1$, it remains unchanged. The default is 0.

DPotSmooth This is a value equal or greater than zero which is used in exponential smoothing of the potential updates before summing to the previous potential functions. The default is 0.5.

PotSmooth This is a value equal or greater than zero which is used in exponential smoothing of the potential functions before using in the next iteration. The default is 0.

PressTarget The target pressure of the system used in the pressure correction method. The default is 0 bar.

PressFactor It is the scaling factor in the pressure correction method. If its value is greater than zero, the pressure correction method will be used in the potential refinement. The default is 0.

CorNB If yes, the non-bonded potentials will be refined during the iterative process. The default is yes.

CorB If yes, the bond potentials will be refined during the iterative process. The default is yes.

CorA If yes, the angle potentials will be refined during the iterative process. The default is yes.

CorD If yes, the dihedral potentials will be refined during the iterative process. The default is yes.

inPot The filename of input potential functions to be corrected.

refHist The filename of reference distribution functions.

output The base name used as the output files.

lastConf If yes, the last configuration of each iteration is used as the initial structure of the next iteration.

SaveTraj If yes, the simulation trajectory of each iteration is stored.

SysName System name used in the simulation with LAMMPS software.

References

[1] Emiliano Brini, Elena A Algaer, Pritam Ganguly, Chunli Li, Francisco Rodríguez-Ropero, and Nico FA van der Vegt. Systematic coarse-graining methods for soft matter simulations–a review. *Soft Matter*, 9(7):2108–2119, 2013.

[2] Helgi I Ingólfsson, Cesar A Lopez, Jaakko J Uusitalo, Djurre H de Jong, Srinivasa M Gopal, Xavier Periole, and Siewert J Marrink. The power of coarse graining in biomolecular simulations. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 4(3):225–248, 2014.

[3] Sergei Izvekov and Gregory A Voth. A multiscale coarse-graining method for biomolecular systems. *The Journal of Physical Chemistry B*, 109(7):2469–2473, 2005.

[4] Alexander P Lyubartsev and Aatto Laaksonen. Calculation of effective interaction potentials from radial distribution functions: A reverse monte carlo approach. *Physical Review E*, 52(4):3730, 1995.

[5] Alexander P Lyubartsev, Aymeric Naômé, Daniel P Vercauteren, and Aatto Laaksonen. Systematic hierarchical coarse-graining with the inverse monte carlo method. *The Journal of Chemical Physics*, 143(24):243120, 2015.

[6] WG Noid. Perspective: Coarse-grained models for biomolecular systems. *The Journal of chemical physics*, 139(9):09B201_1, 2013.

[7] Dirk Reith, Mathias Pütz, and Florian Müller-Plathe. Deriving effective mesoscale potentials from atomistic simulations. *Journal of computational chemistry*, 24(13):1624–1636, 2003.

[8] David Rosenberger, Martin Hanke, and Nico FA van der Vegt. Comparison of iterative inverse coarse-graining methods. *The European Physical Journal Special Topics*, 225(8-9):1323–1345, 2016.

[9] M Scott Shell. The relative entropy is fundamental to multiscale and inverse thermodynamic problems. *The Journal of chemical physics*, 129(14):144108, 2008.