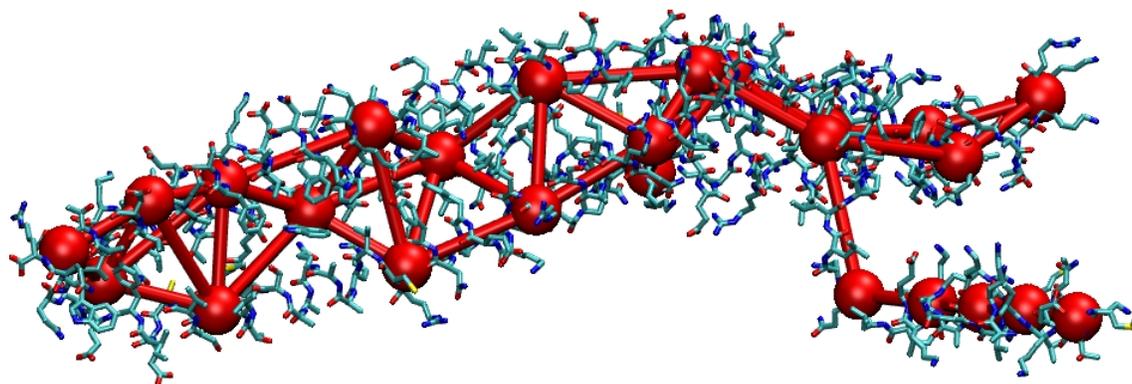


University of Illinois at Urbana-Champaign
Beckman Institute for Advanced Science and Technology
Theoretical and Computational Biophysics Group
Computational Biophysics Workshop

Shape-Based Coarse Graining



Anton Arkhipov

Ying Yin

Danielle Chandler

Jen Hsin

Kirby Vandivort

A current version of this tutorial is available at
<http://www.ks.uiuc.edu/Training/Tutorials/>

Contents

1	Coarse-graining an atomic structure	8
1.1	Coarse-graining of a BAR domain monomer.	9
1.2	Mapping the coarse-grained monomer structure onto a different copy of the monomer.	13
2	Parameterizing SBCG force field	16
2.1	Non-bonded interaction parameters.	16
2.2	Obtaining initial guess for bonded interaction parameters from all-atom simulations.	19
2.3	Iterative refinement of bonded interaction parameters.	21
3	Building a shape-based coarse-grained membrane	27
3.1	Generate a patch of coarse-grained membrane.	28
3.2	Add charged lipids to the membrane patch.	29
4	Combining proteins and membrane for a simulation	30
5	Running a coarse-grained simulation	32
5.1	Preparing a configuration file.	32
5.2	Simulation outputs.	34

Introduction

In this session, we will learn about coarse-grained (CG) molecular dynamics (MD) simulations. Coarse-graining refers to making a simplified model of a molecular system, e.g., reducing groups of atoms to point masses (“beads”). As a result, systems too large and processes too slow for all-atom MD simulations with current computing resources still can be studied using CG models. Of course, this comes at a price of reduced accuracy.

This tutorial presents one CG method that has been quite successful in a number of applications, termed shape-based coarse-graining (SBCG; see Arkhipov et al., *Structure*, **14**:1767, 2006; *Biophys. J.*, **95**:2806, 2008). In this method, a small number of CG beads are used to represent overall shapes of proteins or lipid membranes, with typical ratio of 200-500 atoms per bead. The tutorial introduces tools for SBCG modeling that are provided in VMD as plugins (<http://www.ks.uiuc.edu/Research/vmd/plugins/cgtools/>).

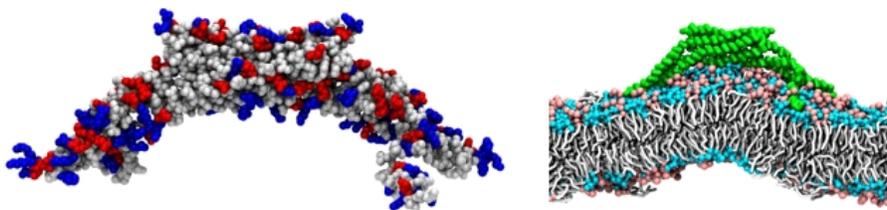


Figure 1: Atomic structure of a BAR domain dimer. Left, charged residues are shown in blue (positive) and red (negative). Right, a BAR domain bending negatively charged membrane in an all-atom MD simulation.

For exercises, we will use proteins called BAR domains (Peter et al., *Science*, **303**:495, 2004). BAR domains are α -helical bundles capable of forming homodimers, featuring a high density of positively charged residues on their curved surface. Accordingly, they can bind to and bend negatively charged membrane, which makes them key players in membrane remodeling processes in cells. In experiments, multiple BAR domains are often observed acting in concert, forming regular lattices on the membrane surface, which enhances membrane bending. Due to the large size of a lattice involving multiple BAR domains, and long time scales needed to observe membrane bending, all-atom MD falls short of revealing many important characteristics of the process, and one has to employ CG approaches, such as SBCG (Arkhipov et al., *Biophys. J.*, **95**:2806, 2008).

Since CG approaches employ greatly simplified models of molecules, one should be very careful in applying them. The SBCG method represents shapes of large molecules consisting of thousands of atoms with a small number of beads, typically 10 to 50. Furthermore, the arrangement of the beads is tuned to reproduce the shape of the molecule, such as that available from an X-ray crystal structure; assemblies of beads representing individual molecules behave

as near-rigid, although elastic, bodies. Thus, processes where significant changes in structures of individual molecules may happen, or where fine atomic-level interactions are important, are not good candidates for SBCG studies.

The SBCG model has been designed to permit handling of SBCG structures and simulations in the same way as it is done for all-atom structures and MD simulations. That is, operating in an SBCG representation, one uses VMD and NAMD without any changes in comparison with the all-atom case, and works with the same file types as for all-atom modeling, such as PSF and PDB for structures, and topology, parameter, and configuration files for simulations (see VMD and NAMD tutorials, <http://www.ks.uiuc.edu/Training/Tutorials/>). However, these SBCG PSF, PDB, parameter and topology files first need to be created according to the all-atom model that one desires to coarse-grain. In this tutorial, we will learn how to use the SBCG plugins of VMD to build such files, and to refine parameters of the SBCG models for subsequent simulations.

One should keep in mind that with the sizes of systems that are typically addressed using SBCG models, limitations of PDB/PSF file formats (how large a number the file can hold in the coordinate, mass, and charge fields) may easily become an issue. For example, the coordinate field in PDB files only allows values less than 10,000 Å, or 1 μm. If a larger system is considered, in cases when PSF/PDB files are necessary one should scale (diminish) coordinates to reduce the system size, while for actual NAMD simulations or for work with VMD one should use binary file formats, such as those of `.coor` or `.dcd` formats, which do not have size limitations. In this tutorial, we will not have such a problem as we consider a relatively small system for the exercises.

Required Programs

The following programs are required for this tutorial:

- **VMD:** The tutorial assumes that you already have a working knowledge of VMD, which is available at <http://www.ks.uiuc.edu/Research/vmd/> (for all platforms). The VMD tutorial is available at <http://www.ks.uiuc.edu/Training/Tutorials/vmd/tutorial-html/>
- **NAMD:** In order to perform simulations with the CG model in this tutorial, NAMD should be correctly installed on your computer. For installation instructions, please refer to the NAMD Users' Guide. The NAMD tutorial is available in both Unix/MacOSX and Windows versions: <http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/> <http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/>
- **Plotting Program:** Under Unix/MacOSX, one can use program `xm-grace`, available at <http://plasma-gate.weizmann.ac.il/Grace/> (Free download), or `gnuplot`, <http://www.gnuplot.info/> (Free download). Under Windows, one can use Microsoft Excel, available at <http://office.microsoft.com/en-us/FX010858001033.aspx> (Purchase required), or `scilab`, available at <http://www.scilab.org/> (Free download). Other useful graphing programs, with versions available for both Unix/MacOSX and Windows, are Mathematica, <http://www.wolfram.com/> (Purchase required) and Matlab, <http://www.mathworks.com/> (Purchase required).

Most of the exercises in the tutorial are performed using Shape-Based Coarse-Graining (SBCG) Tools in VMD. The Tools are implemented as a set of plugins available with their Graphical User Interfaces (GUIs) through VMD menu:

Extensions → Modeling → CG Builder

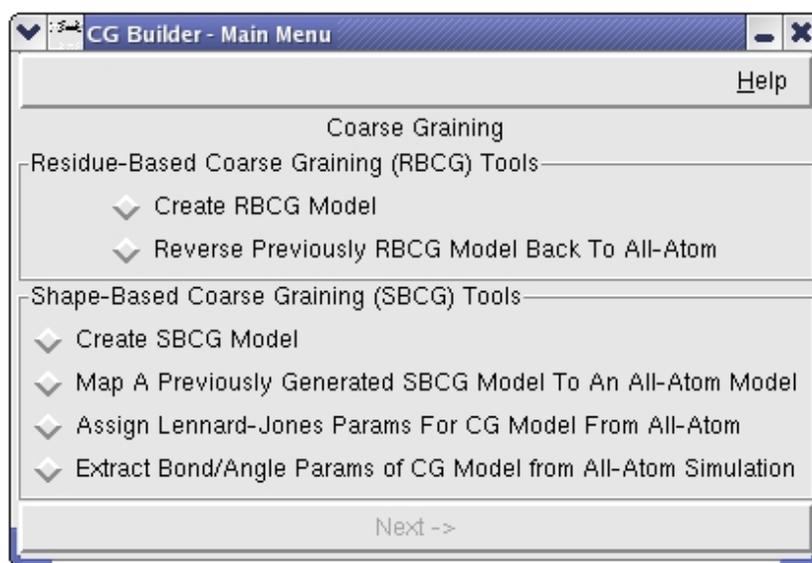


Figure 2: Main Graphical User Interface for the CG Builder Tools in VMD. Available are several tools for two CG models, one of which is the SBCG model addressed in this tutorial.

Getting Started

If you are performing this tutorial at a Computational Biophysics Workshop offered by the Theoretical and Computational Biophysics Group, a copy of the files needed for this tutorial have been set up for you. They can be found in your home directory, under the path `~/Workshop/sbcg-tutorial/files/`.

- **Unix/Mac OS X Users:** In a Terminal window type:

```
cd <path to the directory sbcg-tutorial/files/>
```

You can list the content of this directory by using the command `ls`.

- **Windows Users:** Navigate to the `sbcg-tutorial` → `files` directory using Windows Explorer.

If you downloaded the tutorial from the web you will also need to download the appropriate files, unzip them, and place them in a directory of your choosing. You should then navigate to that directory in a similar manner as described directly above. The files for this tutorial are available at <http://www.ks.uiuc.edu/Training/Tutorials/>

In the figure below, you can see the structure of the directory containing files for each of the tutorial exercises.

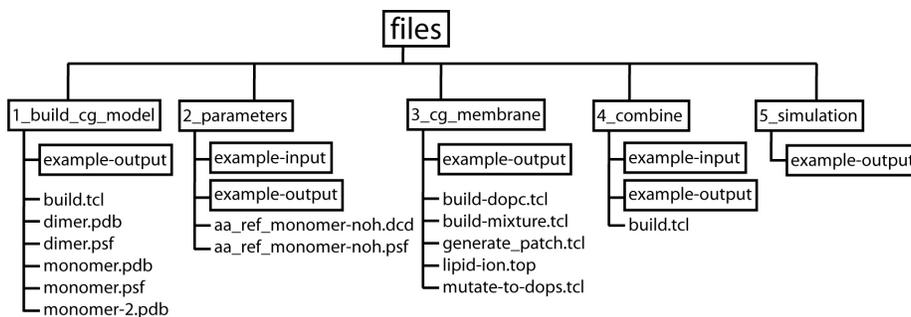


Figure 3: Directory Structure for tutorial exercises. Sample output for each exercise is provided in an “example-output” subdirectory within each folder. When possible, exemplary output files from one exercise are used as exemplary input files for the next exercise; they can be found in “example-input” subdirectories.

1 Coarse-graining an atomic structure

We will work with the amphiphysin BAR domain dimer from *Drosophila* (Peter et al., *Science*, **303**:495, 2004). It is a homodimer, i.e., it consists of two identical monomers. It is natural to employ exactly the same SBCG model for each monomer, which can be done by coarse-graining one monomer first, and then copying the resulting SBCG model and aligning it with the orientation and position of the second monomer. In this section, we will learn how to use SBCG VMD plugins for both steps.

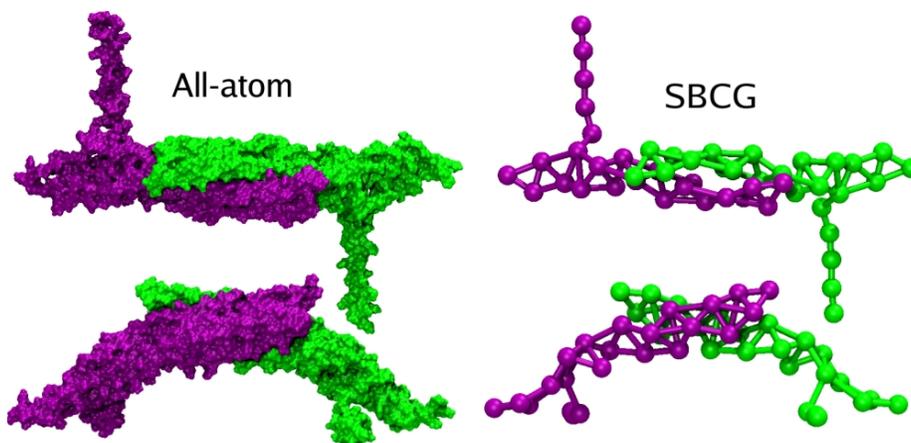


Figure 4: BAR domain homodimer. The two monomers are shown in green and purple. The all-atom structure is shown on the left, and an example of a SBCG structure is shown on the right. Both all-atom and SBCG structures are shown from the top and from the side.

Mapping an SBCG structure created for one protein onto other copies of that protein is a common task in coarse-graining of large macromolecular assemblies, which often contain multiple copies of one protein. A good example are viral capsids, the protein shells enclosing genetic material of viruses. Most known viral capsids are highly symmetric (e.g., icosahedral) structures, composed of multiple copies of a few proteins (see, e.g., Arkhipov et al., *Structure*, **14**:1767, 2006).

Navigate to the directory `1_build_cg_model/`. You can examine the whole dimer in VMD (files `dimer.pdb` and `dimer.psf` in `1_build_cg_model/`). One monomer is designated as `segname P1`, and the other as `segname P2`. You can save each monomer from VMD to separate PDB files using the `writpdb` command for atom selections `segname P1` or `segname P2`, and a PSF using the `writesf` command (one PSF file will work for either monomer, since they are identical except for the atoms' positions). Such PDB and PSF are already created: see `monomer.psf`, `monomer.pdb` and `monomer-2.pdb` in the directory `1_build_cg_model/`. Note that both `dimer.psf` and `monomer.psf` contain in-

formation about individual atoms and bonds between them, but not about angles, since they were created using the VMD command `writepsf` (not to be confused with the `psfgen` command `writepsf`), and VMD does not store information about angles, dihedrals, etc. Because of that, these PSF files cannot be used for MD simulations, but they are sufficient for SBCG conversions.

1.1 Coarse-graining of a BAR domain monomer.

Let us now coarse-grain a BAR domain monomer.

1. Start VMD and load the all-atom monomer structure (load `monomer.psf` and `monomer.pdb` into the same molecule).
2. Open the CG Builder in VMD (`Extensions` → `Modeling` → `CG Builder`), and choose the option “Create SBCG Model”. This will bring you to the SBCG Builder GUI.
3. Make sure that in the GUI you choose “Molecule”, and not “Electron Density Map”. The latter allows one to construct a SBCG model from a density map in CITUS or `.dx` format (such maps can be obtained from cryo-electron microscopy).
4. Choose the monomer from the dropdown for “Molecule”.
5. We do not need to specify the mass of the molecule, since VMD obtains that information from the PSF file you loaded. Without a PSF, VMD makes a good guess of atomic masses based on atom names in the PDB file, but quality of the guess can be compromised if the PDB file contains non-conventional atom names. Thus, it is usually better to specify the mass of the molecule if the PSF is not available, and especially in case you are working with a density map.
6. Set “Number of CG Beads” to 25. This corresponds to approximately 150 atoms per CG bead. Commonly used ratios in SBCG applications are 150 to 500 atoms per CG bead.

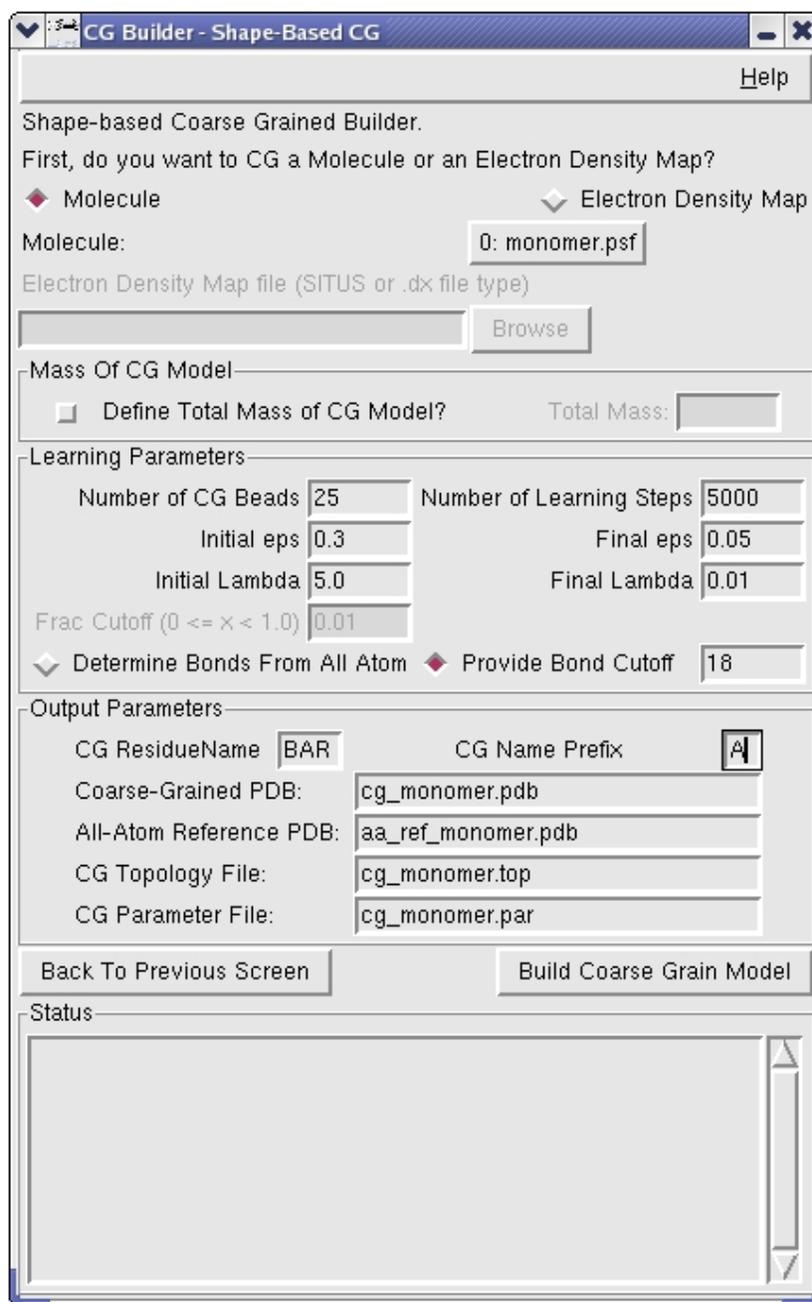


Figure 5: SBCG Builder GUI.



Learning parameters in the SBCG Builder. In SBCG models, CG beads are distributed in space occupied by the all-atom protein using a self-organizing neural network. A few parameters, such as “eps” and “Lambda” are used by the network to go through iterations of the learning algorithm (see Arkhipov et al., *Structure*, 14:1767, 2006). The SBCG Builder will adjust values of those parameters automatically for optimal convergence, depending on the number of CG beads requested by the user, but the user can also modify them, if desired. **The learning algorithm is stochastic, i.e., each time it is used to create a CG model of the same all-atom structure, the result will be slightly different.** However, the overall shape of the protein is maintained each time.

7. Once CG bead positions are assigned, the algorithm connects some of them by bonds. By default, a bond between two beads is established if the parts of the protein represented by each bead are directly connected by the protein backbone (“Determine Bonds From All Atom”). Toggle the other switch on, **Provide Bond Cutoff**, and set the cutoff value to 18. Now, a bond between two beads will be established if the beads are within 18 Å of one another. Which of the two options to choose depends on the application. Choosing connectivity according to the protein backbone is more realistic, but for the exercise with BAR domains this choice does not matter much for the end result.

8. Change the “CG Residue Name” to “BAR”, and “CG Name Prefix” to “A”. Names of the CG beads will be “A1”, “A2”, “A3”, and so on, up to “A25”.

9. Hit the “Build Coarse Grain Model” button. Completion of the SBCG algorithm will take a few moments.

10. The main result of running the algorithm is the production of output files that are written on the hard drive, namely the SBCG topology, parameter, and PDB files, and an all-atom reference PDB file. If you want to have specific names for those files, they can be changed in the SBCG Builder GUI before hitting “Build Coarse Grain Model” button. The output PDB file containing the newly constructed SBCG model is automatically loaded in VMD as a new molecule, overlapped with the original all-atom model.

SBCG Builder in the text mode. All SBCG tools in VMD can also be used in the text mode, which is very convenient for employing these tools in scripting. For a single-time use, source the tools in the VMD Tk Console. First, make sure that the SBCG tools package is loaded, by typing in the Tk Console

```
> package require cgtools
```

The command to run SBCG Builder has the following syntax:

```
> ::cgnetworking::networkCGMolecule statusProc
inMolId cgResName cgPrefix outPDB outAllAtom outTop
outParm numBeads numSteps epsInit epsFinal lambdaInit
lambdaFinal bondCutMethod bondCutDist massValue
```



Here, `statusProc` tells the program where to write the comments, e.g., set it to `puts` if you want all comments printed out in the Tk Console. The other variables mostly correspond to the fields in the SBCG Builder GUI that we have just used. The variable `inMolId` tells the SBCG Builder which molecule from those loaded in VMD to use for the coarse-graining; `bondCutMethod` should be set 0 to use backbone connectivity, or 1 to use the cutoff distance for assigning bonds; `massValue` defines the total mass of the molecule, and if it is negative, the masses of atoms are used directly from VMD. The exemplary usage, corresponding to what we just did with the GUI, is as follows:

```
> ::cgnetworking::networkCGMolecule puts 0 BAR
A cg_monomer.pdb aa_ref_monomer.pdb cg_monomer.top
cg_monomer.par 25 5000 0.3 0.05 5.0 0.01 1 18.0 -1
```

If you want to suppress the log messages, define an empty procedure, e.g.,

```
> proc ps logText
```

and use “ps” instead of “puts”.

11. Sometimes, the SBCG algorithm does not converge well during the allocated learning steps, or the obtained CG model does not look as you like. One common problem may be that the algorithm did not assign positions to all the beads, in which case one or more beads are left “empty” (a warning will appear in the bottom part of the SBCG Builder GUI if this is the case). The simplest solution is just to re-run the SBCG Builder, which usually solves such problems immediately.

12. The SBCG output PDB and topology files determine the structure of the coarse-grained protein model. To obtain the complete structure for display in

VMD, or for subsequent simulations, we need to make a PSF file. This can be done the same way as commonly achieved for all-atom files, namely, using a PSFgen script or employing the AutoPSF VMD plugin. An example PSFgen script is provided: `build.tcl`. Just run the following command in the VMD Tk Console: `source build.tcl`. Note that the script uses the SBCG PDB and topology files you have just created, `cg_monomer.pdb` and `cg_monomer.top`; if you did not place these files in the directory where `build.tcl` is located, you will need to edit `build.tcl` and specify correct paths to the files. If you choose to employ the AutoPSF plugin (**Extensions** → **Modeling** → **Automatic PSF Builder**), remember to delete the default topology file from the list of topologies in the plugin, and add the CG topology file that you created.

SBCG Builder output files. Sample SBCG Builder output files are provided in the folder `1.build_cg_model/example-output`, including also the output files from running the PSFgen script `build.tcl`. Note that all these output files are generally going to be somewhat different from those you create, due to the probabilistic nature of the SBCG algorithm. The SBCG parameter file created by the SBCG Builder is rather a template for the parameter file and needs modification before being used for simulations. The bond lengths, equilibrium angles, and Lennard-Jones radii provided for CG beads in this file are based on a quite reasonable estimate, but the energy constants for bonds, angles, and non-bonded interactions, are set uniformly as constants. For serious CG applications, the latter values should be refined, which is a relatively time-consuming procedure that may involve all-atom simulations used for input, and multiple iterations of CG simulations. We will learn basics of this procedure later in the tutorial.

Please also note that if you are creating a CG structure using a density map as an input, there is usually much less information available to parameterize the CG force field than in the case of an all-atom structure used as an input. For a density map, one usually does not know, for example, the charge distribution over the map. One would have to guess and tune most of the CG force-field parameters based on some assumptions about, e.g., the structure stiffness, which is usually unknown. In the case of an all-atom structure, many characteristics, e.g., charge distribution, can be obtained easily. Such characteristics as the structure stiffness can be estimated using all-atom simulations and employed to parameterize the SBCG force field, as shown in the next section.

1.2 Mapping the coarse-grained monomer structure onto a different copy of the monomer.

We will now create the CG model for the second BAR domain monomer, which is structurally identical to the CG model of the first monomer, by mapping the first model onto the position and orientation of the second monomer.

1. In the CG Builder window, go back from the SBCG Builder GUI to the main CG menu, by hitting the button “Back To Previous Screen”.
2. Choose the option “Map A Previously Generated SBCG Model To An All-Atom Model”, and hit the button Next->.
3. The Mapping GUI requires you to choose the original CG model, reference all-atom structure, and the all-atom model to map onto from the list of molecules currently loaded into VMD. For the **Coarse-Grained Molecule**, choose `cg_monomer.pdb` that you have just created (you can load it to VMD or use the one that has been already loaded automatically after running the SBCG Builder).

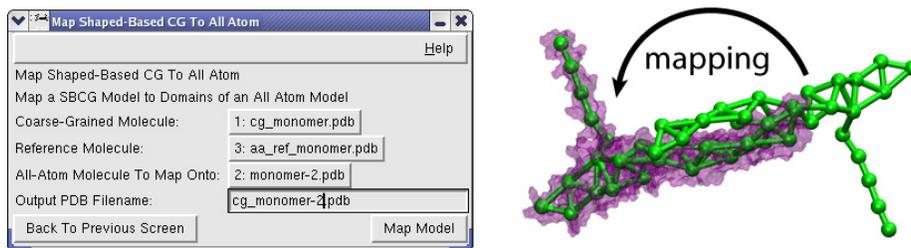


Figure 6: Mapping the SBCG structure of one BAR domain monomer onto the position and orientation of the other monomer. The all-atom structure of the second monomer is shown as a transparent purple surface.

4. For the **Reference Molecule**, load into VMD and choose in the Mapping GUI `aa_ref_monomer.pdb`, which was created by the SBCG Builder. This so-called reference all-atom PDB file is very important for mapping and for fine-tuning the SBCG parameter file (see next section). The reference PDB file contains the same all-atom structure that was used as an input for the SBCG Builder, but its beta-field is filled with numbers that show for each atom, which CG bead it belongs to. Thus, the reference PDB file provides the information about direct mapping of atoms to CG beads for the specific instance of the coarse-graining; a reference all-atom PDB file is always created by the SBCG Builder when a new SBCG model is constructed.
5. For the **All-Atom Molecule To Map Onto**, load into VMD and choose in the Mapping GUI `monomer-2.pdb`, the all-atom PDB file for the second monomer.
6. Set the **Output PDB Filename** to `cg_monomer-2.pdb`.
7. Hit the **Map Model** button. The program will produce the file `cg_monomer-2.pdb`, which will be automatically loaded into VMD. You can create a PSF/PDB pair

for this second monomer's CG model in the same way as was done for the first monomer. Use the same CG topology file. If you want to use the PSFgen script `build.tcl`, do not forget to change `cg_monomer.pdb` to `cg_monomer-2.pdb` there, and call output files differently, e.g., `cg_monomer-2-psfgen.psf` and `cg_monomer-2-psfgen.pdb`.

SBCG Mapping in the text mode. The mapping command has the following syntax:

```
> ::cgtools::mapCGMolecule statusProc AAId CGId refId  
outPDB
```



(see explanation for the SBCG Builder in the text mode, above). `AAId`, `CGId`, and `refId` are the IDs of the molecules loaded in VMD, for the all-atom structure that we want to map the CG structure to, the CG structure, and the all-atom reference file, respectively. For the mapping example above, we would use the following line:

```
> ::cgtools::mapCGMolecule puts 2 1 3 cg_monomer-2.pdb
```

2 Parameterizing SBCG force field

As we mentioned above, the original coarse-graining procedure creates an SBCG parameter file, but the energies associated with various interactions are assigned uniformly and arbitrarily. Refining these interaction parameters to the extent where they realistically reproduce general properties of the original all-atom system is the most important, and perhaps the most difficult, part of the coarse-graining. In this section, we will learn basic techniques for refining the CG parameters, using information about the original static all-atom structure, or an MD simulation of this structure, as an input.

Navigate to the directory `2_parameters/`. Most files that serve as input for this section have been prepared at the steps described in the previous section (check the contents of the folder `example-input`). Remember that the sample CG PDB, PSF, topology and parameter files provided in `example-input` will not be compatible with your own analogous files, because each run of the SBCG Builder contains stochastic elements, so that the final distribution of CG beads in the SBCG model is never exactly the same.

2.1 Non-bonded interaction parameters.

In the SBCG model, non-bonded interactions are represented by the Coulomb and 6-12 Lennard-Jones (LJ) potentials, just as it is commonly done for all-atom simulations. Since each CG bead inherits the mass and charge of the group of atoms it represents, the charges are already assigned, and we do not need to worry about Coulomb interactions. The LJ interactions appear to be a more complex issue. In early applications of the SBCG model, the interaction strength of the LJ potential was set to a uniform value (see Arkhipov et al., *Structure*, **14**:1767, 2006; *Biophys. J.*, **91**:4589, 2006). Later (Arkhipov et al., *Biophys. J.*, **95**:2806, 2008), the procedure was extended to introduce more specificity for each CG bead. In NAMD, the LJ energy constant ϵ_{ij} for the pair of beads i and j is computed as $\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j}$, where ϵ_i and ϵ_j are the strengths for each bead. In the SBCG model, the value of ϵ_i is assigned for each bead i based on the hydrophobic solvent accessible surface area (SASA) for the protein domain represented by the bead,

$$\epsilon_i = \epsilon_{max} \left(\frac{SASA_i^{hphob}}{SASA_i^{tot}} \right)^2, \quad (1)$$

where $SASA_i^{hphob}$ and $SASA_i^{tot}$ are the hydrophobic and total SASA of the domain i ; ϵ_{max} is the user-defined constant. SASA for an all-atom domain is computed in the context of the whole protein, i.e., atoms that are at the surface between two domains, but are buried inside the protein, do not contribute to the computed value. The idea behind using the SASA to determine ϵ_i is to let hydrophobic beads aggregate and hydrophilic beads dissolve in the solvent, which is only implicitly present in SBCG simulations. For a pair of completely hydrophilic beads, $\epsilon_{ij} = 0$, in which case the two beads are free to dissociate

unless they are bound to other particles; ϵ_{ij} for two completely hydrophobic beads is ϵ_{max} , which should be tuned to represent hydrophobic aggregation realistically. For 150 atoms per CG bead, an appropriate value is approximately $\epsilon_{max} = 10$ kcal/mol.

The formula and parameter values used here (such as $\epsilon_{max} = 10$ kcal/mol) were found to be optimal for certain SBCG applications (Arkhipov et al., *Biophys. J.*, **95**:2806, 2008). For SBCG applications to significantly different systems, one may need to modify the formula or parameters, or both, which can be done by editing the plugin scripts that are delivered together with VMD.

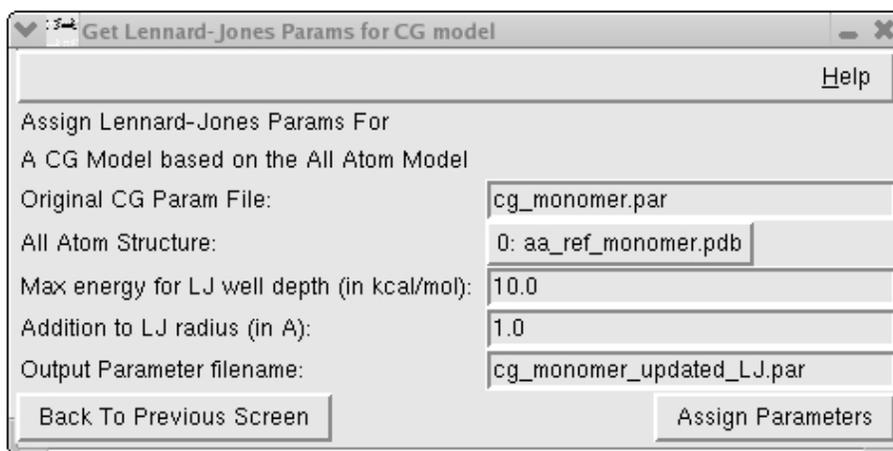


Figure 7: GUI for assigning parameters for non-bonded (LJ) interactions for the SBCG model.

Assigning these parameters is easily done with another GUI in the VMD CG tools.

1. Start VMD if you have closed it, and load the reference all-atom monomer PDB `aa_ref_monomer.pdb`.
2. Start the SBCG LJ GUI in VMD: `Extensions` → `Modeling` → `CG Builder` → `Assign Lennard-Jones Params For CG Model From All-Atom`.
3. In the field “Original CG Param File”, provide the path to the CG parameter file, `cg_monomer.par`, that was created when you ran SBCG Builder. The easiest way is to copy this file directly in the folder you are working in, `2_parameters/`. Then, you can just type “`cg_monomer.par`” in the corresponding field in the GUI (make sure that VMD’s working directory is `2_parameters/` - you can check it by using the command “`pwd`” in the VMD’s Tk Console, and if necessary navigate to the right directory using command “`cd`”). Otherwise,

provide the full path in the GUI's field.

4. For “All Atom Structure”, choose the molecule `aa_ref_monomer.pdb`.
5. “Max energy for LJ well depth” is the aforementioned ϵ_{max} ; set it to 10 kcal/mol.
6. The LJ radius σ_{ij} for the interaction between beads i and j is obtained in NAMD by default as $\sigma_{ij} = (\sigma_i + \sigma_j)/2$, where σ_i and σ_j are radii associated with each atom. In SBCG model, each σ_i is obtained as a radius of gyration of the groups of atoms represented by the CG bead, increased by a constant value $\Delta\sigma$, which accounts for the fact that each atom has a radius (typically, 1-2 Å). This value is chosen by the user in the GUI field “Addition to LJ radius”. Set $\Delta\sigma = 1$ Å.
7. Choose the Output Parameter filename” to be `cg_monomer_updated.LJ.par` and hit the “Assign Parameters” button.
8. Compare entries for non-bonded interactions in the originally produced parameter file `cg_monomer.par` with those in the new file, `cg_monomer_updated.LJ.par`. The LJ radii σ_i are very similar between the two files, because the original SBCG Builder algorithm uses almost the same procedure as we just employed to assign these values ($\Delta\sigma = 1$ Å in both cases). However, LJ energies in `cg_monomer.par` are set uniformly to 4 kcal/mol by the SBCG Builder, and now they are changed significantly in the updated file, accounting for the choice $\epsilon_{max} = 10$ kcal/mol and for the specificity of each CG bead.

Assigning LJ Parameters in the text mode. The syntax for this command is

```
> ::cgtools::sasa_LJ_networking statusProc par_CG
  pdbrefID f_out ELJ RLJ
```



For the example that we have just considered, we would need to run this command using the following values:

```
> ::cgtools::sasa_LJ_networking cg_monomer.par 0
  cg_monomer_updated.LJ.par 10.0 1.0
```

Make sure that you are using the correct VMD molecule ID for the all-atom reference file!

2.2 Obtaining initial guess for bonded interaction parameters from all-atom simulations.

The terms for bonded interactions in the SBCG method are described by potentials $V_{bond}(r) = K_b(L - L_0)^2$ and $V_a(\theta) = K_a(\theta - \theta_0)^2$ for bond length L and angle θ , where K_b , L_0 , K_a , and θ_0 are the force-field parameters. The procedure for extracting the values for these parameters from an all-atom simulation is based on the concept of the so-called Boltzmann inversion: for each variable x (such as i -th bond length L_i), one obtains the distribution $\rho(x)$ from the all-atom simulation, and uses the Boltzmann relation $\rho(x) = \rho_0 \exp[-V(x)/k_B T]$ to obtain $V(x)$. This procedure can be illustrated by the simple example of a one-dimensional harmonic oscillator, with a particle moving along the x coordinate in the potential $V(x) = f(x - x_0)^2$ (note that there is no $1/2$ factor, according to the CHARMM force-field notation). With the system in equilibrium at temperature T , the average position $\langle x \rangle$ is equal to x_0 , and the root mean square deviation of x (RMSD) is given by $\langle x^2 \rangle - \langle x \rangle^2 = k_B T / (2f)$ (k_B is the Boltzmann constant). Using an MD simulation, one can compute $\langle x \rangle$ and the RMSD, thus obtaining x_0 and f ; note that in this case it is not necessary to compute complete $\rho(x)$. These formulas are used in the SBCG method to obtain an initial guess for K_b , L_0 , K_a , and θ_0 .

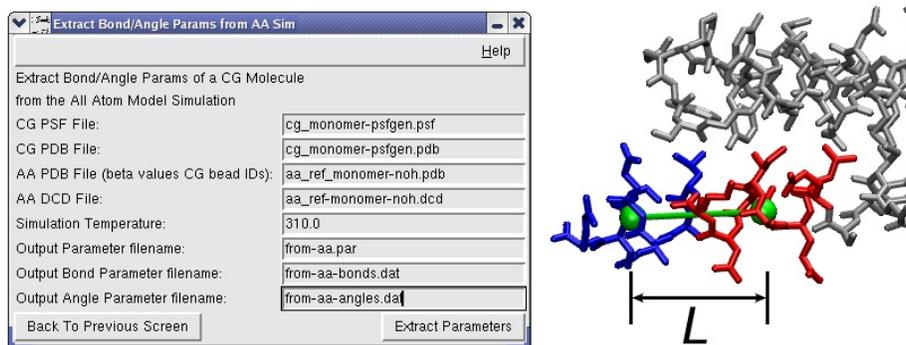


Figure 8: Extracting bonds and angles for the CG model from an all-atom simulation. The GUI is shown on the left. An example of analyzing a CG bond is shown on the right. L is the distance between the centers of mass of all-atom domains (blue and red) that correspond to the two CG beads being analyzed (green).

The procedure consists of computing positions of centers of mass for each all-atom domain that is represented by one CG bead, at each time frame from the all-atom trajectory. Then, for each pair of beads that forms a bond, or triple of beads that form an angle, the distance L or angle θ between the respective centers of mass is computed and averaged over all time frames; RMSDs are also calculated.

To obtain the initial guess for the bonded CG parameters, we need to perform the following actions.

1. Start the SBCG Bonds GUI in VMD: **Extensions** → **Modeling** → **CG Builder** → **Extract Bond/Angle Params of CG Model from All-Atom Simulation**.
2. In the GUI, provide the path to the CG PSF and PDB files, `cg_monomer-psfgen.psf` and `cg_monomer-psfgen.pdb`. Those are the files that you created after coarse-graining the BAR domain monomer. For convenience, you may want to copy these files directly into the folder you are working in, `2_parameters/`. Then, you can just type, e.g., “`cg_monomer-psfgen.psf`” in the respective field of the GUI. If you want to do this, make sure that VMD’s working directory is `2_parameters/`, as in the previous example. Otherwise, provide the full path in the GUI’s field.
3. The all-atom trajectory is provided: `2_parameters/aa_ref-monomer-noh.dcd`. Add this file name to the GUI’s field “**AA DCD File**”.
4. To save space, the all-atom trajectory contains only heavy protein atoms, i.e., no hydrogen atoms. To match this with a reference PDB file, we need to rid it of hydrogen atoms as well. This can be easily done in VMD by loading the full reference all-atom PDB, `aa_ref_monomer.pdb`, which you created at the first stage of the coarse-graining, and saving only non-hydrogen atoms (selection “`noh`”) in a new PDB file (using either **File** → **Save Coordinates** or the Tk Console command `writpdb`). Call this new file `aa_ref_monomer-noh.pdb`, and add its name to the corresponding field in the GUI.
5. Set **Simulation Temperature** to 310 K (this is the temperature at which the all-atom simulation was run).
6. Set **Output Parameter Filename**, and filenames for the bond and angle datafiles, to `from-aa.par`, `from-aa-bonds.dat`, and `from-aa-angles.dat`. The latter two files are for analysis purposes only. They contain the same parameters for bonds and angles that the output parameter file does, but in addition they also contain the corresponding values of average L for each bond, its RMSD, and the same for angles. All these values are conveniently organized in columns, which makes it easy to visualize the values of interest using plotting programs. This may become necessary for further refinement of the parameters (see below).
7. Hit the **Extract Parameters** button. Since we have not provided a parameter file as an input, the output file `from-aa.par` does not contain any other information besides the parameters for bonds and angles. To rectify this, just append the entries for the non-bonded parameters from the file `cg_monomer_updated_LJ.par` to `from-aa.par`.

Extracting Bond and Angle Parameters in the text mode. For bond and angle parameters, one should run the following command in the Tk console:



```
> ::cgnetworking::all_ba statusProc psf_CG pdb_CG
pdb_ref dcd_ref T f_out dat_bond dat_angle
```

For the example above, this becomes

```
> ::cgnetworking::all_ba puts cg_monomer-psfgen.psf
cg_monomer-psfgen.pdb aa_ref_monomer-noh.pdb
aa_ref_monomer-noh.dcd 310.0 from-aa.par
from-aa-bonds.dat from-aa-angles.dat
```

2.3 Iterative refinement of bonded interaction parameters.

The Boltzmann relation for a single variable x , which we used in the previous step, holds only if x is an independent variable not affected by other potentials. For a network of bonds, which is often the case for SBCG protein models, the bond lengths and angles are not independent, and when parameters for each of them are derived individually using Boltzmann inversion, the stiffness of the structure may be overestimated. Therefore, the direct Boltzmann inversion can be used only to obtain the initial values for force constants K_b and K_a , which need to be further adjusted until the stiffness of the CG model becomes closer to that of the all-atom model. In this section, we will learn an exemplary (and simplified) procedure used to adjust K_b and K_a . For the equilibrium bond length L_0 and angle θ_0 , Boltzmann inversion usually provides a very good guess, so that these parameters do not require any further modification.

1. Let us first run a SBCG simulation using the initial guess for bonds and angles in the parameter file `from-aa.par`. This will be our iteration 1. We will use NAMD for the simulation. Set up a folder `iteration1`, similar to that found in `2.parameters/example-output/`.
2. In your folder `iteration1`, create an empty folder `output`, and a folder `input`; fill the latter one with the files that you created at previous steps: `cg_monomer-psfgen.pdb`, `cg_monomer-psfgen.psf`, and `from-aa.par`. Copy the NAMD configuration file `iteration1.conf` from `2.parameters/example-output/iteration1/` to your folder `iteration1`. Check the contents of the configuration file. You will see that many settings are similar to those used for all-atom simulations, but others are different. For example, the time step used is 100 fs, instead of 1 fs commonly used for all-atom simulations. These parameters can be, in principle, changed depending on what you want to simulate.
3. Now, use NAMD to run the simulation.

4. We will now use the SBCG Bonds GUI to analyze the SBCG simulation trajectory and obtain RMSDs of bonds and angles, just the same way as we did before for an all-atom trajectory. The SBCG Bonds GUI requires a reference PDB file (see above), where the beta-values of atoms are assigned according to the number of CG bead that the atom corresponds to. Here, the structure we analyze is the same as the SBCG model. Thus, the beta-values for each CG bead in the reference PDB file should be the same as the number of that bead. Note that this number starts from 1, i.e., it is offset by one from the bead index in VMD, which starts from 0. You can copy your CG PDB file `cg_monomer-psfgen.pdb` to create the reference PDB, `cg_monomer-beta.pdb`; fill the beta fields in the file using a text editor (see the example in the folder `example-input/`). Alternatively, you can use the simple script, `example-input/beta.tcl`, to do this automatically.

5. Employ the SBCG Bonds GUI as you did before for an all-atom simulation. Use `cg_monomer-psfgen.psf` and `cg_monomer-psfgen.pdb` for CG PSF and PDB files. For the trajectory to analyze (“AA DCD File”), use the CG trajectory that you just obtained from the simulation: `iteration1/output/iteration1.dcd`. For the reference file, type the name of the just created CG reference PDB, `cg_monomer-beta.pdb`. Set Simulation Temperature to 310 K. Set Output Parameter Filename, and filenames for the bond and angle datafiles, to `iteration1/from-iter1.par`, `iteration1/from-iter1-bonds.dat`, and `iteration1/from-iter1-angles.dat`. Hit the Extract Parameters button.

6. The extracted parameters are in the folder `iteration1`. We can ignore the file `from-iter1.par`. Let us look at the files `from-iter1-bonds.dat` and `from-iter1-angles.dat`, which contain columns of data obtained from the CG simulation, in the following order. For the file `from-iter1-bonds.dat`: name of bead 1, name of bead 2, average distance between the two beads, its RMSD, and the bond spring constant obtained from the RMSD using Boltzmann inversion (see above). For the file `from-iter1-angles.dat`: name of bead 1, name of bead 2, name of bead 3, average angle between the three beads, its RMSD, and the angle spring constant similarly obtained from the RMSD.

7. We can now compare these data with those obtained from the all-atom simulation using a plotting program (e.g., compare `iteration1/from-iter1-bonds.dat` with `from-aa-bonds.dat`). One can see that average bond lengths and angles, which for CG simulation are mainly defined by L_0 and θ_0 in the CG parameter file, are essentially the same in the all-atom and CG simulations. Thus, the original assignment of L_0 and θ_0 is appropriate, and we do not need to tune them further.

8. We can further compare the RMSDs of bonds and angles, which demonstrate the structure flexibility. However, practically it is more convenient to compare inverse RMSDs, which, in Boltzmann inversion procedure, are proportional to K_b and K_a . Thus, we can compare the last columns of the `*dat` files for bonds

and angles obtained from the CG simulation with those from the all-atom simulation. Note again the difference: what we compare is not the parameter values used in the two simulations, but stiffnesses of bonds and angles observed in those simulations.

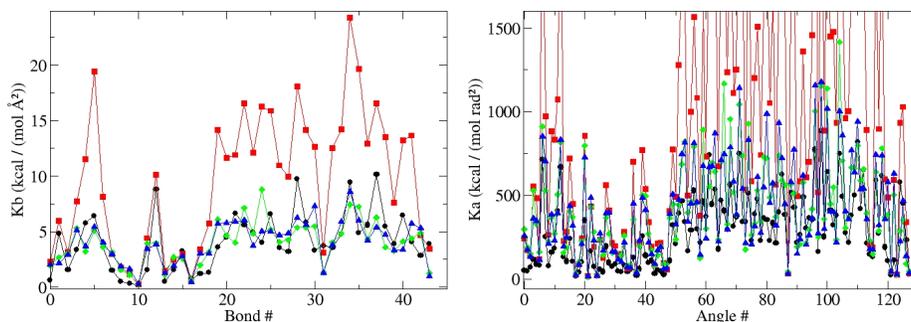


Figure 9: Comparison of bond and angle strengths, K_b and K_a , obtained using Boltzmann inversion from simulations with successively scaled parameter file entries. Since the Boltzmann procedure is used, the values of K_b and K_a are inversely proportional to the RMSDs of corresponding bond lengths and angles, as observed in simulations. Thus, essentially, here we compare stiffnesses of the structure in simulations with different parameters. Comparing RMSDs directly is not as informative as comparing their inverses, or K_b and K_a . Left: bonds; right: angles. Black circles: from all-atom simulation. Red squares: from SBCG, iteration 1. Green diamonds: from SBCG, iteration 2. Blue triangles: from SBCG, iteration 3. The iteration numbers correspond to the files provided in the folder `2_parameters/example-output/`.

9. You will see that the protein structure is much stiffer in the SBCG simulation than it was in the all-atom simulation (i.e., bond and angle RMSDs are smaller, or, inverse RMSDs are higher). This is due to the strong interconnection between beads in the SBCG model, which makes the direct Boltzmann inversion procedure not quite adequate, as mentioned above. To overcome this, we need to decrease K_b and K_a in our parameter file. One can change these parameters one-by-one, and run many CG simulations to find the set of values that gives the right stiffness for every bond and angle; this is, however, very tedious, and, for such coarse model as ours, is probably unnecessary. Below, we follow a simpler approach, where K_b and K_a are scaled uniformly over all bonds and angles, to match the structure stiffness roughly.

10. To scale K_b and K_a , we will use the Scaling GUI in the SBCG tool set (entitled “Scale Bond/Angle Spring Constants”). Note that this GUI can be used for any CHARMM-style parameter file, not necessarily for CG only. Remember, the parameter file we want to scale is the one we used for the simulation in iteration 1, i.e., the file `from-aa.par`. Do not make a mistake of scaling the file `iteration1/from-iter1.par`; parameters in this file reflect the properties of the CG model in iteration 1, while we are interested in reproducing the prop-

erties of the all-atom simulation.

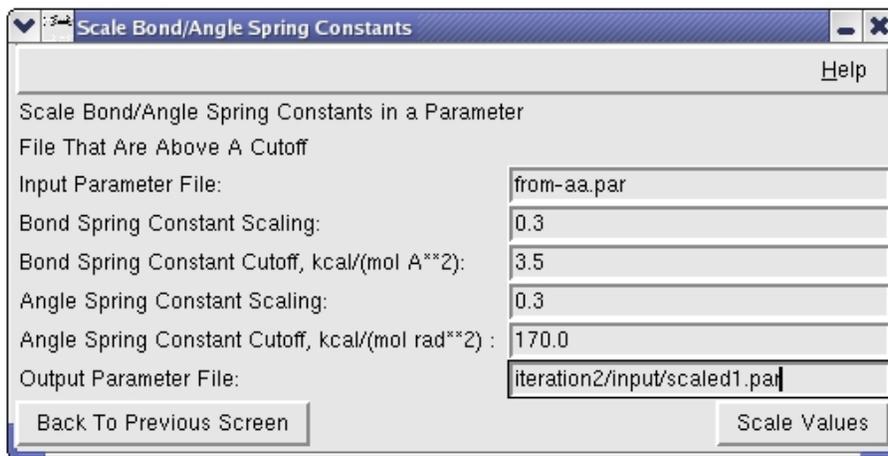


Figure 10: GUI for scaling bonds and angles in a parameter file.

11. Set the Input Parameter File in the GUI to `from-aa.par`. Set scaling constants, for example, to 0.3 for both Bond Spring Constant Scaling and Angle Spring Constant Scaling.

12. The problem with the direct Boltzmann inversion is that it usually makes the CG structure too stiff, i.e., bond and angle spring constants are too large. However, we can see that for some of the spring constants that have relatively small values, the results of the all-atom simulation and of the CG one in iteration 1 agree quite well. It is, therefore, counterproductive to scale K_b and K_a for such weak bonds and angles, since that would make them much weaker than we want. Thus, we do not want to apply scaling to the bonds and angles for which K_b and K_a are below certain threshold. This can be achieved by setting the cutoff level for scaling in the GUI, for example, setting the **Bond Spring Constant Cutoff** to $3.5 \text{ kcal}/(\text{mol } \text{Å}^2)$ and **Angle Spring Constant Cutoff** to $170 \text{ kcal}/(\text{mol } \text{rad}^2)$. You should choose the actual values for scaling and cutoffs according to the agreement between your CG and all-atom simulations.

13. Make a new folder, `iteration2`, for the new CG simulation. Add the same files and subfolders into this folder as you did for `iteration1`. The only difference is that we will need to replace the parameter file `from-aa.par` in folder `iteration2/input` by the one with scaled K_b and K_a . In the GUI, set Output Parameter File to `iteration2/input/scaled1.par`, and hit the **Scale Values** button.

(You must again copy the LJ parameters into the new file.)

14. We can now run the new CG simulation in the folder `iteration2`. After it is done, analyze the CG trajectory using the SBCG Bonds GUI, as in the step 5 above (you can reuse the SBCG reference PDB file that you created earlier). This procedure will give you the following files in the folder `iteration2`: `from-iter2.par`, `from-iter2-bonds.dat`, and `from-iter2-angles.dat`. Plot the data from these files and compare them with the data from the all-atom simulation and from iteration 1. The bond and angles stiffness in interaction 2 should be closer to those observed in the all-atom simulation than the stiffness from iteration 1.

15. The scaling steps should be repeated several times, possibly, separately for bonds and angles. Usually, for the next iteration it is convenient to scale the values from the previous iteration, e.g., scale values from the file `from-aa.par` to obtain `scaled1.par`, then scale values from `scaled1.par` to obtain `scaled2.par`, and so on.

16. The final stiffness of CG bonds and angles does not have to be exactly the same as that from the all-atom simulation. If the average deviation is $\pm 25\%$, this already can be considered a reasonable agreement for such a coarse model. In the `example-output`, we performed three iterations. In iteration 1, the original file `from-aa.par`, with parameters obtained directly from the all-atom simulation, was used. For iteration 2, the values in `from-aa.par` were scaled as described in steps 11-12, to yield `scaled1.par`. For iteration 3, the `Bond Spring Constant Scaling` was set to 1.0, `Bond Spring Constant Cutoff` to 100.0 kcal/(mol \AA^2), `Angle Spring Constant Scaling` to 0.7, and `Angle Spring Constant Cutoff` to 300 kcal/(mol rad^2), i.e., bonds were not scaled.

17. Now the SBCG protein structure is established and the protein force-field is fully parameterized. We can run production SBCG simulations! Note that, depending on the stiffness of the bonds and angles in the resulting model, the time step for simulations can be larger or smaller. The time step of 100 fs, which we used during the parameterization run, can be potentially increased, resulting in faster simulations.

Scaling Bond and Angle Spring Constants in the text mode.

To scale bonds and angles, run the following command in the Tk console:



```
> ::cgnetworking::scaleParameterConstants statusProc  
bondScaleFactor bondCutoff angleScaleFactor  
angleCutoff outParFilename
```

For the example of scaling considered above, namely, the scaling after iteration 1, this becomes

```
> ::cgnetworking::scaleParameterConstants puts  
cg_monomer.par 0.3 3.5 0.3 170.0 scaled1.par
```

3 Building a shape-based coarse-grained membrane

A building block of the SBCG lipid model consists of two beads - a “head” bead and a “tail” bead - connected by a harmonic bond. This is the simplest representation that preserves the elongated shape of a real lipid molecule as well as the separation of hydrophobic and hydrophilic parts. Even so, if we were to represent a single all-atom lipid, which contains less than 150 atoms, in this way, the resulting beads would be too light to accommodate the long timesteps desired in a SBCG simulation. We therefore have to view SBCG lipid “molecules” in a more abstract sense, allowing that one SBCG “molecule” represents a small patch of all-atom lipids, rather than one real all-atom lipid molecule.

At this level of coarse-graining, the differences between lipids of different types (e.g. POPE vs. POPC or DOPC) become almost negligible. However, SBCG lipids can be matched to all-atom simulations by considering qualities such as the bilayer thickness and area per lipid. Since the lipid bilayer thickness is approximated to be 50 Å, each bead therefore has a diameter of 12.5 Å. One SBCG lipid occupies then a cross-sectional area of approximately 156 Å². In this tutorial, we will be working mainly with DOPC, which occupies 72.5 Å² per lipid. One SBCG DOPC lipid must then represent approximately 2.2 all-atom DOPC lipids. The mass of the 2.2 lipids is divided equally among the “head” and the “tail” beads, meaning that the “head” bead actually represents both the head groups of the lipids and some portion of the tails, and the “tail” bead represents the remainder of the lipid tails. Since one DOPC lipid contains 138 atoms, 2.2 lipids corresponds to ~300 atoms, or ~150 atoms per SBCG bead, consistent with the level of coarse graining used for the BAR domain proteins. In many circumstances, one will want to include charged as well as neutral lipids. We therefore will also define negatively charged SBCG lipids to represent DOPS to use in a mixed DOPC/DOPS membrane. Since DOPC is the main lipid, and we have already determined that one SBCG lipid represents 2.2 all-atom lipids, this convention is kept for DOPS. Since DOPS is slightly heavier, the extra mass is added to the DOPS “head” bead so that the “tail” beads of the two lipids can remain identical. The DOPS “head” bead is given a charge of -2.2 to account for the fact that it incorporates 2.2 all-atom DOPS lipids. All these characteristics are reflected in the provided topology file for SBCG lipids, `3_cg_membrane/lipid-ion.top`.

The bond parameters and nonbonded parameters for the SBCG lipids were chosen to consistently reproduce the bilayer thickness and area per lipid values in all-atom simulations, as described in detail in Arkhipov et al., *Biophys. J.*, **95**:2806, 2008. Since we do not run simulations in this section, a parameter file is not necessary at this point. We will need the parameter file later, when we run a CG simulation of a combined lipid-protein system.

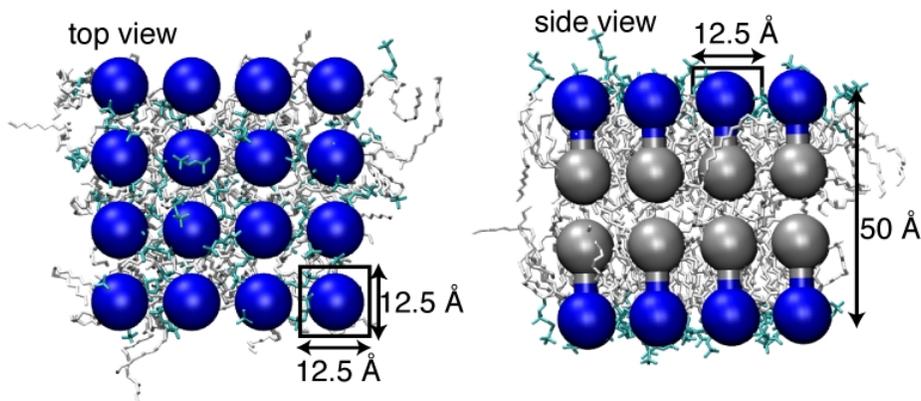


Figure 11: Shape-based coarse-grained lipid bilayer. Each SBCG lipid is comprised of a “head” bead (blue) and a “tail” bead (gray) connected by a bond. The diameter of one bead is 12.5 Å, and the thickness of the bilayer is 50 Å.

3.1 Generate a patch of coarse-grained membrane.

In this section, we will create a patch of SBCG DOPC lipids. The script `3_cg_membrane/generate_patch.tcl` is provided for this task. The usage of this script is

```
generate_patch Nx Ny dx dy hname tname rname outPDB
```

where N_x and N_y are the number of lipids in the x and y directions, dx and dy are the lipid spacings in the x and y directions (which will be 12.5 for our case), `hname` is the name of the head bead (PCH), `tname` is the name of the tail bead (DOT), `rname` is the residue name of the lipid (DOPC), and `outPDB` is the desired name of the output PDB file.

1. To build a patch of DOPC lipids with this script, navigate to the folder `3_cg_membrane` and type the following in the VMD Tk Console:

```
> source generate_patch.tcl
> generate_patch 37 9 12.5 12.5 PCH DOT DOPC dopc.pdb
```

This will create a 37×9 DOPC patch to be used later in this tutorial.

2. To create a corresponding psf file, use the provided psfgen script `3_cg_membrane/build-dopc.tcl` by typing the following into the Tk Console:

```
> source build-dopc.tcl
```

3. One may alternatively use the AutoPSF plugin in VMD to create a psf file. To do this, navigate to **Extensions** → **Modeling** → **Automatic PSF Builder**. In Step 1, check that `dopc.pdb` is set as the input molecule, and set the desired output file name. Under “topology files” delete the default all-atom topology and then add the SBCG topology file `lipid-ion.top` by clicking “Add” and selecting `lipid-ion.top`. In Step 2, Select “Everything” and click “Guess

and split chains using current selections”. In Step 3, click “Create Chains” to create the psf file.

4. Load the resulting pdb and psf files into VMD. You should see a 37×9 array of SBCG lipids.

3.2 Add charged lipids to the membrane patch.

We will now create a combined DOPC/DOPS membrane by changing some of the DOPC lipids to DOPS. The provided script `3_cg_membrane/mutate-to-dops.tcl` selects 30% of the lipids at random and changes them to DOPS.

1. To use this script, type the following into the Tk Console:

```
> source mutate-to-dops.tcl
```

2. Now create a psf file either by using the provided psfgen script `3_cg_membrane/build-mixture.tcl` by typing

```
> source build-mixture.tcl
```

or by using the AutoPSF plugin.

3. Load the resulting pdb and psf files into VMD - you should see that about 30% of the lipids are now DOPS lipids. For example, you can color lipids by `rename` to distinguish between DOPC and DOPS.

You will need this mixed DOPC/DOPS membrane patch for the simulations in following sections.

4 Combining proteins and membrane for a simulation

In this section, you will build a system containing 6 BAR domain dimers on top of a planar membrane. A prepared tcl script, `build.tcl`, which is located at directory `4_combine` will be used to generate this system. In the previous sections, you have generated the building blocks of this system: the BAR domain dimer and the lipid membrane patch. You will also need ions to neutralize the total charge of the system.

1. In a Terminal window in your `4_combine` directory, copy the input files that you have created while working on the previous sections, using the following commands:

```
> cp ../1_build_cg_model/cg_monomer.pdb .
> cp ../1_build_cg_model/cg_monomer-2.pdb .
> cp ../1_build_cg_model/cg_monomer.top .
> cp ../3_cg_membrane/mixture.pdb .
> cp ../3_cg_membrane/lipid-ion.top .
```

The last input files needed is a pdb of ions, `ions.pdb`, which is located at your current directory `4_combine/example-input`.



How ions are treated in SBCG. Here “ions” are SBCG beads carrying a charge of $+2.2|e|$ and mass of 1,000 amu, representing 8 ions of mixed nature (such as both Na^+ and Cl^-), with their hydration shells. Such choice is made to match the coarse-graining of the charged lipids. Other schemes of introducing charges that represent ions in the SBCG model are possible, and the choice of the charge and mass of an SBCG “ion” is done by the researcher, depending on the specifics of the simulated system.

2. Start VMD and in the VMD Tk Console window, type the following command:

```
> source build.tcl
```

This command calls the script `build.tcl`, which will replicate six copies of BAR domain dimers and combine the BAR domains, lipids and ions together.

3. The output of the script `build.tcl` will be a psf file (`6bar.psf`) and a pdb file `6bar.pdb` containing six BAR domain dimers on top of a membrane patch, neutralized by ions.

4. Take a look at `build.tcl`. This script contains the following steps: (1) Load the two monomers of a BAR domain dimer. (2) Move the monomers to appropriate locations, to form a lattice. (3) Call VMD plugin `psfgen` and input SBCG topology files for BAR domains, lipids and ions. (4) Input coordinates of the six BAR domain dimers, lipid membrane, and ions. (5) Generate psf and pdb files for the combined system. More detailed descriptions of the commands are included in `build.tcl` as comments beginning with “#”. You can view the commands and comments in `build.tcl` using any text editor, or, e.g., using `ls` command in the VMD Tk Console.

5. Load the output system into VMD using the following command in the Tk Console:

```
> mol load psf 6bar.psf pdb 6bar.pdb
```

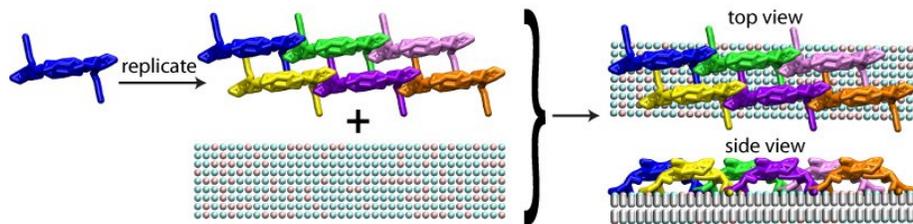


Figure 12: Combining proteins and membrane for a simulation. Ions are not shown for clarity.

6. Measure the center and dimension of the system by following commands:

```
> set sel [atomselect top all]
> measure center $sel
> measure minmax $sel
```

Record the above values for use in the next section.

5 Running a coarse-grained simulation

We are now almost ready to simulate the system of SBCG BAR domains with SBCG membrane. In this section we will discuss first how to write a NAMD configuration file for an SBCG system. We will then perform the simulation, and discuss the file outputs and simulation result.

To perform exercises, navigate to the folder `5.simulation/`.

5.1 Preparing a configuration file.

Since SBCG was designed to be compatible with NAMD, an SBCG configuration file looks similar to a normal, all-atom, NAMD configuration file that you might have used before.

1. A sample configuration file `sim.conf` has been prepared for you in the directory `example-output/`. Copy it to the folder where you want to run the simulation, and open it with a text editor. Remember to create in the same folder subfolder `output` and `input`, and add you CG files to the folder `input` analogously to how it is done in `example-output/`. Note that you will need here an SBCG parameter file for lipids, which we have not used before. For this purpose, copy the file `example-output/input/lipid-ion.par` to your `input` folder.

2. The configuration file contains many options (entries in the first column), followed by their parameters (entries in the second column) specifically chosen for the simulated system. Assuming readers already have experience with NAMD simulations, here we will only go through those options that require special adjustments for an SBCG system. New NAMD users are encouraged to consult the NAMD Tutorial and NAMD User's Guide.

3. In the text editor displaying the content of `sim.conf`, scroll down to the section `# Force-Field Parameters`. Note all lines beginning with `#` are comments ignored by NAMD.

Under `# Force-Field Parameters`, you will find six simulation options that might need different parameters than those of an all-atom simulation. These options define how you want the interactions between beads to be computed. Since each SBCG bead represents a cluster of atoms, parameters such as `cutoff`, `switchdist`, and `pairlistdist` would typically have bigger values than those for an all-atom simulation. The values listed here have been tested to work well for the SBCG BAR domain system, and can be used as starting values for other SBCG systems. Note, however, the `cutoff` parameter should always be bigger than the longest bond length in your simulation system, otherwise your simulation will crash.

4. Scroll down to the section # **Integrator Parameters**.

The parameter `timestep` has a value of `100.0`, implying that the integration timestep of the simulation is 100 fs/step. A typical all-atom simulation uses 1 or 2 fs/step, hence the SBCG gives a speedup of 100 from the choice of integration timestep alone. The choice of the timestep depends on how fast beads are moving in the simulation, and, thus, the maximal timestep possible (so that the simulation does not crash) is determined by the strength of interactions, e.g., stiffness of bonds, as mentioned above. If your simulation crashes with a timestep of 100 fs/step, starting the simulation with a shorter timestep might fix the problem. Then timestep can be increased when the system becomes stable later in the simulation.

5. In the `cellBasisVector1`, `cellBasisVector2`, `cellBasisVector3`, and `cellOrigin` parameters, one specifies the periodic boundary of the system. Compare these values to the system size you have measured in the previous section.

You should see that only `cellBasisVector2` matches the y-dimension of your system, while `cellBasisVector1` and `cellBasisVector3` are both much larger than the x- and z-dimension of the system. This is because we only want the system to be continuous in the y-direction, while in the other two directions we want to allow the membrane to bend freely.

6. Constant temperature is maintained in this SBCG simulation using Langevin dynamics, as usually done in all-atom simulations. You can take a look at these parameters under # **Constant Temperature Control**. In addition to the temperature control, the Langevin dynamics provides means to simulate the solvent effect implicitly. Namely, the Langevin dynamics introduces viscous drag and random forces acting on each CG bead, which can be used to mimic the viscosity of the solvent and the Brownian motion due to random hits from the molecules of the solvent. A single parameter, `langevinDamping`, is used to account for these effects. Here, `langevinDamping` is set to 2ps^{-1} , as this value was found to reproduce the viscosity of water for the coarse-graining level used (150 atoms per CG bead).

7. In the last few lines of `sim.conf`, you can see that the simulation is designed to be minimized by 1000 steps, and then to run for 50,000,000 steps. This corresponds to $50,000,000\text{ steps} \times 100\text{ fs/step} = 5\text{ }\mu\text{s}$ simulation time.

8. Close the text editor displaying `sim.conf`. Run the simulation by typing `namd2 sim.conf > sim.log` in a terminal window.

5.2 Simulation outputs.

On a one-processor machine, this simulation will take about eight days to complete, but actually we do not need to run the full $5\mu\text{s}$. The general trend is obvious already after about the first 10 ns, which can be achieved within half an hour or so. If you do not wish to run the simulation yourself, you can use the files provided in `example-output/` for the following discussion on file outputs and results.

1. Open the logfile of the simulation, `sim.log`, with a text editor. If you did not run the simulation, use `example-output/sim.log`.

The logfile of a simulation contains useful information. When your simulation crashes, checking the logfile for the error message is the first step of fixing the problem. The logfile can also give you an estimate on how long a simulation will run. Find the words “Benchmark time” in `sim.log`, here you can find the speed of the simulation. Now let’s examine the system via VMD.

2. Close the text editor. Open VMD, and load the psf file of the system, `6bar.psf`. If you did not run the simulation, make sure you use the provided `example-output/input/6bar.psf`.

3. Load the output dcd file, `sim.dcd`. If you did not run the simulation, use `example-output/output/sim.dcd`. In this case, you will have 150 frames loaded in VMD, one frame for each nanosecond of the simulation.

Use VMD to take a look at the simulation result. The BAR domain system originally sits on a flat membrane, and after 150 ns of simulation time, the membrane is curved into an arch. A simple SBCG simulation used here demonstrates very well how BAR domain proteins perform their job of sculpting membrane shape. For more information on BAR domain-induced membrane curvature, please see Arkhipov et al., *Biophys. J.*, **95**:2806, 2008.

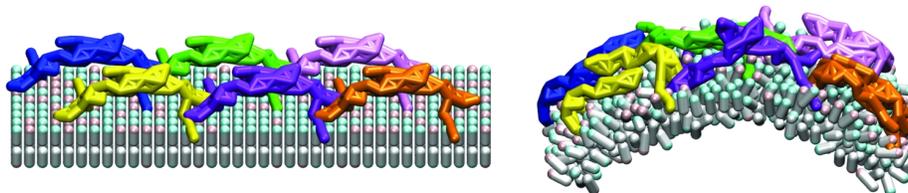


Figure 13: Simulation result of the BAR domain system. Left: beginning of the simulation. Right: system at 150 ns.

4. In your VMD session, use the VMD animation tool to display the last frame, which should show a highly curved BAR domain/membrane system. Open the

Graphical Representations window via **Graphics** \rightarrow **Representations**, and choose the Periodic tab in the lower half of the GUI window. Check on the box of **+X**.

This step should create an additional periodic image of the system in the **+x**-direction. In the VMD OpenGL window, you can see that you now have two systems separated by a large distance. This is the result of giving `cellBasisVector1` a value much larger than the actual x-dimension of the system. This was done to cut off the system from its x-direction periodic neighbors, such that the system is effectively isolated in this direction, making simulation of curvature formation feasible. Same thing was done in the z-direction.

5. Uncheck the box of **+X**, and check **+Y**, and increase the value in “Number of images” on the bottom of the Graphical Representations window to, for example, 10.

The y-direction is continuous, hence you can see in the VMD OpenGL window a long membrane patch, curved as to form a membrane tube.

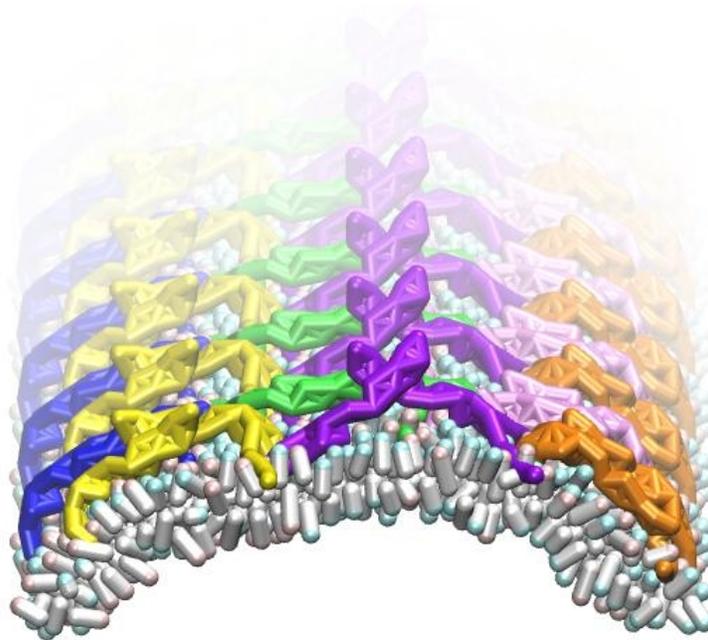


Figure 14: Periodic image in the y-direction.

6. This is the end of the tutorial. You are now ready to use SBCG!