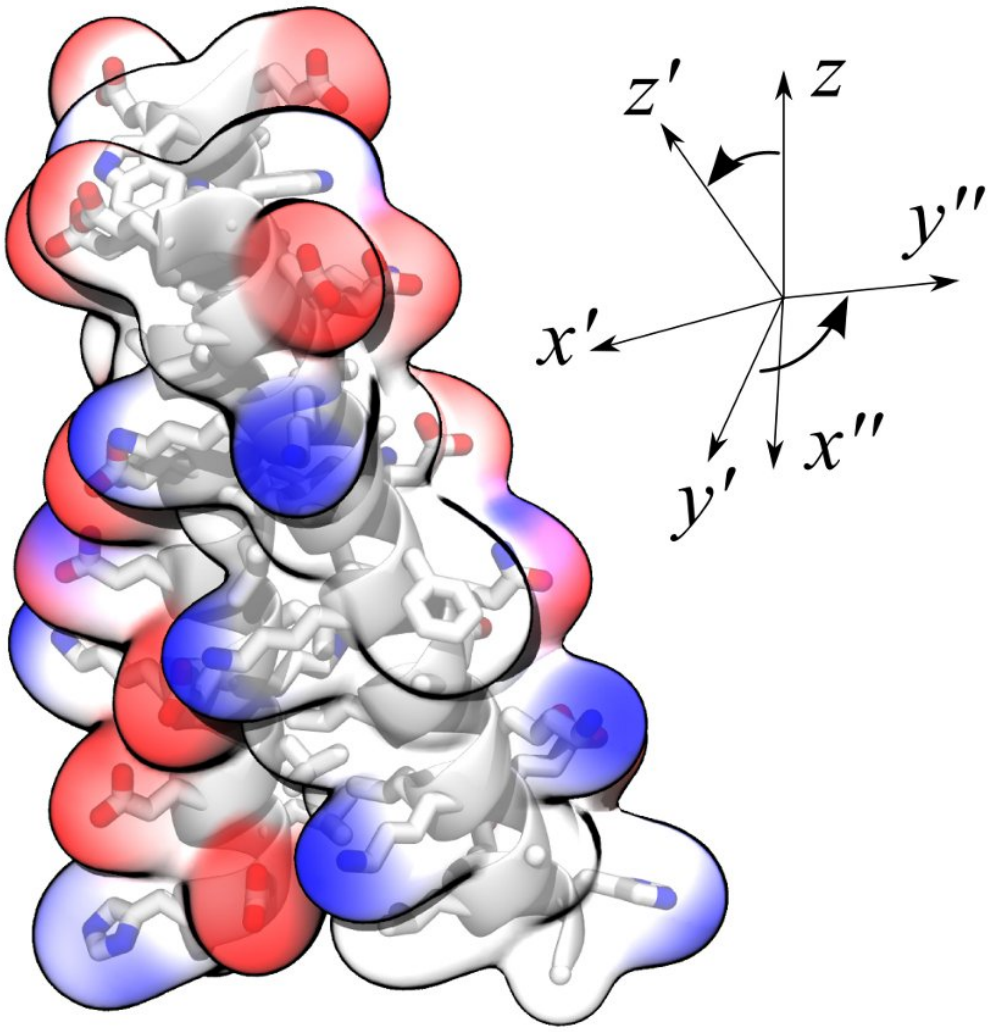
COLLECTIVE VARIABLES MODULE
Reference manual for NAMD

Code version: 2017-09-18



Giacomo Fiorin, Jérôme Hénin

Contents

1 Introduction

In molecular dynamics simulations, it is often useful to reduce the large number of degrees of freedom of a physical system into few parameters whose statistical distributions can be analyzed individually, or used to define biasing potentials to alter the dynamics of the system in a controlled manner. These have been called 'order parameters', 'collective variables', '(surrogate) reaction coordinates', and many other terms.

Here we use primarily the term 'collective variable' (shortened to *colvar*), which indicates any differentiable function of atomic Cartesian coordinates, x_i, with i between 1 and N, the total number of atoms:

$$\xi(t) = \xi\left(x_i(t), x_j(t), x_k(t), \ldots\right) \ , \ \ 1 \leq i, j, k \ldots \leq N \tag{1}$$

This manual documents the collective variables module (**Colvars**), a portable software that interfaces multiple MD simulation programs, with a focus on flexibility, robustness and high performance. The module is designed to perform multiple tasks concurrently during or after a simulation, the most common of which are:

- apply restraints or biasing potentials to multiple colvars, tailored on the system by choosing from a wide set of basis functions, without limitations on their number or on the number of atoms involved; while this can in principle be done through a TclForces script, using the Colvars module is both easier and computationally more efficient;
- calculate potentials of mean force (PMFs) along any set of colvars, using different enhanced sampling methods, such as Adaptive Biasing Force (ABF), metadynamics, steered MD and umbrella sampling; variants of these methods that make use of an ensemble of replicas are supported as well;
- calculate statistical properties of the colvars, such as running averages and standard deviations, correlation functions of pairs of colvars, and multidimensional histograms: this can be done either at run-time without the need to save very large trajectory files, or after a simulation has been completed using VMD and the `cv` command.

Detailed explanations of the design of the Colvars module are provided in reference [1]. Please cite this reference whenever publishing work that makes use of this module.

2 A crash course

Suppose that we want to run a steered MD experiment where a small molecule is pulled away from a binding site. In Colvars terms, this is done by applying a moving restraint to the distance between the two objects. The configuration will contain two blocks, one defining the distance variable (see section4, section6), and the other the moving harmonic restraint (subsection7.4).

```
colvar {
  name dist
  distance {
    group1 { atomNumbersRange 42-55 }
    group2 {
      psfSegID PR
      atomNameResidueRange CA 15-30 }
    }
  }
}
```

```
harmonic {
  colvars dist
  forceConstant 20.0
  centers 4.          # initial distance
  targetCenters 15.   # final distance
  targetNumSteps 500000
}
```

Reading this input in plain English: the variable here named *dist* consists in a distance function between the centers of two groups: the ligand (atoms 42 to 55) and the alpha carbon atoms (CA) of residues 15 to 30 in the protein (segment name PR). The atom selection syntax is detailed in section5.

To the "*dist*" variable, we apply a `harmonic` potential of force constant 20 kcal/mol/Å^2, initially centered around a value of 4 Å, which will increase to 15 Å over 500,000 simulation steps.

3 General parameters and input/output files

Here, we document the syntax of the commands and parameters used to set up and use the Colvars module in NAMD. One of these parameters is the configuration file or the configuration text for the module itself, whose syntax is described in subsection3.2 and in the following sections.

3.1 NAMD parameters

To enable a Colvars-based calculation, two parameters must be added to the NAMD configuration file, `colvars` and `colvarsConfig`. An optional third parameter, `colvarsInput`, can be used to continue a previous simulation.

- `colvars` ⟨Enable the Colvars module⟩
  Context: NAMD configuration file
  Acceptable values: boolean
  Default value: `off`
  Description: If this flag is on, the Colvars module within NAMD is enabled; the module requires a separate configuration file, to be provided with `colvarsConfig`.

- `colvarsConfig` ⟨Configuration file for the collective variables⟩
  Context: NAMD configuration file
  Acceptable values: UNIX filename
  Description: This file contains the definition of all collective variables and their biasing or analysis methods. This file can also be provided by the Tcl command `cv configfile`; alternatively, the contents of the file itself can be given as an argument to the command `cv config`.

- `colvarsInput` ⟨Input state file for the collective variables⟩
  Context: NAMD configuration file
  Acceptable values: UNIX filename
  Description: When continuing a previous simulation run, this file contains the current state of all collective variables and of their associated algorithms. It is written automatically at the end of any simulation with collective variables. This file can also be provided by the Tcl command `cv load`.

3.2 Configuration syntax for the Colvars module

All the parameters defining the colvars and their biasing or analysis algorithms are read from the file specified by the configuration option `colvarsConfig`, or by the Tcl commands `cv config` and `cv configfile`. Hence, none of the keywords described in this section and the following ones are available as keywords for the NAMD configuration file. The syntax of the Colvars configuration is "`keyword value`", where the keyword and its value are separated by any white space. The following rules apply:

- keywords are case-insensitive (`upperBoundary` is the same as `upperboundary` and `UPPERBOUNDARY`): their string values are however case-sensitive (e.g. file names);
- a long value or a list of multiple values can be distributed across multiple lines by using curly braces, "{" and "}": the opening brace "{" must occur on the same line as the keyword, following a space character or other white space; the closing brace "}" can be at any position after that;
- many keywords are nested, and are only meaningful within a specific context: for every keyword documented in the following, the "parent" keyword that defines such context is also indicated;
- the '=' sign between a keyword and its value, deprecated in the NAMD main configuration file, is not allowed;
- Tcl syntax is generally not available, but it is possible to use Tcl variables or bracket expansion of commands within a configuration string, when this is passed via the command `cv config . . .` ; this is particularly useful when combined with parameter introspection, e.g. `cv config "colvarsTrajFrequency [DCDFreq]"`;
- if a keyword requiring a boolean value (`yes|on|true` or `no|off|false`) is provided without an explicit value, it defaults to '`yes|on|true`'; for example, '`outputAppliedForce`' may be used as shorthand for '`outputAppliedForce on`';
- the hash character # indicates a comment: all text in the same line following this character will be ignored.

The following keywords are available in the global context of the colvars configuration, i.e. they are not nested inside other keywords:

- `colvarsTrajFrequency` ⟨Colvar value trajectory frequency⟩
  Context: global
  Acceptable values: positive integer
  Default value: `100`
  Description: The values of each colvar (and of other related quantities, if requested) are written to the file *outputName*`.colvars.traj` every these many steps throughout the simulation. If the value is `0`, such trajectory file is not written. For optimization the output is buffered, and synchronized with the disk only when the restart file is being written.

- `colvarsTrajAppend` ⟨Append to trajectory file?⟩
  Context: global
  Acceptable values: boolean
  Default value: `off`
  Description: If this flag is enabled, and a file with the same name as the trajectory file is already present, new data is appended to that file. Otherwise, a new file is created with the same name that