INDUS for PLUMED
User Manual

Sean M. Marks
(semarks@seas.upenn.edu)
Amish J. Patel*
(amish.patel@seas.upenn.edu)

Patel Group
University of Pennsylvania

October 14, 2018

Contents

1 Introduction **2**
1.1 Units 2

2 Getting the Code **2**

3 Installation **2**
3.1 Patching INDUS into PLUMED 2
3.2 Patching PLUMED+INDUS into an MD Code 3
3.3 Installing the Code as a Standalone Program 3

4 Using the Code **3**
4.1 As Part of PLUMED 4
4.2 As a Standalone Program (XTC files only) 4
4.3 Limitations of the Code and Warnings to Users 4

5 Input Options **4**
5.1 Input File Structure 5
5.2 Target Atom Selection 5
5.2.1 By Index 6
5.3 Probe Volume Geometries 6
5.3.1 Sphere 6
5.3.2 Box 6
5.3.3 Cylinder 6
5.4 Biasing $\tilde{N}_v$ Using the INDUS Code's Potential 7
5.5 Standalone Mode Options 7
5.6 Examples 8
5.6.1 As Part of PLUMED 8
5.6.2 Standalone Mode 9

6 Choice of Biasing Parameters **9**
6.1 Example: Bulk Water 10

1 Introduction

The INDUS code described here computes the number of particles, N_v, and the coarse-grained number of particles, $\tilde{N}_v$, in probe volumes, v, of several different geometries. It is primarily intended to be used as an extension to PLUMED, a popular plugin for molecular dynamics (MD) engines such as GROMACS, LAMMPS, AMBER, and NAMD. As part of PLUMED, the INDUS code can be used to post-process trajectories stored in a variety of formats (see `driver`). If PLUMED+INDUS is subsequently patched into an MD engine, it can be used to perform biased simulations with $\tilde{N}_v$.

Note that INDUS can also be compiled as a standalone code. However, this is intended primarily for development purposes.

For more on the INDUS method in theory and practice, here is a (non-exhaustive) list of topics and relevant references:

- INDUS for beginners [1]
- INDUS in probe volumes of all shapes and sizes [2]
- INDUS for biological systems, especially proteins [3, 4]
- INDUS and the sparse sampling method, an alternative to traditional umbrella sampling for efficiently computing $F_v(\tilde{N})$ [5, 6]

1.1 Units

The INDUS code uses the same basic set of units as PLUMED, which also happens to be the same as GROMACS.

Quantity	**Units**
Length	nm $= 10^{-9}$ m $= 10$ Å
Time	ps $= 10^{-12}$ s
Mass	amu
Energy	kJ/mol

2 Getting the Code

The code is available on GitHub (link).

3 Installation

3.1 Patching INDUS into PLUMED

It is recommended that you make use of the INDUS code as an extension to PLUMED. Make sure that you perform the following steps **before** configuring and compiling PLUMED!

A shell script to aid you in the patching process is available in the INDUS code repository at `plumed_patch/patch_plumed.s` When you run the script, simply pass to it the location of the root directory of the PLUMED repository (*i.e.* the one that contains the `Makefile` and so on):

```
./patch_plumed.sh <location-of-PLUMED-source-code>
```

Then simply configure, compile, and install PLUMED normally. For instructions on how to install PLUMED, see here.

Once installation is complete, a new action, `INDUS`, will be available to you the next time you run PLUMED. You can check that the procedure was successful by running the following command:

```
plumed --no-mpi manual --vim --action INDUS
```

This will print out a list of all the actions and command-line tools registered in PLUMED, followed by options which are specific to the `INDUS` action. (*Note*: The `-no-mpi` option flag is to prevent PLUMED from looking for an MPI library, since none is required for this check. The `-vim` flag causes output to be printed in a console-friendly format.)

3.2 Patching PLUMED+INDUS into an MD Code

Once you have installed PLUMED+INDUS, follow the instructions here to add it to your MD engine of choice. Note that a given version of PLUMED only supports certain versions of each MD code. To check which MD codes and versions are supported by each version of PLUMED, check out the online manual (link).

3.3 Installing the Code as a Standalone Program

The INDUS code can be compiled as a standalone program that can analyze XTC files. Before proceeding, you will need to install the `xdrfile` library written by the GROMACS development team (see here). Make note of where you install it: you will need this information later.

Once you have installed `xdrfile`, go to the main directory of the INDUS code repository and change the following lines in the `Makefile`:

```
# This variable indicates where you installed xdrfile (i.e. the directory containing the
   folders bin, include, and lib).
XDR_DIR=$(HOME)/programs/xdrfile/1.1.4

# If you aren't using a compiler that supports MPI, change the following lines
# - Mac OSX: CC=clang, CXX=clang++
# - Linux: CC=gcc, CXX=g++
CC=mpicc
CXX=mpic++

# Change this variable to 0 (false) if you aren't using MPI compilers
is_mpi_enabled=1
```

4 Using the Code

This section discusses how to use the INDUS code as part of PLUMED, and as a standalone code. Note that the INDUS code uses its own input file (`indus.input`), which is separate from the normal PLUMED input file (`plumed.dat`) and has its own syntax. Section 5 will discuss the details of the INDUS input file.

4.1 As Part of PLUMED

Below is a sample PLUMED input file. Observe that the INDUS input file, `indus.input`, is an input to the `INDUS` action.

```
# plumed.dat - PLUMED input file

# Create an instance of the INDUS action, which governs the interface between
# PLUMED and the INDUS code
indus: INDUS INPUTFILE=indus.input

# Put a harmonic bias on Ntilde_v of the form U = KAPPA/2*(Ntilde_v - AT),
# where KAPPA=1.0 kJ/mol and AT=0.0
restraint: RESTRAINT ARG=indus.ntilde KAPPA=1.0 AT=0.0

# Print some important quantities
# - indus.n        = N_v
# - indus.ntilde   = Ntilde_v
# - restraint.bias = value of the biasing potential
# - STRIDE=500     = print every 500 steps (i.e. every 1 ps when dt=0.002ps)
PRINT ...
  LABEL=print
  ARG=indus.n,indus.ntilde,restraint.bias
  STRIDE=500
  FILE=plumed.out
... PRINT
```

4.2 As a Standalone Program (XTC files only)

When compiled as a standalone program, the INDUS code can be used to analyze XTC files. Invoke from the command line using the following syntax:

```
./indus <indus.input>
```

See Section 5.5 for input options particular to running in standalone mode.

4.3 Limitations of the Code and Warnings to Users

1. Only orthorhombic boxes are supported (i.e. rectangular box where the side lengths may differ from each other)
2. Only one probe volume may be specified per instance of INDUS.
3. You should be able to pass `indus.ntilde` and `indus.ubias` to any other PLUMED action with no problems. However, I have not tested this aside from passing `indus.ntilde` and `indus.ubias` to `RESTRAINT`.
4. I used PLUMED 2.4.0 and GROMACS 2016.3 when writing and testing the code. Any other versions of PLUMED that are 2.2 or higher should work just fine, but I have not tested these and cannot guarantee that they will work. The program should run fine with any simulation package that your favorite PLUMED version supports, but—again—I haven't tested it myself.

5 Input Options

This section discusses the syntax of the INDUS input file, which is typically named `indus.input`.

Note: In the following section, angle brackets (`<>`) are used to indicate places where a value should be placed. The angle brackets themselves are not part of the actual input syntax. Vertical lines (`|`) are used to enumerate options when only a limited number of values are permitted. For example, `axis = <x|y|z>` indicates that you must choose `axis` to be either `x`, `y`, or `z`.

5.1 Input File Structure

The INDUS input file is organized into key-item pairs, where the *item* can be one of the following:

1. a *value*: a string or number
2. a *vector*: a sequence of values enclosed by square brackets, [ ]. Sometimes I will also use the term *array*, which is simply a vector that has a fixed number of values.
3. a *block*: an arbitrary grouping of values, vectors, and possibly other blocks, all enclosed by curly braces, . A block functions like an object (in the sense of object-oriented programming) and all of the entries within its curly braces are said to be in its *scope*.

It is instructive to consider an example:

```
MyBlock = {
  # This is a comment
  some_value  = 2.718
  some_vector = [ 1.0  2.0  3.0 ]   # This is a comment at the end of a line

  # Presumably, MyBlock corresponds to an object in the code that owns
  # another object of type NestedBlock
  NestedBlock = {
    another_value = 3.14159
    AnotherBlock = { another_vector = [ we are values ] }
  }
}
```

Observe that each key is separated from its corresponding item by an equals sign (`=`). Comments are indicated by the hash character (`#`). Each token in the input file (e.g. keys, values, equals signs, brackets, braces, comment characters, etc.) must be separated from the others by whitespace (single/multiple spaces or a new line).

Aside from the aforementioned restrictions, the input syntax is quite flexible. Indentation is not necessary, and braces/brackets/equals signs need not be on the same line as any other token. For example, the following is an eyesore but perfectly legal:

```
MyBlock
= { some_vector = [
1.0  2.0
3.0 ] }
```

Each key-item pair exists within a scope. The highest scope in the input file is the *file scope*. This encompasses all the values, vectors, and blocks that are not enclosed in other blocks. For instance, `MyBlock` in the previous examples is a block at file scope. In contrast, `NestedBlock` and `some_vector` are within the scope of `MyBlock`. In this way of thinking, the input file *itself* is like an object at the top of the hierarchy.

5.2 Target Atom Selection

Target atoms are indicated by the key `Target` at file scope. This is a unique key (multiple target selections are not permitted).

5.2.1 By Index

Atoms are selected as ranges of indices of the form `<start>-<stop>:<stride>`, the same as in PLUMED. This selects every `<stride>`th atom from index `start` to `stop`. All indices start at 1. if `:<stride>` is omitted, the stride is assumed to be 1.

Here are some examples.

```
# Select every 4th atom from index 1 to index 16500
Target = [ atom_index 1-16500:4 ]

# Select every atom from index 500 to index 1000
Target = [ atom_index 500-1000 ]

# Ranges can be combined using a comma (no spaces)
Target = [ atom_index  1-16500:4,32000-64000:3 ]
```

5.3 Probe Volume Geometries

Probe volumes are defined at global scope using the keyword `ProbeVolume`. Only one probe volume geometry may be specified per instance of INDUS. To define multiple probe volumes using PLUMED, one must create multiple instances of the `INDUS` action.

5.3.1 Sphere

```
ProbeVolume = {
  type   = sphere
  radius = <r>
  center = [ <x> <y> <z> ]
  # Coarse-graining
  sigma   = <σ>
  alpha_c = <α_c>
}
```

5.3.2 Box

```
ProbeVolume = {
  type   = box
  # Boundaries of box along each axis
  x_range = [ <x_low> <x_high> ]
  y_range = [ <y_low> <y_high> ]
  z_range = [ <z_low> <z_high> ]
  # Coarse-graining
  sigma   = <σ>
  alpha_c = <α_c>
}
```

5.3.3 Cylinder

```
ProbeVolume = {
  type   = cylinder
  # Cylinder will have its axis parallel to this axis
  # - Default: parallel to z-axis
  axis    = <x|y|z>
  # Location of the center of the base
  base    = [ <x> <y> <z> ]
  radius  = <r>
  height  = <h>
  # Coarse-graining
  sigma   = <σ>
  alpha_c = <α_c>
}
```

5.4 Biasing $\tilde{N}_v$ Using the INDUS Code's Potential

As an alternative to PLUMED's implementation, the INDUS code makes available its own biasing potential of the following general form:

$$\mathcal{U}_{\text{bias}}(x) = \mathcal{U}_{\text{harmonic}}(x) + \mathcal{U}_{\text{linear}}(x) + \mathcal{U}_{\text{left harmonic}}(x) + \mathcal{U}_{\text{right harmonic}}(x) \tag{1}$$

$$= \frac{1}{2}\kappa(x - x^*)^2 + (\phi x + c) + k_{\text{left}}\Theta(x_{\text{left}} - x) + k_{\text{right}}\Theta(x - x_{\text{right}}) \tag{2}$$

where $\Theta(x)$ is the unit step function, which is zero for $x < 0$ and one for $x \geq 1$. The corresponding syntax for the INDUS input file is:

```
Bias = {
  # Tells INDUS that x=Ntilde_v
  order_parameter = ntilde

  # Harmonic
  x_star = <x*>
  kappa  = <κ>

  # Linear
  phi      = <φ>
  constant = <c>

  # Left one-sided harmonic
  x_left = <x_left>
  k_left = <k_left>

  # Right one-sided harmonic
  x_right = <x_right>
  k_right = <k_right>
}
```

Any parameters not specified in the input file are set to zero automatically.

Then, in the PLUMED input file, $\mathcal{U}_{\text{bias}} = \mathcal{U}_{\text{bias}}(\tilde{N}_v)$ is passed to `RESTRAINT` as its `ARG`:

```
restraint: RESTRAINT ARG=indus.ubias SLOPE=1.0 AT=0.0 KAPPA=0.0
```

5.5 Standalone Mode Options

The following options are supported at file scope.

```
# The XTC file to analyze
XtcFile = <path_to_file>

# Range of times for which calculations are performed (in ps)
t0 = <t_0>
tf = <t_f>
```

All of these options are ignored when running the INDUS code as part of PLUMED.

5.6 Examples

5.6.1 As Part of PLUMED

Suppose we want to compute N_v and $\tilde{N}_v$ in a cylindrical probe volume with the center of its base at $\mathbf{r}_0 = (2.0, 2.0, 2.0)$ nm with radius $r = 1.0$ nm and height $h = 0.5$ nm. The cylinder is parallel to the y-axis and extends from $y_{\text{low}} = 2.0$ nm to $y_{\text{high}} = y_{\text{low}} + h = 2.5$ nm. The target atoms have indices 1, 4, 7, ... , 16497 (i.e. every 3 atoms from 1 to 16,500, indexed from 1). The coarse-graining parameters are $\sigma = 0.01$ nm and $\alpha_c = 2\sigma = 0.02$ nm. Use the INDUS code's `Bias` feature to apply a bias of the form $\mathcal{U}_{\text{bias}} = \phi\tilde{N}$ with $\phi = 0.6$ kJ/mol.

The INDUS input file will be:

```
# indus.input

# Atoms targeted by INDUS
Target = [ atom_index 1-16500:3 ]

# Probe volume
ProbeVolume = {
  type   = cylinder
  axis   = y
  # Dimensions and location
  center = [ 2.0 2.0 2.0 ]
  radius = 1.0
  height = 0.5
  # Coarse-graining
  sigma = 0.01
  alpha_c = 0.02
}

# Add a linear bias
Bias = {
  order_parameter = ntilde
  phi = 0.6
}
```

The corresponding PLUMED input file is:

```
# plumed.dat

# Create an instance of the INDUS action, which computes N_v and Ntilde_v
indus: INDUS INPUTFILE=<indus.input>

# Pass the bias computed by the INDUS code, U_bias, to PLUMED's RESTRAINT
# - This is required to invoke the part of PLUMED that passes the biasing forces
#   to the MD engine
# - Using this approach, ARG = U_bias(Ntilde_v) = restraint.bias
restraint: RESTRAINT ARG=indus.ubias SLOPE=1.0 AT=0.0 KAPPA=0.0

```

```
# Print N_v, Ntilde_v, and U_bias evert 500 steps
PRINT ...
  LABEL=print
  ARG=indus.n,indus.ntilde,restraint.bias
  STRIDE=500
  FILE=plumed.out
... PRINT
```

5.6.2 Standalone Mode

Compute N_v and $\tilde{N}_v$ in a spherical probe volume center at $\mathbf{r}_0 = (2.5, 2.5, 2.5)$ nm with radius $r = 0.6$ nm. The target atoms have indices 1, 5, 9, ... , 16497 (i.e. every 4 atoms from 1 to 16,500, indexed from 1). The coarse-graining parameters are $\sigma = 0.01$ nm and $\alpha_c = 2\sigma = 0.02$ nm. Print the time series of N_v and $\tilde{N}_v$ for $t \geq 100$ ps and $t \leq 500$ ps. Also print the forces that would be obtained for these frames under a harmonic potential on $\tilde{N}_v$ with $\kappa = 0.89$ kJ/mol and $N^* = -5$.

```
# indus.input

# Atoms targeted by INDUS
Target = [ atom_index 1-16500:4 ]

# Probe volume
ProbeVolume = {
  type   = sphere
  center = [ 2.5 2.5 2.5 ]
  radius = 0.6
  # Coarse-graining
  sigma = 0.01
  alpha_c = 0.02
}

# Add a harmonic bias
Bias = {
  order_parameter = ntilde
  x_star = -5
  kappa  = 0.89
}

# XTC file to analyze
XtcFile = my_trajectory.xtc

# Production phase (ps)
t0 = 100
tf = 500

# Print time series of N_v and Ntilde_v, and biasing forces
PrintOutput = yes
PrintForces = yes
```

6 Choice of Biasing Parameters

When performing umbrella sampling with $\tilde{N}_v$, we use the statistics of N_v in the liquid basin in an unbiased simulation to guide our choice of biasing parameters. This is important for obtaining accurate results.

Let the average and variance of N_v in the liquid basin be $\langle N_v \rangle_0$ and $\langle (\delta N_v)^2 \rangle_0$, respectively. Assuming that the liquid basin is approximately Gaussian, its curvature is

$$\kappa_0 = \frac{k_\mathrm{B}T}{\langle(\delta N_v)^2\rangle_0} \tag{3}$$

Suppose that we want to bias $\tilde{N}_v$ using a harmonic potential,

$$\mathcal{U}_\mathrm{bias}(\tilde{N}_v) = \frac{1}{2}\kappa(\tilde{N}_v - N^*)^2, \tag{4}$$

where κ and N^* are constants (the biasing parameters). We choose κ as a multiple of κ_0:

$$\kappa = \alpha\kappa_0 \tag{5}$$

where $\alpha = 2$ or 3.

Accurate WHAM results require overlap between adjacent windows. For a desired overlap of $\Delta F_\mathrm{overlap}$, assuming that the biased ensembles for each N^*-value are Gaussian are leads an estimated spacing between N^*-values of

$$\Delta N^* = 2\sqrt{\frac{(\beta\Delta F_\mathrm{overlap})\langle(\delta N_v)^2\rangle_0}{1+\alpha}} \tag{6}$$

where $\beta = (k_\mathrm{B}T)^{-1}$. For an initial guess, we typically choose $\alpha = 2$ and $\Delta F_\mathrm{overlap} = 4\ k_\mathrm{B}T$. This leads to:

$$\kappa = 2\kappa_0 \tag{7}$$

$$\Delta N^* = \frac{4}{\sqrt{3}}\sqrt{\langle(\delta N_v)^2\rangle_0} \tag{8}$$

Note that sampling $N_v = 0$ may require you to use windows with $N^* < 0$.

6.1 Example: Bulk Water

Figure 1 illustrates $F_v(\tilde{N})$ in a small sphere in bulk water. The free energy distribution was produced using UWHAM [7]. The aforementioned heuristics were used to choose the biasing parameters.

References

[1] Amish J Patel, Patrick Varilly, and David Chandler. Fluctuations of water near extended hydrophobic and hydrophilic surfaces. *The Journal of Physical Chemistry B*, 114(4):1632–1637, 2010.

[2] Amish J Patel, Patrick Varilly, David Chandler, and Shekhar Garde. Quantifying density fluctuations in volumes of all shapes and sizes using indirect umbrella sampling. *Journal of statistical physics*, 145(2):265–275, 2011.

[3] Amish J Patel, Patrick Varilly, Sumanth N Jamadagni, Michael F Hagan, David Chandler, and Shekhar Garde. Sitting at the edge: How biomolecules use hydrophobicity to tune their interactions and function. *The Journal of Physical Chemistry B*, 116(8):2498–2503, 2012.

[4] Amish J Patel and Shekhar Garde. Efficient method to characterize the context-dependent hydrophobicity of proteins. *The Journal of Physical Chemistry B*, 118(6):1564–1573, 2014.

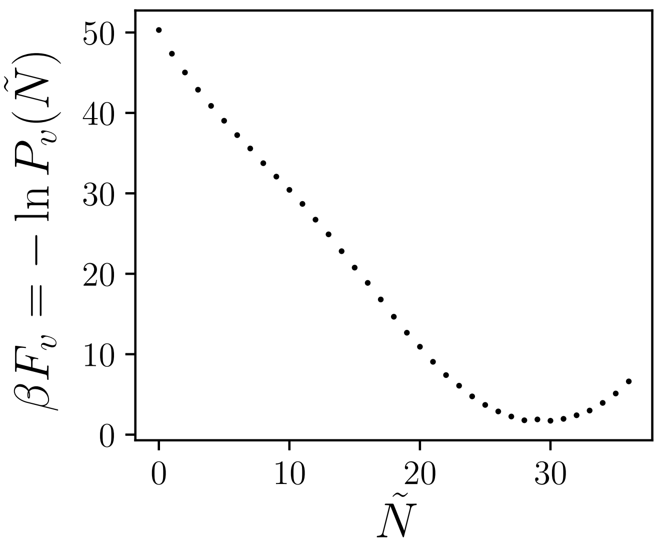

Figure 1: The free energy landscape for the coarse-grained number of waters, $\tilde{N}$, in a small probe volume in bulk water at $T = 300$ K and $P = 1$ bar. The probe volume, v, is a sphere of radius $r = 0.6$ nm in a box of about $4,000$ SPC/E water molecules.

[5] Erte Xi, Richard C Remsing, and Amish J Patel. Sparse sampling of water density fluctuations in interfacial environments. *Journal of chemical theory and computation*, 12(2):706–713, 2016.

[6] Erte Xi, Sean M Marks, Suruchi Fialoke, and Amish J Patel. Sparse sampling of water density fluctuations near liquid-vapor coexistence. *Molecular Simulation*, pages 1–12, 2018.

[7] Zhiqiang Tan, Emilio Gallicchio, Mauro Lapelosa, and Ronald M Levy. Theory of binless multi-state free energy estimation with applications to protein-ligand binding. *The Journal of chemical physics*, 136(14):04B608, 2012.