Steered Molecular Dynamics/Jarzynski Equality Calculator

Generated by Doxygen 1.8.13

Contents

1 Class Index **1**

1.1 Class List 1

2 Class Documentation **3**

2.1 JarzynskiFreeEnergy Class Reference 3

2.1.1 Detailed Description 3

2.1.2 Constructor & Destructor Documentation 3

2.1.2.1 JarzynskiFreeEnergy() [1/2] 3

2.1.2.2 JarzynskiFreeEnergy() [2/2] 4

2.1.2.3 ∼JarzynskiFreeEnergy() 4

2.1.3 Member Function Documentation 4

2.1.3.1 JEprocessVector() 4

2.1.3.2 JERaw() 4

2.1.3.3 JETaylor() 5

2.1.3.4 read() 5

2.1.3.5 resetIndex() 5

2.1.3.6 vecProcess() 6

Index **7**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

JarzynskiFreeEnergy 3

Chapter 2

Class Documentation

2.1 JarzynskiFreeEnergy Class Reference

`#include <JE_compute.hpp>`

Public Member Functions

- JarzynskiFreeEnergy ()
- JarzynskiFreeEnergy (int, double)
- ~JarzynskiFreeEnergy ()
- void vecProcess ()
- void resetIndex ()
- void read (std::string input)
- double JERaw (std::vector< double > *JEVector)
- double JETaylor (std::vector< double > *JEVector)
- double JEprocessVector (int, double(JarzynskiFreeEnergy::*f)(std::vector< double > *VectorInput), std::vector< double > *JEVector)
- double **alpha** (double, double, double)

2.1.1 Detailed Description

The main class

2.1.2 Constructor & Destructor Documentation

2.1.2.1 JarzynskiFreeEnergy() [1/2]

`JarzynskiFreeEnergy::JarzynskiFreeEnergy ( )`

Default Constructor

2.1.2.2 JarzynskiFreeEnergy() [2/2]

```
JarzynskiFreeEnergy::JarzynskiFreeEnergy (
            int ,
            double  )
```

Default Constructor with parameters

2.1.2.3 ~JarzynskiFreeEnergy()

```
JarzynskiFreeEnergy::~JarzynskiFreeEnergy ( )
```

Destructor

2.1.3 Member Function Documentation

2.1.3.1 JEprocessVector()

```
double JarzynskiFreeEnergy::JEprocessVector (
            int position,
            double(JarzynskiFreeEnergy::*)(std::vector< double > *VectorInput) f,
            std::vector< double > * JEVector )
```

< Compute free energies with algorithm plugged in from JarzynskiFreeEnergy::*f, with the work values from std::vector<double> *VectorInput

< Integer iterator

< Instance of the class to get the function method

< Empty vector before doing mean calculations - as we are working with a pointer to a vector, and not creating a copy each time, this is required

< If the work values are within an angstrom range, add to vector

< store work values

< Assess the free energy with the function as pointed to by sample.f*

2.1.3.2 JERaw()

```
double JarzynskiFreeEnergy::JERaw (
            std::vector< double > * JEVector )
```

Free energy calculation functions from work distribution The Raw Jarzynski Equality computer

< Vector to copy the value into, as to not change the values of the elements inside the vector pointer

< Free energy (Gibbs)

< Boltzmann Factor, at 303K

< iterator for vector

< compute the raw JE

2.1.3.3 JETaylor()

```
double JarzynskiFreeEnergy::JETaylor (
            std::vector< double > * JEVector )
```

The Taylor Series Jarzynski Equality computer

< Vector to copy the work value into, as to not change the values of the elements inside the vector pointer

< Vector to store the squared work values.

< Free Energy

< Boltzmann Factor

< double iterator

Store the raw work values from the JEVector

Store the squared work values

< Average work

< Average squared work

< Taylor series interpreter

2.1.3.4 read()

```
void JarzynskiFreeEnergy::read (
            std::string input )
```

IO - Input the column data values as computed from LAMMPS

< Line index

< z coordinates of the Nanoparticle/molecule

< z coordinate of the bilayer COM

< Force values

< Work values

< Add index for next line

2.1.3.5 resetIndex()

```
void JarzynskiFreeEnergy::resetIndex ( )
```

Clear all bins before reading in anything

< Clear file line index

< Clear z coordinates

< Clear COM of the bilayer

< Clear Force values

< Work values

2.1.3.6 vecProcess()

```
void JarzynskiFreeEnergy::vecProcess ( )
```

This function uses the boost::tuple and create a vector of tuples; it stores the free energy values with the first index storing the coordinate value where we base the search. The second and third elements store the minimum and maximum range from which the work distribution was analyzed. Finally, the last element stores the free energy values.

< Define minimum z coordinate

< Define maximum z coordinate

We want to accumulate the values via the bins: */

< Make bins for storing the work values between i - 0.5 and i + 0.5 - i.e. a 1 angstrom interval, using the raw JE interpreter

< Make bins for storing the work values between i - 0.5 and i + 0.5 - i.e. a 1 angstrom interval, using the taylor series JE interpreter

< Store free energy values from JERaw algorithm

< Push back values in each

Print out to 5 decimal places

Print out to 5 decimal places

The documentation for this class was generated from the following files:

- include/JE_compute.hpp
- src/JE_alg.cxx

Index

~JarzynskiFreeEnergy
    JarzynskiFreeEnergy, 4

JERaw
    JarzynskiFreeEnergy, 4
JETaylor
    JarzynskiFreeEnergy, 4
JEprocessVector
    JarzynskiFreeEnergy, 4
JarzynskiFreeEnergy, 3
    ~JarzynskiFreeEnergy, 4
    JERaw, 4
    JETaylor, 4
    JEprocessVector, 4
    JarzynskiFreeEnergy, 3
    read, 5
    resetIndex, 5
    vecProcess, 5

read
    JarzynskiFreeEnergy, 5
resetIndex
    JarzynskiFreeEnergy, 5

vecProcess
    JarzynskiFreeEnergy, 5