Tutorials

From PIMC++

In this tutorial, we will first calculate the energy of a single particle in a box with periodic boundary conditions. Although not of physical interest, this calculation will introduce you to the PIMC++ software suite, including its input format and output analysis and processing tools.

In this first exercise, you will be exposed to a great deal of terminology and syntax related to path integrals and used in PIMC++. We will alert you to important points; occasionally, links embedded in the text will take you to more detailed information about technical or expert-level topics. However, many aspects of the simulation input and software design will be explored in more detail later, so do not worry about understanding every aspect right away.

Specifically, we will calculate the energy of a single particle in a box of size $10\ \mathring{A}$ at a temperature of $1.666K$. For consistency with the Helium simulations we'll do, we're working in a system of units in which energy is in Kelvin, length is in $\mathring{A}$, and $\hbar = 1$. In these units, the free particle mass will be $0.5\ K^{-1}\mathring{A}^{-2}$, which corresponds to $4.034 * 10^{-26}\ kg$.
In this simple case, it will be possible to compare the simulation with the analytical result. If you do this, you should get an energy of $2.51\ K$.

We will first describe how the input file is organized and how important aspects of the simulation are controlled. The first input file looks like this:
You can view a fully annotated version of this file here in order to see more detail about input syntax and options.

```
double tau=.05;
Section (Parallel)
{
  int ProcsPerClone = 1;
}
Section (System)
{
  int NumTimeSlices=12;
  Array<double,1> Box(3)=[10.0,10.0,10.0];
  Array<bool, 1 > IsPeriodic(3)=[true,true,true];
  Section (Particles)
  {
    Section (Species)
    {
      string Name="free";
      string Type="free";
      double lambda=1.0;
      string Statistics="BOLTZMANNON";
      int NumParticles=1;
      int NumDim=3;
      string InitPaths="BCC";
    }
  }
 }
Section (Action)
{
  int NumImages=0;
  int MaxLevels = 2;
  Array<string,1> PairActionFiles(1) = ["zero.PairAction"];

}

Section (Observables)
{
  string OutFileBase = "SingleParticle";
```

```
  Section (Observable)
    {
      string Type = "Energy";
      string Name = "Energy";
       string Description="Total Energy";
      int Frequency=1;
    }
}


Section (Moves){
  Section (Move) {
    string Type="BisectionBlock";
    string Name="BisectionBlock";
    string PermuteType="NONE";
    string Species="free";
     int NumLevels=2;
    int StepsPerBlock=2;
  }
  Section (Move)
    {
      string Type="ShiftMove";
      string Name="Shift";
    }
 }

Section (Algorithm)
{
  Section (Loop){
    int Steps=1000;
    Section (Loop){
      int Steps=50;
      Section (Move) {string Name="BisectionBlock";}
      Section (Observe) {string Name = "Energy"; }
      Section (Move) {string Name = "Shift"; }
    }
    Section (WriteData){}
  }
}
```

Helpful editing tip for input files: The input files conform to C++ syntax. If you turn on C++ syntax highlighting in your editor (Meta-x c++-mode followed by Meta-x font-lock-mode in emacs), they will be colored and the braces will be matched in a friendly way.

Let's briefly examine the input file. Notice the content is grouped by *Section*, each of which contains variables that define the input to the simulation.
In summary, the sections are :

- Parallel: Specifies how many processors over which each path is distributed.
- System: Information about the physical system to be simulated. Contains number of time slices, tau, box-size, and the particle Section.
    - Particle Section: Contains a section for each Species.
        - Species Section: Contains lambda ($\lambda = \frac{\hbar^2}{2m}$), statistics, number of particles, etc. A Species can be an element, ion, molecule, or any object with uniform physical properties.
- Moves: Defines the moves of your Monte Carlo
- Observables: Defines what properties of the system you are measuring
- Action: The action (dimensionless energy) is used to evaluate whether to accept or reject a Monte Carlo move. By specifying which actions to use and their properties, the description of the system's physical interactions is fixed (i.e. free particles, Coulomb, Lennard-Jones, etc.).
- Algorithm: Specifies the simulation sequence of events (when each Move and Observable is called)

A note about units: PIMC++ has no intrinsic units and any system of units can be used by specifying consistent values in the input file. $\beta = M\tau = (k_B T)^{-1}$, where M is the number of time slices. β has units of inverse energy and establishes the temperature scale. For the tutorial, input files will use units of Kelvin and Angstrom with $\hbar = 1$.

Be sure to understand which lines of the input file establish that you are simulating a particle of mass 0.5, temperature $1.66K$ and box $10 \times 10 \times 10$ $\AA$.

The mass is defined through lambda ($\lambda = \frac{\hbar^2}{2m}$) (Section: System/Species/Particle). The temperature is defined implicitly by $\frac{1}{NumTimeSlices * tau}$ with NumTimeSlices (Section:System) and tau (Section: System) specified in the input.

Now, let's run PIMC++ with this input file. Copy it to a file in your working directory and name it SingleParticle.in. On Tungsten, we have provided a script to submit batch jobs. Enter

```
runPIMC SingleParticle.in
```

Please see these instructions for more details about running PIMC++ on Tungsten or on a local desktop or laptop.

On Tungsten, you can use the command

```
bjobs
```

to check the status of your job in the queue and during runtime.

Once it's through the queue, this run should take about 30 seconds and produce an output file called SingleParticle.0.h5. In the "Observables" section of the input file, there is variable called "OutFileBase". This specifies the prefix of the output file(s). Because "OutFileBase" (Section: Observables) was set to "SingleParticle" in this case, the output is named "SingleParticle.0.h5". For a serial run, the output filename will be the prefix plus "0.h5". For a parallel run, each processor generates an output file by appending its processor number and ".h5" to the prefix. The output files are written in a portable, hierarchical file format known as HDF5 (http://hdf.ncsa.uiuc.edu/HDF5/) .

Let us take a look at the output we've generated. In order to do this, we will run the analysis script Report.py on the output. On Tungsten or the NCSA laptops, the script will be in your path and you can just enter

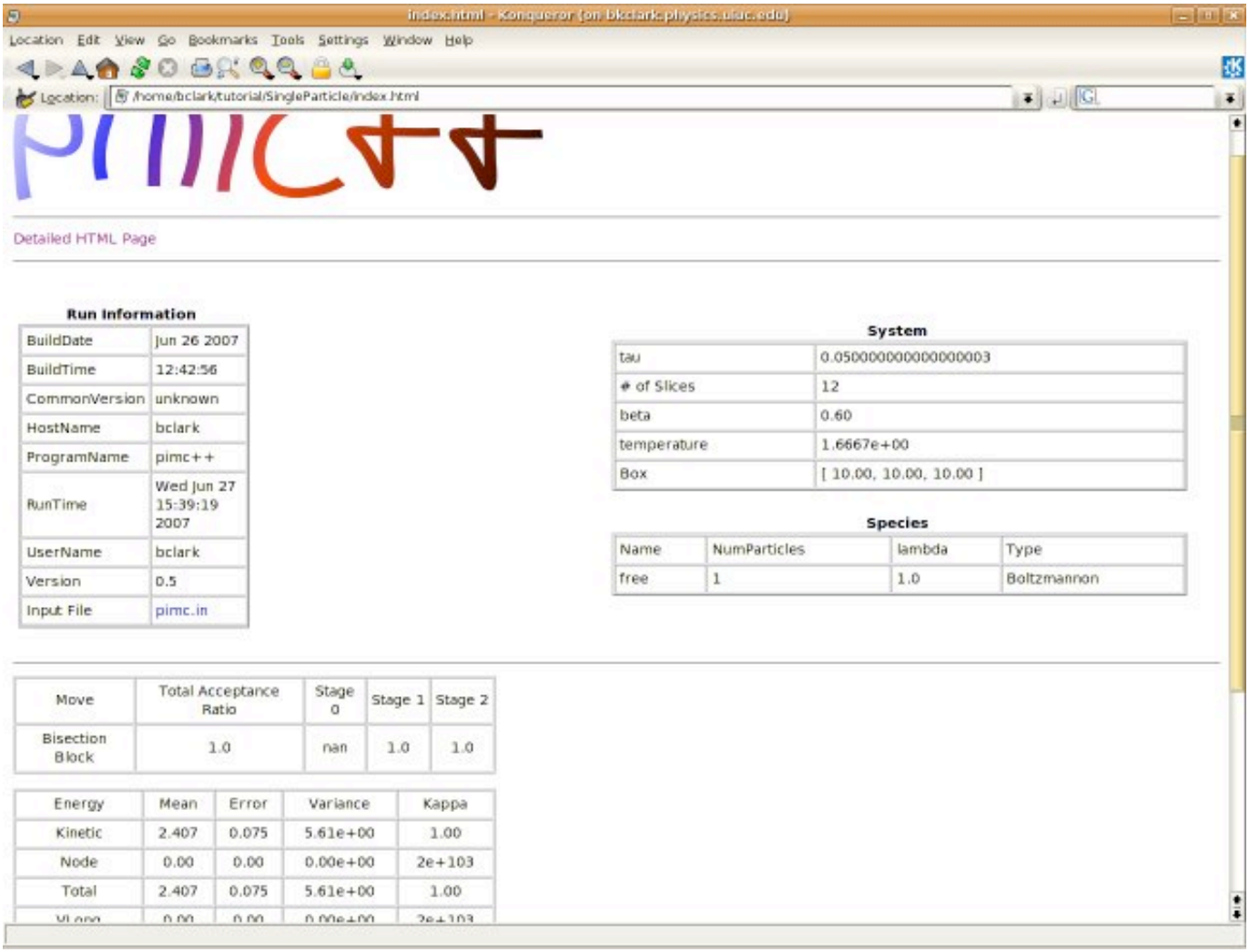
```
Report.py SingleParticle
```

If you have a different setup where the top-level PIMC++ installation directory is PIMC++, the analysis script can be found at PIMC++/src/analysis/Report.py.

This script will produce an html file that will give us some information about our run. Examine SingleParticle/index.html (http://www.pathintegrals.info/tutorial/SingleParticle/index.html) with a web browser. On Tungsten, we recommend using

```
konqueror SingleParticle/index.html &
```

Your screen should be similar to the following:



Let us take note of the following information in the html summary.

- Run information: (Upper left table) Specifies where and when it was run, who ran it, etc.
- System Data: (Upper right table) System Parameters including temperature, species information, etc.
- Move information: Table specifying acceptance ratio of moves and center of mass drift (add this to your input). You will notice that in this simulation, the acceptance ratio is 1.00. Why is this?. There is additional information on the bisection move here. You can ignore this for now. Note: There is another "move" in the simulation, "ShiftMove", that isn't in the table. It is not a real move, but simply rearranges the storage of the path in memory. Thus it has to acceptance ratio, but it is nonethless required to ensure ergodicity. Go here to understand the technical reason behind this.
- Observable Information: We have measured the **Energy** observable in this simulation. Notice the energy table with the different components of the energy and their respective variance, error and autocorrelation time. We see that the energy is close to $2.5\ K$ as anticipated above. Of course, as we lower the temperature, we would see quantum effects becoming more important as the energy approaches the ground state for a single particle in a box. (Note: there is a subtlety to getting the correct energy if your box is small compared to $\sqrt{4\lambda\tau}$ in PIMC++ caused by an approximation to the kinetic action. This can be skipped on the first pass through the tutorial, but experts can explore this NumImage approximation here) Although a summary of the energy is convenient, it is often useful to be able to see a trace plot of the energy (for example to choose an equilibration time). We can get such detailed information by clicking on the "Detailed HTML Page" at the top of index.html. Do that now. This page also allows us to see that the data was output 1000 times. By examining the Algorithm section of our input file, we see that the data gets written every time a WriteData is encountered in the algorithm (nb: this is NOT true for the PathDump observable which is special!). Looking at the number of iterations given for each of the Loop sections, we see that WriteData will be called 1000 times.

Once you feel familiar with this introductory simulation, we can move towards a more interesting system:

- Liquid helium
- PIMC++ applications