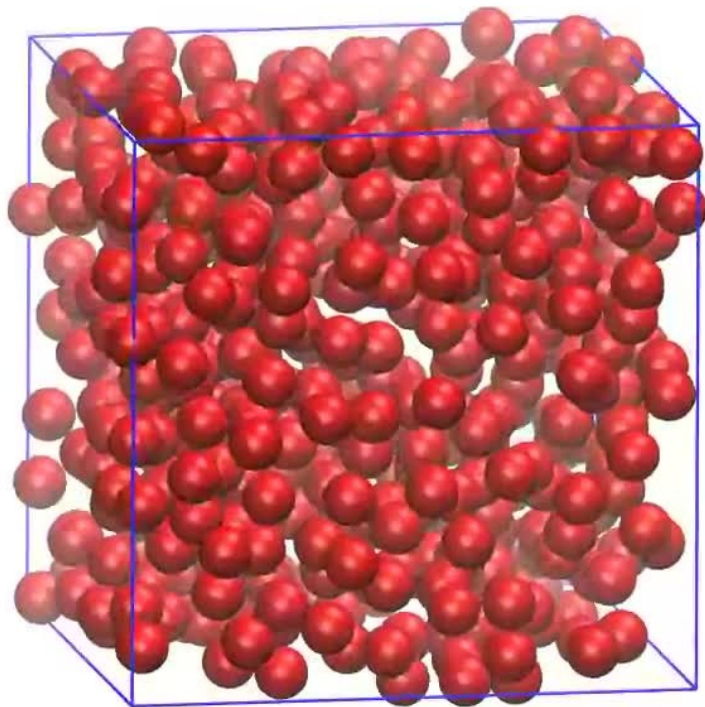


# Molecule Dynamics Project Coupling Python with C



Pablo Alcain  
Matias Factorovich  
Md. Enamul Hoque

# The Challenge:

Make an easy to use MD program and adaptable program that could be used in example for teaching purpose

The problem:  
It was written in C



```
/* read restart */
fp=fopen(restfile,"r");
if(fp) {
    int natoms;
    natoms=sys.natoms;

    for (i=0; i<natoms; ++i) {
        fscanf(fp,"%lf%lf%lf",sys.pos+i, sys.pos+natoms+i, sys.pos+2*natoms+i);
    }
    for (i=0; i<natoms; ++i) {
        fscanf(fp,"%lf%lf%lf",sys.vel+i, sys.vel+natoms+i, sys.vel+2*natoms+i);
    }
    fclose(fp);
    azzero(sys.frc, 3*sys.nthreads*sys.natoms);
} else {
    perror("cannot read restart file");
    return 3;
}

/* initialize forces and energies.*/

if((int)sys.tempin==0){
    ander=0;
} else {
    ander=1;
}
sys.nfi=0;
sys.clist=NULL;
sys.plist=NULL;
updcells(&sys);
sys.var_andersen=pow(kboltz*sys.tempin/mvsq2e/sys.mass,0.5);
force(&sys);
ekin(&sys);
```

# What are our MD C program capabilities?

- Screen input
- Coordinates and thermodynamic data output to a file
- Just NVE support
- Lennard-Jones potential

## What we wanted to add?

- Easy to use GUI support and intimidate graph
- Thermostat to Run NVT simulations
- Any potential → Look up table .
- Make the code clean enough for future enhanced

- The Solution: Take our base MD program written in C and coupled it to python.
  - Easy program language
  - Build features for making graph
  - Easy coupling with C through c\_types
  - Build features for making GUI



Let's get on with it!!!

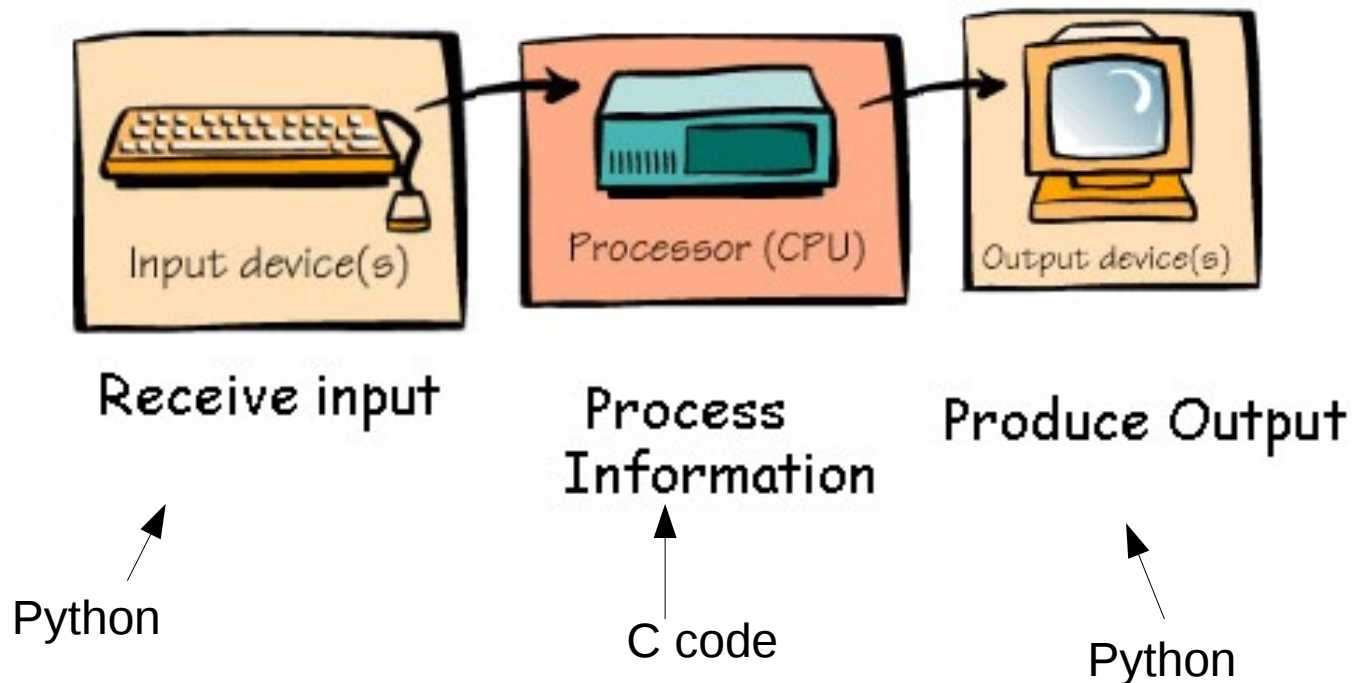


- The Solution: Take our base MD program written in C and coupled it to python.
  - Easy program language
  - Build features for making graph
  - Easy coupling with C through c\_types
  - Build features for making GUI

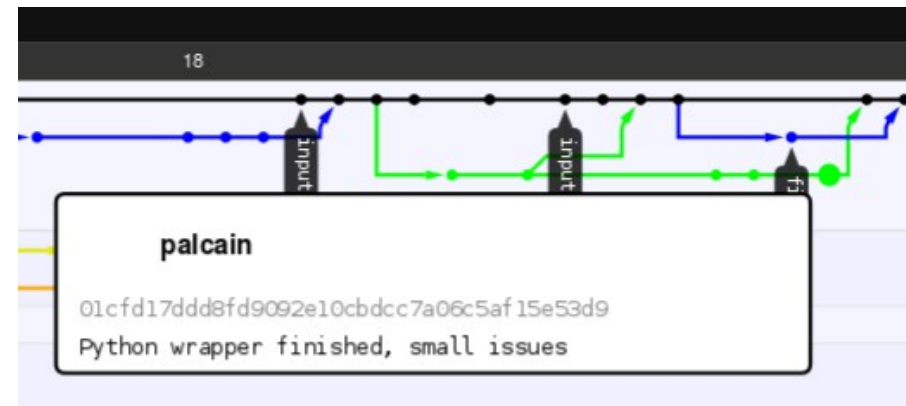
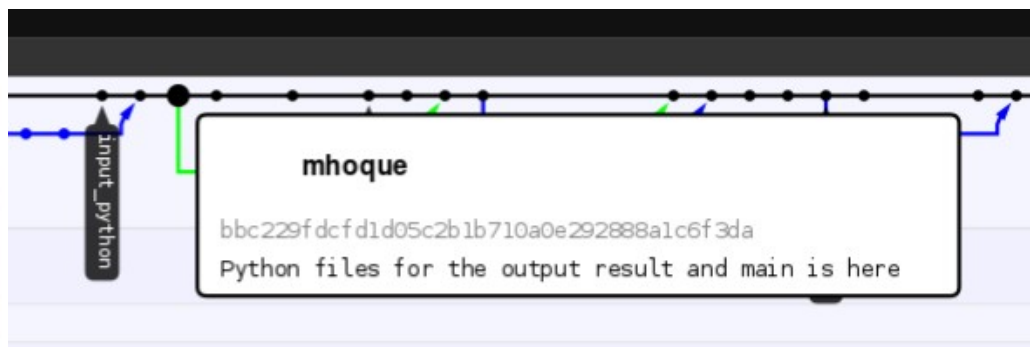
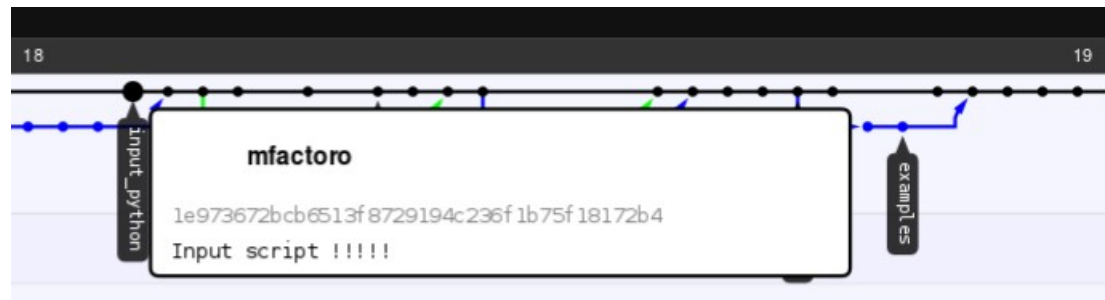
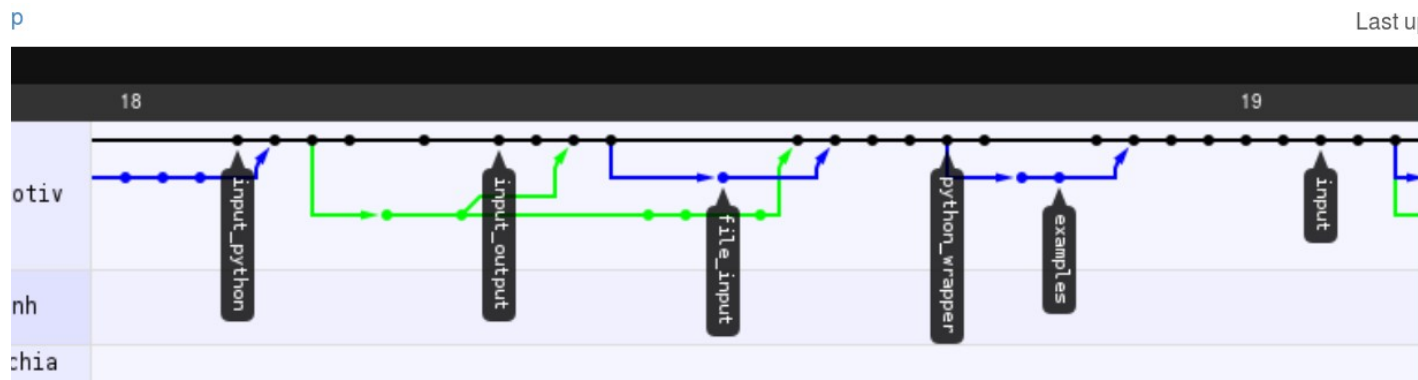


Let's get on with it!!!

- First Goal: Run the MD program as a library  
Called from python
  - Just the heavy calculation will be done in C
    - Cell build, force calculation, Verlet integrator
  - Management in python
    - Input/Output and looping through time-step



# Each one Starts their job



# What we achieved ???

```
for i in range(mdsys.nsteps):  
    ## This is the main loop, integer  
    if (i % mdsys.nprint == 0):  
        mdsys.output(i)  
        time.append(i)  
    md.velverlet(byref(mdsys))  
    md.ekin(byref(mdsys))  
    if (i % cellfreq == 0):  
        md.updcells(byref(mdsys))
```



# Now what ????

[x]: improve input behaviour

(x): sanitize input from python and clean classes

(x): make the proper GUI input (with a flag to mark we want to run GUI, like  
"./main.py -g")

[x]: "portability": figure out a way to ask for the number of threads in main.py

[x]: Makefiles and c-code

(x): remove the warnings and check that Makefiles work properly

(o): refactor mdsys class in both c and python to remove unneeded info (like nfi,  
ncellfreq, and such)

[x]: add a LUT potential

[x]: add different integrators methods

(x)write code

(x)make python use it

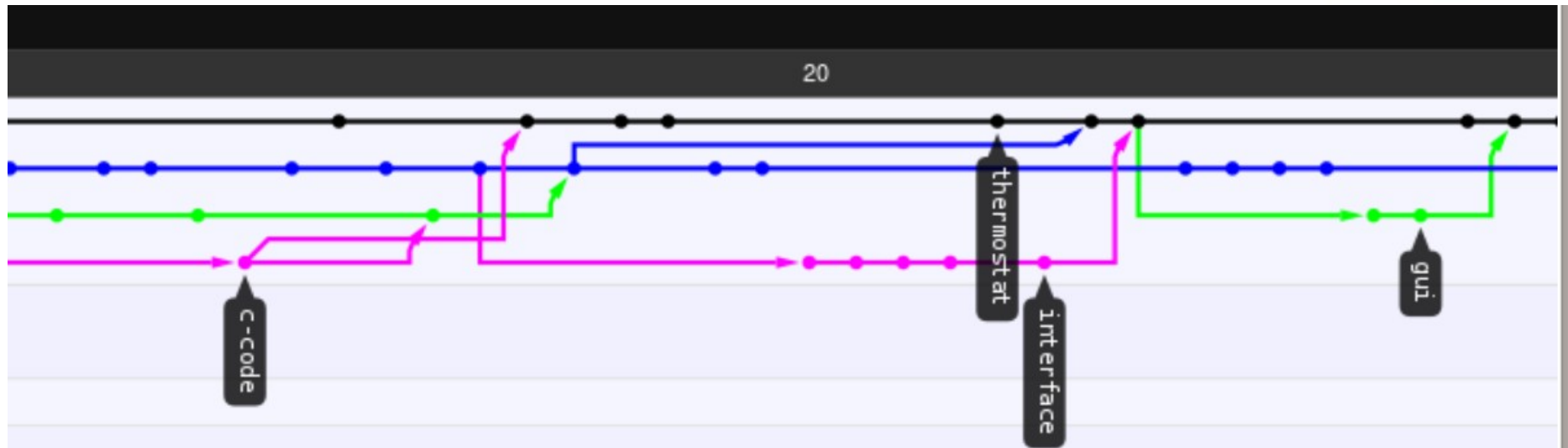
[ ]: clean GUI input file

[ ]: refactor classes like Potential and mdsys in python

[ ]: finish documentation

-----  
[ ]: add different atom types

To Do List !!!



Three main Branches  
GUI and output Graph  
Andersen Thermostat  
Lookup Table

# Results

- GUI

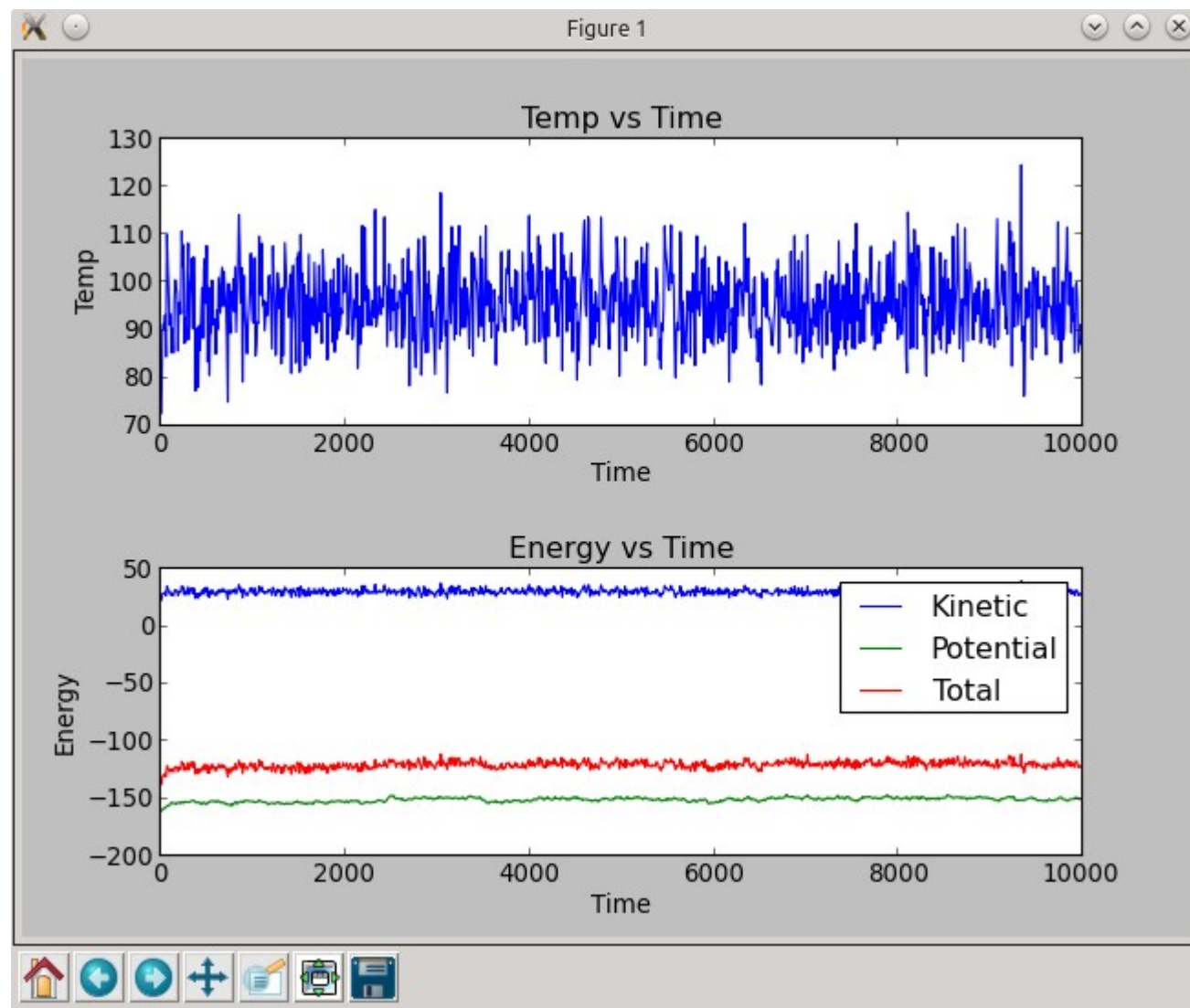
A simple gui interface for LJMD

Please give the input parameter to run in LJMD

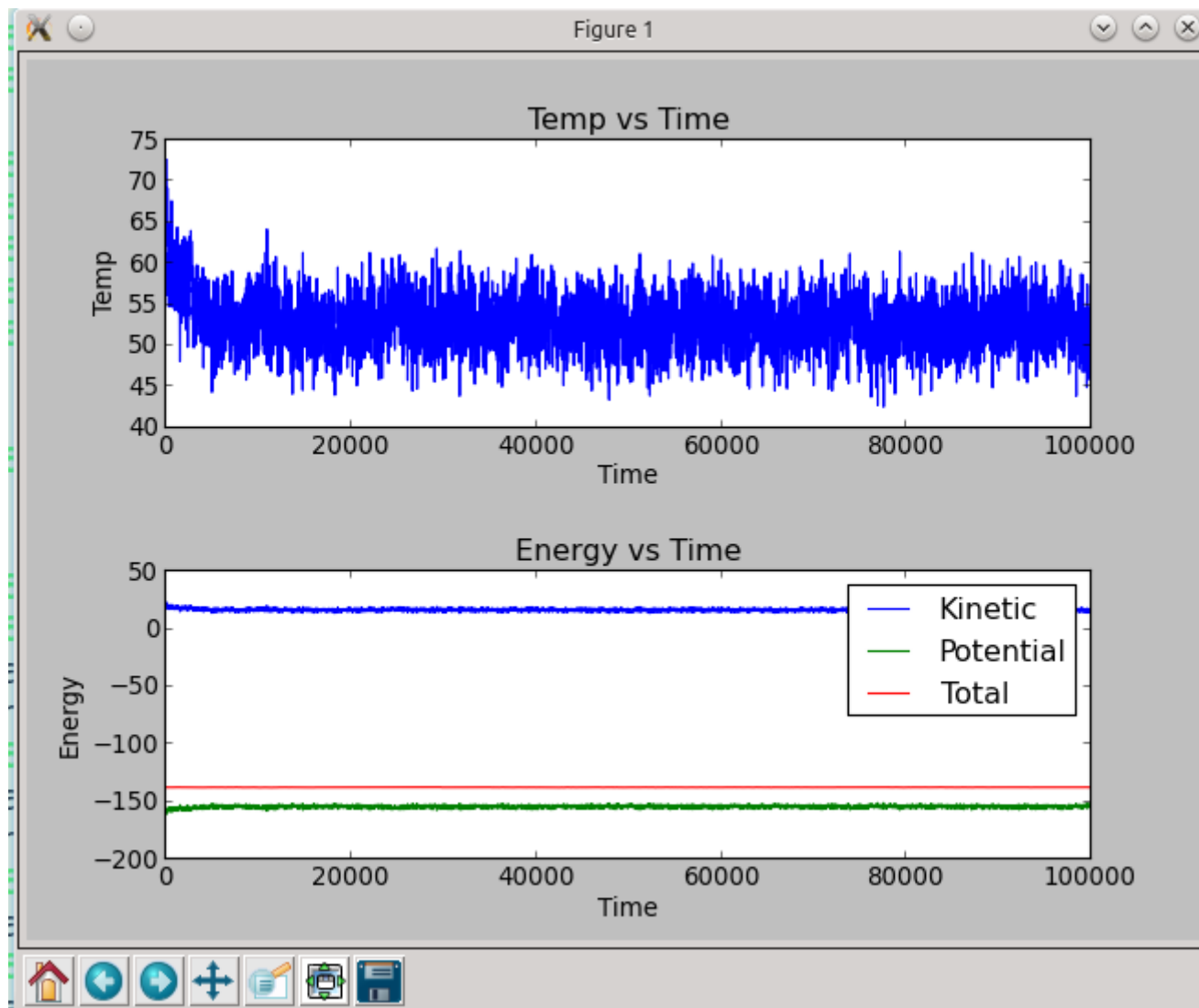
Number of Atoms:	<input type="text" value="108"/>	Restfile (in *.rest):	<input type="text" value="argon_108.rest"/>
Mass of Atoms (in amu):	<input type="text" value="39.948"/>	Trajectory ( in *.xyz):	<input type="text" value="argon_108.xyz"/>
Epsilon (in kj/mol):	<input type="text" value="0.2379"/>	Output energies (in *.dat):	<input type="text" value="argon_108.dat"/>
Sigma (in angstrom):	<input type="text" value="3.405"/>	Choice Potential:	<input checked="" type="radio"/> Lennard Jones
rcut (in angstrom):	<input type="text" value="8.5"/>		<input type="radio"/> Morse
box length (in angstrom):	<input type="text" value="17.158"/>		<input type="radio"/> Home Made
Number of steps in MD:	<input type="text" value="10000"/>	Choice Integrator:	<input checked="" type="radio"/> Verlet
MD time step (in fs):	<input type="text" value="5.0"/>		<input type="radio"/> Integrator 2
output print frequency:	<input type="text" value="100"/>	Choice Thermostat:	<input checked="" type="radio"/> Andersen
		Temperature:	<input type="text" value="95.0"/>
		Frequency:	<input type="text" value="0.1"/>
			<input type="radio"/> None

It is project in the workshop at ICTP

# NVT



# NVE





# How Does the code look?

```
if mdsys.thermostat==False:
    for i in range(mdsys.nsteps):
        ## This is the main loop, integrator and force calculator
        if (i % mdsys.nprint == 0):
            mdsys.output(i)
            time.append(i)
            md.velverlet(byref(mdsys))
            md.ekin(byref(mdsys))
            if (i % cellfreq == 0):
                md.updcells(byref(mdsys))
if mdsys.thermostat==True:
    for i in range(mdsys.nsteps):
        ## This is the main loop, integrator and force calculator
        if (i % mdsys.nprint == 0):
            mdsys.output(i)
            time.append(i)
            md.velverlet(byref(mdsys))
            md.ekin(byref(mdsys))
            md.andersen(byref(mdsys))
            if (i % cellfreq == 0):
                md.updcells(byref(mdsys))
```

- What have we achieved
  - Learning skills with software development tools
    - gdb → debugging
    - git → software sharing and development
    - Code coupling → making use of C power to perform fast math and python simple language and packages to input and proses and present data
    - Having a Tidy(ish) code easy to follow

git-repo: <https://github.com/JLeitmotiv/ljmd-c.git>