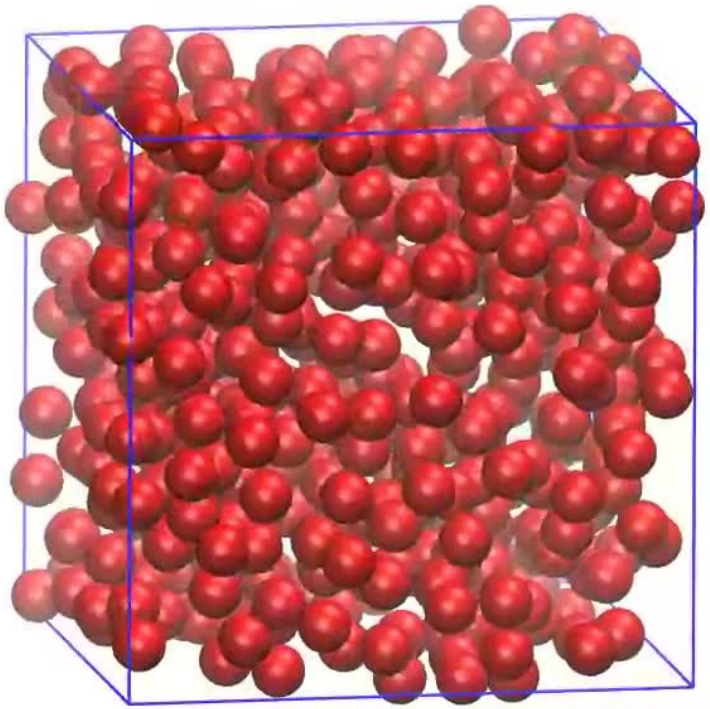
Molecule Dynamics Project
Coupling Python with C



Pablo Alcain
Md. Enamul Hoque
Matias Factorovich

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

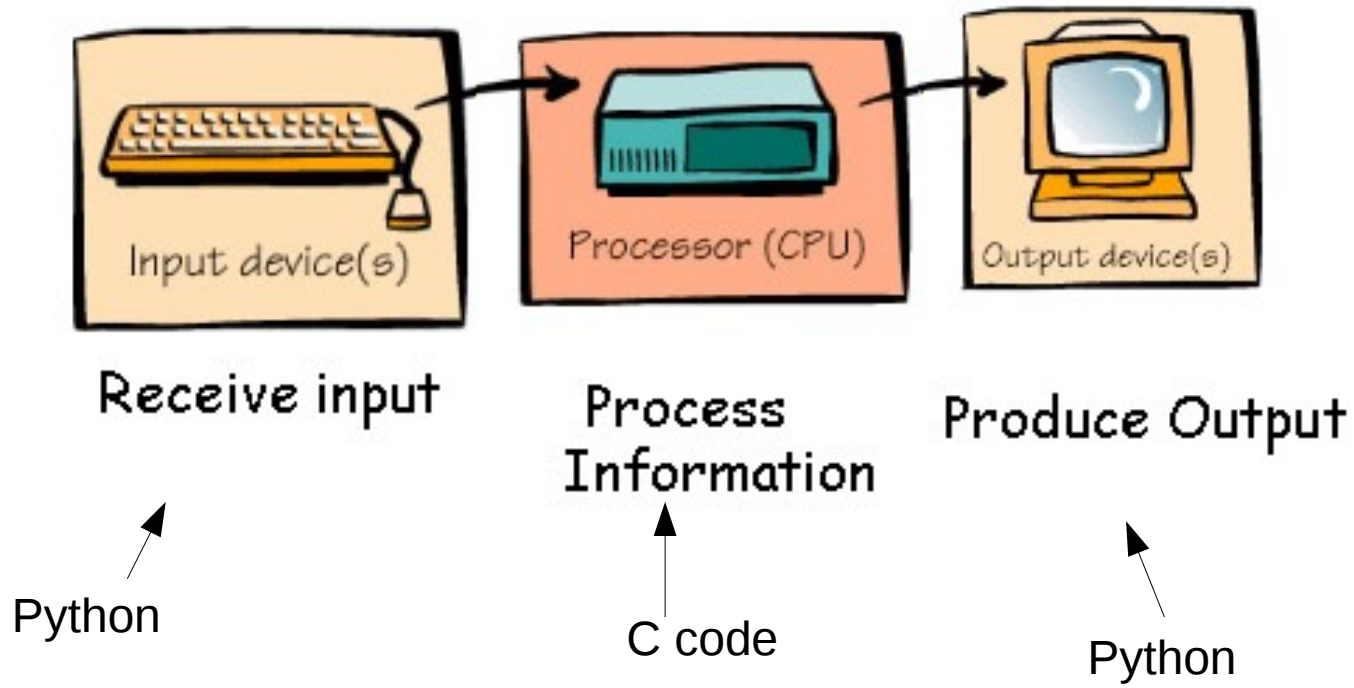
What we wanted to add?

- 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

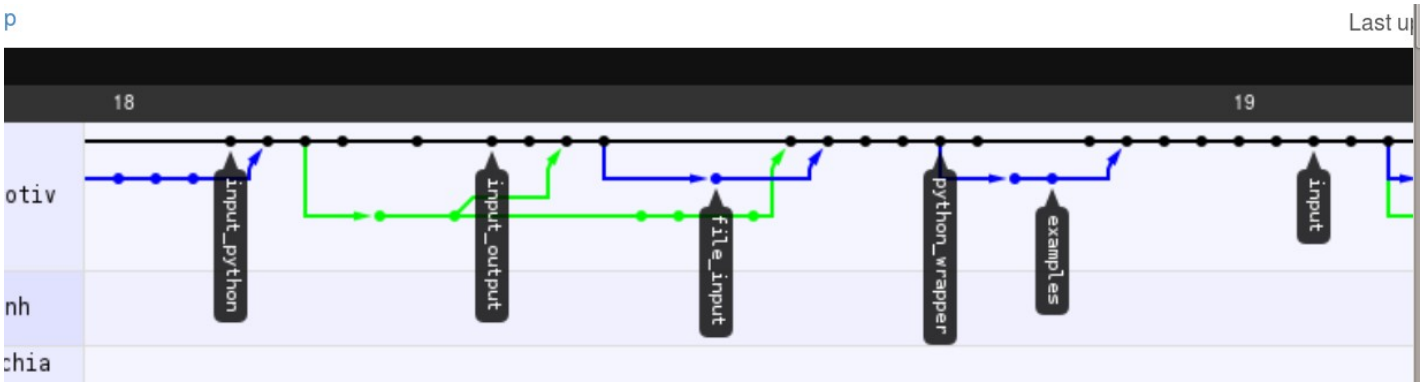
- The Solution:Take our base MD program written in C and coupled it to python.
  - Easy program language
  - Build features for making graph
  - Easy coupling with C through c_types
  - Build features for making GUI

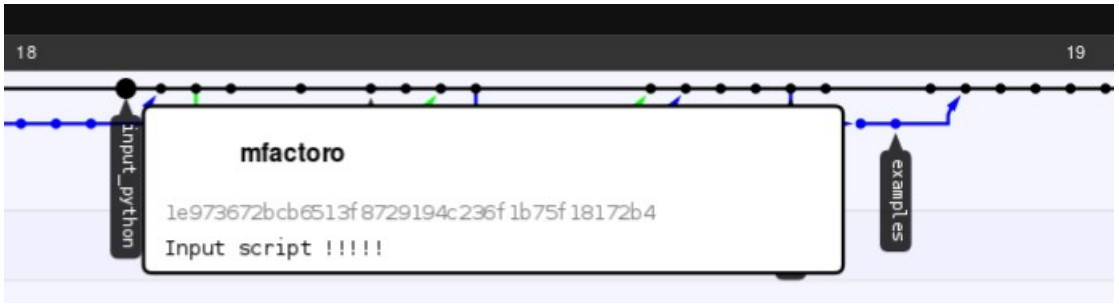
Let's get on with it!!!

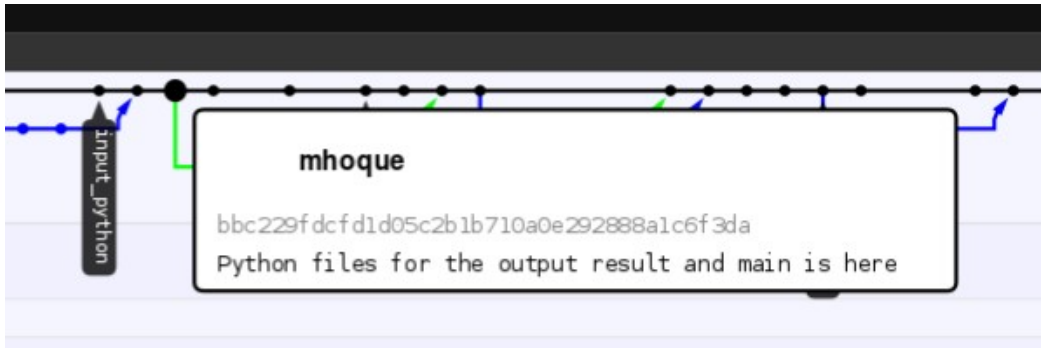
- First Goal: Run the MD program as a library Called from python
  - Just the heavy calculation will be done in C
    - Cell build, force calculation, Verlet integrator
  - Management in python
    - Input/Output and looping through time-step

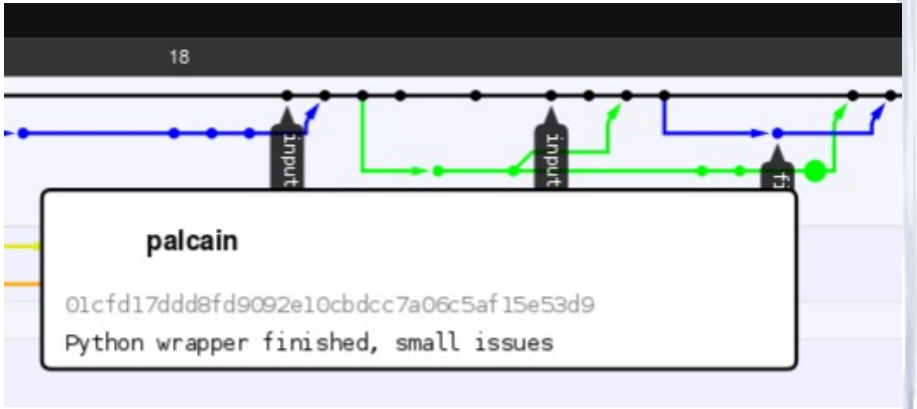


Each one Starts their job









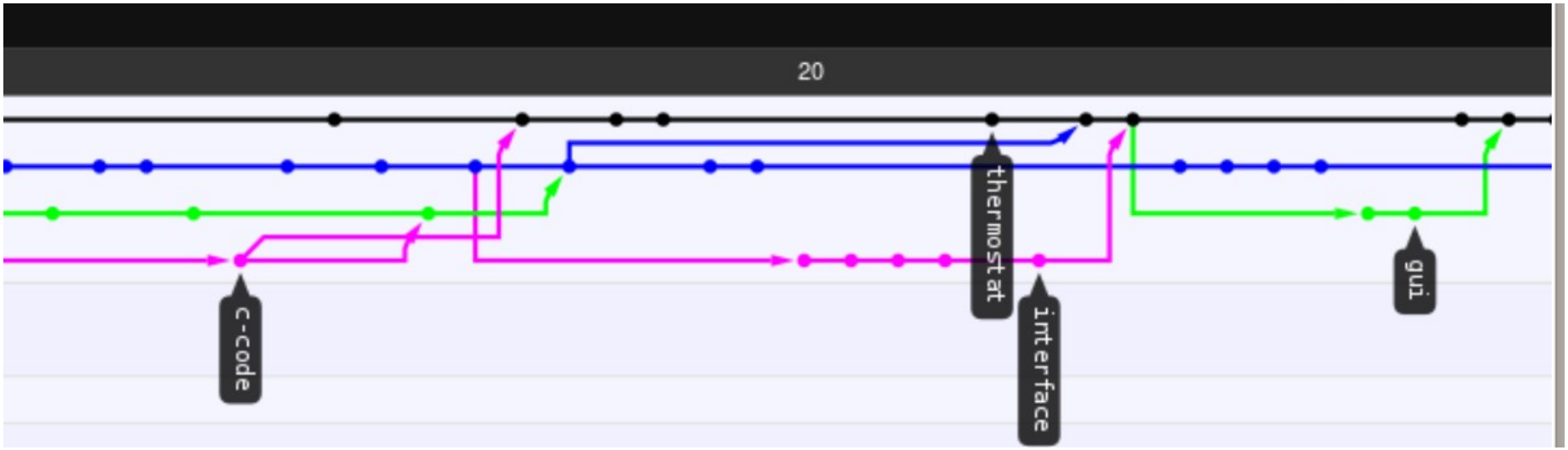
What we achieved ???

```
for i in range(mdsys.nsteps):
  ## This is the main loop, integ
  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 ????

[-]: 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
[ ]: Makefiles and c-code
  ( ): 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)

[ ]: add a LUT potential
[-]: add different integrators methods
  (X)write code
  ( )make python use it
[ ]: add different atom types

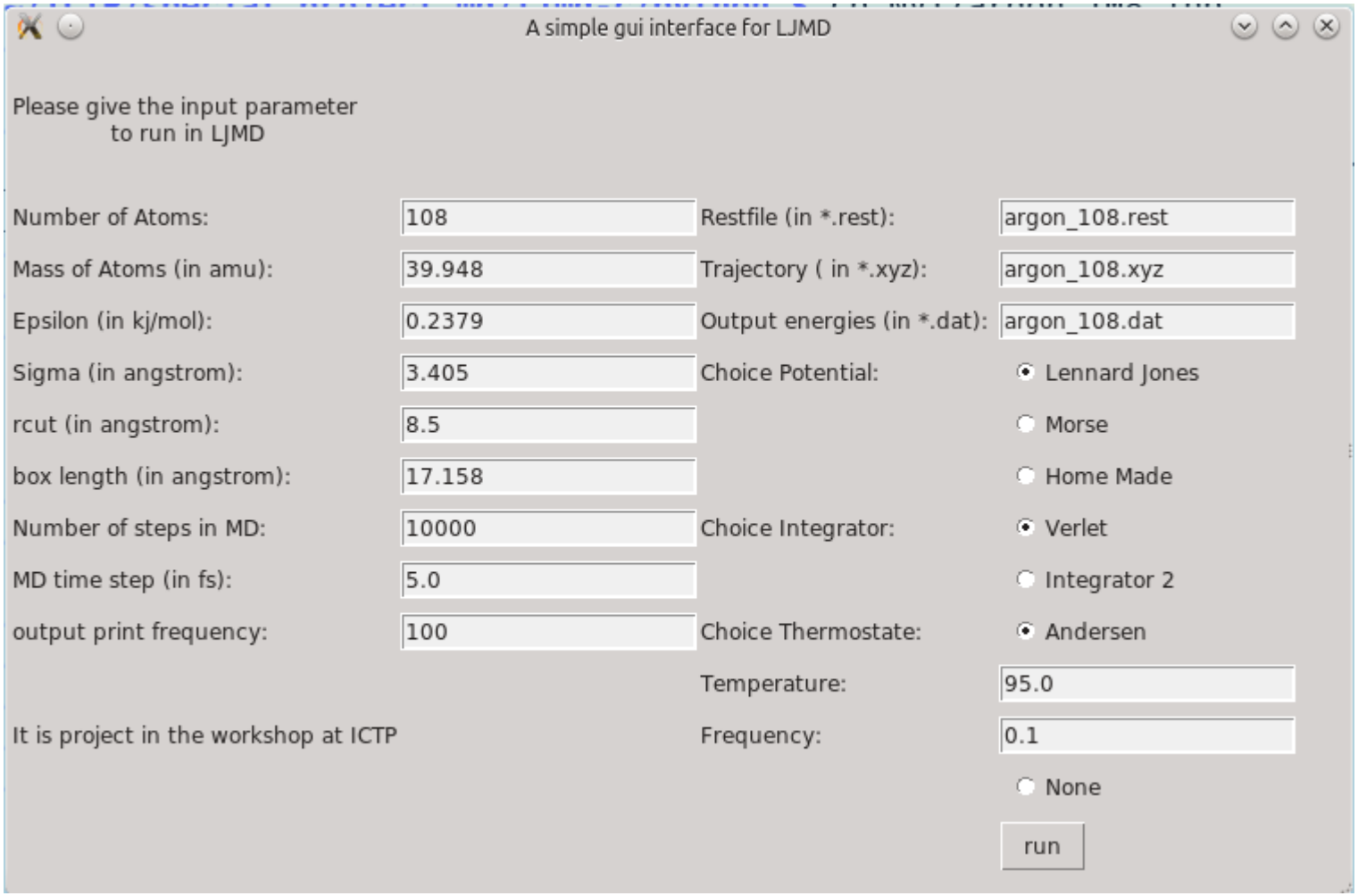


Three main Branches

- 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:
108
Mass of Atoms (in amu):
39.948
Epsilon (in kj/mol):
0.2379
Sigma (in angstrom):
3.405
rcut (in angstrom):
8.5
box length (in angstrom):
17.158
Number of steps in MD:
10000
MD time step (in fs):
5.0
output print frequency:
100
It is project in the workshop at ICTP
Restfile (in *.rest):
argon_108.rest
Trajectory (in *.xyz):
argon_108.xyz
Output energies (in *.dat):
argon_108.dat
Choice Potential:
Lennard Jones
Morse
Home Made
Choice Integrator:
Verlet
Integrator 2
Choice Thermostate:
Andersen
Temperature:
95.0
Frequency:
0.1
None
run

NVT

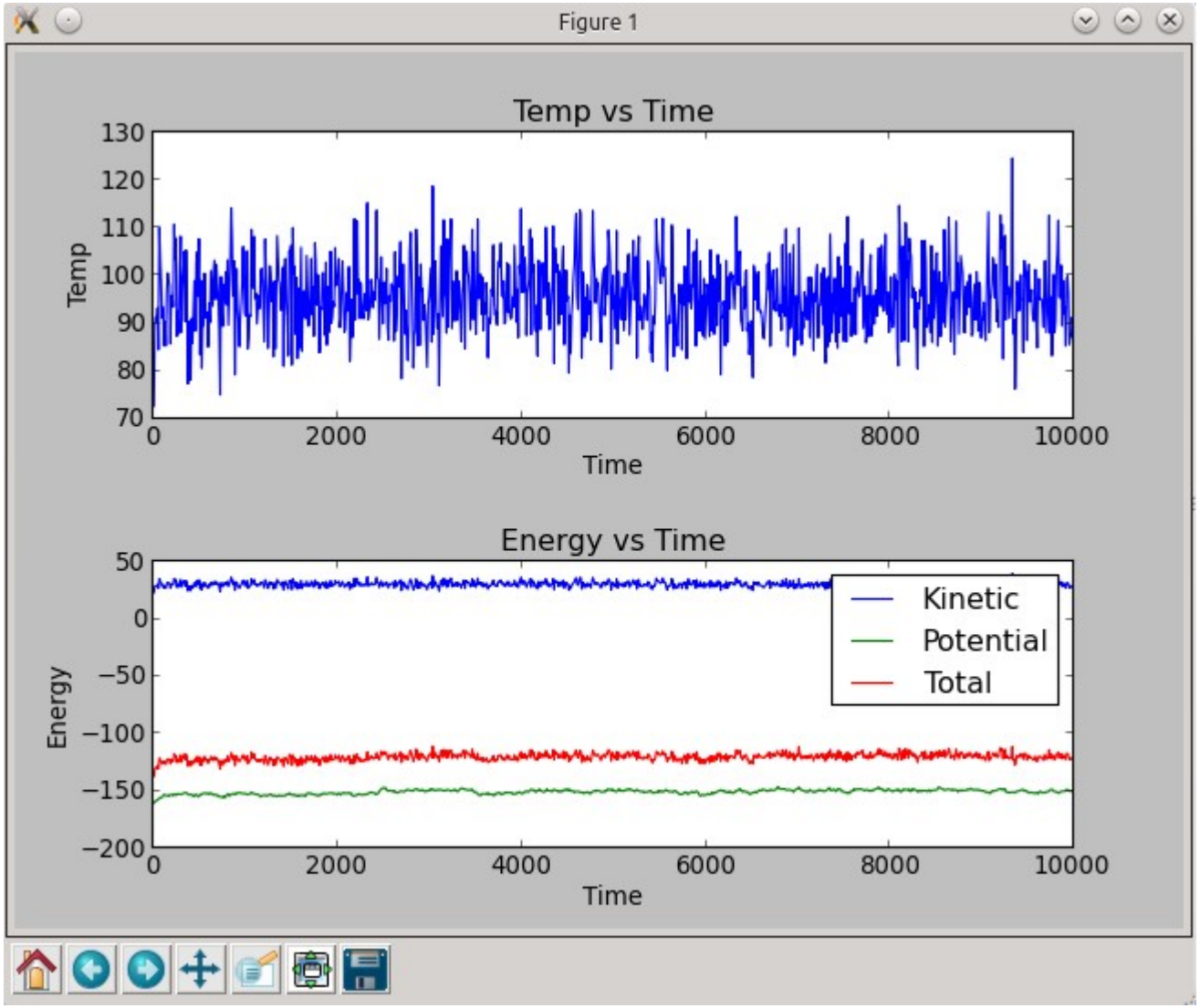


Figure 1
Temp vs Time
Temp
130
120
110
100
90
80
70
0
2000
4000
6000
8000
10000
Time
Energy vs Time
Energy
50
0
−50
−100
−150
−200
Kinetic
Potential
Total
Time

NVE

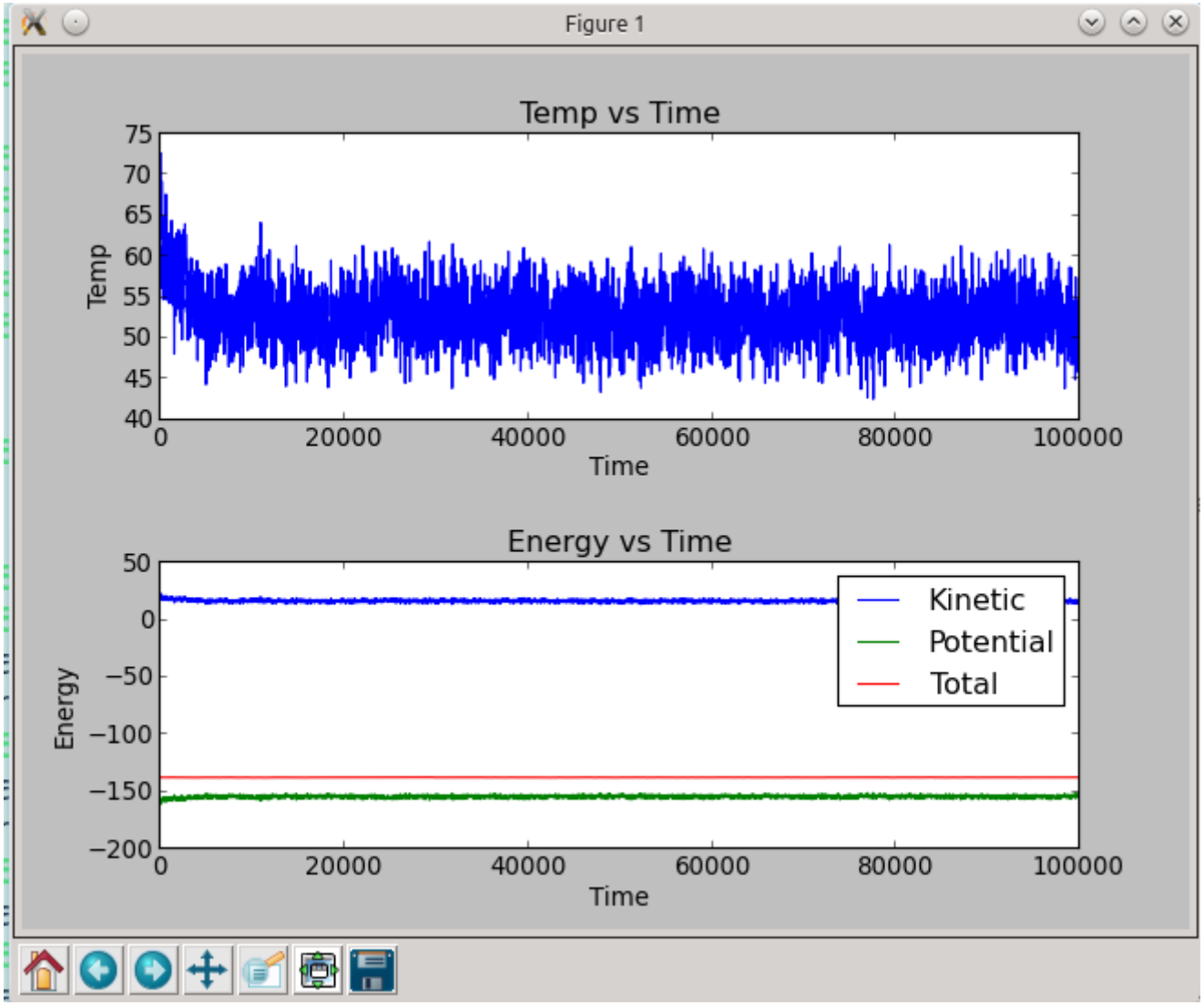


Figure 1
Temp vs Time
Temp
75
70
65
60
55
50
45
40
0
20000
40000
60000
80000
100000
Time
Energy vs Time
Energy
50
0
−50
−100
−150
−200
Kinetic
Potential
Total
0
20000
40000
60000
80000
100000
Time

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