computation of lattice pamrameters of UO2 for temperatures

lattice parameters of uo2 for temperatures (from 250K to 3000K; step size=125K) were computed.
lammps computed each volume corresponding to each temperature.
lattice parameter was computed as fllows:

```
c = (initial volume of system)**(1/3)/5.47      # where 5.47 is an initial lattice parameter.
lattice parameter at this temperature = (volume at this temperature)**(1/3) / c
```

volume at each temperature was computed with following script:

common header, script_part1.txt used in the python code below

```
#Initialization:
atom_style charge
units metal
boundary        p p p    # impose periodic boundary conditions for three directions
dimension       3
newton          on
#default is on
#Atom definition:
read_data data.fcc111    # uo2 system on fcc 111 lattice
mass 1 238
group         1 type 1
mass 2 16
group         2 type 2
#Force fields:
pair_style hybrid/overlay born/coul/long 10.0 morse 10.0
pair_coeff 1 1 born/coul/long 0.013806784 0.32702 3.26 0.0 0.0 10.0
pair_coeff 1 2 born/coul/long 0.013806827 0.327021 3.54 0.0 0.0 10.0
pair_coeff 1 2 morse 0.57745 1.65 2.369 10.0
pair_coeff 2 2 born/coul/long 0.013806869 0.327022 3.82 3.950633264 0.0 10.0
# the potentials above referred to [lammps-users] Uranium dioxide, incorrect net charge of system
kspace_style pppm 1.0e-4      # use ewald method
```

download: script_part1.txt

the python code, lattice.py: it generates lammps scripts and executes them in a loop

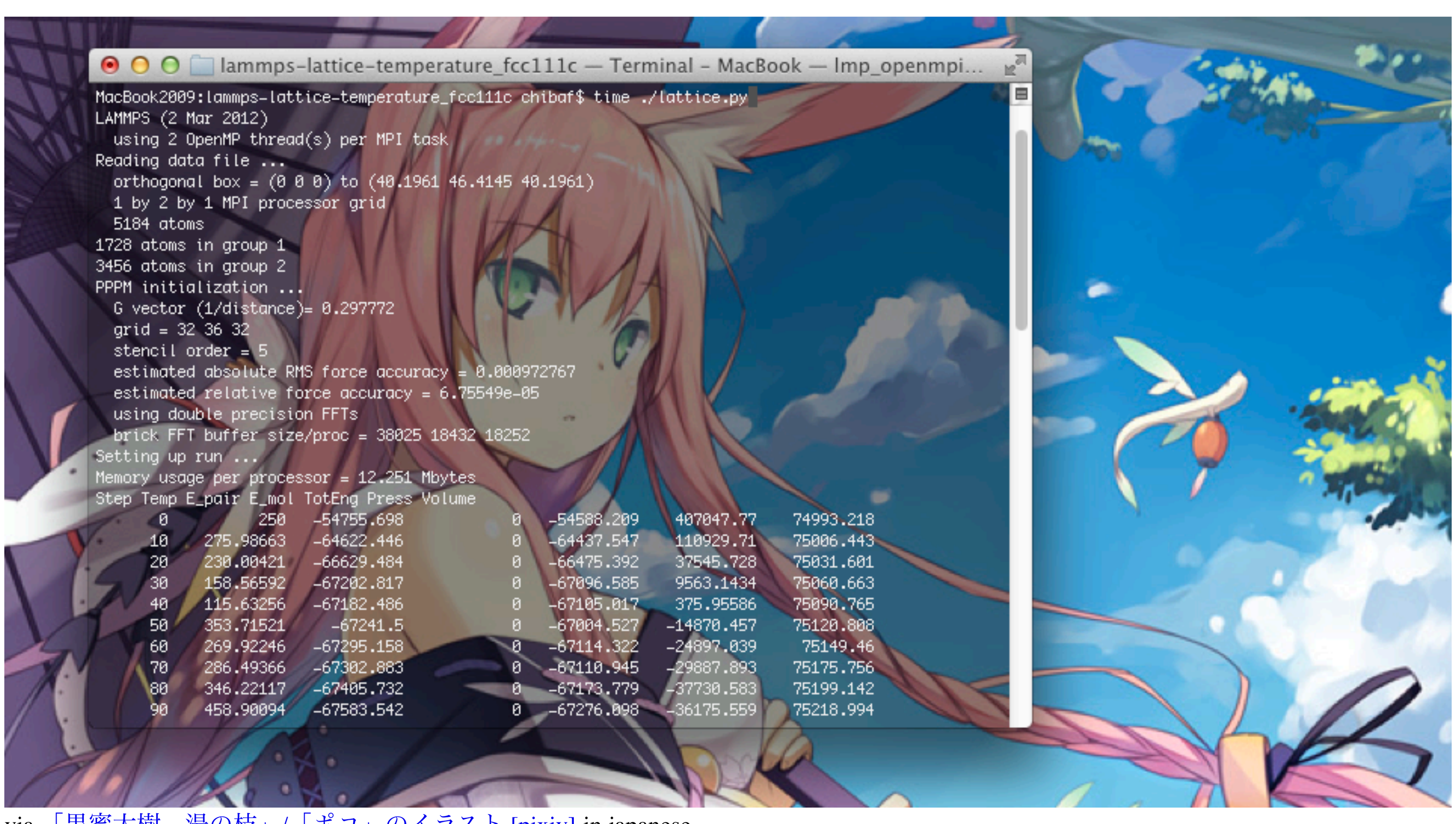
```
#!/usr/bin/python

import sys
from string import *
import os
from subprocess import *

for i in range(0,23):
 temp=250+i*125
 fn="in."+str(temp)
 call("cat script_part1.txt >"+fn, shell=True)   # use the header above
 s="log log."+str(temp)+" "
 call("echo " +s+ ">>"+fn, shell=True)
 s="velocity all create " +str(float(temp))+ " 78621 dist gaussian"
 call("echo " +s+ ">>"+fn, shell=True)
 s="fix NPT all npt temp " +str(float(temp))+ " " +str(float(temp))+ " 0.005 iso 0.0 0.0 10.0 drag 0.2"
 call("echo " +s+ ">>"+fn, shell=True)
 s="thermo 10 "
 call("echo " +s+ ">>"+fn, shell=True)
 call("echo " +s+ ">>"+fn, shell=True)
 s="run 3000 "       # number of time step
 call("echo " +s+ ">>"+fn, shell=True)
 call("time mpirun -np 2 lmp_openmpi -sf opt < "+fn, shell=True)  # use optimization for pair_coeff
```

download: lattice.py

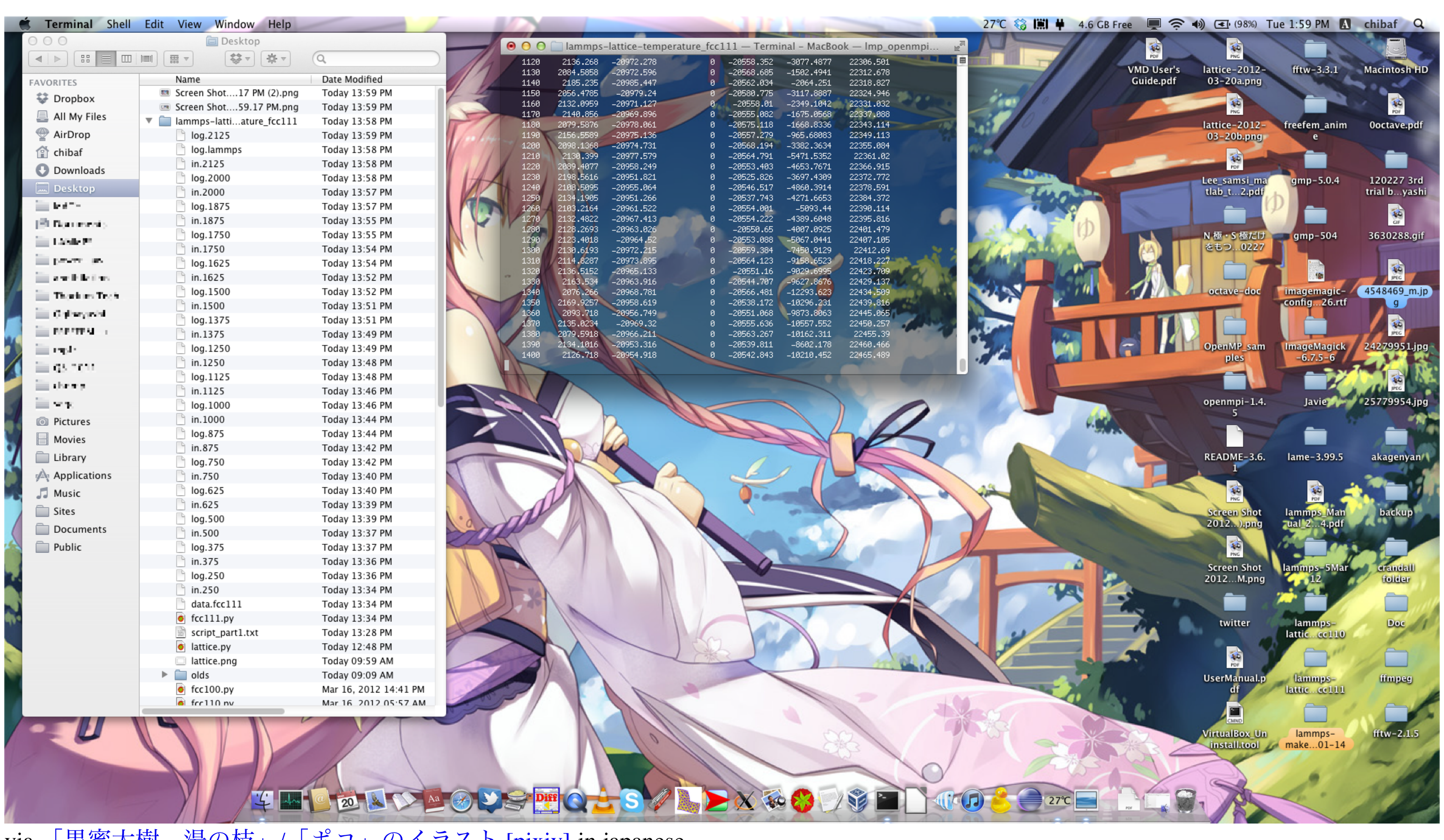
download: fcc111.py

usage:

```
$ ./fcc111.py > data.fcc111
$ ./lattice.py
```



via 「黒蜜大樹　湯の枝」/「ポコ」のイラスト [pixiv] in japanese
"a big tree of black honey and a sprig of hot water" / illustration by poko [pixiv]



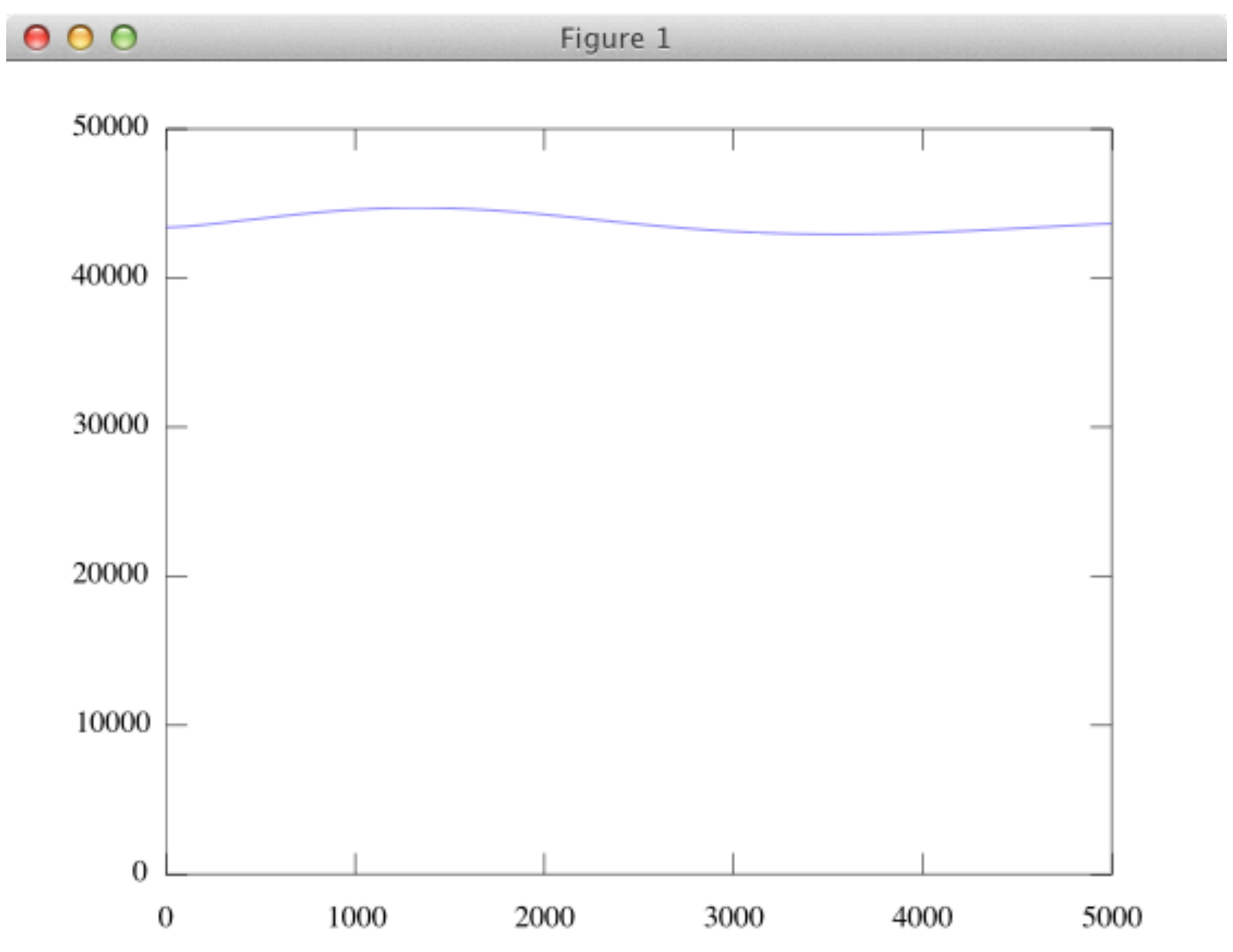
via 「黒蜜大樹　湯の枝」/「ポコ」のイラスト [pixiv] in japanese
"a big tree of black honey and a sprig of hot water" / illustration by poko [pixiv]

the computation was done from 250K to 3000K.
uo2 on fcc111 lattice was employed and its system size was 3000 atoms (U=1000, O=2000).

post-process with octave of lammps computation was done as follows:

```
octave-3.6.1:2> a=readlog("log.1500");
octave-3.6.1:3> a
WARNING: terminal is not fully functional
a =(press RETURN)

  scalar structure containing the fields:

    Chead =
    {
      [1,1] = Step
      [1,2] = Temp
      [1,3] = E_pair
      [1,4] = E_mol
      [1,5] = TotEng
      [1,6] = Press
      [1,7] = Volume
      [1,8] =
    }
    data =
    {
      [1,1] =

       0.0000e+00   1.5000e+03  -3.0472e+04   0.0000e+00  -2.9891e+04   4.7107e+05   4.3399e+04
       1.0000e+01   4.7468e+03  -3.7655e+04   0.0000e+00  -3.5815e+04   1.1956e+05   4.3400e+04
       2.0000e+01   1.3793e+03  -3.7171e+04   0.0000e+00  -3.6637e+04   1.2023e+05   4.3403e+04
       3.0000e+01   3.0126e+03  -3.8240e+04   0.0000e+00  -3.7072e+04   6.7935e+04   4.3408e+04
       4.0000e+01   2.1906e+03  -3.8609e+04   0.0000e+00  -3.7760e+04   6.8932e+04   4.3412e+04
       5.0000e+01   2.4341e+03  -3.9170e+04   0.0000e+00  -3.8226e+04   6.6184e+04   4.3418e+04
       6.0000e+01   2.1992e+03  -3.9549e+04   0.0000e+00  -3.8696e+04   6.6632e+04   4.3424e+04
       7.0000e+01   2.3105e+03  -4.0018e+04   0.0000e+00  -3.9122e+04   7.8593e+04   4.3430e+04
       8.0000e+01   2.1054e+03  -4.0356e+04   0.0000e+00  -3.9540e+04   8.4733e+04   4.3438e+04
-- less -- (f)orward, (b)ack, (q)uit
octave-3.6.1:4> b=a.data(1){};
octave-3.6.1:5> plot(b(:,1),b(:,7))
octave-3.6.1:6> axis([0,5000,0,50000])
```

the above shows the seventh column of log data includes volumes.



then lattice parameters for temperatures is computed as follows.

```
octave-3.6.1:1> for i=0:22;a=num2str(250+125*i);b=readlog(strcat("log.",a));
> c=b.data(1){};n=length(c);f=c(1,7)**(1/3)/5.47;d(i+1,1)=250+125*i;
> d(i+1,2)=(sum(c(n-100+1:n,7))/100)**(1/3)/f;end;  % computation of lattice parameter
octave-3.6.1:2> plot(d(:,1),d(:,2),'-*');
octave-3.6.1:3> axis([250,3000,5.35,5.6])
octave-3.6.1:4> save("-ascii","lattice.dat","d")  % save lattice parameter
octave-3.6.1:5> print -dpng lattice.png;
```

download: readlog.m (a modified version)

a batch mode of octave is also available:

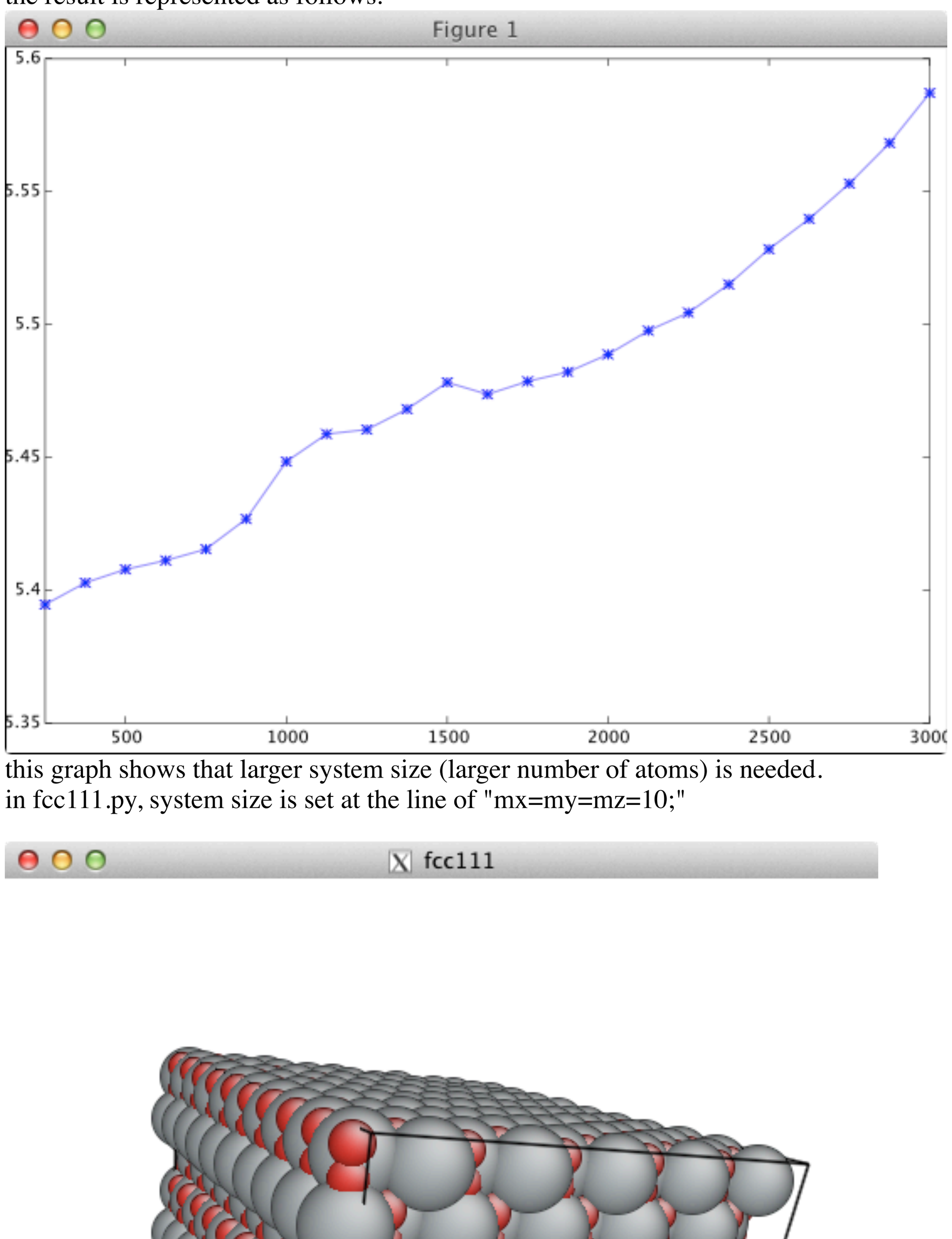
```
$ octave -q lattice.txt       # "-q" is an option for quiet mode.
```

lattice.txt

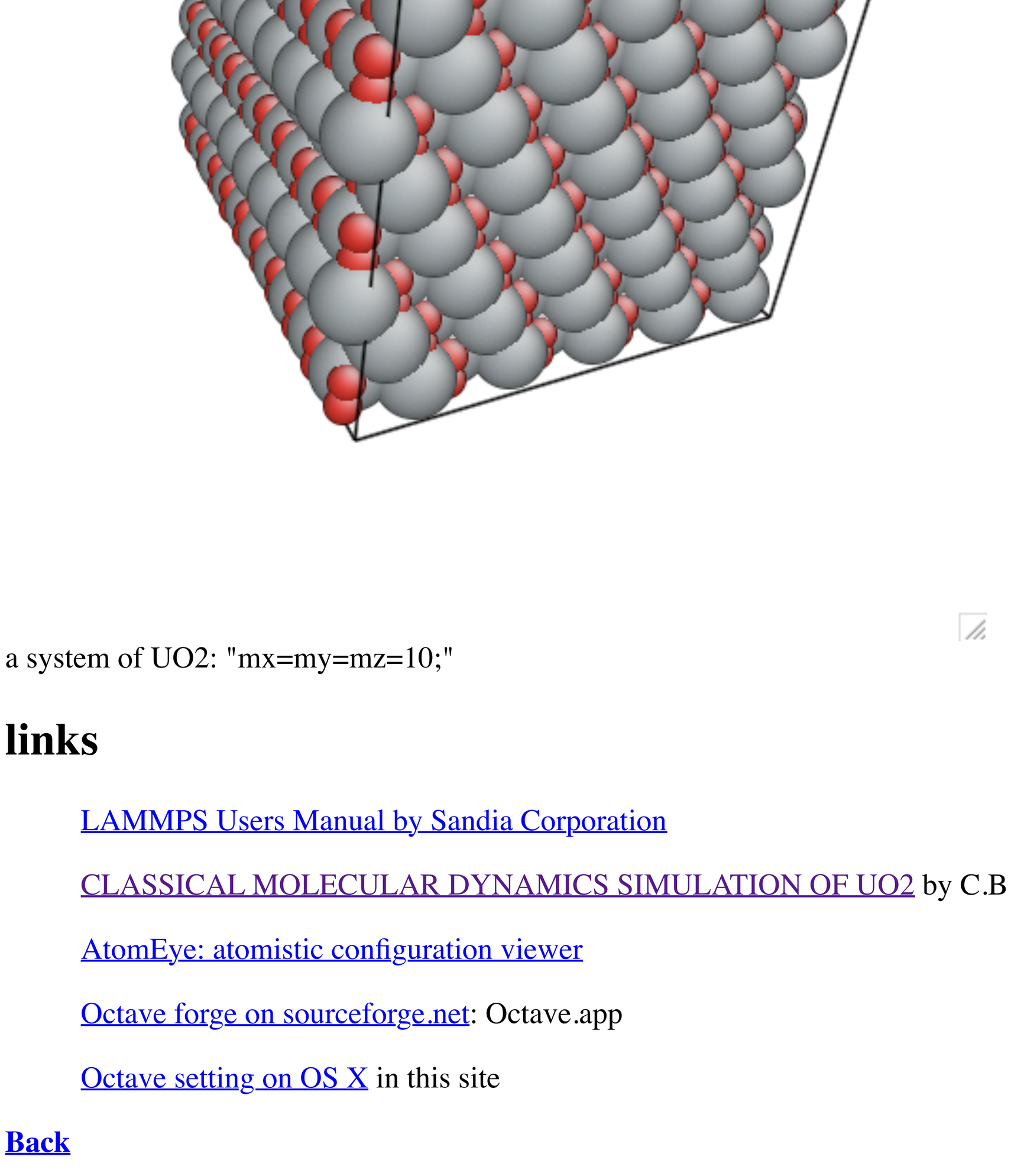
```
for i=0:22;a=num2str(250+125*i);b=readlog(strcat("log.",a));
c=b.data(1){};n=length(c);f=c(1,7)**(1/3)/5.47;d(i+1,1)=250+125*i;
d(i+1,2)=(sum(c(n-100+1:n,7))/100)**(1/3)/f;end;
plot(d(:,1),d(:,2),'-*');
axis([250,3000,5.35,5.6]);
save("-ascii","lattice.dat","d");
print -dpng lattice.png;
```

download: lattice.txt

the result is represented as follows:



this graph shows that larger system size (larger number of atoms) is needed.
in fcc111.py, system size is set at the line of "mx=my=mz=10;"



a system of UO2: "mx=my=mz=10;"

links

- LAMMPS Users Manual by Sandia Corporation
- CLASSICAL MOLECULAR DYNAMICS SIMULATION OF UO2 by C.B. Basak (pdf)
- AtomEye: atomistic configuration viewer
- Octave forge on sourceforge.net: Octave.app
- Octave setting on OS X in this site

Back

Mail to chibaf(if you have some questions or ...)

Last update on 22/May/2012