3 Dimension Particle Simulation: Latent Heat of Evaporating Water Droplet :HW4

David Jedynak

February 19, 2018

Abstract

This paper will demonstrate how to use the Molecular Dynamics simulation to show the heat needed to change a liquid water droplet to a gas at constant temperature(latent heat). The theory behind the simulation will be covered. Algorithms used to implement theory will be documented and explained. A full procedure will be given to show how the simulation was used in this experiment.

1 Introduction

Latent heat is the energy absorbed or released by a mass during a constant temperature phase change. A water droplet in a constant temperature environment will absorb thermal energy and if the energy is enough, the liquid water will turn into a gas. A high enough temperature must be used otherwise a phase change might not occur. The higher the temperature is, the faster the phase change will occur and this will make it more difficult to observe in the simulation.

2 Theory

Latent heat will be the energy absorbed by the liquid water droplet as it turns into a gas. This can be calculated by taking the difference between the total energy (kinetic and potential) when the system was a liquid, and then as it is a gas.

Latent Heat is equal to the difference of the total energy from the start time and the final time.

p particle
i particle index
n total number of particles
PE - potential energy
KE - kinetic energy
t0 - initial time
tf - final time

$$LH = \sum_{i=1}^{n} KE(p_i(v_x, v_y, v_z, t0) + PE(p_i(x, y, z, t0)) - \sum_{i=1}^{n} KE(p_i(v_x, v_y, v_z, tf) + PE(p_i(x, y, z, tf))$$

3 Algorithms, Code

3.1 Important C Functions

Pseudo Code.(See code below for 3D plotting) This function was very important in this simulation so here is a more detailed description of it.

1. this part initializes several variables

```
int size=xdim;
 typedef struct part{
   double x[D]; // position
   int c; //color
 } part;
 part p[N];

 tet+=tetdot;                                                   //increment the
 phi+=phidot;                                                   //increment the
```

2. check to see if this should be a 3d simulation and set the scale

```
if (D!=3){
  printf("3d visualization requires D=3, you have D=%i!",D);
  return;
}
if (ydim<size) size=ydim;
scalefac=0.7*size/L*shift;
```

3. center the cube using the given L

```
// center cube
  for (int n=0; n<N; n++)
    for (int d=0; d<D; d++){
      p[n].x[d]=x[n][d]-L/2;
      p[n].c=n+1;
    }
```

4. Rotate and shift the particles from the viewing perspective

```
// rotate around y
  double c=cos(tet);
  double s=sin(tet);
  for (int n=0; n<N; n++){
    double x=c*p[n].x[0]+s*p[n].x[2];
    p[n].x[2]=-s*p[n].x[0]+c*p[n].x[2];
    p[n].x[0]=x;
  }
  // rotate around x
  c=cos(phi);
  s=sin(phi);
  for (int n=0; n<N; n++){
    double y=c*p[n].x[1]+s*p[n].x[2];
    p[n].x[2]=-s*p[n].x[1]+c*p[n].x[2];
    p[n].x[1]=y;
  }
  // shift box away from origin
  for (int n=0; n<N; n++){
    p[n].x[2]+=shift;
  }
  qsort(&p[0].x[0],N,sizeof(part),&compare);
```

5. set the colors and the sizes of the different particles

```
for (int n=0; n<N; n++){                                 //draw the circles with diffe
    int xx=p[n].x[0]/p[n].x[2]*scalefac;
    int yy=p[n].x[1]/p[n].x[2]*scalefac;
    myfilledcircle(0     ,xdim/2+xx,ydim/2-yy,0.5/p[n].x[2]*scalefac+2);
    myfilledcircle(p[n].c,xdim/2+xx,ydim/2-yy,0.5/p[n].x[2]*scalefac);
  }
}
```

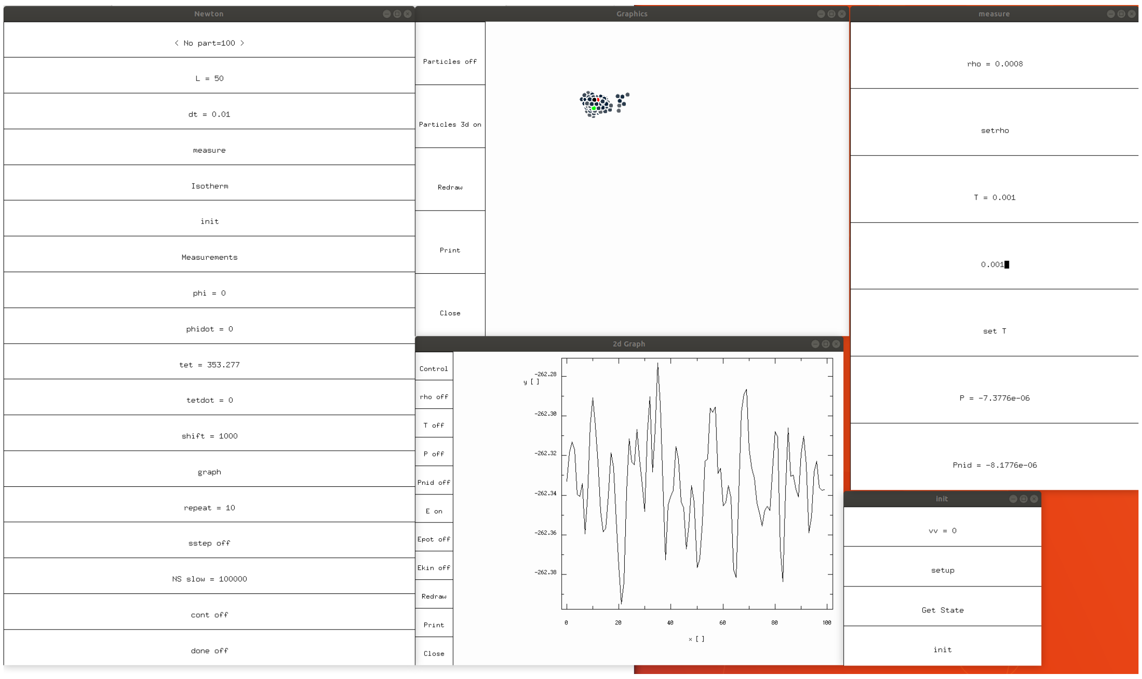
4 Procedure

1. Open terminal and launch the MD program with the following command

```
$ ./MDA_Therm_Exec
```

2. Click Graph, in the graph window, turn off particles and turn on 3D particles. In the main window, click Measurements, turn on E (Energy). In the main window click init and measure. In the measure window, set T to 0.001, In the main window, set No part to 100, dt to 0.1.
3. leave other settings to defaults
4. In main window, click turn continue(cont) you should see the particles quickly group together into a liquid.
5. Note: you don't have to click set T continually because i added the setTemp() function to iterate so it keeps the T as the set point.
6. Once the particles reach a steady state and they are no longer a gas, turn (cont) off (like figure 1). take a screen shot to record initial values.
7. change dt to 0.01
8. set T to 0.5
9. Now we will use the step button to cycle until the liquid/solid has turned into a gas. Observe the E graph the DC change will be the Latent heat energy.
10. Once the energy seems to level off this should mean that the system is now a gas like in figure 2
11. record the final Energy Graph by taking a screen shot.

5 Results

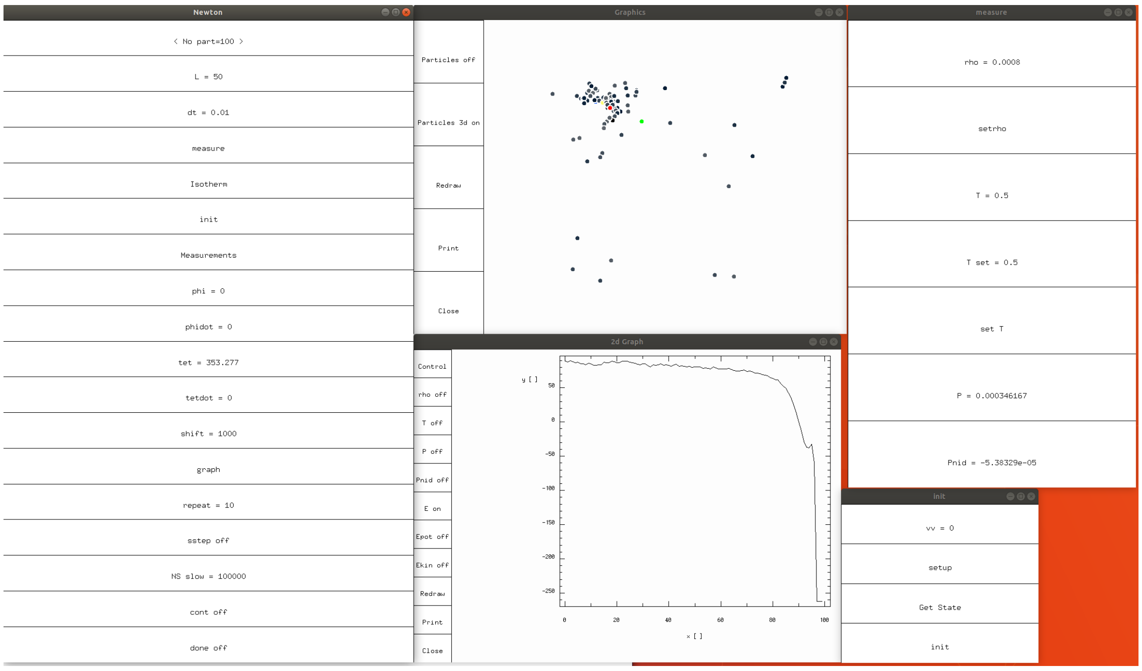


```
< No part=100 >
L = 50
dt = 0.01
phi = 0
phidot = 0
tet = 353.277
tetdot = 0
shift = 1000
repeat = 10
sstep off
NS slow = 100000
cont off
done off

rho = 0.0008
T = 0.001
0.001
P = -7.3776e-06
Pnid = -8.1776e-06

vv = 0
```

Figure 1:
The simulation experiment is set up. All the particles are grouped together. The energy is oscillating but the amplitude is pretty small.



< No part=100 >
L = 50
dt = 0.01
phi = 0
phidot = 0
tet = 353.277
tetdot = 0
shift = 1000
repeat = 10
sstep off
NS slow = 100000
cont off
done off

rho = 0.0008
T = 0.5
T set = 0.5
P = 0.000346167
Pnid = -5.38329e-05

vv = 0

Figure 2:
The temperature has been set to 0.5 and iterated several times

It appears the Latent heat of this phase transition was about 340 Joules.

6 Conclusion

The results were as expected. As the liquid water was put in a constant high temp environment, it absorbed heat*(figure 2 shows this), and the liquid turned into a gas. The transition between phases had a very non linear curve whereas after and before the transition the energy had a linear relationship with Temperature.

7 Simulation Code

```
#include <math.h>
#include <mygraph.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>

#define Nmax 1000
#define D 3
#define MeasMax 100
double  L=50; // size of box
//changed code to always run setTemp() in iterate to keep T constant.

double m[Nmax]; // charges of the particles
double x[Nmax][D],v[Nmax][D]; // State of the system

// parameters
double scalefac=100;
double x0[Nmax][D],v0[Nmax][D],dt=0.01,vv=1;
double rho[MeasMax],Tset=0,Tmeas[MeasMax], ppnid[MeasMax],pp[MeasMax],Etot[MeasMax],Epot[MeasMax],
int N=Nmax,Measlen=MeasMax,iterations=0;

// Global variables for Isotherm
int Thermalize=10000, MeasNo=1000;

// 3d visualization
double tet=0,phi=0,tetdot=0.05,phidot=0,shift=150;
/*The set density function changes the value L (box length) in order to achieve a given density*/
void setdensity(){
  double fact=1/L;
  L=pow(N/rho[0],1./D);
  fact*=L;
  for (int n=0;n<N; n++)
    for (int d=0; d<D; d++)
      x[n][d]*=fact;
}

/*Calculate the temperature based off of the number of particles, kinetic energy, and the density
double density(){
  double r=0;
  r=N;
  for (int d=0; d<D; d++) r/=L;
  return r;
}
/*calculate the average Temperature by summing the kinetic energy, then dividing it by D dimension
double T(double v[N][D]){
  double t=0;
  for (int n=0; n<N; n++)
    for (int d=0; d<D; d++)
      t+=m[n]*v[n][d]*v[n][d];
  return t/N/D;
}
/*every time this is called the Temp gets set to whatever T is set to*/
void setTemp(){
  Tmeas[0]=T(v);
  double fact=sqrt(Tset/Tmeas[0]);
  for (int n=0;n<N; n++)
    for (int d=0; d<D; d++)
      v[n][d]*=fact;
}
/*Non ideal pressure*/
double Pnid(double x[N][D]){
  double p=0;
  for (int n=0; n<N; n++)
    for (int m=n+1; m<N; m++){
      double dr[D],dR=0;
      for (int d=0; d<D; d++){
        dr[d]=x[m][d]-(x[n][d]-L);
        double ddr;
        ddr=x[m][d]-(x[n][d]);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr;
        ddr=x[m][d]-(x[n][d]+L);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr;
        dR+=dr[d]*dr[d];
      }
      double dR6=dR*dR*dR;
      double dR12=dR6*dR6;
      double Fabs=12/dR12-6/dR6;
      p+=Fabs/D;
    }
  for (int d=0; d<D; d++) p/=L;

  return p;
}
/*Potential Energy, calculated using the forces and the distances between each particle*/
double Ep(double x[N][D],double v[N][D]){
  double EE=0;

  for (int n=0; n<N; n++)
    for (int m=n+1; m<N; m++){
      double dr[D];
      double dR=0;
      for (int d=0; d<D; d++){
        dr[d]=x[m][d]-(x[n][d]-L);
        double ddr;
        ddr=x[m][d]-(x[n][d]);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr;
        ddr=x[m][d]-(x[n][d]+L);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr;
        dR+=dr[d]*dr[d];
      }
      double dR6=dR*dR*dR;
      double dR12=dR6*dR6;
      EE += 1/dR12-1/dR6;
    }
  return 2*EE;
}
/*Calculates the force on each particle in each direction
-iterates though every particle calculates the distance between it and other particles*/
void F(double x[N][D], double v[N][D],double FF[N][D]){

  memset(&FF[0][0],0,N*D*sizeof(double)); //zeroize
  for (int n=0; n<N; n++) //iterate particles
    for (int m=n+1; m<N; m++){
      double dr[D],dR=0;
      for (int d=0; d<D; d++){ //iterate dimensions
        dr[d]=x[m][d]-(x[n][d]-L);
        double ddr;
        ddr=x[m][d]-(x[n][d]);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr; // as the
        ddr=x[m][d]-(x[n][d]+L);
        if (fabs(ddr)<fabs(dr[d])) dr[d]=ddr; //as the particle gets far away force levels out
        dR+=dr[d]*dr[d];
      }
      double dR6=dR*dR*dR;
      double dR12=dR6*dR6;
      double Fabs=12/dR12/dR-6/dR6/dR;
      for (int d=0;d<D; d++){
        FF[n][d]-=Fabs*dr[d];
        FF[m][d]+=Fabs*dr[d];
      }
    }
  return;
}
/*primary function, updates dynamics for particles using above functions and also controls temp
-iterate through each velocity and integrate acceleration
```

```
-iterate position integrate velocity*/
void iterate(double x[N][D],double v[N][D],double dt){
  double ff[N][D];
  F(x,v,ff);
  if (iterations==0)
    for (int n=0;n<N;n++)
      for (int d=0;d<D;d++)
        v[n][d]+=0.5*ff[n][d]/m[n]*dt;
  else
    for (int n=0;n<N;n++)
      for (int d=0;d<D;d++)
        v[n][d]+=ff[n][d]/m[n]*dt;

  for (int n=0;n<N;n++)
    for (int d=0;d<D;d++){
      x[n][d]+=v[n][d]*dt;
      if (x[n][d]<0) x[n][d]+=L;
      else if (x[n][d]>=L) x[n][d]-=L;
    }
  setTemp();//added set temp to lower amout of mouse clicks for setting temperature.
  iterations++;
}

/*set initial velocities, masses*/
void setup(){
  int M=pow(N-1,1./D)+1;
  for (int n=0; n<N; n++){
    m[n]=1;
    for (int d=0; d<D; d++){
      int nn=n;
      for (int dd=0; dd<d; dd++) nn/=M;
      x0[n][d]=(nn%M)*L/M;
      if (d==1){
        if (x0[n][0]<L/2)
          v0[n][d]=vv;
        else v0[n][d]=-vv;
      }
      else v0[n][d]=0;
    }
  }
}
/*sets initial positions and velocities for particles*/
void init(){
  for (int n=0; n<N; n++)
    for (int d=0; d<D; d++){
      x[n][d]=x0[n][d];
      v[n][d]=v0[n][d];
    }
  iterations=0;
}
/*saves the current state as the initial state. for future use. pretty nice*/
void GetState(){
  for (int n=0; n<N; n++)
    for (int d=0; d<N; d++){
      x0[n][d]=x[n][d];
      v0[n][d]=v[n][d];
    }
  iterations=0;
```

```
}
/*2D graph of particles*/
void draw(int xdim, int ydim){
  int size=xdim;
  if (ydim<size) size=ydim;
  scalefac=size/L;

  mydrawline(1,0,size,size,size);
  mydrawline(1,size,0,size,size);
  for (int n=0; n<N; n++){
    int xx=x[n][0]*scalefac;
    int yy=x[n][1]*scalefac;
    myfilledcircle(n+1,xx,size-yy,0.5*scalefac);
  }
}
/**/
int compare(const void *x1,const void *x2){
  if (((double *) x1)[2]< ((double *)x2)[2]) return 1;
  else return -1;
}
/*3D graph of particles*/
void draw3d(int xdim, int ydim){
  int size=xdim;
  typedef struct part{
    double x[D]; // position
    int c; //color
  } part;
  part p[N];

  tet+=tetdot;                                                        //increment the vieu
  phi+=phidot;                                                        //increment the vieu

  if (D!=3){
    printf("3d visualization requires D=3, you have D=%i!",D);
    return;
  }
  if (ydim<size) size=ydim;
  scalefac=0.7*size/L*shift;

  // center cube
  for (int n=0; n<N; n++)
    for (int d=0; d<D; d++){
      p[n].x[d]=x[n][d]-L/2;
      p[n].c=n+1;
    }
  // rotate around y
  double c=cos(tet);
  double s=sin(tet);
  for (int n=0; n<N; n++){
    double x=c*p[n].x[0]+s*p[n].x[2];
    p[n].x[2]=-s*p[n].x[0]+c*p[n].x[2];
    p[n].x[0]=x;
  }
  // rotate around x
  c=cos(phi);
  s=sin(phi);
  for (int n=0; n<N; n++){
    double y=c*p[n].x[1]+s*p[n].x[2];
```

```
      p[n].x[2]=-s*p[n].x[1]+c*p[n].x[2];
      p[n].x[1]=y;
    }
    // shift box away from origin
    for (int n=0; n<N; n++){
      p[n].x[2]+=shift;
    }
    qsort(&p[0].x[0],N,sizeof(part),&compare);

    for (int n=0; n<N; n++){                              //draw the circles with differen
      int xx=p[n].x[0]/p[n].x[2]*scalefac;
      int yy=p[n].x[1]/p[n].x[2]*scalefac;
      myfilledcircle(0     ,xdim/2+xx,ydim/2-yy,0.5/p[n].x[2]*scalefac+2);
      myfilledcircle(p[n].c,xdim/2+xx,ydim/2-yy,0.5/p[n].x[2]*scalefac);
    }
  }
  /*store data from simulation for various dynamics*/
  void Measure(){
    memmove(&rho[1],&rho[0],(MeasMax-1)*sizeof(double));
    rho[0]=density();
    memmove(&Tmeas[1],&Tmeas[0],(MeasMax-1)*sizeof(double));
    Tmeas[0]=T(v);
    memmove(&ppnid[1],&ppnid[0],(MeasMax-1)*sizeof(double));
    ppnid[0]=Pnid(x);
    memmove(&pp[1],&pp[0],(MeasMax-1)*sizeof(double));
    pp[0]=rho[0]*Tmeas[0]+ppnid[0];
    memmove(&Ekin[1],&Ekin[0],(MeasMax-1)*sizeof(double));
    Ekin[0]=N*D*Tmeas[0];
    memmove(&Epot[1],&Epot[0],(MeasMax-1)*sizeof(double));
    Epot[0]=Ep(x,v);
    memmove(&Etot[1],&Etot[0],(MeasMax-1)*sizeof(double));
    Etot[0]=Epot[0]+Ekin[0];
  }
  /*isotherm file management / writing*/
  void Isotherm(){
    FILE *res;
    char IsoName[100];

    sprintf(IsoName,"Iso%f_%i.dat",Tset,N);
    res=fopen(IsoName,"w");

    for (rho[0]=density(); rho[0]>0.01;rho[0]/=1.1){
      setdensity();
      // thermalize
      for (int i=0; i<Thermalize; i++){
        setTemp();
        iterate(x,v,dt);
      }
      // measure values
      double PP=0, TT=0;
      for (int i=0; i<MeasNo; i++){
        iterate(x,v,dt);
        double Tmeas=T(v);
        TT+=Tmeas;
        PP+=rho[0]*Tmeas+Pnid(x);
      }
      TT/=MeasNo;
      PP/=MeasNo;
      fprintf(res,"%e %e %e\n",rho[0],PP,TT);
      Events(1);
      DrawGraphs();
    }
    fclose(res);
  }
  /*isotherm routine*/
  void Isotherms(){
    double LStart=L;
    for (Tset=0.05; Tset<2; Tset+=0.05){
      L=LStart;
      init();
      Isotherm();
    }
  }
  /**/
  int main(){
    struct timespec ts={0,1000000};
    int cont=0;
    int sstep=0;
    int repeat=10;
    int done=0;
    char name[50],mname[N][50];
    setup();
    init();
    Measure();

    DefineGraphN_R("rho",&rho[0],&Measlen,NULL);
    DefineGraphN_R("T",&Tmeas[0],&Measlen,NULL);
    DefineGraphN_R("P",&pp[0],&Measlen,NULL);
    DefineGraphN_R("Pnid",&ppnid[0],&Measlen,NULL);
    DefineGraphN_R("E",&Etot[0],&Measlen,NULL);
    DefineGraphN_R("Epot",&Epot[0],&Measlen,NULL);
    DefineGraphN_R("Ekin",&Ekin[0],&Measlen,NULL);

    AddFreedraw("Particles",&draw);
    AddFreedraw("Particles 3d",&draw3d);
    StartMenu("Newton",1);
    DefineMod("No part",&N,Nmax);
    DefineDouble("L",&L);
    DefineDouble("dt",&dt);
    StartMenu("measure",0);
    DefineDouble("rho",&rho[0]);
    DefineFunction("setrho",setdensity);
    DefineDouble("T",&Tmeas[0]);
    DefineDouble("T set",&Tset);
    DefineFunction("set T",setTemp);
    DefineDouble("P",&pp[0]);
    DefineDouble("Pnid",&ppnid[0]);
    EndMenu();
    StartMenu("Isotherm",0);
    DefineInt("Thermalize",&Thermalize);
    DefineInt("MeasNo",&MeasNo);
    DefineFunction("Measure Isotherm",Isotherm);
    DefineFunction("Measure multiple Isotherms",Isotherms);
    EndMenu();
    StartMenu("init",0);
    for (int n=0; n<N; n++){
    }
    if (N<15)
      for (int n=0; n<N; n++){
        sprintf(mname[n],"Particle %i",n);
        StartMenu(mname[n],0);
        DefineDouble("m",&m[n]);
        for (int d=0; d<D; d++){
          sprintf(name,"x[%i]",d);
          DefineDouble(name,&x0[n][d]);
        }
        for (int d=0; d<D; d++){
          sprintf(name,"v[%i]",d);
          DefineDouble(name,&v0[n][d]);
        }
        EndMenu();
      }
    DefineDouble("vv",&vv);
    DefineFunction("setup",&setup);
    DefineFunction("Get State",&GetState);
    DefineFunction("init",&init);
    EndMenu();
    DefineGraph(curve2d_,"Measurements");
    DefineDouble("phi",&phi);
    DefineDouble("phidot",&phidot);
    DefineDouble("tet",&tet);
    DefineDouble("tetdot",&tetdot);
    DefineDouble("shift",&shift);
    DefineGraph(freedraw_,"graph");
    DefineInt("repeat",&repeat);
    DefineBool("sstep",&sstep);
    DefineLong("NS slow",&ts.tv_nsec);
    DefineBool("cont",&cont);
    DefineBool("done",&done);
    EndMenu();
    while (!done){
      Events(1);
      DrawGraphs();
      if (cont||sstep){
        sstep=0;
        for (int i=0; i<repeat; i++) iterate(x,v,dt);
        Measure();
      }
      else      nanosleep(&ts,NULL);
    }
  }
```

References

https://www.ndsu.edu/pubweb/~carswagn/LectureNotes/370/index.html