Molecular Dynamics study of melting in 2D, inverse-twelfth -power interaction

Name: Tanmoy pandit
Email: tanmoypandit163@gmail.com

Contents

1 Introduction **2**

2 Basic introduction for phase transition **2**

3 Algorithm for of pressure vs density with LJ potential **3**

4 The System studied **4**

5 The run time **4**

6 Programme **4**

7 Plot **9**

8 The Explanation of our result **10**

8.1 Equation of state 10

8.2 The Experimental Evidence 10

9 References **11**

---

1 Introduction

Recently there has been a great deal of experimental and theoretical effort devoted to the study of the melting transition in two dimensions (2D). Experimental systems studied include the melting of mono-layers of rare-gas atoms on graphite, 2 electrons on He films, polystyrene spheres on water, and thin liquid-crystal films.Other simulation work on hard disk and L-J potential suggested that 2-D melting is a first order phase transition.There are also different analytic method also to study the phase transition like **Renormalization Group** method etc.The geometric packing fraction idea plays an important rule in study of melting in 2D.The important question is that computer simulation is really capable to capture the physics of phase transition in 2D.It is true indeed that all such simulation even for long run computer simulation corresponds to small time in the time scale of real laboratory.There is an important reason to choose an inverse power r^{-n}.Because there is no characteristic length in potential ,inverse power systems have useful scaling property which allows the entire equation of state to be determined from that of the single isotherm.Yes it is true indeed that the hard-disc system($n \to \infty$) is believed to have first order phase transition.Here we have shown a MD simulation for n=12 system with particular attention near the melting point.We are able to observe the phase separation in the pressure vs density curve in the coexistence region.

2 Basic introduction for phase transition

If you have a homogeneous single molecular species which can exist atmost in 3 phases(let's assume).Generally this kind of system is characterised by thermodyanamic variables(mainly intensive quantities) e.g volume(V), density(ρ) and pressure(P)(there could be other variables as well like specific heat,specific entropy but lets keep it simple).All these are connected by some kind of equation of state. The system is best described if you express the system in terms of any of the two variables and observe the change of one in terms of other.For the better purpose of illustration we are going to show the P-V diagram(isotherms).As you keep on decreasing the temperature ,local interaction starts appearing and these leads to the system from gaseous to liquid phase.

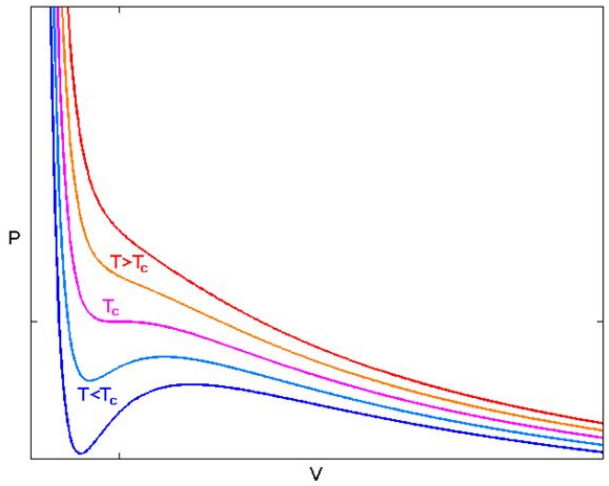

Figure 1: Phase Duagram of vanderwall gas

3 Algorithm for of pressure vs density with LJ potential

- Place the particles in triangular lattice initially, although they will move in continuum.

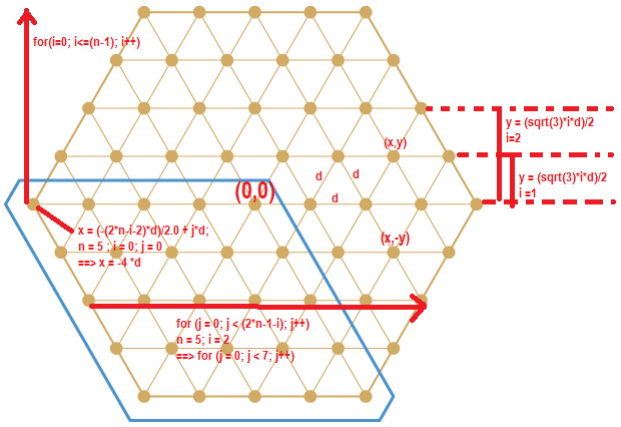

Figure 2: Placing the particle in a triangular lattice

- Calculate potential and force among neighboring particles.
- The update rule should be langevin equation in the overdamped limit i.e. with mass term zero and no change of velocity with time.
- Integration method used is modified euler update.

$$x_{t+1} = x_t + D\beta force_x \delta T + \eta_{gaussian}\sqrt{2D\delta T} \tag{1}$$

δT is the time step which goes inside square root for dimensional consistency. D is the diffusion constant. $Force_x$ is the x-component of force, similarly there will be equation for y.

- Calculate the virial and from that calculate pressure
- variance of the pressure i.e. $< P^2 >$ - $< P >^2$.
- The formula for calculating virial pressure is given by,

$$p = \frac{k_B T N}{V} + \frac{1}{Vd}\overline{\sum_{i<j} f(r_{ij}) r_{ij}}. \tag{2}$$

where,V is the volume ,f(r_{ij}) is the force on particle i exerted by particle j, and r_{ij} is the vector going from i to j: $r_{ij} = r_j$ - r_i ,N is the total number of particles and d is the dimension of the system

- We have used **Langevin Thermostat** to keep the temperature of the system fixed. When we consider the motion of large particles through a continuum of smaller particles, Langevin equation

$$\frac{dp_i}{dt} = (\frac{\partial \phi(q)}{\partial q_i} - \gamma p_i + \delta p) \tag{3}$$

$$\frac{dq_i}{dt} = \frac{p_i}{m} \tag{4}$$

is taken into account. The smaller particles create a damping force to the momenta, -γp_i, as the large particles push the smaller ones out of the way. The smaller(thermal) particles also move with kinetic energy and give random kicks to the large particles.σ, γ are connected by a fuctuation-dissipation relation $\sigma^2 = 2\gamma m_i k_B T$ in order to recover the canonical ensemble distribution.

At each time step Δt the Langevin thermostat changes the equation of motion so that the change in momenta is

$$\Delta p_i = (\frac{\partial \phi(q)}{\partial q_i} - \gamma p_i + \delta p)\Delta t \tag{5}$$

where γp_i damp the momenta and δp is a Gaussian distributed random number with probability

$$\rho(\delta p) = \frac{1}{\sqrt{2\pi}\sigma} exp(\frac{|\delta p|^2}{2\sigma^2}) \tag{6}$$

And standard deviation $\sigma^2 = 2\gamma m_i k_B T$. The random fluctuating force represents the thermal kicks from the small particles. The damping factor and the random force combine to give the correct canonical ensemble.

4 The System studied

We performed MD calculations for a system with 784 (26× 30) particles of mass m interacting with the pair potential

$$u(r) = \epsilon(\sigma/r)^{12} \tag{7}$$

truncated at r =2.50σ. Here σ has dimensions of length and ϵ energy. Assuming periodic-boundary conditions we confine the particles to a rectangular (very nearly square) unit cell whose ratio of height to width is (26/30) $(2/\sqrt{3})$ = 1.00074; this cell accommodates a section of a perfect unstrained hexagonal lattice. We generally used a time step $\Delta t = 0.00001 \times (\frac{\epsilon}{m\sigma^2})^{12}$ (In the following we use reduced units where a = e = m =1.) A typical (Einstein) vibrational period in the solid near the melting density is (80—100)Δt.Einstein assumed, for simplicity, that there was only one vibrational frequency, taking it to be an average if more than one frequency was present. This average frequency is now called the Einstein frequency E and one speaks of the Einstein single oscillator model.

5 The run time

Away from the phase-transition region only rela-tively short runs were required to achieve consistent results. We generally made an equilibration run of 25000δt . Much shorter runs gave stable thermo- dynamic properties but some structural properties very sensitive to long-wavelength fluctuations such as the angular correlation function required the longer equilibration period. In the vicinity of the melting'transition ($0.97 \leq p \leq 1.02$) much longer runs were made, par- ticularly in the middle of this region where we attempted to distinguish between hexatic and two- phase behavior.

6 Programme

```
1
2  /*HERE WE SIMULATED THE DYNAMICS OF BROWNIAN PARTICLE IN A HARMONIC OSCILLATOR POTENTIAL
3  WITH 4/(R^12) REPULSIVE INTERACTION*/
4  /*author:Tanmoy Pandit(MP16007) and Sandip Roy(MP16001),iiser mohali*/
5
6  #include<stdlib.h>
7
8  #include<math.h>
9  #include<stdio.h>
10
11 #define NOP 784                /*NUMBER OF PARTICLES*/
12 #define GAMMA 1.0               /*FRICTIONAL CONSTANT*/
13 #define SIGMA  1.1              /*NOMINAL PARTICLE DIAMETER*/
```

```
#define TIME 2.5          /*TOTAL TIME FOR EVOLUTION*/
#define TEMP 1.0               /*TEMPERATURE OF THE HEAT BATH*/
#define Kb    1.0             /*BOLTZMAN CONSTANT*/
#define deltaT   0.00001        /*TIME steps*/
#define BETA 1/(Kb*TEMP)      /*INVERSE OF ENERGY OF THE HEAT BATH*/
#define D 1/(BETA*GAMMA)      /*DIFFUSION CONSTANT*/
#define T_START    1.5             /*TIME FROM WHICH PRESSURE CALCULATION STARTED*/
#define R_CUT 2.5
#define DIMENSION 2.0


#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)
 /*-------------------------------------------------------------*/
 /* GAUSSIAN RANDOM NUMBER GENERATOR*/
 /*-------------------------------------------------------------*/
float ran2(long idum)
{
  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  float temp;

  if (idum <= 0) {
    if (-(idum) < 1) idum=1;
    else idum = -(idum);
    idum2=(idum);
    for (j=NTAB+7;j>=0;j--) {
      k=(idum)/IQ1;
      idum=IA1*(idum-k*IQ1)-k*IR1;
      if (idum < 0) idum += IM1;
      if (j < NTAB) iv[j] = idum;
    }
    iy=iv[0];
  }
  k=(idum)/IQ1;
  idum=IA1*(idum-k*IQ1)-k*IR1;
  if (idum < 0) idum += IM1;
  k=idum2/IQ2;
  idum2=IA2*(idum2-k*IQ2)-k*IR2;
  if (idum2 < 0) idum2 += IM2;
  j=iy/NDIV;
  iy=iv[j]-idum2;
  iv[j] = idum;
  if (iy < 1) iy += IMM1;
  if ((temp=AM*iy) > RNMX) return RNMX;
  else return temp;
}
#undef IM1
#undef IM2
#undef AM
#undef IMM1
#undef IA1
#undef IA2
#undef IQ1
#undef IQ2
#undef IR1
```

```
#undef IR2
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX
float gasdev(long idum)
{
  float ran1(long idum);
  static int iset=0;
  static float gset;
  float fac,rsq,v1,v2;

  if (idum < 0) iset=0;
  if  (iset == 0) {
    do {
      v1=2.0*ran2(idum)-1.0;
      v2=2.0*ran2(idum)-1.0;
      rsq=v1*v1+v2*v2;
    } while (rsq >= 1.0 || rsq == 0.0);
    fac=sqrt(-2.0*log(rsq)/rsq);
    gset=v1*fac;
    iset=1;
    return v2*fac;
  } else {
    iset=0;
    return gset;
  }
}

struct particles
{
    float x;              /*X COORDINATES OF THE PARTICLES*/
    float y;              /*Y COORDINATES OF THE PARTICLES*/
    float x_euler;        /*X COORDINATES USED DURING EULER UPDATES*/
    float y_euler;        /*Y COORDINATEDS*/
    int n_number;         /*NUMBER OF NEIGHBOR PARTICLES*/
    float max_disp;       /*MAXIMUM DISPLACEMENT OF THE PARTICLES STARTING FROM A PARTICULAR
    TIME*/
    float force1_x;       /*FORCE ALONG THE X-DIRECTION BY THE NEIGHBORING PARTICLES BEFORE
    EULER UPDATE*/
    float force1_y;       /*FORCE ALONG THE Y-DIRECTION BY THE NEIGHBORING PARTICLES BEFORE
    EULER UPDATE*/
    float force2_x;       /*FORCE CALCULATED FROM COORDINATES OF EULER UPDATE*/
    float force2_y;       /*FORCE CALCULATED FROM COORDINATES OF EULER UPDATE*/
};

/*.......Define the L-J potential..........as (4/r^12)*/
float WCA_POT(float r)
{
float f;
f=4.0/pow(r,12.0);
return (f);

}

void main()
{



    FILE *f5;
    f5=fopen("DENSITY_PRESSURE_VARIANCE.DAT","w");
    long seed=1;
    float max=0,min=0, ENERGY=0.0,dr,t,a,h,Lx,Ly,dx,dy;
    int i,j,counter=0,ensemble;
    struct particles arr[NOP];
    float VOLUME,PRESSURE=0.0,DENSITY,VIR=0.0,VAR=0.0,PRESSURE_SQUARED=0.0,force_x=0.0,
    force_y=0.0,force=0.0;


for(DENSITY=0.9;DENSITY<=.9;DENSITY+=0.01)
```

```
152     {
153
154         a=sqrt(2/(sqrt(3)*DENSITY));
155         h=sqrt(3)*a/2;
156
157         a=floorf(a*100)/100;
158         h=floorf(h*100)/100;
159
160          Lx=sqrt(NOP)*a;
161          Ly=sqrt(NOP)*h;
162
163
164          VOLUME=Lx*Ly;
165
166
167          /*-------------------------------------------------------------*/
168          /* DISTRIBUTING THE PARTICLES IN A TRIANGULAR LATTICE*/
169          /*-------------------------------------------------------------*/
170     for(i=0;i<sqrt(NOP);i++)
171     {
172      for(j=0;j<sqrt(NOP);j++)
173      {
174          if(i%2==0)
175          {
176          arr[counter].x=j*a-Lx/2.0;
177          arr[counter].y=h*i-Ly/2.0;
178
179          arr[counter].x=floorf( arr[counter].x*100)/100;
180          arr[counter].y=floorf( arr[counter].y*100)/100;
181
182          }
183         else if(i%2!=0)
184          {
185          arr[counter].x=(j+0.5)*a-Lx/2.0;
186          arr[counter].y=i*h-Ly/2.0;
187          arr[counter].x=floorf( arr[counter].x*100)/100;
188          arr[counter].y=floorf( arr[counter].y*100)/100;
189          }
190          counter++;
191          }
192     }
193 counter=0;
194
195          /*-----------------------------------------*/
196          /*EVOLUTION OF THE SYSTEM THROUGH TIME*/
197          /*-----------------------------------------*/
198 counter=0;
199 for(t=0;t<=TIME;t+=deltaT)
200    {
201         /*INITIALIZATION*/
202  for(i=0;i<NOP;i++)
203  {
204       arr[i].force1_x=0;
205       arr[i].force1_y=0;
206       arr[i].force2_x=0;
207       arr[i].force2_y=0;
208  }
209          /*
    ------------------------------------------------------------------------------------------------------------
    */
210          /*FINDING THE FORCE ALONG THE X AND Y DIRECTION BY ALL  THE  PARTICLES---BEFORE EULER
     UPDATE*/
211          /*
    ------------------------------------------------------------------------------------------------------------
    */
212     for(i=0;i<NOP-1;i++)
213     {
214     for(j=i+1;j<NOP;j++)
215     {
216          dx=arr[i].x-arr[j].x;
217          dx=dx-Lx*rint(dx/Lx);
```

```
218
219             dy=arr[i].y-arr[j].y;
220             dy=dy-Ly*rint(dy/Ly);
221
222             dr=sqrt(pow(dx,2)+pow(dy,2));
223
224             force_x=dx*12/pow(dr,14);
225             force_y=dy*12/pow(dr,14);
226
227             if(dr<R_CUT)
228             {
229             arr[i].force1_x+=force_x;
230             arr[i].force1_y+=force_y;
231
232             arr[j].force1_x+=-force_x;
233             arr[j].force1_y+=-force_y;
234             }
235            }
236        }
237             /*-----------------------------------*/
238             /*EULER UPDATE OF COORDINATES*/
239             /*-----------------------------------*/
240            for(i=0;i<NOP;i++)
241            {
242
243             arr[i].x_euler=arr[i].x+D*BETA*arr[i].force1_x*deltaT+gasdev(seed)*sqrt(2*D*deltaT);
244             arr[i].x_euler=arr[i].x_euler-Lx*rint(arr[i].x_euler/Lx);
245
246             arr[i].y_euler=arr[i].y+D*BETA*arr[i].force1_y*deltaT+gasdev(seed)*sqrt(2*D*deltaT);
247             arr[i].y_euler=arr[i].y_euler-Ly*rint(arr[i].y_euler/Ly);
248
249            }
250             /*
        ------------------------------------------------------------------------------------------------------------------------
        */
251             /*FINDING THE FORCE ALONG THE X AND Y DIRECTION BY THE NEIGHBORING PARTICLES---AFTER
        EULER UPDATE*/
252             /*
        ------------------------------------------------------------------------------------------------------------------------
        */
253
254
255 for(i=0;i<NOP-1;i++)
256 {
257     for(j=i+1;j<NOP;j++)
258     {
259             dx=arr[i].x_euler-arr[j].x_euler;
260             dx=dx-Lx*rint(dx/Lx);
261
262             dy=arr[i].y_euler-arr[j].y_euler;
263             dy=dy-Ly*rint(dy/Ly);
264
265             dr=sqrt(pow(dx,2)+pow(dy,2));
266
267             force_x=dx*12/pow(dr,14);
268             force_y=dy*12/pow(dr,14);
269
270
271             if(dr<R_CUT)
272             {
273
274             arr[i].force2_x+=force_x;
275             arr[i].force2_y+=force_y;
276
277              arr[j].force2_x+=-force_x;
278             arr[j].force2_y+=-force_y;
279             }
280
281     }
282
283 }
```

```
284
285             /*-------------------------------------*/
286 for(i=0;i<NOP;i++)
287         {
288
289          arr[i].x=arr[i].x+D*BETA*0.5*(arr[i].force1_x+arr[i].force2_x)*deltaT+gasdev(seed)*
      sqrt(2*D*deltaT);
290          arr[i].x=arr[i].x-Lx*rint(arr[i].x/Lx);
291
292          arr[i].y=arr[i].y+D*BETA*0.5*(arr[i].force1_y+arr[i].force2_y)*deltaT+gasdev(seed)*
      sqrt(2*D*deltaT);
293          arr[i].y=arr[i].y-Ly*rint(arr[i].y/Ly);
294
295         }
296
297
298
299
300 if(t>=T_START)
301 {
302     /*-------------------------*/
303     /*CALCULATION OF VIRIAL*/
304     /*-------------------------*/
305     for(i=0;i<NOP-1;i++)
306     {
307          for(j=i+1;j<NOP;j++)
308          {
309           dx=arr[i].x-arr[j].x;
310           dx=dx-Lx*rint(dx/Lx);
311
312           dy=arr[i].y-arr[j].y;
313           dy=dy-Ly*rint(dy/Ly);
314
315           dr=sqrt(pow(dx,2)+pow(dy,2));
316
317           force=12.0/pow(dr,13.0);
318
319           if(dr<R_CUT)
320               VIR+=dr*force;
321          }
322     }
323     /*-------------------------*/
324     /*PRESSURE CALCULATION*/
325     /*-------------------------*/
326
327     PRESSURE+=(DENSITY/BETA)+VIR/(DIMENSION*VOLUME);
328     PRESSURE_SQUARED+=pow((DENSITY/BETA)+VIR/(DIMENSION*VOLUME),2);
329     VIR=0.0;
330 }
331 }//TIME LOOP
332     VAR=sqrt(PRESSURE_SQUARED/((TIME-T_START)/deltaT)-pow(PRESSURE/((TIME-T_START)/deltaT),2)
      );
333     fprintf(f5,"%f %f %f\n",DENSITY,PRESSURE/((TIME-T_START)/deltaT),VAR);
334
335 VIR=0.0;
336 PRESSURE=0.0;
337 PRESSURE_SQUARED=0.0;
338
339
340      }//DENSITY LOOP
341
342 }//MAIN
```

7 Plot

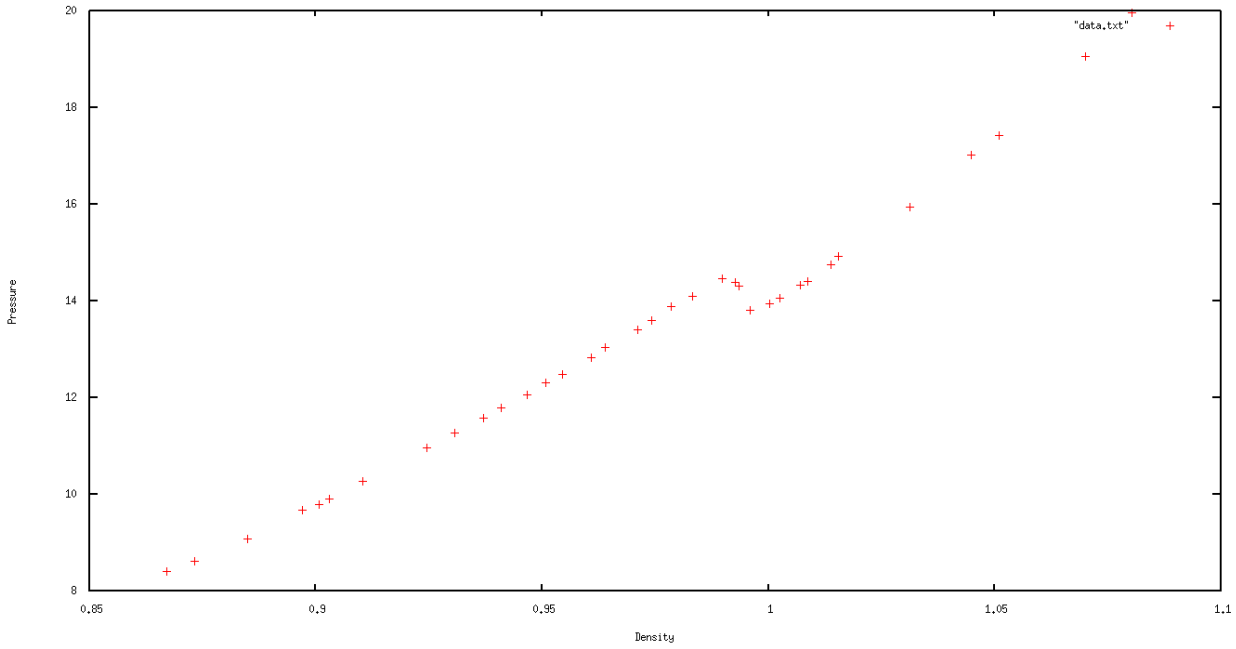

Figure 3: Density vs pressure data

8 The Explanation of our result

8.1 Equation of state

In this section we examine the behavior of the pressure along the T = 1 isotherm to try to determine the nature of the phase transition. There are some practical advantages in analyzing thermodynamic functions along an isotherm rather than an isochore since distinctions between behavior expected for first and higher-order transitions are more apparent. In an infinite system the constant pressure in the two-phase region along an isotherm implies a very large change in slope of the P-ρ curve. This change is less noticeable along an isochore and is more easily confused with the behavior expected for a continuous transition. These advantages persist for finite systems. Mayer and Wood have analyzed the behavior to be expected along an isotherm at a first-order transition for a finite system at equilibrium with periodic-boundary conditions. They find there should be a symmetric loop in the P-ρ curve in the two-phase region. Because of the nonzero interface tension between the fluid and solid phases, the finite system initially overshoots the infinite system's coexistence pressure in the F traverse to avoid forming a "droplet" of the solid phase. Similar arguments apply to the S traverse. The size of the overshoot is an increasing function of $\frac{\tau}{N^{12}}$ where τ is the interface tension, and it vanishes in the limit $N \to \infty$. There can be regions in the loop where $\frac{\partial P}{\partial \rho}_{\tau} < 0$, which of course violates stability for an infinite system. Such a region in the P-ρ plane seems very hard to explain except by assuming a nonzero interface tension between two phases. The analysis of Mayer and Wood ignores fluctuations and hence probably overestimates the size of the overshoot and is by no means rigorous, but does give a reasonable first approximation to the behavior that might be expected at a classical first-order transition for a finite system.

8.2 The Experimental Evidence

Melting is considered to be one of the most fundamental problems in physical science. Generally, dimensionality plays an important role in melting. In three-dimension, it's well known that a crystal melts directly into a liquid via a first-order transition. In two-dimension (2D), however, the melting process has been widely debated whether it is a first-order transition or a two-step transition with an intermediate hexatic

phase. Experimentally 2D melting has been intensively studied in equilibrium systems such as molecular and colloidal crystals, but rarely been explored in non-equilibrium system such as granular materials. In this paper, people experimentally studied the 2D melting in a driven granular model system at single particle level using video recording and particle tracking techniques. Measurements of orientational/translational correlation functions show evidences that the melting is a two-step transition. A novel concept of orientational/translational susceptibilities enable us to clearly resolve the intermediate hexatic phase. Our results are in excellent agreement with the two-step melting scenario predicted by KTHNY theory, and demonstrate that the KTHNY melting scenario can be extended to non-equilibrium systems.The following plot confirm you there is phase transition from solid to liquid phase

The structure factor F_{hkl} is a mathematical function describing the amplitude and phase of a wave diffracted from crystal lattice planes characterised by Miller indices h,k,l.

The structure factor may be expressed as

$$\begin{aligned} F_{hkl} =& F_{hkl} exp(i\alpha_{hkl}) \\ =& \sum_j f_j exp[2\pi i(hx_j + ky_j + lz_j)] \\ =& \sum_j f_j cos[2\pi(hx_j + ky_j + lz_j)] + i\sum_j f_j sin[2\pi(hx_j + ky_j + lz_j) \\ =& A_{hkl} + iB_{hkl} \end{aligned} \tag{8}$$

where the sum is over all atoms in the unit cell, x_j,y_j,z_j are the positional coordinates of the jth atom, f_j is the scattering factor of the jth atom, and α_{hkl} is the phase of the diffracted beam. Structure factor physically

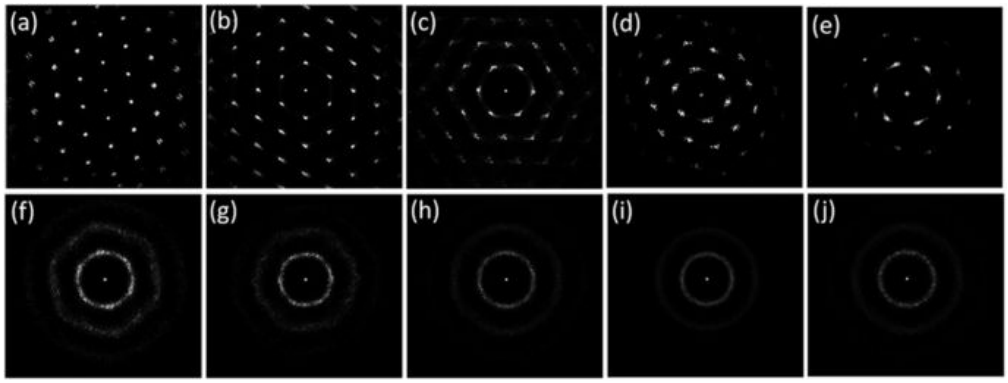

Figure 4: Change of structure factor from the solid to liquid transition

means that how closely the lattice is packed,the intensity in structure factor in solid phase is larger than liquid phase.The above diagram shows that intensity decrease with the decrease of density.

9 References

- Molecular Dynamics study of melting in two dimensions, inverse- twelfth -power interaction by Jeremy Q. Broughton, George H. Gilmer, and John D. Weeks
- Berezinskii–Kosterlitz–Thouless transition and two-dimensional melting* by V N Ryzhov, E E Tareyeva, Yu D Fomin and E N Tsiok