2016/1/14(Wed)

ウィグナー結晶の

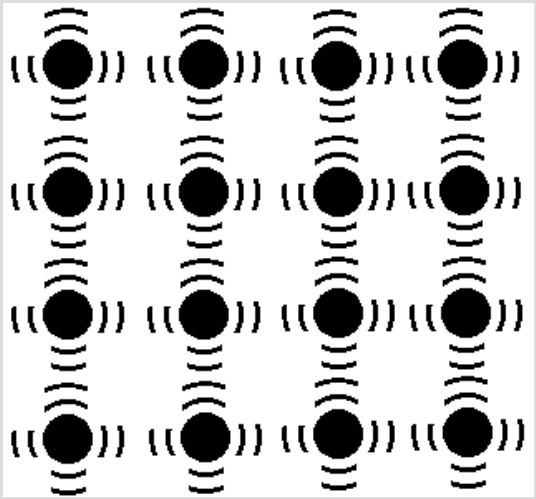
再現とその条件について

香取研究室
白木 俊平

～目次～

1．ウィグナー結晶とは

- 電子ガスが取るとされている結晶状態のこと
- 非常に低密度な領域では、電子は互いにクーロン斥力を及ぼしあっているにも関わらず、結晶化することが予想されている



[Wikipedia より引用]

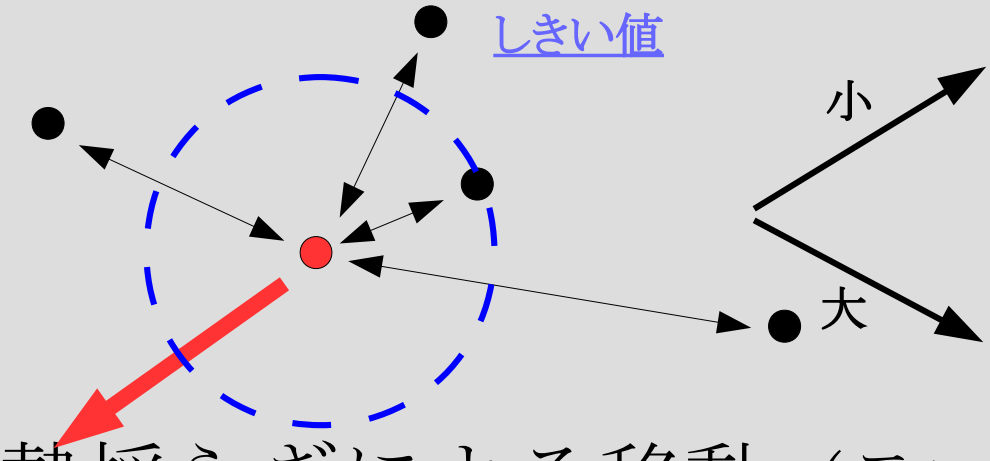
２．シミュレーション　～その１～

- 四角形の領域を用意し、その内部に粒子（電子）を一定数配置する
- 粒子が、熱揺らぎで動き回っている中でも、粒子（電子）同士が互いにクーロン斥力を及ぼし合うことで結晶化するのかを調べる

２－１．設定（動き）

動きは２種類

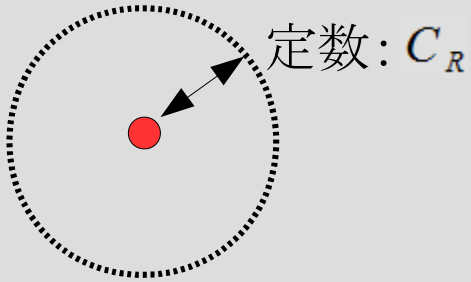
１．クーロン斥力による移動



クーロン斥力による移動
. . . なし

合力の向きに、一定量
の移動を行う C_C

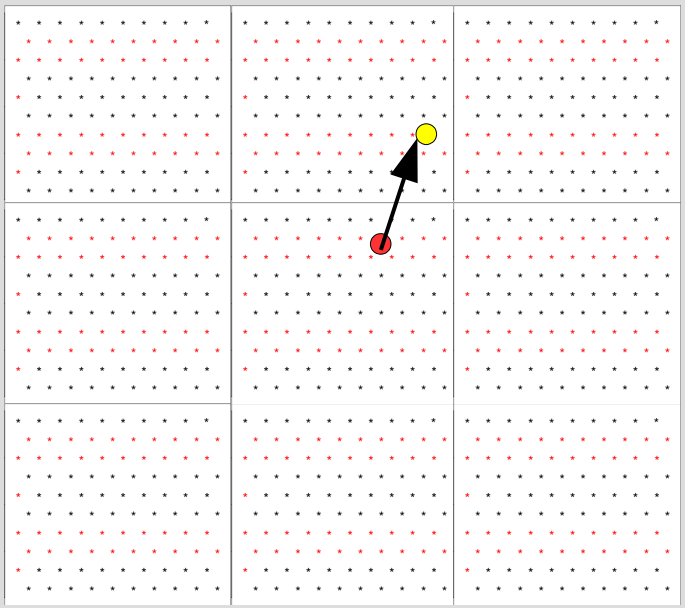
２．熱揺らぎによる移動（ランダムウォーク）



360°　全方位に対し
等確率で、距離 C_R だけ
移動する

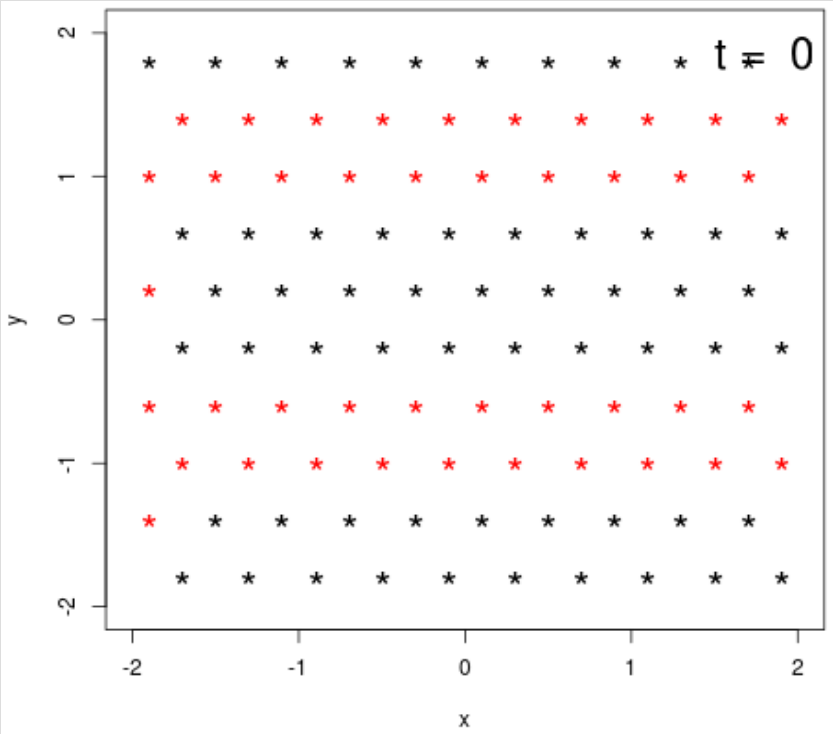
２－１．設定（配置）

- 領　域　　：１辺　４　の正方形
- 粒子数　　：１００個
- 初期配置　：三角格子
- ステップ数　：１０００



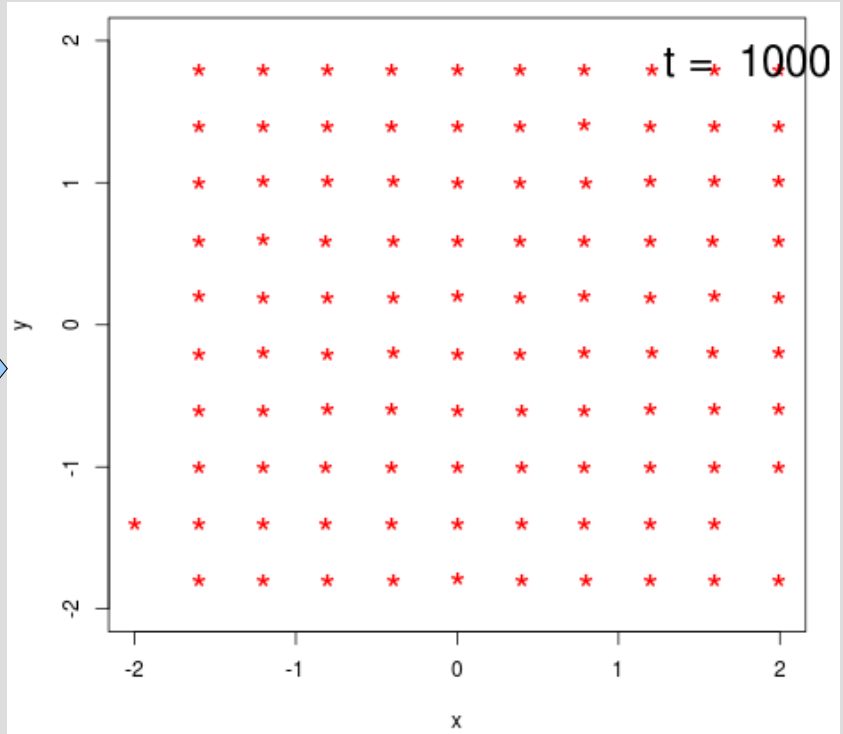
２－２．結果

初期配置（ｔ＝０）



三角格子

ｔ＝１０００



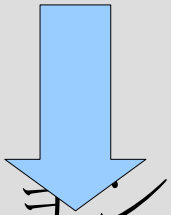
正方格子

安定配置は正方格子！？

疑問

１．電子ガスの温度を上昇させたらどうか？
⇒熱揺らぎ（ランダムウォーク）による
移動幅を大きくする

２．電子ガスの密度を変化させたらどうか？
⇒領域の大きさを変化させる



シミュレーション　～その２～

３．シミュレーション　～その２～

温度変化

- **動き**
  - クーロン斥力による移動量：C_C　（一定）
  - 熱揺らぎによる移動量　　：C_R　（変動）

⇒動きの比：C で表現　　$C = \dfrac{C_R}{C_C}$

密度変化

- **領域**
  - 横は４で固定
  - 縦は変動する

アスペクト比：h で 表 現

$$h = \frac{\text{縦}}{\text{横}\ (=4)}$$

３－２．結果

結果は５パターン

１）　線状

２）四角格子（正方格子を含む）

３）ズレ四角格子

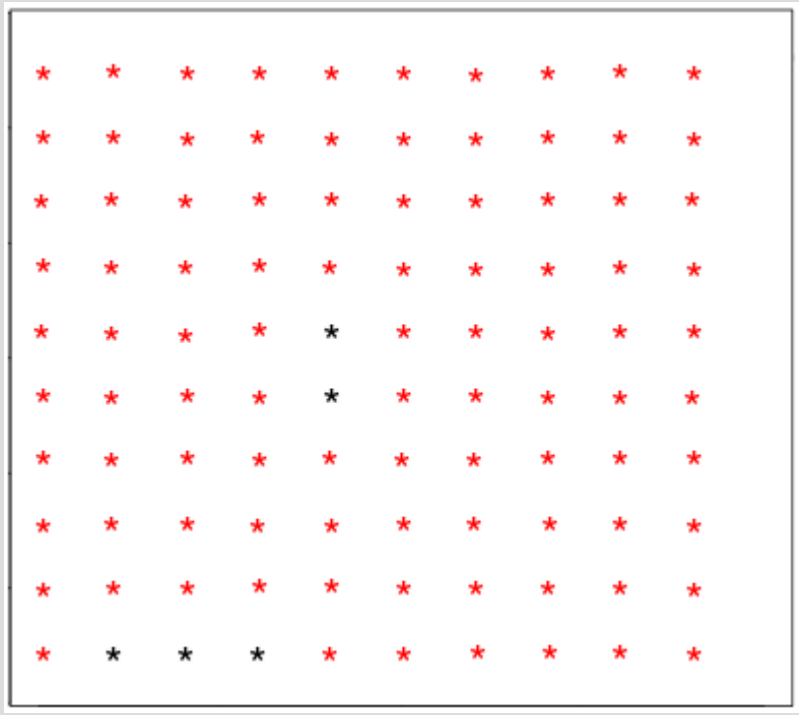
４）三角格子

５）ランダム

パターン

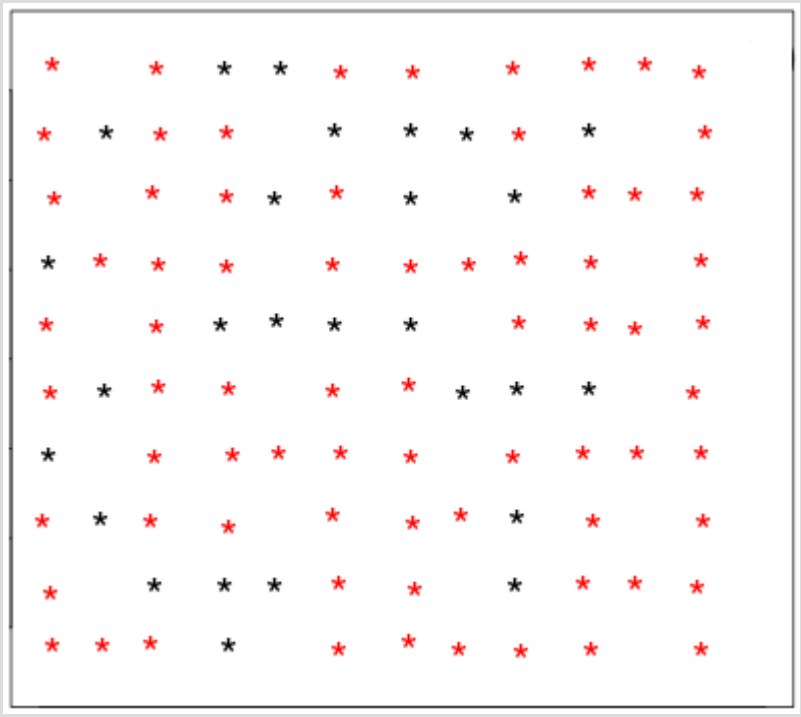
1）線状

2）四角格子

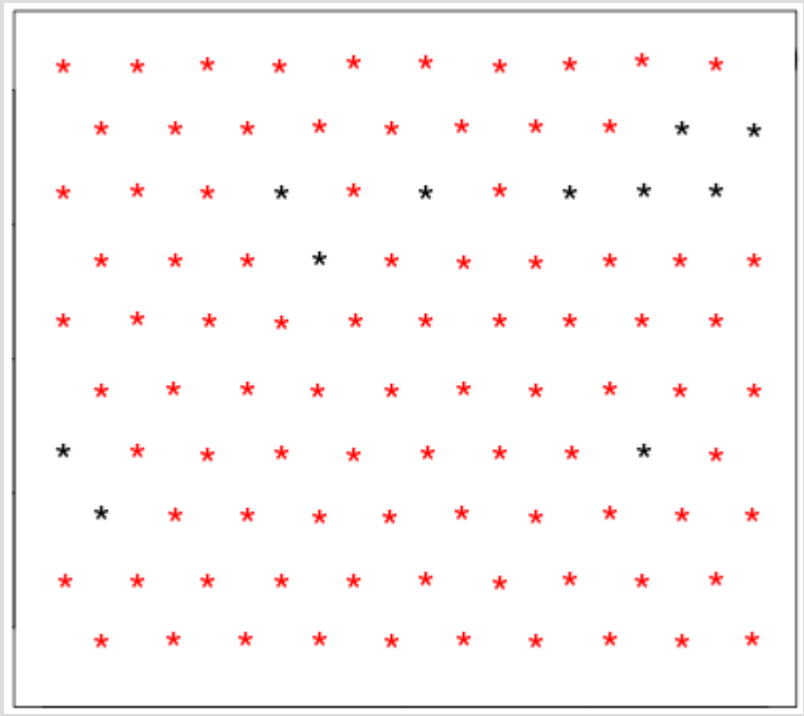


パターン

3）ズレ四角格子

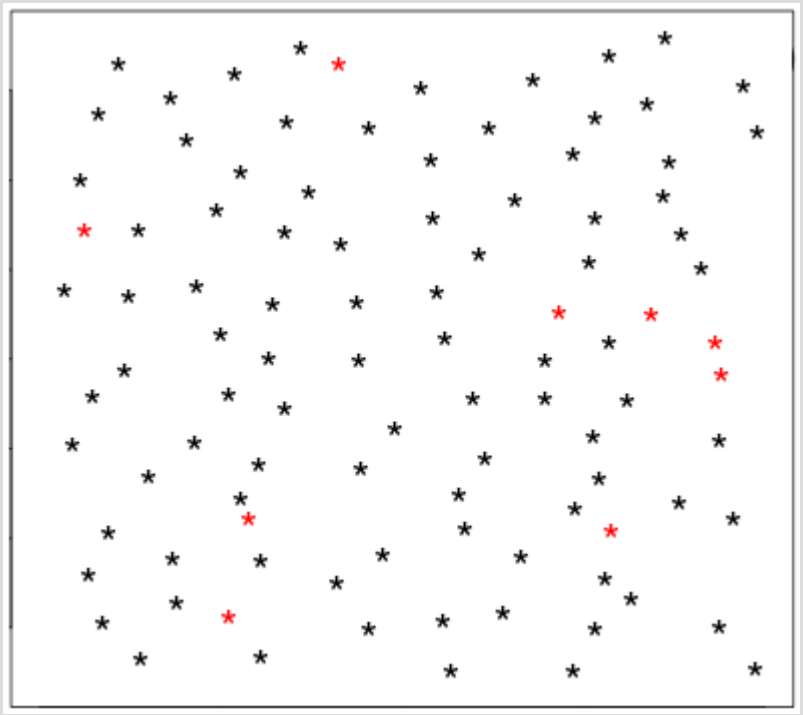


4）三角格子



パターン

5）ランダム



３－２．結果

結果は５パターン

１）　線状

２）四角格子（正方格子を含む）

３）ズレ四角格子

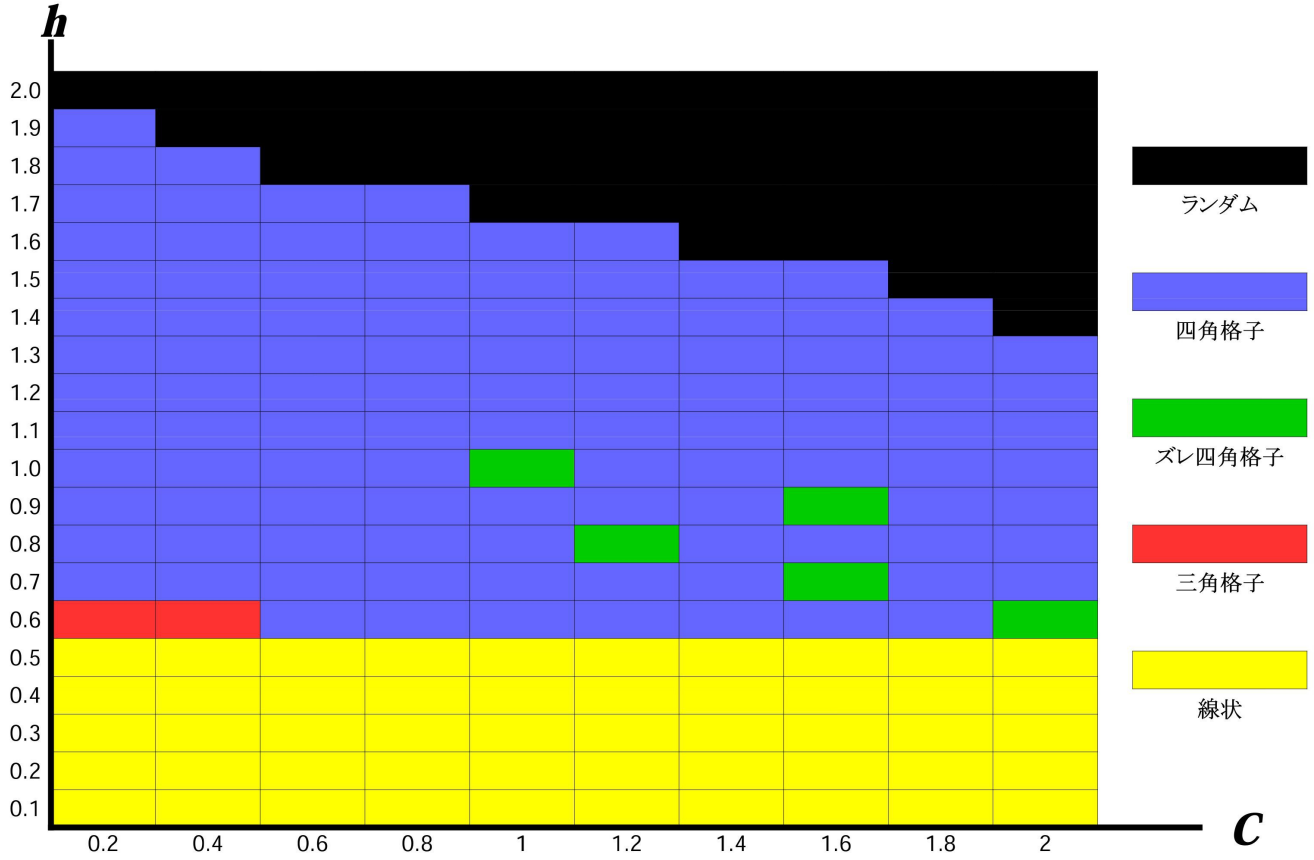
４）三角格子

５）ランダム

横軸：動きの比　$C = \frac{C_R}{C_C}$

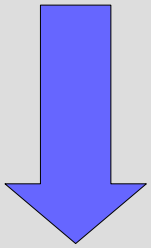
縦軸：アスペクト比　$h = \frac{\text{縦}}{\text{横}\ (=4)}$

３－２．結果（相図）


Result of the movement (t=1000)
h
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
c
ランダム
四角格子
ズレ四角格子
三角格子
線状

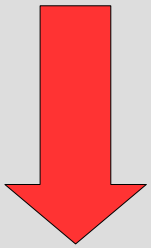
本当に正しいか？

・ステップ数：１０００まで計算を行った



しかし．．．

・時間経過と共に、安定配置へ向かっているのか？



そこで！

・ポテンシャルエネルギー：V の時間変化を調べる

４．クーロンポテンシャルエネルギー

クーロンポテンシャルエネルギー：V

$$V = \frac{1}{n}\sum_{i=1}^{n} V_i$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j\neq i}^{n} \log r_{ij}\right)$$

これを、

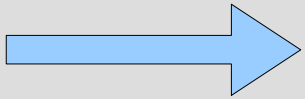
先の全てのh, Cにおいて計算する

（シミュレーション）

４．クーロンポテンシャルエネルギー

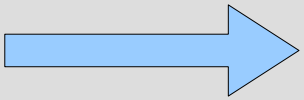
結果は、以下の３通りと予想される

１）三角格子　⇒　〇〇
　ポテンシャル：V は減少



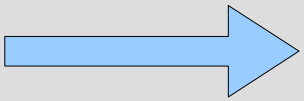
四角格子
ズレ四角格子

２）三角格子　⇒　〇〇
　ポテンシャル：V は増加

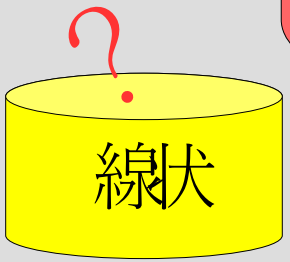


ランダム

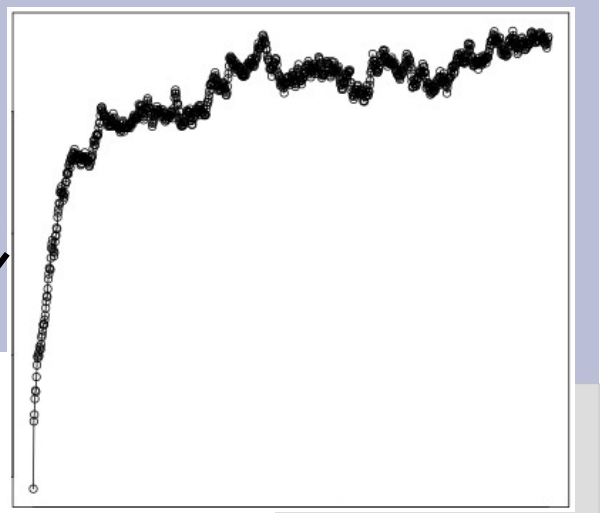
３）三角格子　⇒　三角格子
　ポテンシャル：V は変化なし



三角格子



3－1．結果（相図）

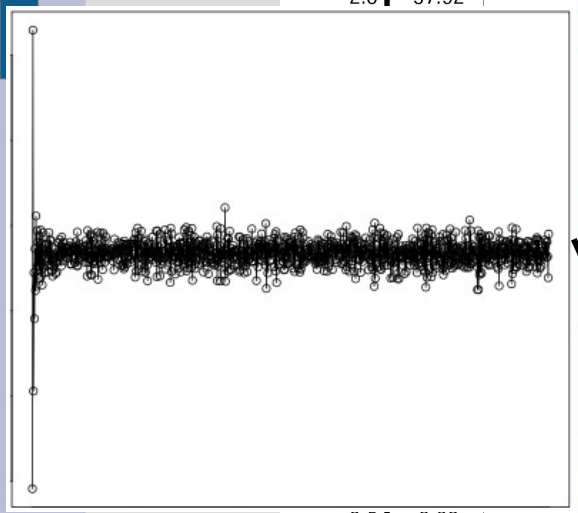


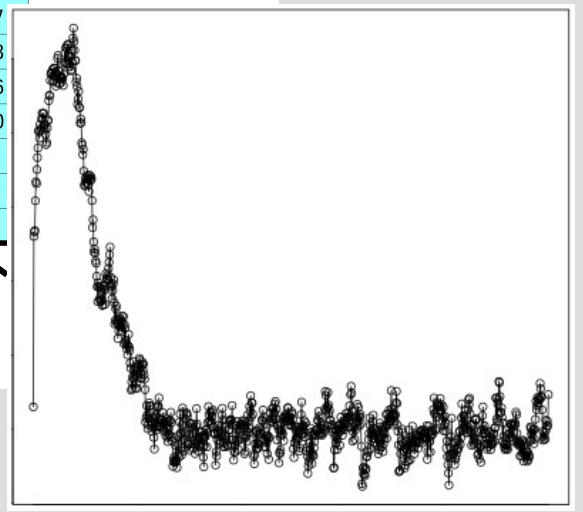
Coulomb Potential Energy

h	初期値	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
2.0	-37.92	-37.89	-37.85	-37.81	-37.80	-37.82	-37.80	-37.79	-37.80	-37.85	-37.80
		-36.28	-36.12	-36.09	-36.09	-36.08	-36.07	-36.06	-36.06	-36.05	-36.05
		-34.50	-34.50	-34.32	-34.29	-34.25	-34.25	-34.28	-34.27	-34.28	-34.32
		-32.66	-32.66	-32.66	-32.66	-32.65	-32.45	-32.42	-32.41	-32.42	-32.41
		-30.75	-30.75	-30.75	-30.75	-30.74	-30.74	-30.52	-30.52	-30.59	-30.55
		-28.78	-28.78	-28.77	-28.77	-28.76	-28.75	-28.74	-28.73	-28.55	-28.52
		-26.72	-26.72	-26.72	-26.72	-26.71	-26.70	-26.69	-26.67	-26.66	-26.45
		-24.58	-24.58	-24.58	-24.58	-24.57	-24.56	-24.55	-24.54	-24.52	-24.51
		-22.35	-22.35	-22.35	-22.34	-22.34	-22.33	-22.32	-22.31	-22.30	-22.28
		-20.02	-20.03	-20.03	-20.02	-20.02	-20.01	-20.00	-19.99	-19.97	-19.96
		-17.60	-17.60	-17.60	-17.60	-17.60	-17.58	-17.57	-17.56	-17.55	-17.53
		-15.07	-15.07	-15.07	-15.07	-15.06	-15.05	-15.04	-15.03	-15.01	-15.00
		-12.35	-12.42	-12.42	-12.42	-12.41	-12.40	-12.39	-12.38	-12.36	-12.35
		-9.64	-9.65	-9.65	-9.64	-9.64	-9.62	-9.58	-9.60	-9.59	-9.57
		-6.67	-6.67	-6.74	-6.73	-6.72	-6.71	-6.70	-6.68	-6.68	-6.68
0.5	-3.62	-3.77	-3.77	-3.77	-3.76	-3.76	-3.75	-3.73	-3.71	-3.69	-3.66
0.4	-0.44	-0.61	-0.61	-0.61	-0.60	-0.59	-0.58	-0.56	-0.54	-0.52	-0.50
0.3	2.85	2.76	2.73	2.72	2.72	2.73	2.75	2.76	2.79	2.73	2.76
0.2	6.33	6.22	6.22	6.10	6.10	6.10	6.13	6.15	6.18	6.21	6.24
0.1	10.00	9.50	9.50	9.50	9.55	9.55	9.60	9.65	9.70	9.75	9.80

減少

変化なし

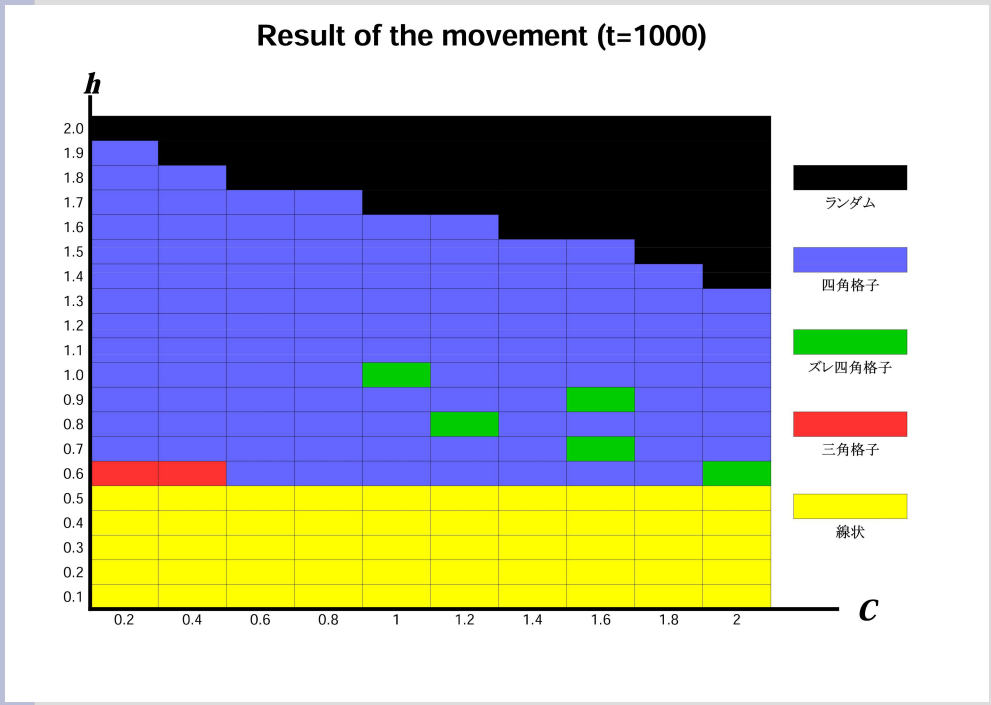


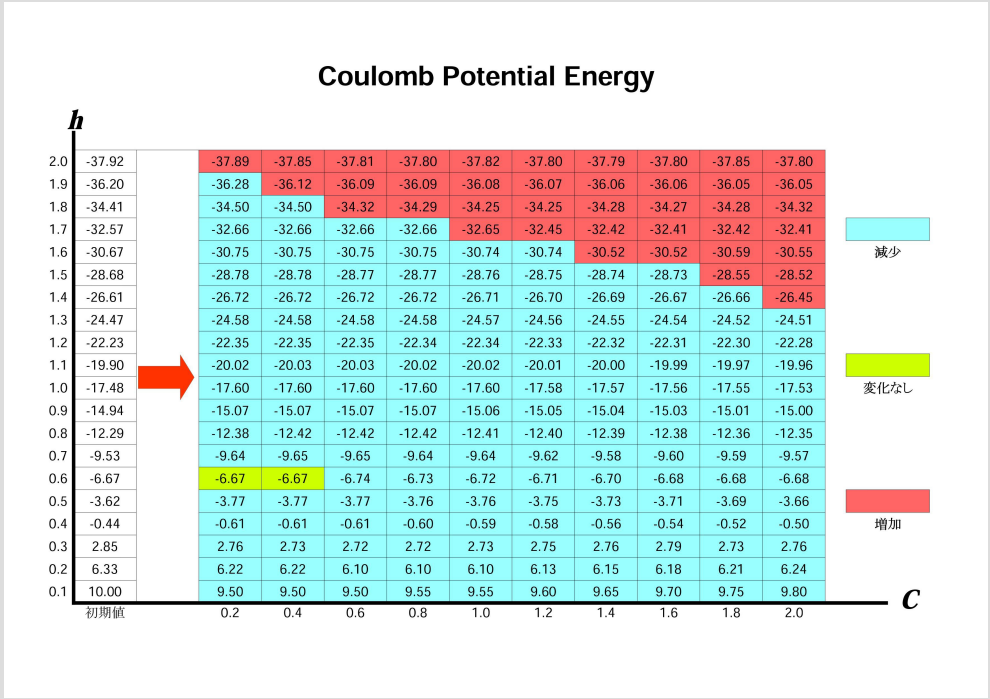


まとめ

ｔ＝１０００での配置

ポテンシャルクーロンエネルギー：V





h	初期値	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
2.0	-37.92	-37.89	-37.85	-37.81	-37.80	-37.82	-37.80	-37.79	-37.80	-37.85	-37.80
1.9	-36.20	-36.28	-36.12	-36.09	-36.09	-36.08	-36.07	-36.06	-36.06	-36.05	-36.05
1.8	-34.41	-34.50	-34.50	-34.32	-34.29	-34.25	-34.25	-34.28	-34.27	-34.28	-34.32
1.7	-32.57	-32.66	-32.66	-32.66	-32.66	-32.65	-32.45	-32.42	-32.41	-32.42	-32.41
1.6	-30.67	-30.75	-30.75	-30.75	-30.75	-30.74	-30.74	-30.52	-30.52	-30.59	-30.55
1.5	-28.68	-28.78	-28.78	-28.77	-28.77	-28.76	-28.75	-28.74	-28.73	-28.55	-28.52
1.4	-26.61	-26.72	-26.72	-26.72	-26.72	-26.71	-26.70	-26.69	-26.67	-26.66	-26.45
1.3	-24.47	-24.58	-24.58	-24.58	-24.58	-24.57	-24.56	-24.55	-24.54	-24.52	-24.51
1.2	-22.23	-22.35	-22.35	-22.35	-22.34	-22.34	-22.33	-22.32	-22.31	-22.30	-22.28
1.1	-19.90	-20.02	-20.03	-20.03	-20.02	-20.02	-20.01	-20.00	-19.99	-19.97	-19.96
1.0	-17.48	-17.60	-17.60	-17.60	-17.60	-17.60	-17.58	-17.57	-17.56	-17.55	-17.53
0.9	-14.94	-15.07	-15.07	-15.07	-15.07	-15.06	-15.05	-15.04	-15.03	-15.01	-15.00
0.8	-12.29	-12.38	-12.42	-12.42	-12.42	-12.41	-12.40	-12.39	-12.38	-12.36	-12.35
0.7	-9.53	-9.64	-9.65	-9.65	-9.64	-9.64	-9.62	-9.58	-9.60	-9.59	-9.57
0.6	-6.67	-6.67	-6.67	-6.74	-6.73	-6.72	-6.71	-6.70	-6.68	-6.68	-6.68
0.5	-3.62	-3.77	-3.77	-3.77	-3.76	-3.76	-3.75	-3.73	-3.71	-3.69	-3.66
0.4	-0.44	-0.61	-0.61	-0.61	-0.60	-0.59	-0.58	-0.56	-0.54	-0.52	-0.50
0.3	2.85	2.76	2.73	2.72	2.72	2.73	2.75	2.76	2.79	2.73	2.76
0.2	6.33	6.22	6.22	6.10	6.10	6.10	6.13	6.15	6.18	6.21	6.24
0.1	10.00	9.50	9.50	9.50	9.55	9.55	9.60	9.65	9.70	9.75	9.80

まとめ

- 四角格子の箇所は、ポテンシャルも低く安定しており、ウィグナー結晶であると予想される
- 高温・低密度における、ランダムな配置では、ポテンシャルも高くなっており、結晶化しているとは言えない
- 結晶化は、低温・中密度の時に起きる

今後の課題

- h- C グラフの所々に出てきた「ズレ四角格子」の解明
- ポテンシャル V に関するグラフの、最初の数ステップのばらつきの理由づけ
- C の値と、実際の温度との関連付け
- 高密度状態における、線状配置の安定性の正体
- より高速に計算できる言語の使用

```
#実行は"GS_14.R"

# ------------------------------------------------------------------ #
# 初期配置を三角格子にした #
# 各粒子の相互作用効果範囲を、全粒子に広げる #
# シミュレーション領域のアスペクト比 "h" を変える #

# ------------------------------------------------------------------ #
# [試行回数(時間)]、[粒子数] の設定 #

args <- commandArgs()
if(is.na(args[6]) == T){
   h <- 1
} else {
   h <- as.real(args[6])
}
if(is.na(args[7]) == T){
   time <- 1000
} else {
   time <- as.integer(args[7])
}
if(is.na(args[8]) == T){
   m <- 0
} else {
   m <- as.integer(args[8]) - 1
}
if(is.na(args[9]) == T){
   nn <- 10
} else {
   nn <- as.integer(args[9])
}

cat((m+1),"/",nn,"回目","¥n")
print(Sys.time())

n_START <- 1
go_R_START <- 0.002 * n_START
go_R_STEP <- 0.002
START_TIME <- rep(0,length=(nn+1))
stock <- proc.time()
START_TIME[1] <- stock[3]
lap <- rep(0,length=2)

n <- 100
const <- 1
go_e <- 0.01
const2 <- 0.05

# ------------------------------------------------------------------ #
# width : 移動可能範囲の設定 #

width <- 2
length <- width * h
x_max <- width
x_min <- -width
y_max <- length
y_min <- -length

n2 <- sqrt(n)
r2 <- 2*width/n2
x2 <- r2
x2_sub <- r2/4
y2 <- 2*length/n2
y2_sub <- y2/2

f_criteria <- const/(x2)

scale_time <- 0:time

# ------------------------------------------------------------------ #
# 粒子の位置(x,y)の設定 #

go_R <- go_R_START + go_R_STEP * m
scale_sub <- go_R/go_e

x <- matrix(0:0,n,(time+1))
y <- matrix(0:0,n,(time+1))
x_cp <- matrix(0:0,n,8)
y_cp <- matrix(0:0,n,8)
V <- rep(0,length=(time+1))

# ------------------------------------------------------------------ #
# 粒子の初期値の設定 #
```



```
for(i in 1:n2){
   for(j in 1:n2){
      if(i%%2==1){
         x[(i-1)*n2+j,1] <- x_min + x2_sub + (j-1)*x2
         y[(i-1)*n2+j,1] <- y_max - (y2_sub + (i-1)*y2)
      }else{
         x[(i-1)*n2+j,1] <- x_min + x2_sub*3 + (j-1)*x2
         y[(i-1)*n2+j,1] <- y_max - (y2_sub + (i-1)*y2)
      }
   }
}

# ------------------------------------------------------------------ #
# 時刻tにおける移動距離、力の設定 #
# f_x : 各粒子間における相互の x 方向の力 #
# f_y : 各粒子間における相互の y 方向の力 #
# f[A,(1~2)] : (A...i番目の粒子、B...1はx,2はy) に働く力の大きさ #
# f_i : i 番目の粒子に働く力の合計の大きさ abs{sum{f_i}} #
# e_r : クーロン力の影響によって移動したか(1 or 0) #

f_x <- matrix(0,n,n)
f_y <- matrix(0,n,n)
f <- matrix(0,n,2,byrow=T)
f_i <- rep(0:0,length=n)
e_r <- matrix(0,n,(time+1),byrow=T)

# ------------------------------------------------------------------ #
# 時刻tにおける粒子間の力の計算 #

for(k in 1:time){
   f_x <- matrix(0:0,n,n)
   f_y <- matrix(0:0,n,n)
   f <- matrix(0,n,2,byrow=T)
   for(i in 1:n){
      x_cp[i,1] <- x[i,k] - 2*width
      y_cp[i,1] <- y[i,k] + 2*length
      x_cp[i,2] <- x[i,k]
      y_cp[i,2] <- y[i,k] + 2*length
      x_cp[i,3] <- x[i,k] + 2*width
      y_cp[i,3] <- y[i,k] + 2*length
      x_cp[i,4] <- x[i,k] - 2*width
      y_cp[i,4] <- y[i,k]
      x_cp[i,5] <- x[i,k] + 2*width
      y_cp[i,5] <- y[i,k]
      x_cp[i,6] <- x[i,k] - 2*width
      y_cp[i,6] <- y[i,k] - 2*length
      x_cp[i,7] <- x[i,k]
      y_cp[i,7] <- y[i,k] - 2*length
      x_cp[i,8] <- x[i,k] + 2*width
      y_cp[i,8] <- y[i,k] - 2*length
   }
   for(i in 1:(n-1)){
      for(j in (i+1):n){
         r <- sqrt((x[i,k]-x[j,k])^2 + (y[i,k]-y[j,k])^2)
         f_x[i,j] <- const/(r^2) * (x[i,k]-x[j,k])
         f_y[i,j] <- const/(r^2) * (y[i,k]-y[j,k])
         f_x[j,i] <- const/(r^2) * (x[j,k]-x[i,k])
         f_y[j,i] <- const/(r^2) * (y[j,k]-y[i,k])
         for(l in 1:8){
            if(r > sqrt((x[i,k]-x_cp[j,l])^2 + (y[i,k]-y_cp[j,l])^2)){
               r <- sqrt((x[i,k]-x_cp[j,l])^2 + (y[i,k]-y_cp[j,l])^2)
               f_x[i,j] <- const/(r^2) * (x[i,k]-x_cp[j,l])
               f_y[i,j] <- const/(r^2) * (y[i,k]-y_cp[j,l])
               f_x[j,i] <- const/(r^2) * (x_cp[j,l]-x[i,k])
               f_y[j,i] <- const/(r^2) * (y_cp[j,l]-y[i,k])
               check <- l
            }
         }
         V[k] <- V[k] - log(r)
      }
   }
   V[k] <- V[k]/n

# ------------------------------------------------------------------ #
# 時刻tにおける粒子のおける力の合計、移動 #

   random_1 <- runif(n)
   for(i in 1:n){
      for(j in 1:n){
         f[i,1] <- f[i,1] + f_x[i,j]
         f[i,2] <- f[i,2] + f_y[i,j]
      }
      f_i[i] <- sqrt(f[i,1]^2 + f[i,2]^2)
# 粒子間反発力 #
```

```
      if(f_i[i] > f_criteria){
         e_r[i,k] <- 1
         x[i,k+1] <- x[i,k] + go_e*f[i,1]/f_i[i]
         y[i,k+1] <- y[i,k] + go_e*f[i,2]/f_i[i]
      }else{
         x[i,k+1] <- x[i,k]
         y[i,k+1] <- y[i,k]
      }
# ランダムウォーク #
      x[i,k+1] <- x[i,k+1] + go_R*cos(random_1[i]*2*pi)
      y[i,k+1] <- y[i,k+1] + go_R*sin(random_1[i]*2*pi)
      x_cp[i,1] <- x[i,k+1] - 2*width
      y_cp[i,1] <- y[i,k+1] + 2*length
      x_cp[i,2] <- x[i,k+1]
      y_cp[i,2] <- y[i,k+1] + 2*length
      x_cp[i,3] <- x[i,k+1] + 2*width
      y_cp[i,3] <- y[i,k+1] + 2*length
      x_cp[i,4] <- x[i,k+1] - 2*width
      y_cp[i,4] <- y[i,k+1]
      x_cp[i,5] <- x[i,k+1] + 2*width
      y_cp[i,5] <- y[i,k+1]
      x_cp[i,6] <- x[i,k+1] - 2*width
      y_cp[i,6] <- y[i,k+1] - 2*length
      x_cp[i,7] <- x[i,k+1]
      y_cp[i,7] <- y[i,k+1] - 2*length
      x_cp[i,8] <- x[i,k+1] + 2*width
      y_cp[i,8] <- y[i,k+1] - 2*length
      if(x[i,k+1] > x_max){
         x[i,k+1] <- x_min + (x[i,k+1] - x_max)
      }else if(x[i,k+1] < x_min){
         x[i,k+1] <- x_max - (x_min - x[i,k+1])
      }
      if(y[i,k+1] > y_max){
         y[i,k+1] <- y_min + (y[i,k+1] - y_max)
      }else if(y[i,k+1] < y_min){
         y[i,k+1] <- y_max - (y_min - y[i,k+1])
      }
   }
}
for(i in 1:(n-1)){
   for(j in (i+1):n){
      r <- sqrt((x[i,k]-x[j,k])^2 + (y[i,k]-y[j,k])^2)
      for(l in 1:8){
         if(r > sqrt((x[i,k]-x_cp[j,l])^2 + (y[i,k]-y_cp[j,l])^2)){
            r <- sqrt((x[i,k]-x_cp[j,l])^2 + (y[i,k]-y_cp[j,l])^2)
         }
      }
      V[time+1] <- V[time+1] - log(r)
   }
}
V[time+1] <- V[time+1]/n
# ------------------------------------------------------------------ #
# .pngファイルへの出力 #

file_name <- sprintf("GS_14_%02d.png",m+1)
title <- paste(" h = ",h,"  , C = ",scale_sub)
png(file_name)
plot(scale_time,V,xlab="time",ylab="V_potential",type="o",main=title)
dev.off()

# ------------------------------------------------------------------ #
# 計算時間の計測 #

stock <- proc.time()
START_TIME[m+2] <- stock[3]
lap[1] <- START_TIME[m+2] - START_TIME[m+1]
lap[2] <- START_TIME[m+2] - START_TIME[1]
for(i in 1:2){
   hour <- 0
   minute <- 0
   second <- 0
   m_second <- 0
   if(lap[i] >= 3600){
      hour <- as.integer(lap[i]/3600)
      lap[i] <- lap[i] - hour * 3600
   }
   if(lap[i] >= 60){
      minute <- as.integer(lap[i]/60)
      lap[i] <- lap[i] - minute * 60
   }
   second <- as.integer(lap[i])
   m_second <- (lap[i] - second) * 1000
   if(i == 1) cat("lap : ",hour,"[h]",minute,"[m]",second,"[s]",m_second,"[ms]","¥t¥t")
   if(i == 2) cat("elapsed : ",hour,"[h]",minute,"[m]",second,"[s]",m_second,"[ms]","¥n")
```



```
}
print(Sys.time())
cat("¥n")

if((m+1) == 1) system("Rscript GS_14.R 1.7 1000 2")
if((m+1) == 2) system("Rscript GS_14.R 1.7 1000 3")
if((m+1) == 3) system("Rscript GS_14.R 1.7 1000 4")
if((m+1) == 4) system("Rscript GS_14.R 1.7 1000 5")
if((m+1) == 5) system("Rscript GS_14.R 1.7 1000 6")
if((m+1) == 6) system("Rscript GS_14.R 1.7 1000 7")
if((m+1) == 7) system("Rscript GS_14.R 1.7 1000 8")
if((m+1) == 8) system("Rscript GS_14.R 1.7 1000 9")
if((m+1) == 9) system("Rscript GS_14.R 1.7 1000 10")
```