Aluno: João Victor Alcantara Pimenta 11820812

Professor: Francisco Castilho Alcaraz

# 1 Prolegômenos

Simularemos um modelo de gás bidimensional composto por $N$ moléculas. Sua interação será aproximada pelo potencial de Lennard-Jones,

$$V(r) = 4\epsilon \left[ \left(\frac{\sigma}{r}\right)^1 2 - \left(\frac{\sigma}{r}\right)^6 \right]$$

Sabemos da dinâmica newtoniana que

$$a_{i,x} = \frac{1}{m} \sum_{i \neq k=1}^{N} f_{k,i} \cos(\theta_{k,i})$$

$$a_{i,y} = \frac{1}{m} \sum_{i \neq k=1}^{N} f_{k,i} \sin(\theta_{k,i})$$

Onde, do potencial, fica claro que

$$f_{k,i} = 24 \left( \frac{2}{r_{k,i}^{13}} - \frac{1}{r_{k,i}^{7}} \right)$$

Onde $r$ é a distância usual entre as partículas. Mais especificamente, o algoritmo iterativo que usaremos em nosso programa serão

$$x_i(n+1) = 2x_i(n) - x_i(n-1) + a_{i,x}(n)(\Delta t)^2$$

$$y_i(n+1) = 2y_i(n) - y_i(n-1) + a_{i,y}(n)(\Delta t)^2$$

De condições iniciais

$$x_i(1) = x(0) + v_{i,x}(0)\Delta t, \quad y_i(1) = y(0) + v_{i,y}(0)\Delta t$$

Para as velocidades usaremos

$$v_{i,x}(n) = \frac{x_i(n+1) - x_i(n-1)}{2\Delta}$$

$$v_{i,y}(n) = \frac{y_i(n+1) - y_i(n-1)}{2\Delta}$$

Para todos os programas, consideraremos $L$ o tamanho da caixa (em unidades de $\sigma$) e $N$ o número de partículas.

Alguns pontos,

- A condição inicial será desordenada mas as partículas são alocadas em células igualmente dividas da grade com algum ruído de posicionamento;
- A condição de contorno será periódica;
- Não consideramos interação de partículas muito distantes;

# 2 O algoritmo

Dadas as premissas do programa podemos discutir implementação.

## 2.1 SETUPBODYS

Definiremos brevemente uma função que define a posição inicial e velocidade inicial das partículas na malha. A forma que a função faz isso é a discutida, divida a malha em células posiciona-se cada partícula em uma célula com um deslocamento aleatório interno à célula.

A velocidade total é constante para todas e é dada em uma direção aleatória.

## 2.2 SAVECONDITIONS

Esta função serve somente para atualizar os vetores que armazenam as posições, atualizando quem guarda cada passo temporal.

## 2.3 UPDATECONDITIONS

Esta função coordena o processo de atualizar as condições do sistema. Inicialmente, ela chama uma função auxiliar chamada $CALCULATEACC$ que retorna as condições de aceleração em cada partícula devidamente atualizadas.

Com as acelerações podemos atualizar as posições e velocidades da forma já descrita no algoritmo.

## 2.4 CALCULATEACC

Feita para definir a aceleração de cada partícula, esta função precisa de uma chamada ainda para ter seus parâmetros. A função faz o cálculo da contribuição de aceleração sobre cada partícula.

## 2.5 CALCULATEFORCE

Esta função calcula a distância entre partícula e a utiliza para, seguindo o algoritmo, calcular a força sobre as partículas. Utiliza-se a simetria da dinâmica de Newton para otimizar o cálculo.

## 2.6 ENERGY

A função de energia calcula a energia, não para o algoritmo, mas para confirmar a validade da conservação da grandeza.

## 2.7 O código

```
      SUBROUTINE SETUPBODYS(VMAX)
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02,PI=3.1415926535897)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2)
     +      ,DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS

            ICOEF = CEILING(SQRT(DBLE(NBODY)))
            BSIZE = SIZEL/ICOEF

            DO I=1,NBODY
C                 POSITION IN LATTICE
                  X = MOD(I,ICOEF)*(BSIZE) + BSIZE/2
                  Y = (CEILING(DBLE(I)/DBLE(ICOEF))) * (BSIZE) - BSIZE/2
C                 DEFINE ANGLE
                  ANGLE=2*PI*RAND()
                  SIZE=BSIZE/4
C                 DEFINE POSITION
                  OLD(-2,I,1,1)=X+SIZE*COS(ANGLE)
                  OLD(-2,I,1,2)=Y+SIZE*SIN(ANGLE)
C                 DEFINE ANGLE
                  ANGLE=2*PI*RAND()
C                 RANDOM VELOCITY
                  OLD(-2,I,2,1)=VMAX*COS(ANGLE)
                  OLD(-2,I,2,2)=VMAX*SIN(ANGLE)
            END DO

C           Set old new conditions using velocity
            DO I=1,NBODY
            OLD(-1,I,1,1)=OLD(-2,I,1,1)+OLD(-2,I,2,1)*DT
            OLD(-1,I,1,2)=OLD(-2,I,1,2)+OLD(-2,I,2,2)*DT
            OLD(-1,I,2,1)=OLD(-2,I,2,1)
            OLD(-1,I,2,2)=OLD(-2,I,2,2)
            END DO

      END SUBROUTINE SETUPBODYS

      SUBROUTINE SAVECONDITIONS()
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2)
     +      ,DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS

            DO I=1,NBODY
                  DO J=1,2
                        DO K=1,2
                              OLD(-2,I,J,K)=OLD(-1,I,J,K)
                              OLD(-1,I,J,K)=CONDITIONS(I,J,K)
                        END DO
                  END DO
            END DO
```

```
      END SUBROUTINE SAVECONDITIONS

      SUBROUTINE CALCULATEFORCE()
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +FS(1:NBODY,1:NBODY),RS(1:NBODY,1:NBODY),DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /DISTS/ RS

            DO I=1,NBODY
                  DO K=I+1,NBODY
c                       DISTANCE BETWEEN BODYS in x(circular conditions)
                        RX = OLD(-1,K,1,1) - OLD(-1,I,1,1)
                        XF =  INT(FLOOR(2*(RX+SIZEL)/SIZEL)-1.5)
                        RX = RX - SIZEL*XF
                        DISTS(I,K,1) = RX
c                       DISTANCE BETWEEN BODYS in y(circular conditions)
                        RY = OLD(-1,K,1,2) - OLD(-1,I,1,2)
                        YF =  INT(FLOOR(2*(RY+SIZEL)/SIZEL)-1.5)
                        RY = RY - SIZEL*YF
                        DISTS(I,K,2) = RY
c                       DISTANCE BETWEEN BODYS
                        RS(I,K) = SQRT(RX**2+RY**2)
c                       K-I ~ I-K
                        RS(K,I) = RS(I,K)
                        DISTS(K,I,1) = - DISTS(I,K,1)
                        DISTS(K,I,2) = - DISTS(I,K,2)
                  END DO
            END DO

            DO I=1,NBODY
                  DO K=I+1,NBODY
c                       ignore if rs > 3
                        IF (RS(I,K) .GT. 3) THEN
                              FS(I,K) = 0
                              FS(K,I) = 0
                              CYCLE
                        END IF
                        FS(I,K)= 24*((2/(RS(I,K)**13))-(1/(RS(I,K)**7)))
                        FS(K,I) = FS(I,K)
                  END DO
            END DO

      END SUBROUTINE CALCULATEFORCE

      SUBROUTINE CALCULATEACC()
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02, P=1, PI=3.1415)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +   ACC(1:NBODY,2), FS(1:NBODY,1:NBODY), DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /ACCELERATIONS/ ACC
            COMMON /FORCES/ FS

            CALL CALCULATEFORCE()

            DO I=1,NBODY
                  ACC(I,1)=0
                  ACC(I,2)=0
                  DO K=1,NBODY
```

```
                        IF (I.EQ.K) GOTO 10
                        FAC = 1
C                       Angle in circular conditions
                        ANGLE = ATAN(DISTS(I,K,2)/DISTS(I,K,1))
                        FAC = SIGN(FAC,DISTS(I,K,1))
C                       IN X
                        ACC(I,1) = ACC(I,1) - FS(I,K)*COS(ANGLE)*FAC
C                       IN Y
                        ACC(I,2) = ACC(I,2) - FS(I,K)*SIN(ANGLE)*FAC
 10                     CONTINUE
                    END DO
                    ACC(I,1) = ACC(I,1)/P
                    ACC(I,2) = ACC(I,2)/P
                END DO

      END SUBROUTINE CALCULATEACC

      SUBROUTINE UPDATECONDITIONS()
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      ACC(1:NBODY,2), DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /ACCELERATIONS/ ACC

            CALL CALCULATEACC()

            DO I=1,NBODY
c                   UPDATE POSITION
                    AUXX = 2*OLD(-1,I,1,1)-OLD(-2,I,1,1)+ACC(I,1)*DT**2
                    AUXY = 2*OLD(-1,I,1,2)-OLD(-2,I,1,2)+ACC(I,2)*DT**2

c                   UPDATE VELOCITY
                    CONDITIONS(I,2,1)=(AUXX-OLD(-2,I,1,1))/(2*DT)
                    CONDITIONS(I,2,2)=(AUXY-OLD(-2,I,1,2))/(2*DT)

c                   CIRCULAR CONDITIONS
                    sigx = -FLOOR(AUXX/SIZEL)
                    sigy = -FLOOR(AUXY/SIZEL)

                    CONDITIONS(I,1,1) = AUXX + sigx*SIZEL
                    CONDITIONS(I,1,2) = AUXY + sigy*SIZEL

c                   CIRCULAR CONDITION ON OLD POSITION
                    OLD(-1,I,1,1) = OLD(-1,I,1,1) + sigx*SIZEL
                    OLD(-1,I,1,2) = OLD(-1,I,1,2) + sigy*SIZEL

            END DO

      END SUBROUTINE UPDATECONDITIONS

      SUBROUTINE CALCULATEENERGY(ENERGY)
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02,P=1)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      ACC(1:NBODY,2), DISTS(1:NBODY,1:NBODY,1:2),
     +      ENERGY, RS(1:NBODY,1:NBODY)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

            ENERGY = 0

```

```
            DO 10 I=1,NBODY
                  ENERGY = ENERGY +
C                 KINETIC ENERGY
     +            0.5*P*(CONDITIONS(I,2,1)**2 + CONDITIONS(I,2,2)**2)
                  DO 20 K=I+1,NBODY
C                       ignore if rs > 3
                        IF (RS(I,K) .GT. 3) CYCLE
C                       POTENTIAL ENERGY
                        ENERGY = ENERGY + 4*((1/(RS(I,K)**12))-
     +                  (1/(RS(I,K)**6)))
 20                   END DO
 10          END DO

      END SUBROUTINE CALCULATEENERGY
```

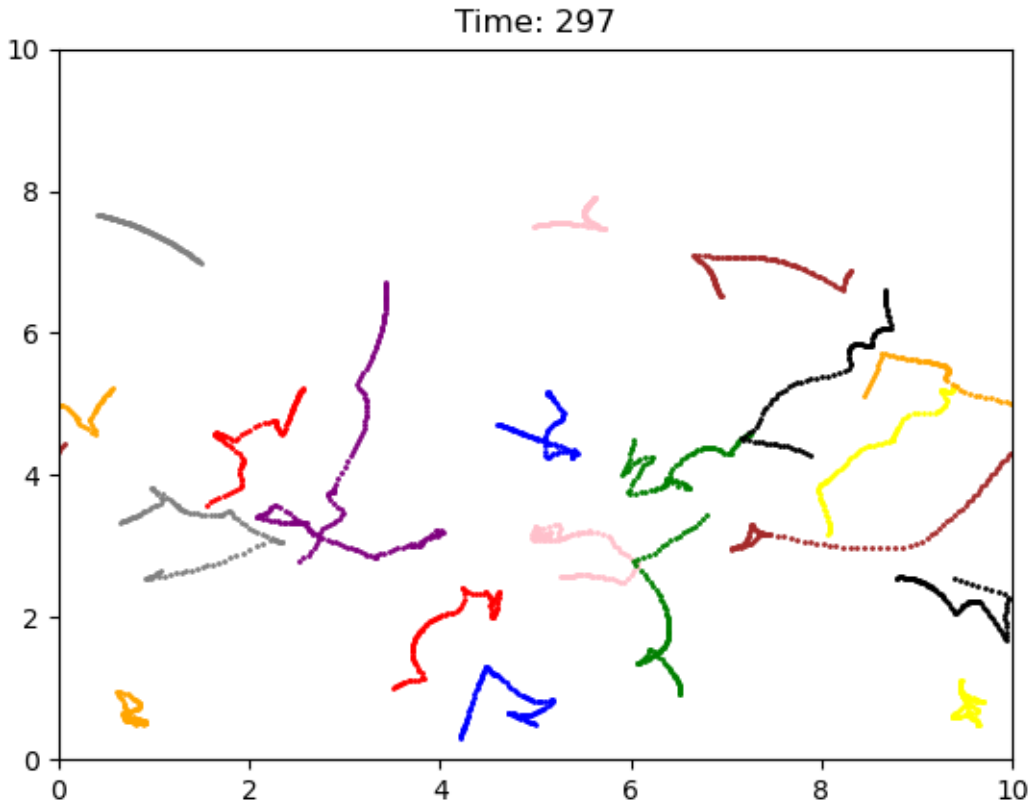
# 3 Tarefa A

Tomaremos $L = 10$,$N = 20$,$dt = 0.02$,$v_0 = 1$. Nosso programa principal será

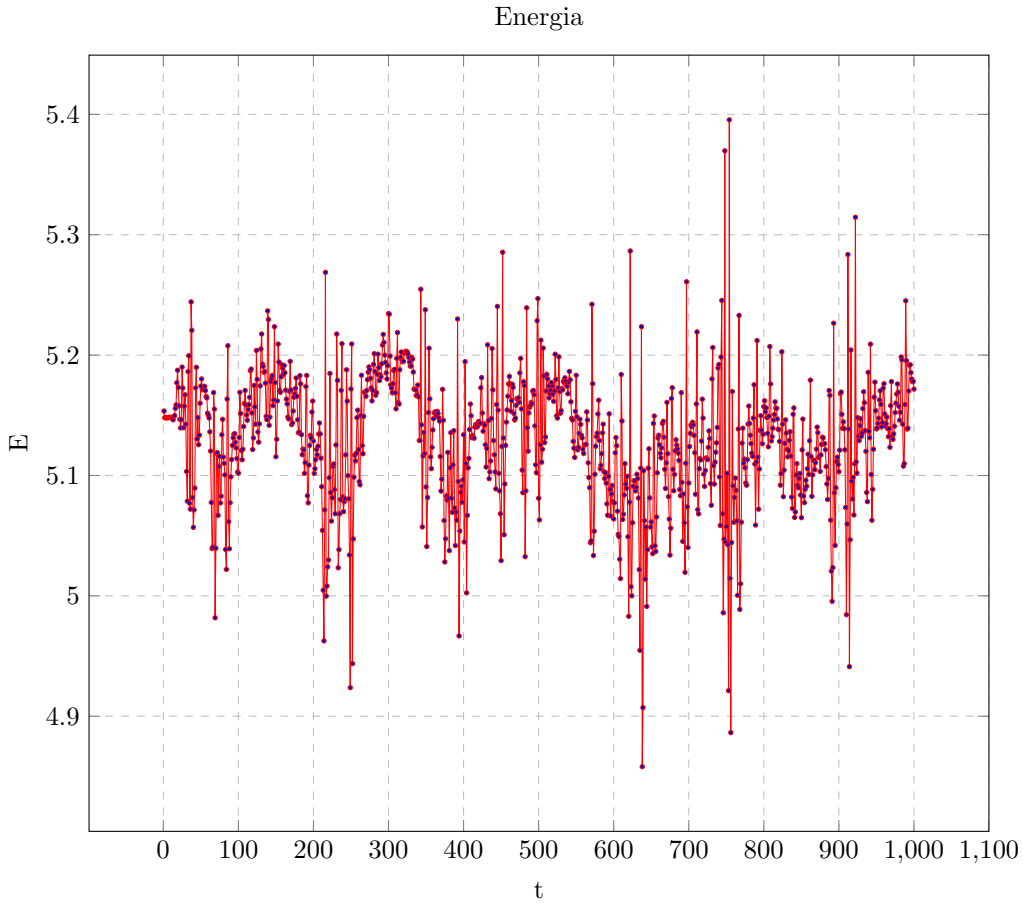
```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

C           FILE TO SAVE THE DATA
            OPEN(UNIT=1,FILE='DATAx.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=2,FILE='DATAy.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=3,FILE='ENERGY.DAT',STATUS='UNKNOWN')

c           INITIALIZE THE RANDOM NUMBER GENERATOR
            CALL SRAND(1)

C           INCITIALIZE OUR SYSTEM CONDITIONS
            VMAX = 1
            CALL SETUPBODYS(VMAX)

C           START THE TIME LOOP
            DO ITIME=1,10000

C                 CALCULATE THE NEW CONDITIONS
                  CALL UPDATECONDITIONS()

C                 WRITE THE NEW CONDITIONS
                  WRITE(*,*) ITIME
                  DO I=1,NBODY
                        WRITE(1,*) CONDITIONS(I,1,1)
                        WRITE(2,*) CONDITIONS(I,1,2)
                  END DO

C                 SAVE THE OLD CONDITIONS
                  CALL SAVECONDITIONS()
```

```
39
40  c                 Calculate energy
41                    CALL CALCULATEENERGY(ENERGY)
42
43                    WRITE(3,*) ITIME,ENERGY
44
45              END DO
46
47        END PROGRAM VERLET
```

Vamos avaliar os caminhos que nossas partículas seguiram. Para os primeiros momentos, antes que o gráfico fique muito carregado, podemos visualizar em uma imagem estática:




Mas para completeza, teremos o arquivo de animação da situação descrita disponível na pasta do projeto e guardado na subpasta da tarefa $A$. Podemos avaliar nossa conservação de energia para reafirmar a validade da simulação. Se colocarmos em um gráfico pelo tempo:




Tudo parece razoável para a dinâmica das partículas.

# 4 Tarefa B

Queremos avaliar a distribuição de velocidades das partículas no nosso sistema. Mantendo as condições de inicialização anteriores da tarefa $A$ sabemos que nossas partículas inicializaram com a velocidade 1 e direção aleatória. Com o tempo, espera-se que a distribuição das velocidades seja de alguma forma similar à distribuição de velocidades de Boltzmann dada por

$$P(v) \approx \frac{v^2}{K_B T} \exp\left(-\frac{mv^2}{2K_B T}\right)$$

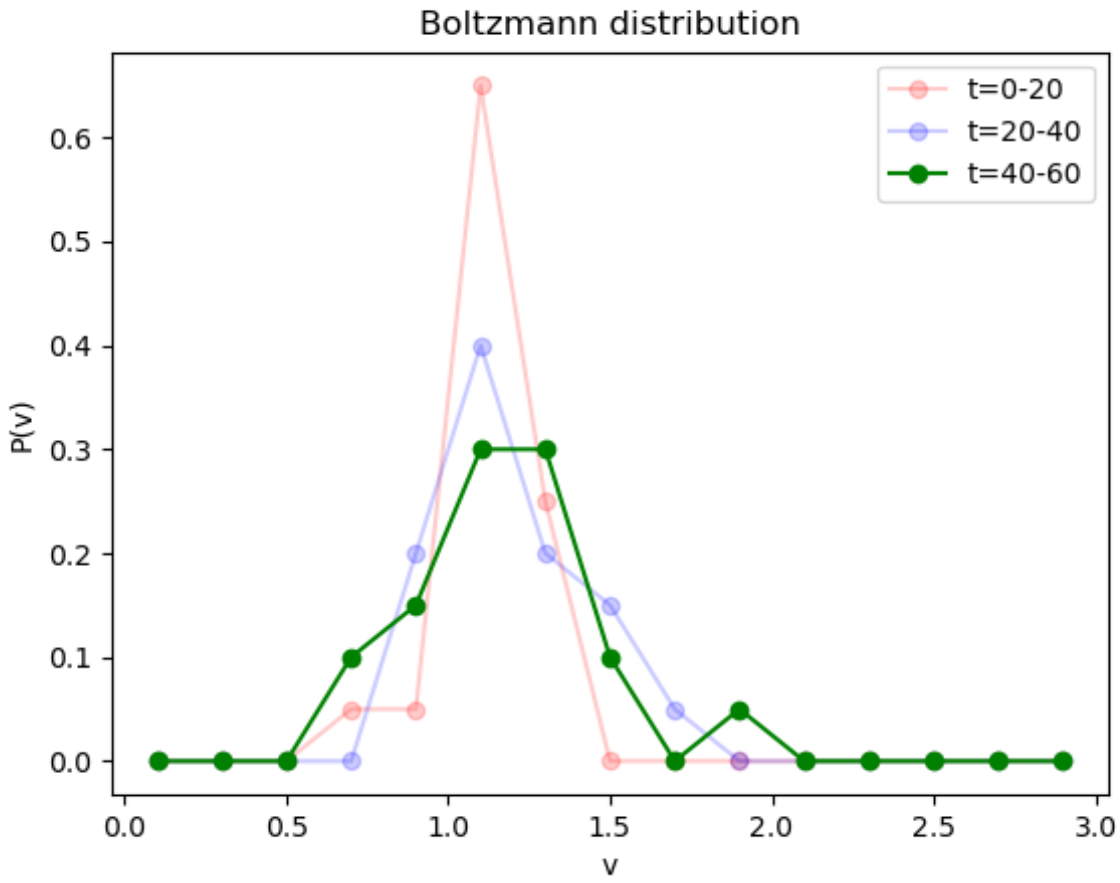
$$P(v_{x,y}) \approx \frac{1}{\sqrt{K_B T}} \exp\left(-\frac{mv_{x,y}^2}{2K_B T}\right)$$

Escrevemos nosso programa,

```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=100,SIZEL=10,DT=0.0001)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
```

```
8             COMMON /FORCES/ FS
9             COMMON /ACCELERATIONS/ ACC
10            COMMON /DISTS/ RS
11
12    C       FILE TO SAVE THE DATA
13            OPEN(UNIT=1,FILE='DATAx.DAT',STATUS='UNKNOWN')
14            OPEN(UNIT=2,FILE='DATAy.DAT',STATUS='UNKNOWN')
15            OPEN(UNIT=3,FILE='ENERGY.DAT',STATUS='UNKNOWN')
16            OPEN(UNIT=4,FILE='VELOCIDADEX.DAT',STATUS='UNKNOWN')
17            OPEN(UNIT=5,FILE='VELOCIDADEY.DAT',STATUS='UNKNOWN')
18            OPEN(UNIT=7,FILE='VELOCIDADE.DAT',STATUS='UNKNOWN')
19
20    c       INITIALIZE THE RANDOM NUMBER GENERATOR
21            CALL SRAND(123)
22
23    C       INCITIALIZE OUR SYSTEM CONDITIONS
24            VMAX = 1
25            CALL SETUPBODYS(VMAX)
26
27    C       START THE TIME LOOP
28            DO ITIME=1,10000
29
30    C             CALCULATE THE NEW CONDITIONS
31                  CALL UPDATECONDITIONS()
32
33    C             WRITE THE NEW CONDITIONS
34                  WRITE(*,*) ITIME
35
36                  DO I=1,NBODY
37                  WRITE(1,*) CONDITIONS(I,1,1)
38                  WRITE(2,*) CONDITIONS(I,1,2)
39                  WRITE(4,*) OLD(-1,I,2,1)
40                  WRITE(5,*) OLD(-1,I,2,2)
41                  WRITE(7,*) SQRT(OLD(-1,I,2,1)**2 + OLD(-1,I,2,2)**2)
42                  END DO
43
44    C             SAVE THE OLD CONDITIONS
45                  CALL SAVECONDITIONS()
46
47    c             Calculate energy
48                  CALL CALCULATEENERGY(ENERGY)
49
50                  WRITE(3,*) ITIME,ENERGY
51
52            END DO
53
54        END PROGRAM VERLET
```

Agora, tendo as velocidades para os momentos da simulação salvos, podemos fazer a visualização das curvas. Para a velocidade total, plotamos




Que para um sistema de 20 partículas parece bem razoavelmente distribuído como Boltzmann de média em torno do esperado. Faremos análises mais precisas sobre a forma quando determinada a temperatura em uma tarefa seguinte.

Podemos repetir o gráfico para as velocidades em $x$ e $y$ que terão distribuições bem parecidas.

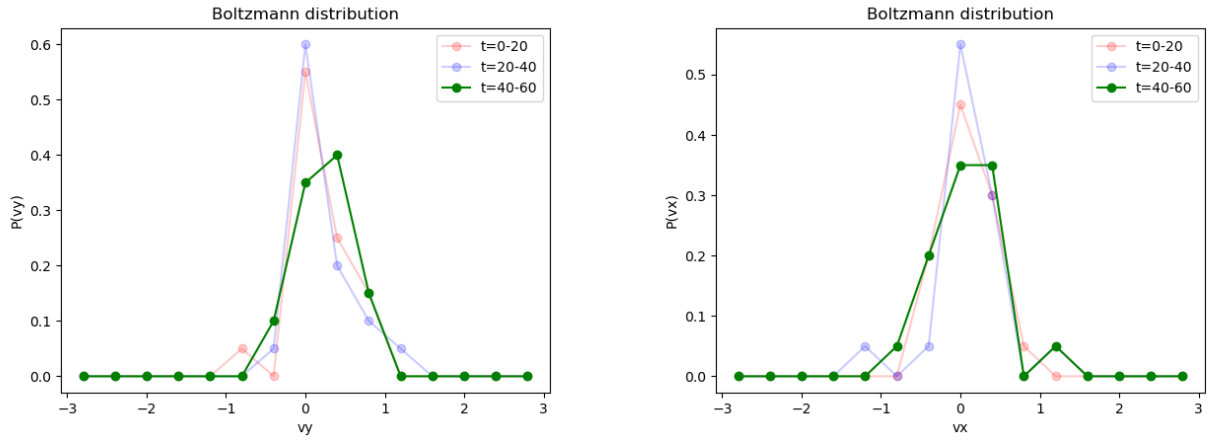



Figure 1: Visualização da distribuição de velocidades

Que lembram as distribuições esperadas.

# 5 Tarefa C

Para esta tarefa modificaremos a função de $SETUP$ de tal forma que dê metade das partículas a velocidade em x e outra metade a velocidade em y.

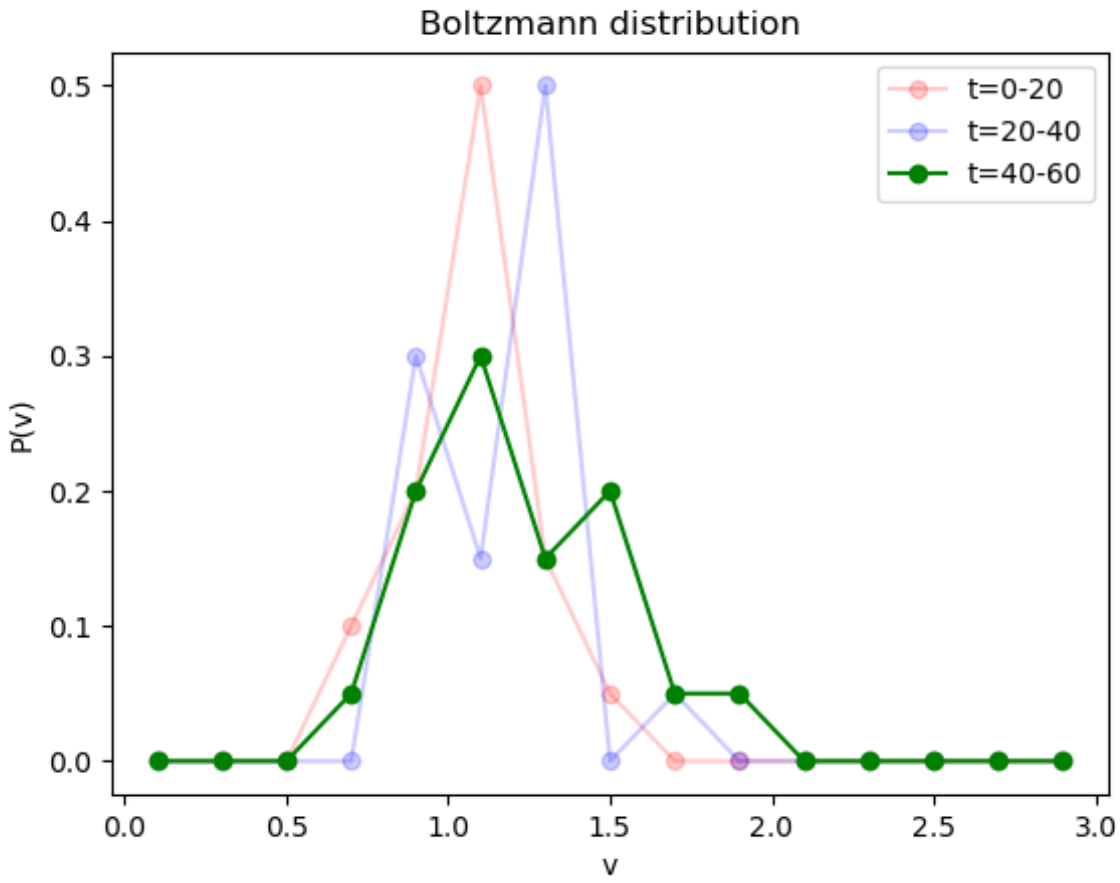
Fazemos isso para confirmar se ainda assim teríamos a convergência observada na tarefa anterior para a distribuição de Boltzmann. Sendo esta uma distribuição inicial tão não natural que, funcionando para esta, espera-se que funcione para múltiplas situação não convencionais e valha certa generalidade.

Escrevemos com uma alteração na inicialização

```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

C           FILE TO SAVE THE DATA
            OPEN(UNIT=1,FILE='DATAx.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=2,FILE='DATAy.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=3,FILE='ENERGY.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=4,FILE='VELOCIDADEX.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=5,FILE='VELOCIDADEY.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=7,FILE='VELOCIDADE.DAT',STATUS='UNKNOWN')


c           INITIALIZE THE RANDOM NUMBER GENERATOR
            CALL SRAND(123)

C           INCITIALIZE OUR SYSTEM CONDITIONS
            VMAX = 1
            CALL SETUPBODYS(VMAX)

C           START THE TIME LOOP
            DO ITIME=1,10000

C                 CALCULATE THE NEW CONDITIONS
                  CALL UPDATECONDITIONS()

C                 WRITE THE NEW CONDITIONS
                  WRITE(*,*) ITIME

                  DO I=1,NBODY
                        WRITE(1,*) CONDITIONS(I,1,1)
                        WRITE(2,*) CONDITIONS(I,1,2)
                        WRITE(4,*) OLD(-1,I,2,1)
                        WRITE(5,*) OLD(-1,I,2,2)
                  WRITE(7,*) SQRT(OLD(-1,I,2,1)**2 + OLD(-1,I,2,2)**2)
                  END DO

C                 SAVE THE OLD CONDITIONS
                  CALL SAVECONDITIONS()

c                 Calculate energy
```

```
                  CALL CALCULATEENERGY(ENERGY)

                  WRITE(3,*) ITIME,ENERGY

            END DO

      END PROGRAM VERLET

      SUBROUTINE SETUPBODYS(VMAX)
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02,PI=3.1415926535897)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2)
     +      ,DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS

            ICOEF = CEILING(SQRT(DBLE(NBODY)))
            BSIZE = SIZEL/ICOEF

            DO I=1,NBODY
C                 POSITION IN LATTICE
                  X = MOD(I,ICOEF)*(BSIZE) + BSIZE/2
                  Y = (CEILING(DBLE(I)/DBLE(ICOEF))) * (BSIZE) - BSIZE/2
C                 DEFINE ANGLE
                  ANGLE=2*PI*RAND()
                  SIZE=BSIZE/4
C                 DEFINE POSITION
                  OLD(-2,I,1,1)=X+SIZE*COS(ANGLE)
                  OLD(-2,I,1,2)=Y+SIZE*SIN(ANGLE)
C                 DEFINE ANGLE
                  vx = MOD(I,2) * VMAX
                  vy = (1 - MOD(I,2)) * VMAX
C                 RANDOM VELOCITY
                  OLD(-2,I,2,1) = vx
                  OLD(-2,I,2,2) = vy
            END DO

C           Set old new conditions using velocity
            DO I=1,NBODY
            OLD(-1,I,1,1)=OLD(-2,I,1,1)+OLD(-2,I,2,1)*DT
            OLD(-1,I,1,2)=OLD(-2,I,1,2)+OLD(-2,I,2,2)*DT
            OLD(-1,I,2,1)=OLD(-2,I,2,1)
            OLD(-1,I,2,2)=OLD(-2,I,2,2)
            END DO

      END SUBROUTINE SETUPBODYS
```

Podemos visualizar no nosso novo cenário.




Que para um sistema de 20 partículas parece bem razoavelmente distribuído como Boltzmann de média em torno do esperado.

Podemos repetir o gráfico para as velocidades em $x$ e $y$ que terão distribuições bem parecidas.

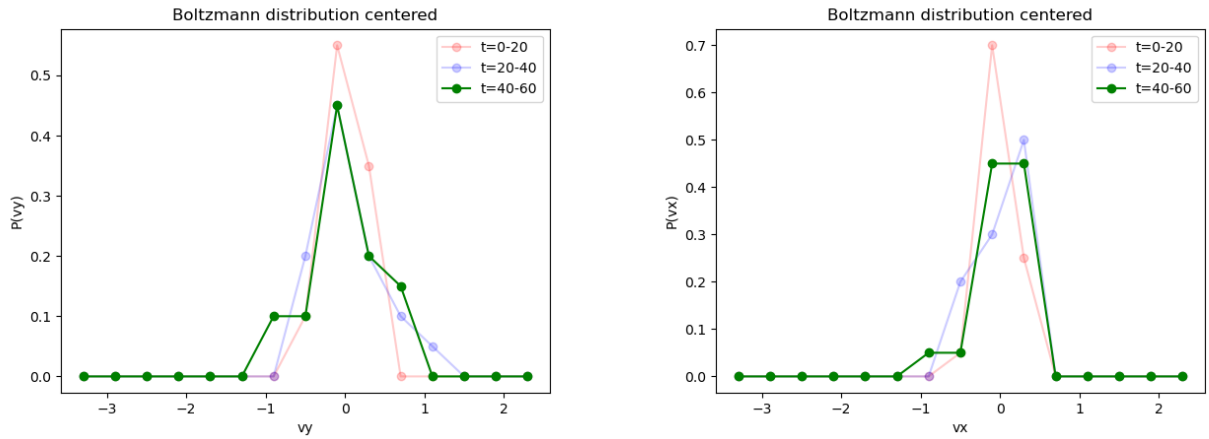



Figure 2: Visualização da distribuição de velocidades

Retornamos à distribuição mesmo com a inicialização dada de curiosa forma.

# 6 Tarefa D

Para esta tarefa basta deixar o sistema entrar em equilíbrio e determinar $k_BT$ definido por

$$K_BT = \langle \frac{1}{m}(v_x^2 + v_y^2) \rangle$$

Escrevemos então o nosso programa principal,

```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY=20,SIZEL=10,DT=0.02)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            REAL*16 kbt
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

c           INITIALIZE THE RANDOM NUMBER GENERATOR
            CALL SRAND(123)

C           INCITIALIZE OUR SYSTEM CONDITIONS
            VMAX = 1
            CALL SETUPBODYS(VMAX)

C           START THE TIME LOOP
            DO ITIME=1,100000

C                 CALCULATE THE NEW CONDITIONS
                  CALL UPDATECONDITIONS()

C                 SAVE THE OLD CONDITIONS
                  CALL SAVECONDITIONS()

c                 Calculate energy
                  CALL CALCULATEENERGY(ENERGY)

            END DO

c           write KBT = <m/2 * (vx^2 + vy^2)>
            kbt = 0
            DO I=1,NBODY
                  kbt = kbt+(OLD(-1,I,2,1)**2 + OLD(-1,I,2,2)**2)/NBODY
                  WRITE(*,*) kbt
            END DO
            kbt = (kbt)/2

            WRITE(*,*) kbt

      END PROGRAM VERLET
```

Calculando para os cenários em $B$ e $C$ temos

Em $C$

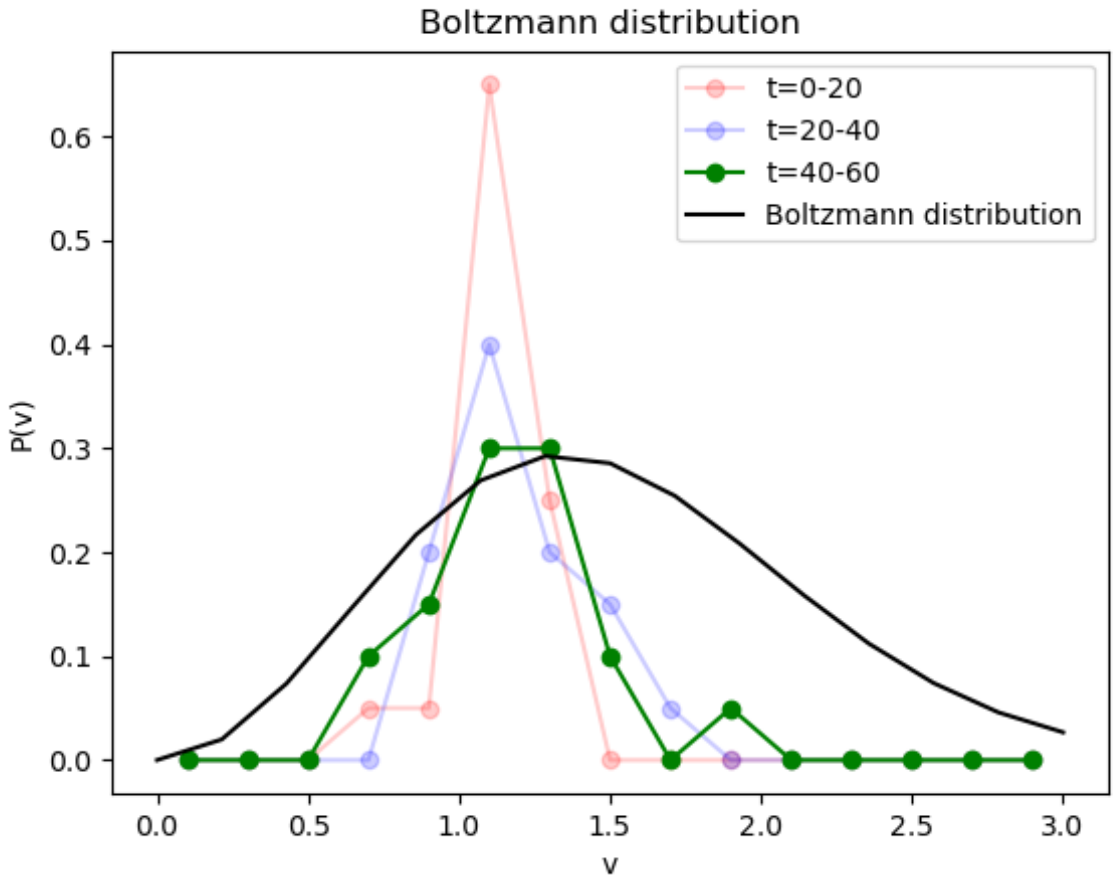
$$K_B T \approx 0.957$$

Em $B$

$$K_B T \approx 0.898$$

queremos visualizar como o fit destas curvas se comportam com os dados encontrados. Podemos plotar a curva de boltzmann esperada junto com a distribuição de velocidades medida.

## 6.1 Em B




Neste cenário a distribuição de velocidades das partícula parece concordar bem com a distribuição esperada para as partículas na temperatura medida do reservatório.

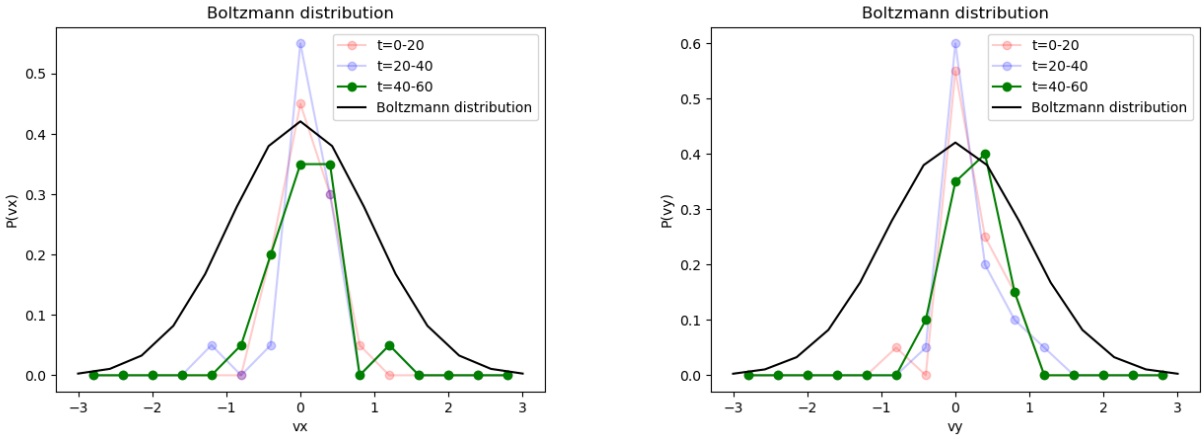
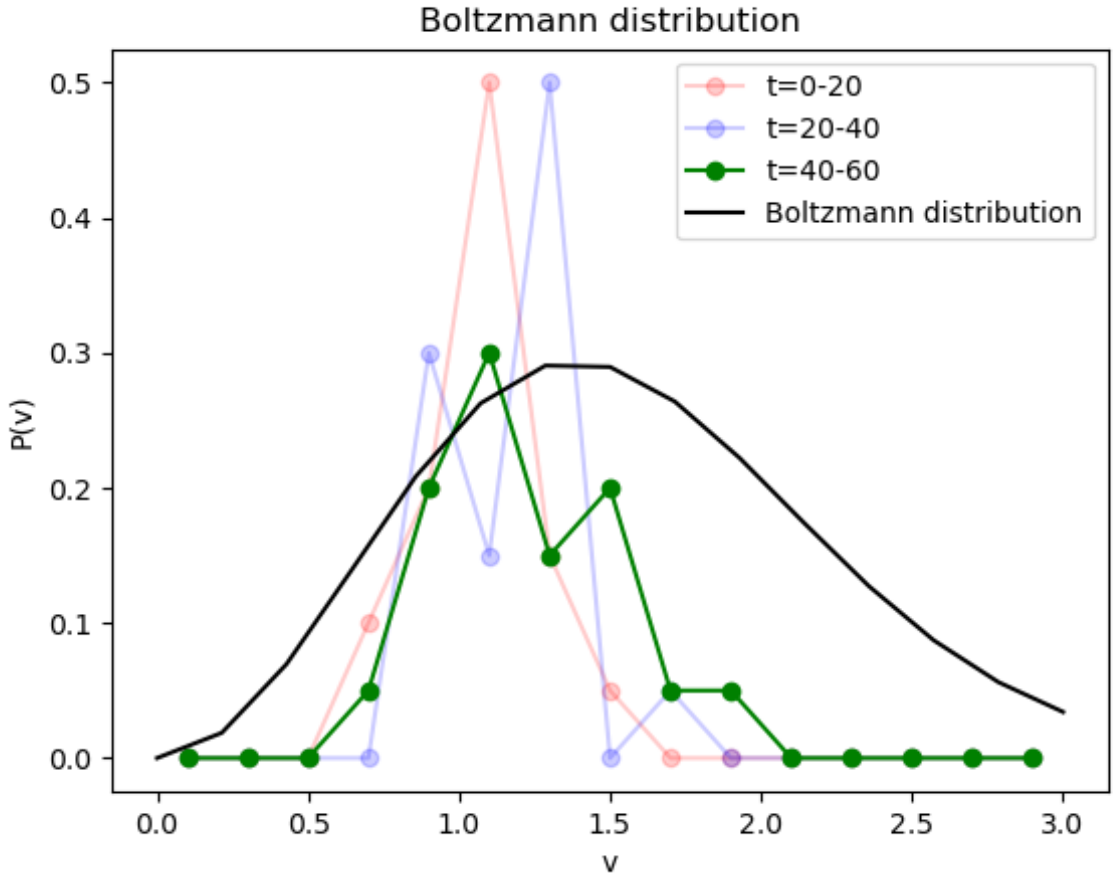



Figure 3: Visualização da distribuição de velocidades

## 6.2 Em C




Neste cenário a distribuição de velocidades das partícula parece concordar bem com a distribuição esperada para as partículas na temperatura medida do reservatório.

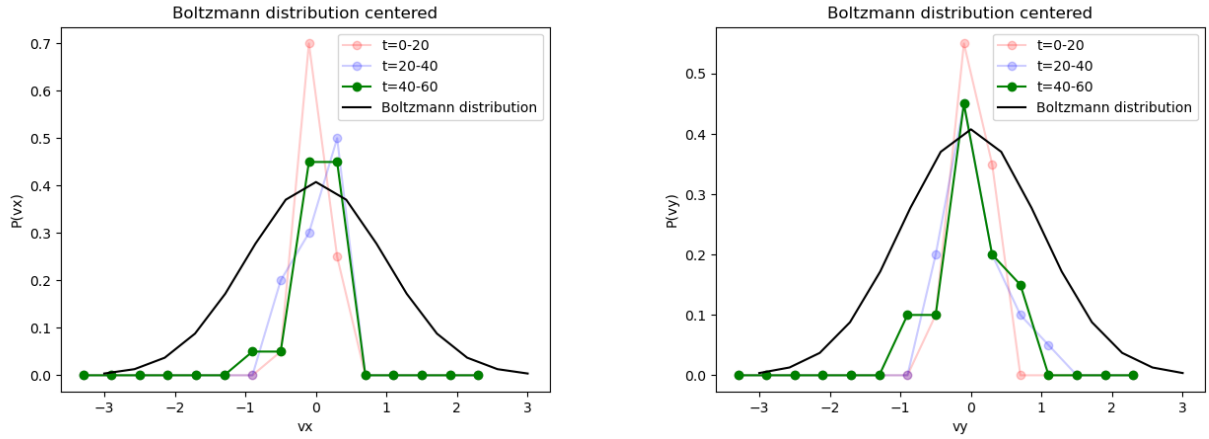



Figure 4: Visualização da distribuição de velocidades

# 7 Tarefa E

Para simularmos uma situação de alta densidade faremos uma mudança nos nosso parâmetros. Tomaremos $N = 16$, $L = 4$, $dt = 0.005$ e manteremos $v_0 = 1$.

Escrevemos,

```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY= 16,SIZEL= 4,DT=0.005)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

C           FILE TO SAVE THE DATA
            OPEN(UNIT=1,FILE='DATAx.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=2,FILE='DATAy.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=3,FILE='ENERGY.DAT',STATUS='UNKNOWN')

c           INITIALIZE THE RANDOM NUMBER GENERATOR
            CALL SRAND(1)

C           INCITIALIZE OUR SYSTEM CONDITIONS
            VMAX = 0.5
            CALL SETUPBODYS(VMAX)

C           START THE TIME LOOP
            DO ITIME=1,10000

C                 CALCULATE THE NEW CONDITIONS
                  CALL UPDATECONDITIONS()

C                 WRITE THE NEW CONDITIONS
                  WRITE(*,*) ITIME
                  DO I=1,NBODY
                        WRITE(1,*) CONDITIONS(I,1,1)
                        WRITE(2,*) CONDITIONS(I,1,2)
```

```
                    END DO

C                   SAVE THE OLD CONDITIONS
                    CALL SAVECONDITIONS()

c                   Calculate energy
                    CALL CALCULATEENERGY(ENERGY)

                    WRITE(3,*) ITIME,ENERGY

                END DO

        END PROGRAM VERLET
```

Podemos visualizar para os períodos especificados como as partículas estão se comportando.

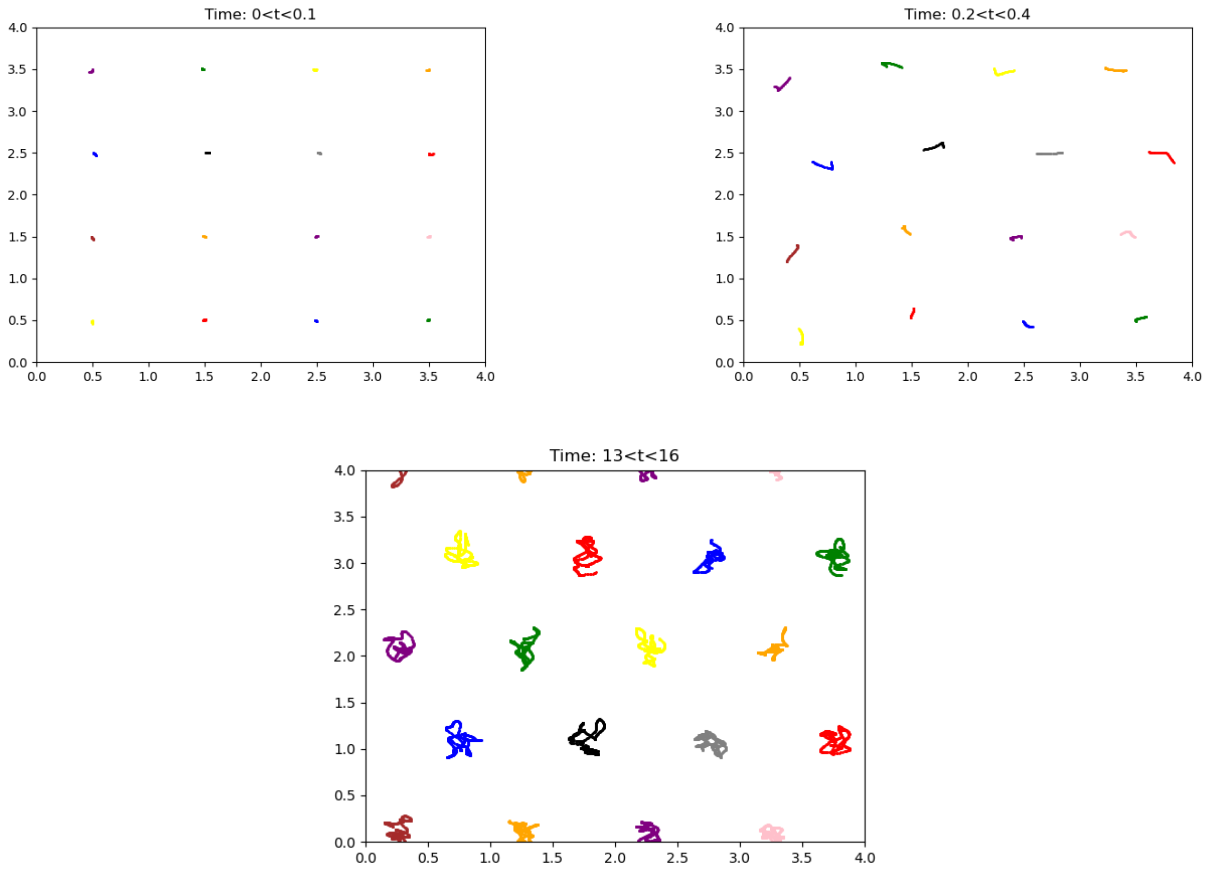
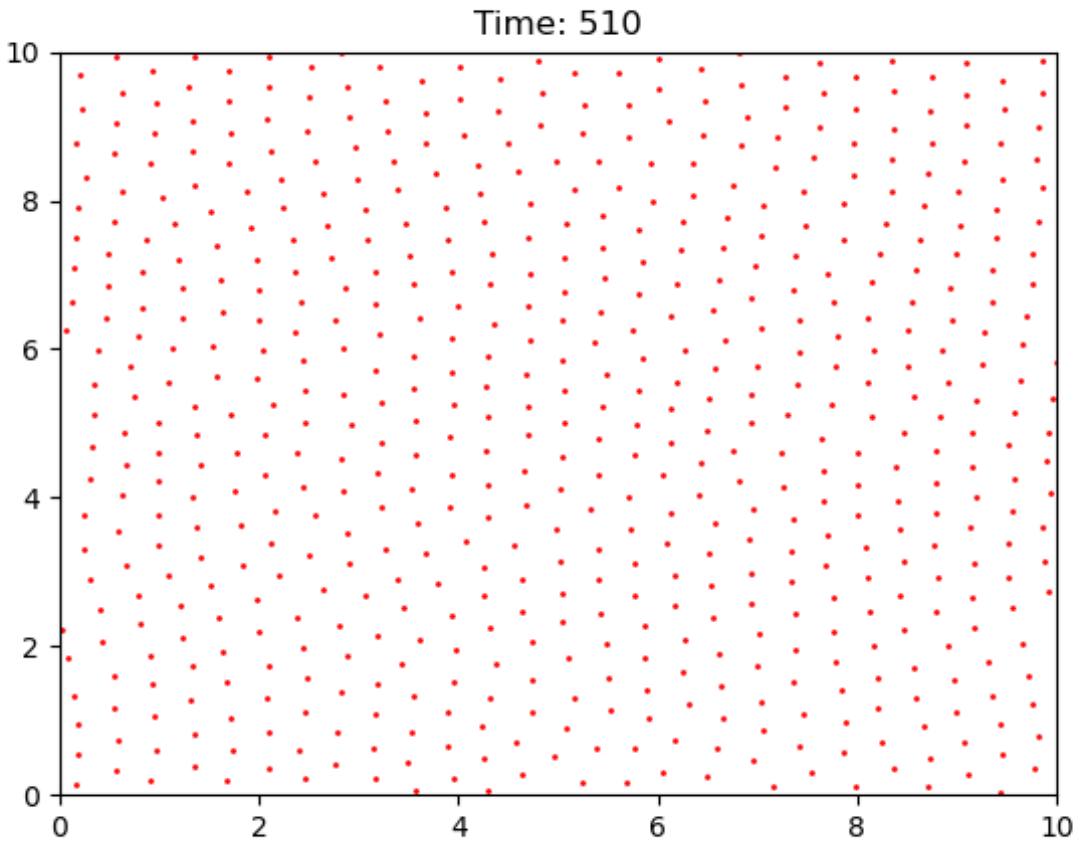



Figure 5: Visualização da Cristalização

Fica fácil visualizar que as moléculas acabam por ficar confinadas à regiões em uma malha triangular de maior estabilidade no plano.

O vídeo da situação se encontra também na pasta do trabalho para melhor visualização. Na subpasta da tarefa *E*. Na pasta também tem uma extensão da tarefa para uma grade mais densa. Podemos visualizar as posições.



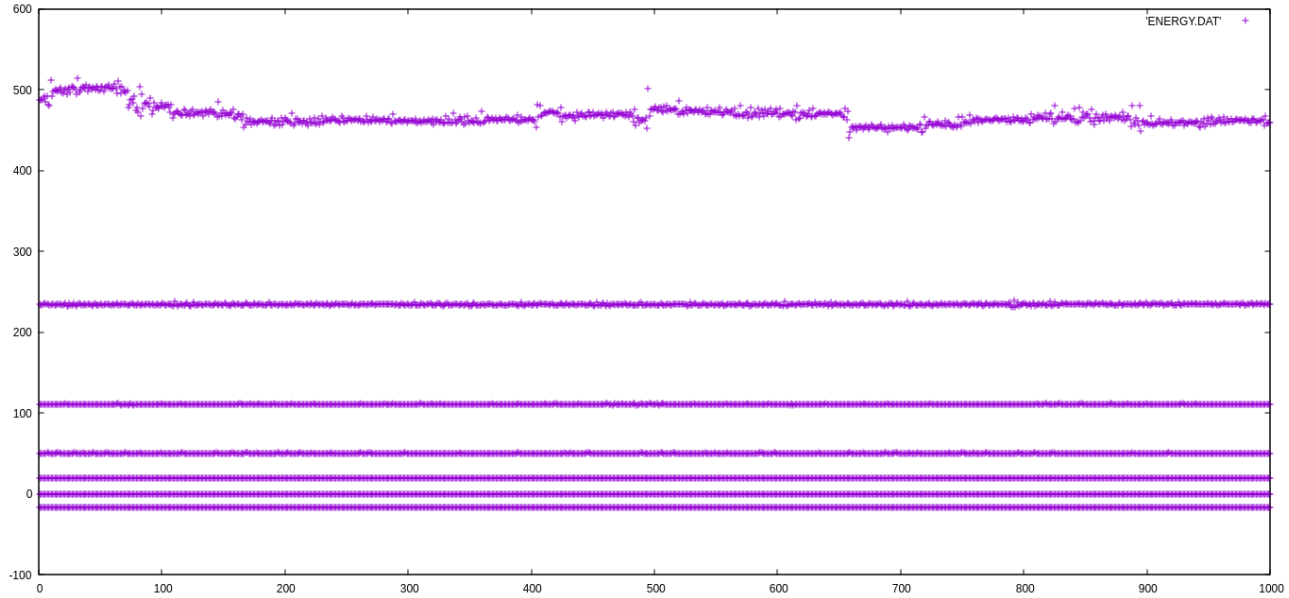

## 8 Tarefa F

Temos um sólido na tarefa anterior. Vamos desfazê-lo. Vamos esquentar nossa simulação aumentando repentinamente todas velocidades por um fator 1.5. Escrevemos,

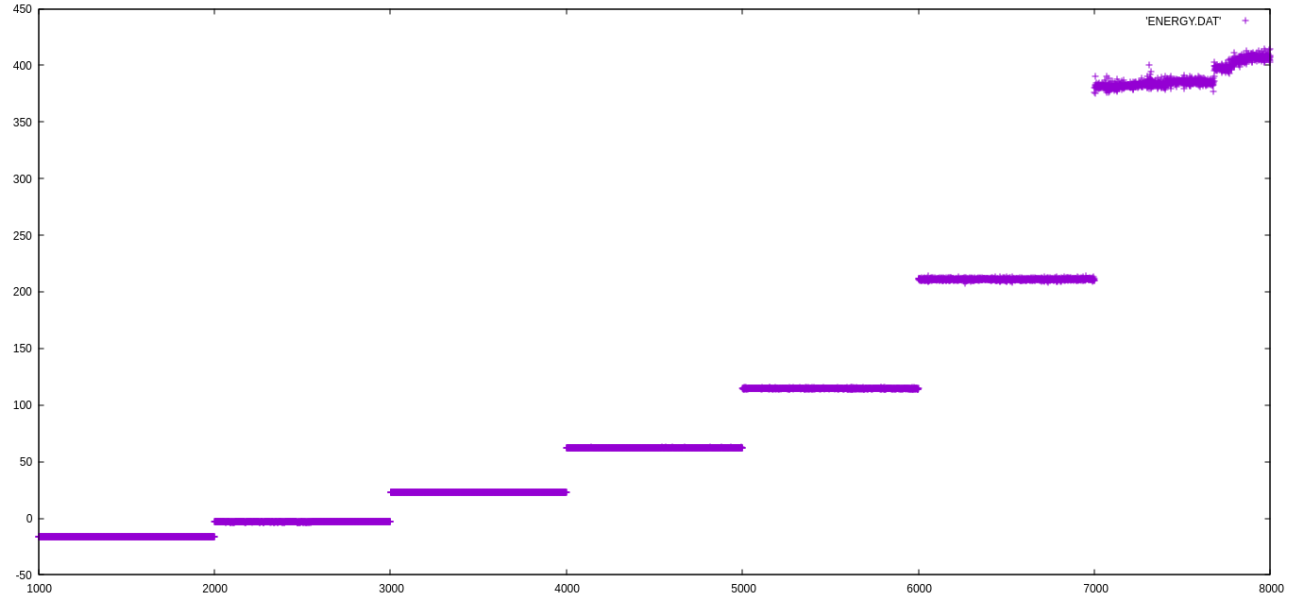
```
      PROGRAM VERLET
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY= 16,SIZEL= 4,DT=0.005)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2),ENERGY
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

C           FILE TO SAVE THE DATA
            OPEN(UNIT=1,FILE='DATAx.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=2,FILE='DATAy.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=3,FILE='ENERGY.DAT',STATUS='UNKNOWN')
            OPEN(UNIT=4,FILE='DATAV.DAT',STATUS='UNKNOWN')

c           INITIALIZE THE RANDOM NUMBER GENERATOR
            CALL SRAND(1)

```

```
C           INCITIALIZE OUR SYSTEM CONDITIONS
            VMAX = 0.1
            R = 1.5
            CALL SETUPBODYS(VMAX)

            do ifase=1,7
C           START THE TIME LOOP
                  DO ITIME=1,1000

C                 CALCULATE THE NEW CONDITIONS
                        CALL UPDATECONDITIONS()

                        WRITE(4,*) ITIME,CONDITIONS(1,1,1),
     +CONDITIONS(1,1,2), SQRT(CONDITIONS(1,2,1)**2+CONDITIONS(1,2,2)**2)

C                 WRITE THE NEW CONDITIONS
                        DO I=1,NBODY
                              WRITE(1,*) CONDITIONS(I,1,1)
                              WRITE(2,*) CONDITIONS(I,1,2)
                        END DO

C                 SAVE THE OLD CONDITIONS
                        CALL SAVECONDITIONS()

c                 Calculate energy
                        CALL CALCULATEENERGY(ENERGY)

                        WRITE(3,*) ITIME,ENERGY

                  END DO

                  CALL UPDATEVELOCITY(R)

            end do

      END PROGRAM VERLET

      SUBROUTINE UPDATEVELOCITY(R)
            IMPLICIT REAL*4(A-H,O-Z)
            PARAMETER(NBODY= 16,SIZEL= 4,DT=0.005)
            REAL*4 CONDITIONS(1:NBODY,2,2),OLD(-2:-1,1:NBODY,2,2),
     +      FS(1:NBODY,1:NBODY),ACC(1:NBODY,2),RS(1:NBODY,1:NBODY),
     +      DISTS(1:NBODY,1:NBODY,1:2)
            COMMON /BODYS/ CONDITIONS, OLD, DISTS
            COMMON /FORCES/ FS
            COMMON /ACCELERATIONS/ ACC
            COMMON /DISTS/ RS

            DO I=1,NBODY
                  DO J=1,2
                        FATOR = (OLD(-1,I,1,J) - OLD(-2,I,1,J))*R
                        OLD(-2,I,1,J) = OLD(-1,I,1,J) - FATOR
                  END DO
            END DO

      END SUBROUTINE UPDATEVELOCITY
```

Note que vamos alterar a energia a cada mudança de velocidade. Vale uma análise das energias usadas aumento a velocidade da forma descrita. Apresentamos as energias em níveis e em ordem sequencial , vê se o aumento da energia com as mudanças de velocidade.



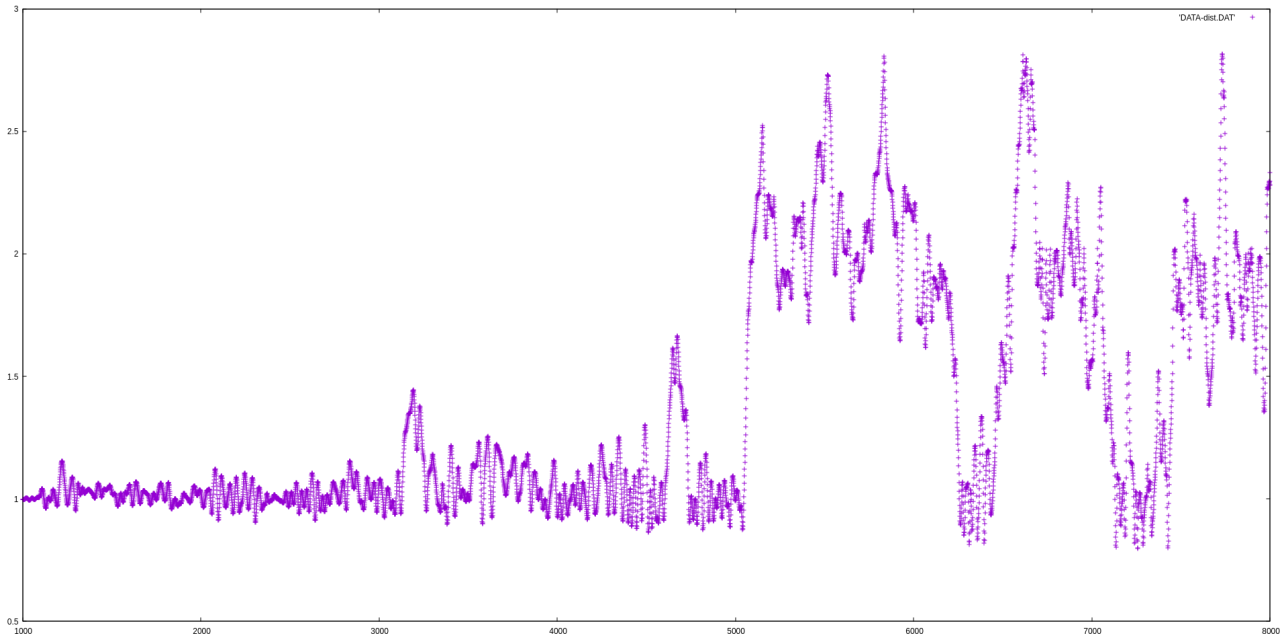

Que representa os níveis de energia usados




Que representa as energias em sequencia temporal

Figure 6: Visualização das Energias usadas

Nosso programa se comporta da forma esperada, vamos analisar nossas partículas em diversos regimes de tempo durante essas mudanças. Uma outra forma de visualizar a liquefação é utilizando da distância entre duas partículas. Observe o seguinte gráfico,




Fica fácil ver no gráfico quando a grade se desfaz completamente, junto com uma transição de energia. Escolhemos quatro momentos na evolução para demonstrar aqui. O vídeo da evolução do sistema se encontra na subpasta da tarefa.

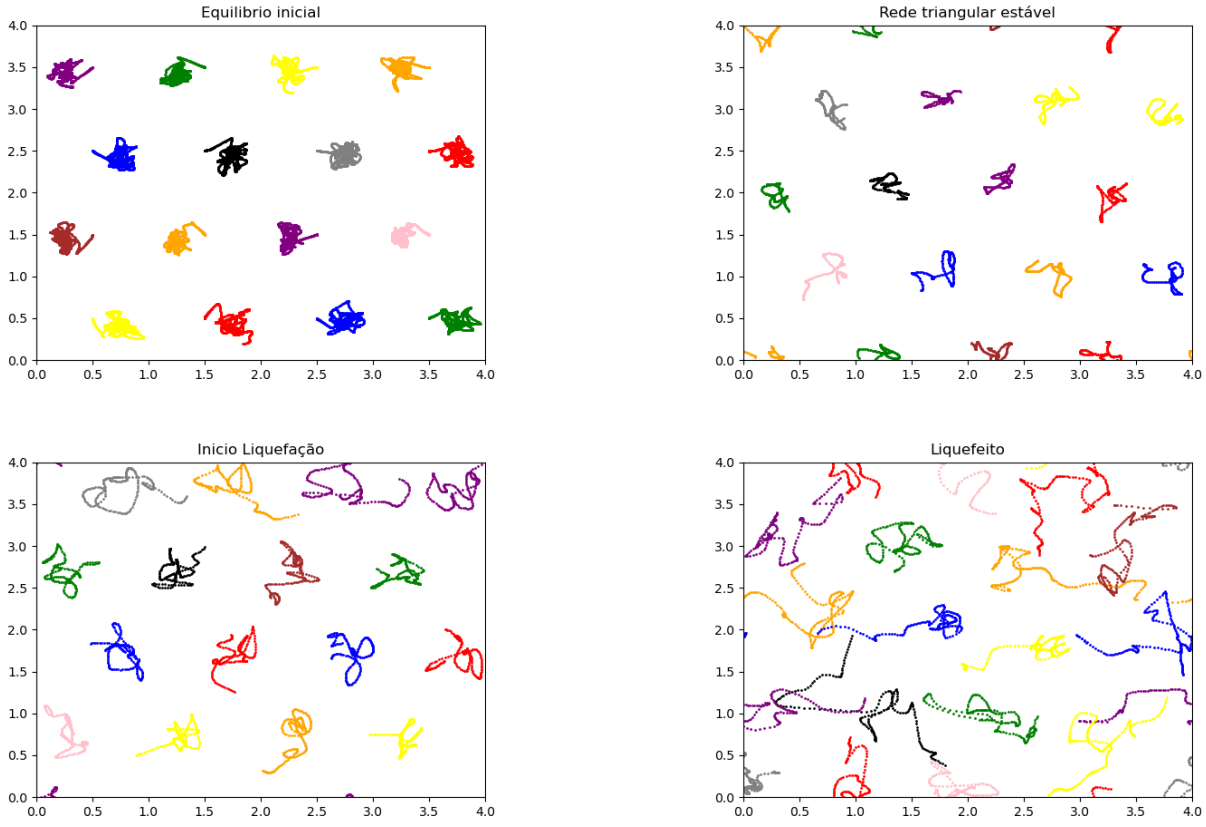



Figure 7: Visualização das Posições