DinamicaMolecular_Tarefa_3

August 1, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
```

0.1 1 - Explorando o passo de tempo a ser usado.

Investigue esse aspecto determinando como o tamanho das flutuações de energia, ou seja, o valor quadrático médio da energia $\sqrt{\langle E^2 \rangle + \langle E \rangle^2}$ depende do intervalo de tempo, dt, utilizado. Confira, também se para longos tempos (~100.000 passos) não surgem tendências de variação no valor médio da energia. De fato, pequenas flutuações na energia sempre estarão presentes, mas é essencial eliminar quaisquer vestígios de desvio na energia total ao longo de períodos de dezenas de milhares de passos de tempo, se a simulação se propor a amostrar corretamente o ensemble microcanônico.

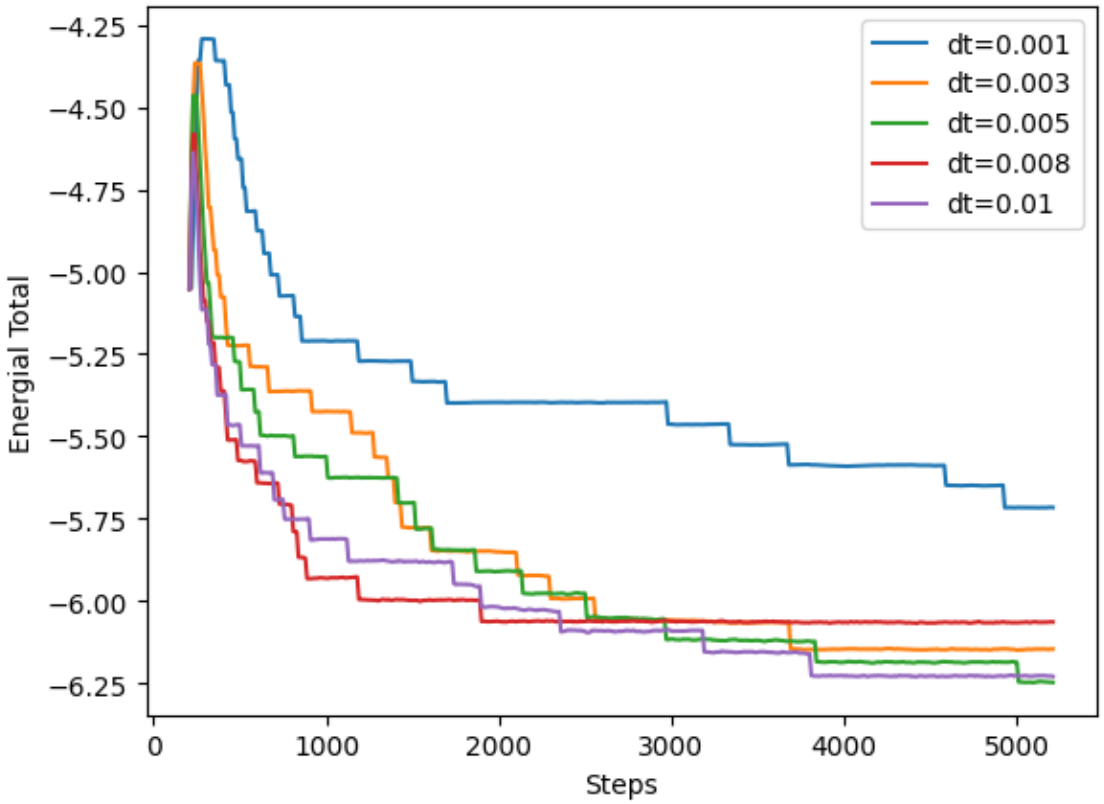
Utilizaremos passos de tempo $dt = \{0.001, 0.003, 0.005, 0.008, 0.01\}$ para a temperatura de $T = 0.5$ e $\rho = 1.0$ com `5000 passos`.

[2]:
```
def step_vs_energy(data_frame, dt):
    "Plotar energia total em função dos passos"
    x = data_frame.columns[0] # Steps
    y = data_frame.columns[5] # Total Energy
    plt.xlabel('Steps')
    plt.ylabel('Energial Total')
    plt.plot(data_frame[x], data_frame[y], label=f'{dt=}')
    plt.legend()

def step_vs_temperature(data_frame):
    "Plotar energia total em função dos passos"
    x = data_frame.columns[0] # Steps
    y = data_frame.columns[1] # Temperature
    plt.xlabel('Steps')
    plt.ylabel('Temperatura')
    plt.ylim(0.3, 0.8)
    plt.plot(data_frame[x], data_frame[y])

def step_vs_input(data_frame):
    user_choice = input(['Step', 'Temp', 'Press', 'PotEng', 'KinEng', 'TotEng',␣
 ↪'c_MSD[1]',
```

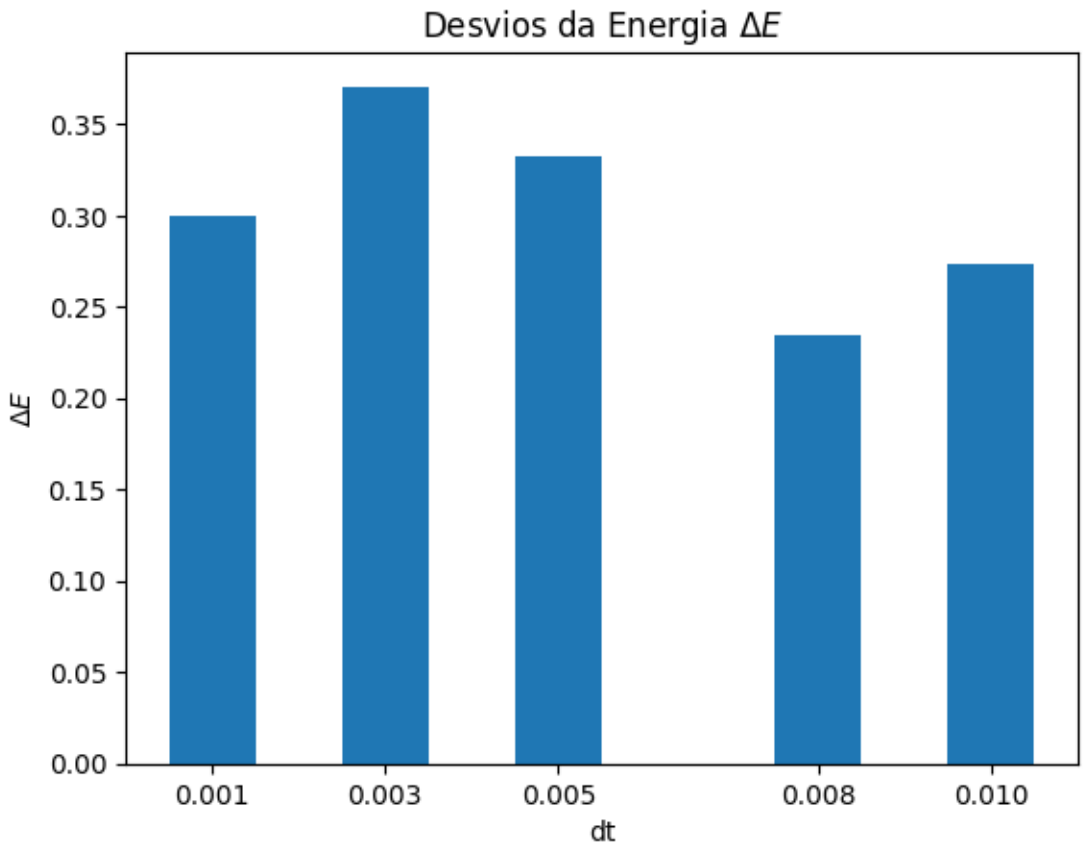
```
        'c_MSD[2]', 'c_MSD[3]'])
    x = data_frame.columns[0] # Steps
    y = data_frame.columns[int(user_choice)] # input
    plt.xlabel('Steps')
    plt.ylabel(y)
    plt.plot(data_frame[x], data_frame[y])
```

[3]:
```
# Fixando  =0.8 e T e varrendo dt
sim_1 = pd.read_csv(os.path.join('lammps', 'build','questao01',  "dt0001.csv"))␣
 ↪# dt =0.001
sim_2 = pd.read_csv(os.path.join('lammps', 'build','questao01' ,"dt0003.csv"))␣
 ↪# dt =0.003
sim_3 = pd.read_csv(os.path.join('lammps', 'build','questao01' ,"dt0005.csv"))␣
 ↪# dt =0.005
sim_4 = pd.read_csv(os.path.join('lammps', 'build','questao01' ,"dt0008.csv"))␣
 ↪# dt =0.008
sim_5 = pd.read_csv(os.path.join('lammps', 'build','questao01',"dt001.csv")) #␣
 ↪dt =0.008
```

[4]:
```
step_vs_energy(sim_1, dt=0.001)
step_vs_energy(sim_2, dt=0.003)
step_vs_energy(sim_3, dt=0.005)
step_vs_energy(sim_4, dt=0.008)
step_vs_energy(sim_5, dt=0.01)
```

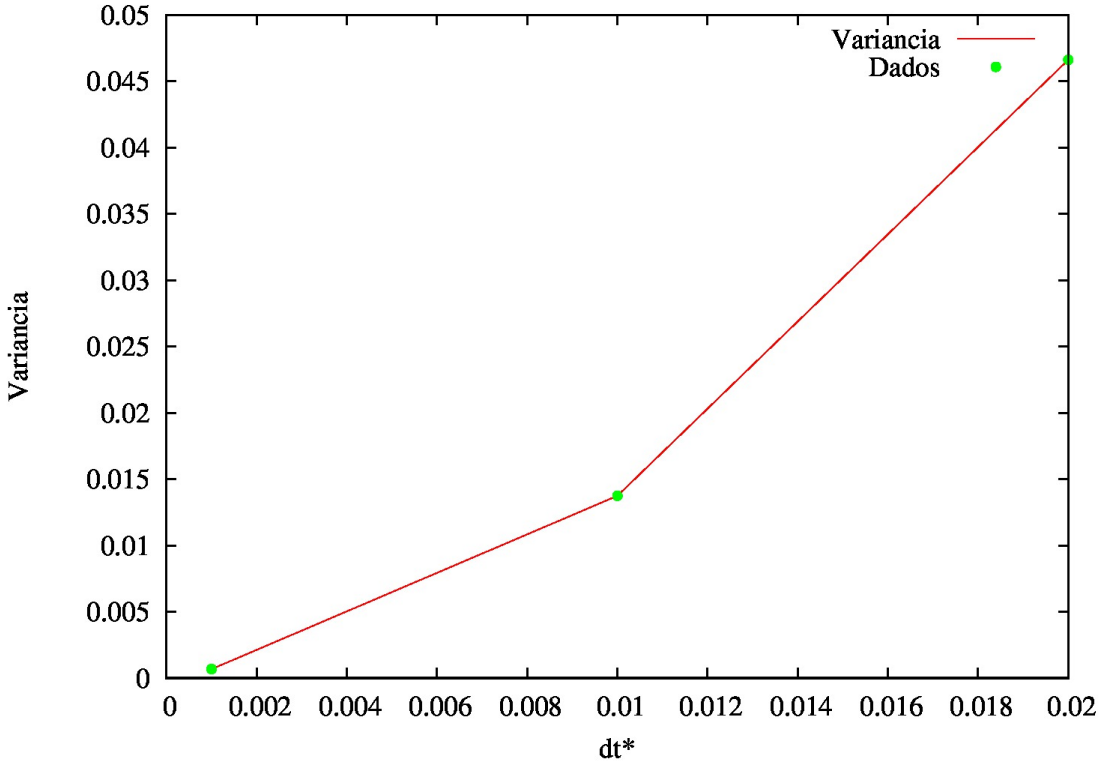


[41]:
```
dt_s = [0.001, 0.003, 0.005, 0.008, 0.01]
desvios =[sim_1.std()['TotEng'] ,sim_2.std()['TotEng'], sim_3.std()['TotEng'],␣
 ↪sim_4.std()['TotEng'], sim_5.std()['TotEng']]
plt.title(r"Desvios da Energia $\Delta E$")
plt.xticks(dt_s);
plt.xlabel('dt')
plt.ylabel(r'$\Delta E$')
plt.bar(dt_s, desvios, width=0.001);
```



Para as simulações acima, a simulação com $dt = 0.008$ apresentou o menor desvio na energia, ficando por um maior tempo com a energia E fixa, como desejado para o ensemble microcanônico. Apesar de ser um passo de tempo `maior`, ainda apresentou resultados melhores que os passos de tempos menores.

Esse resultado não é o esperado para simulações de dinâmica molecular. Para o intervalos de `dt's` utilizados, era de se esperar que o ΔE aumentasse (quase linearmente) com o dt. Deve haver nesses sistemas então, erros de programa ou uma condição inicial não desejável. À critério de comparação, segue um comportamento esperado:



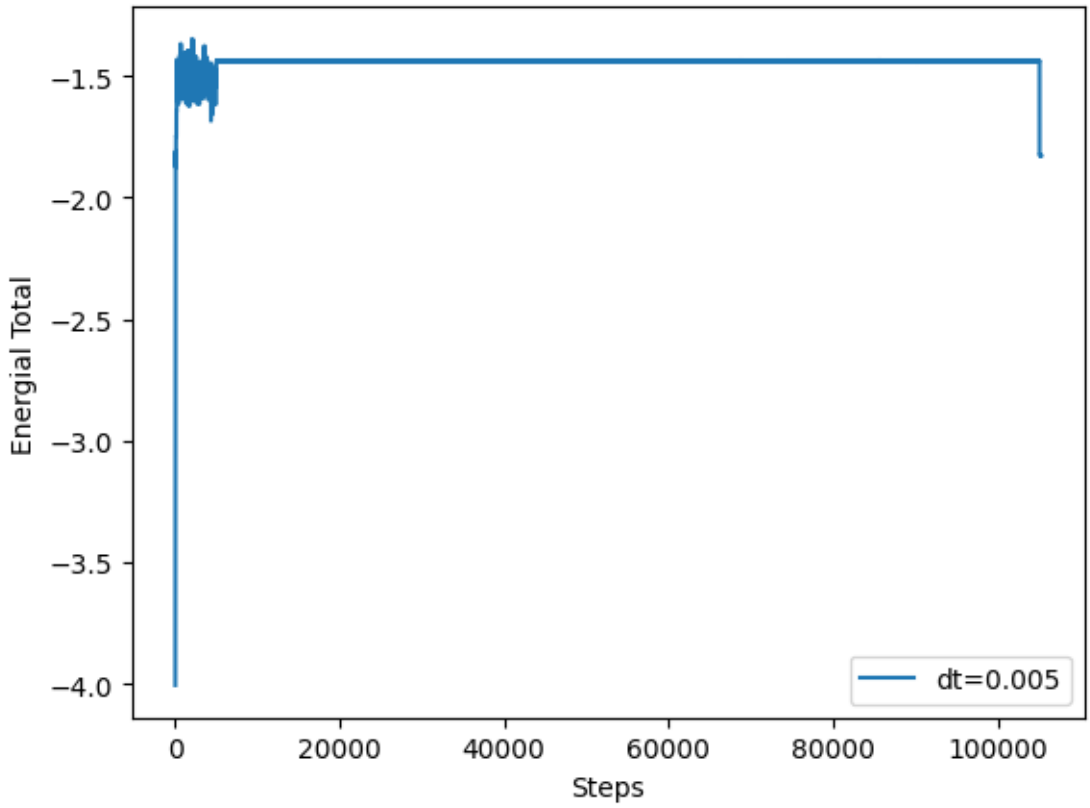
fonte: https://fiscomp.if.ufrgs.br/index.php/Arquivo:Var.jpg

0.1.1 Passos longos - Dinâmica de 100 000 passos

Temperatura $T = 2.0$, $\rho = 0.8$, $dt = 0.005$.

[46]:
```
thousand_steps = pd.read_csv(os.path.join('lammps', 'build', "thousand_steps.
↪csv")) # dt =0.001
```

[47]:
```
step_vs_energy(thousand_steps, dt=0.005)
```



1 2 - Equilibração do Sistema

Faça gráficos mostrando o processo de equilibração e estime o tempo necessário para equilibrar o sistema em, pelo menos, três conjuntos diferentes de valores de temperatura e densidade.

Temperaturas típicas LJ : $T \rightarrow 0.7 - 1.2$
Densidades típicas LJ: $\rho \rightarrow 0.6 - 1.0$

(T = 1.0 ρ = 1.0), (T = 0.5 ρ = 1.2) , (T = 1.5 ρ = 0.6)

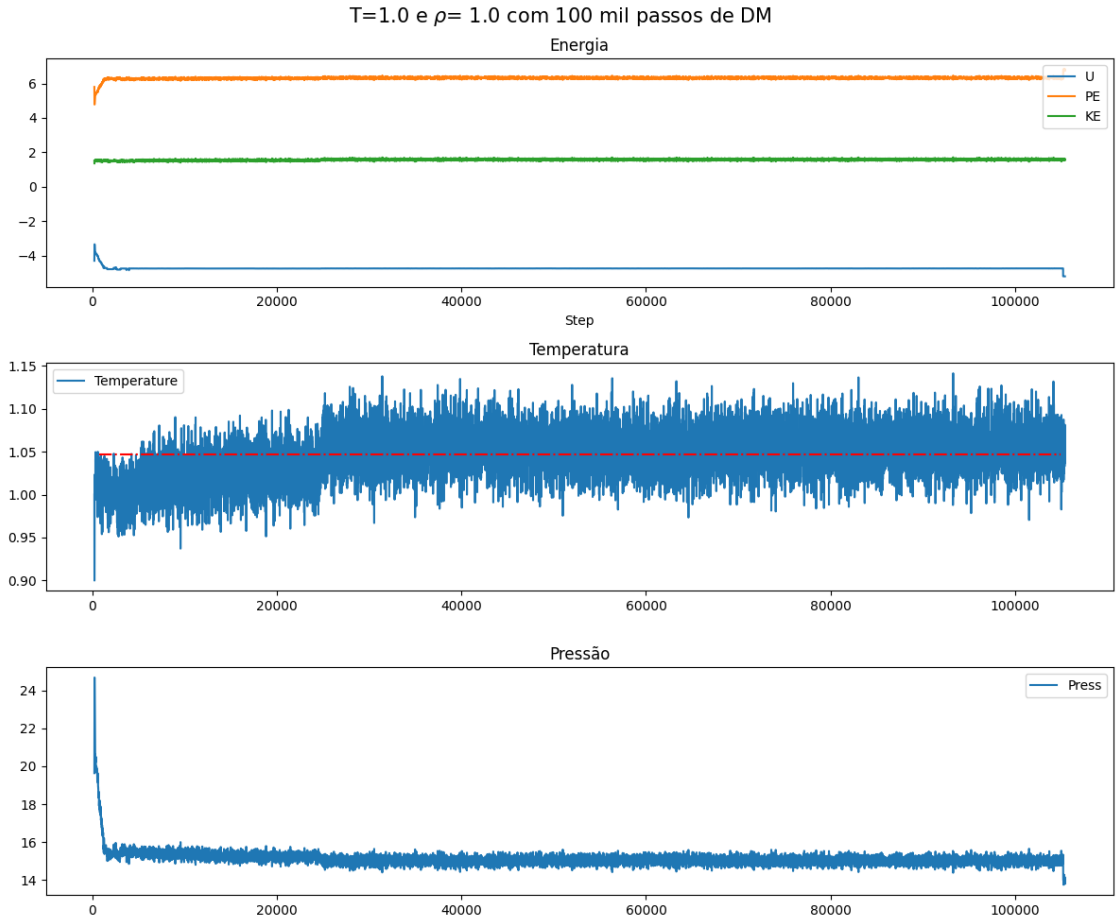
[48]:
```
def plot_questao2(simulacao, T, rho):
    fig, ax = plt.subplots(3,1, figsize=(12, 10))

    plt.suptitle(f"T={T} e " + r"$\rho$" +f"= {rho} com 100 mil passos de DM",␣
 ↪size=15)
    # Energia
    ax[0].set_title("Energia")
    ax[0].plot(simulacao['Step'] , simulacao['TotEng'], label='U')
    ax[0].plot(simulacao['Step'] , -simulacao['PotEng'], label='PE') # Módulo
    ax[0].plot(simulacao['Step'] , simulacao['KinEng'], label='KE')
    ax[0].set_xlabel('Step')
```

```
    ax[0].legend()


    # Temperatura
    ax[1].set_title("Temperatura")
    # ax[1].scatter(simulacao['Step'] , simulacao['Temp'], label='Temperature',␣
↪s=0.08)
    ax[1].plot(simulacao['Step'] , simulacao['Temp'], label='Temperature')
    mean_temperature = simulacao['Temp'].mean()
    ax[1].axhline(mean_temperature, xmin=0.05, xmax=0.95, color='red',␣
↪ls='dashdot')
    ax[1].legend()

    # Press
    ax[2].set_title("Pressão")

    # ax[2].scatter(simulacao['Step'] , simulacao['Press'], label='Press', s=0.
↪05)
    ax[2].plot(simulacao['Step'] , simulacao['Press'], label='Press')
    ax[2].legend()

    plt.tight_layout()
    plt.show()
```

[49]:
```
equilibrium_1 = pd.read_csv(os.path.join('lammps', 'build', "equilibrium1.csv"))
equilibrium_1.head(5)
```

[49]:
```
   Step      Temp      Press    PotEng    KinEng    TotEng  c_MSD[1]  \
0   209  1.000000  19.639216 -5.800773  1.497414 -4.303359  0.000000
1   210  0.989435  20.072866 -5.718225  1.481593 -4.236632  0.000063
2   220  0.899809  23.886826 -4.965801  1.347386 -3.618414  0.001761
3   230  0.954691  24.683839 -4.775390  1.429568 -3.345822  0.004266
4   240  1.015095  24.034536 -4.864918  1.520017 -3.344902  0.009093

   c_MSD[2]  c_MSD[3]
0  0.000000  0.000000
1  0.000063  0.000066
2  0.002144  0.002406
3  0.004847  0.005822
4  0.008102  0.010919
```

[50]:
```
plot_questao2(equilibrium_1, T=1.0, rho=1.0)
```



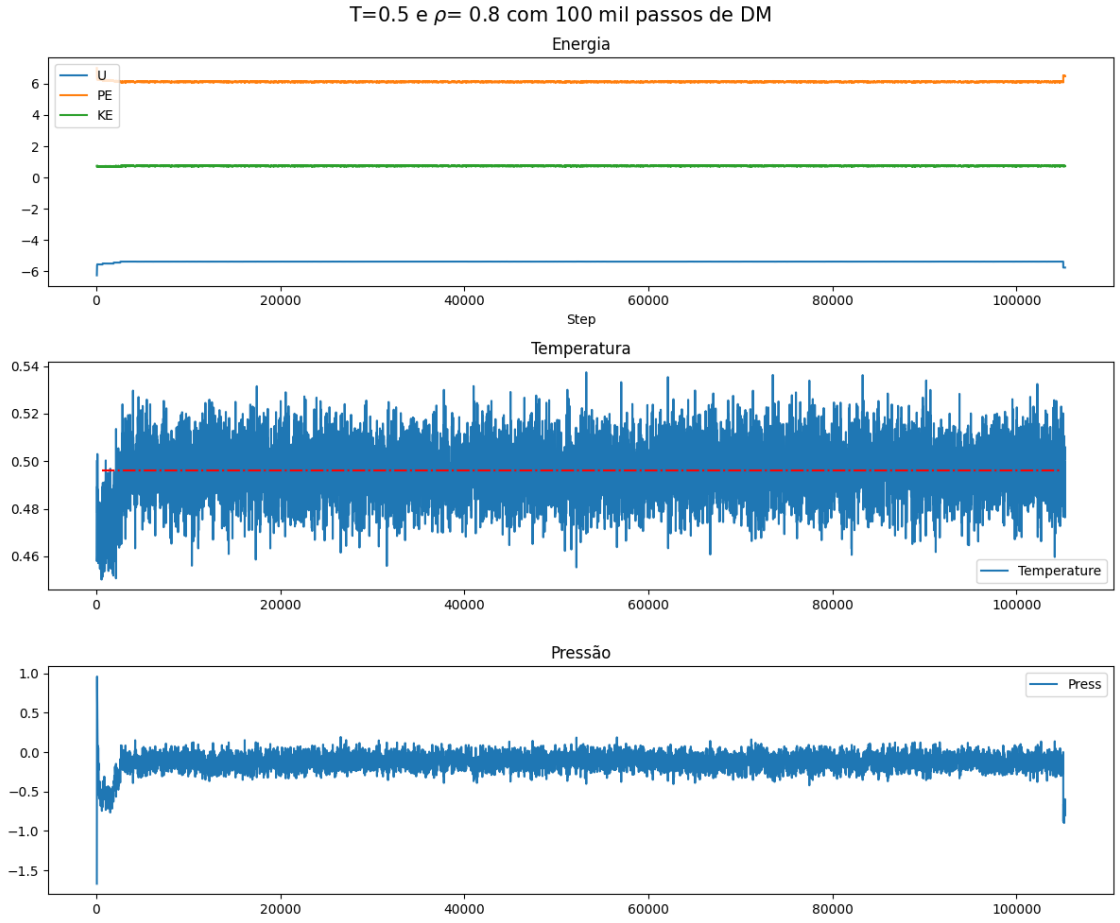
Para essa configuração, a temperatura deve ter começado a oscilar em torno do valor médio no passo 30 mil. Para energia, logo no início da simulação. Já a pressão, por volta dos 40 mil passos.

[51]:
```
equilibrium_2 = pd.read_csv(os.path.join('lammps', 'build', "equilibrium2.csv"))
equilibrium_2.head(5)
```

[51]:

	Step	Temp	Press	PotEng	KinEng	TotEng	c_MSD[1]	c_MSD[2]	c_MSD[3]
0	74	0.500000	-1.671468	-6.999982	0.748828	-6.251154	0.000000	0.000000	0.000000
1	80	0.457988	-0.097217	-6.674934	0.685909	-5.989025	0.000967	0.000966	0.000978
2	90	0.460243	0.857168	-6.440324	0.689286	-5.751038	0.003778	0.003931	0.003598
3	100	0.476390	0.961397	-6.323005	0.713469	-5.609536	0.006183	0.006841	0.006331
4	110	0.489416	0.782183	-6.278276	0.732978	-5.545299	0.009003	0.010590	0.009351

[52]:
```
plot_questao2(equilibrium_2, T=0.5, rho=0.8)
```



A energia não demorou muita a estabilizar, a pressão equilibrou em torno do 3000 passos o que parecer ter acontecido o mesmo para a temperatura.

[53]:
```
# T=1.8   =1.3
equilibrium_3 = pd.read_csv(os.path.join('lammps', 'build', "equilibrium3.csv"))
equilibrium_3.head(5)
```

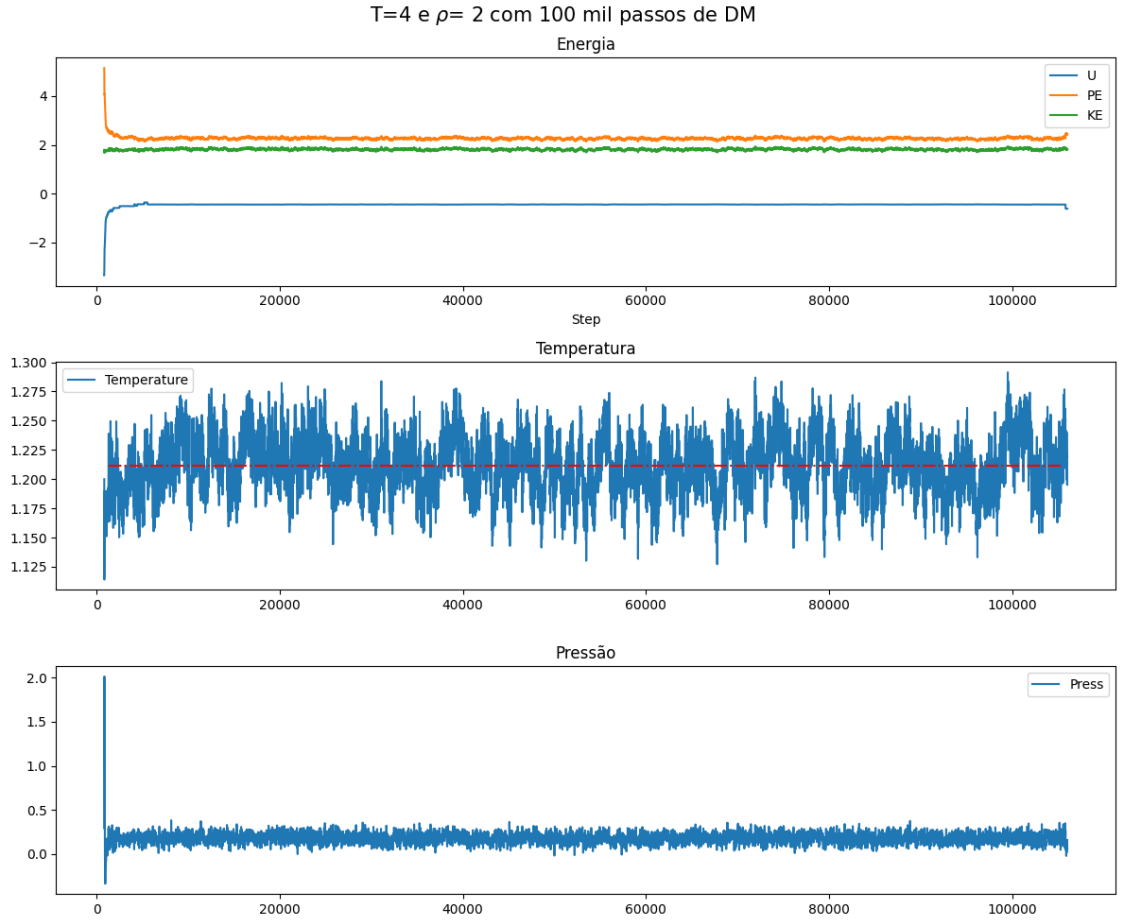
[53]:

	Step	Temp	Press	PotEng	KinEng	TotEng	c_MSD[1]	c_MSD[2]	\
0	829	1.200000	0.292595	-5.151320	1.796897	-3.354423	0.000000	0.000000	
1	830	1.195995	0.303325	-5.145326	1.790899	-3.354427	0.000019	0.000019	
2	840	1.114091	1.407741	-4.507422	1.668255	-2.839167	0.001978	0.001969	
3	850	1.124321	2.015403	-4.069361	1.683573	-2.385788	0.005832	0.006047	
4	860	1.188413	1.523074	-4.093430	1.779546	-2.313884	0.010250	0.011900	

	c_MSD[3]
0	0.000000

```
1  0.000020
2  0.002060
3  0.006316
4  0.012607
```

[54]:
```
plot_questao2(equilibrium_3, T=4, rho=2)
```

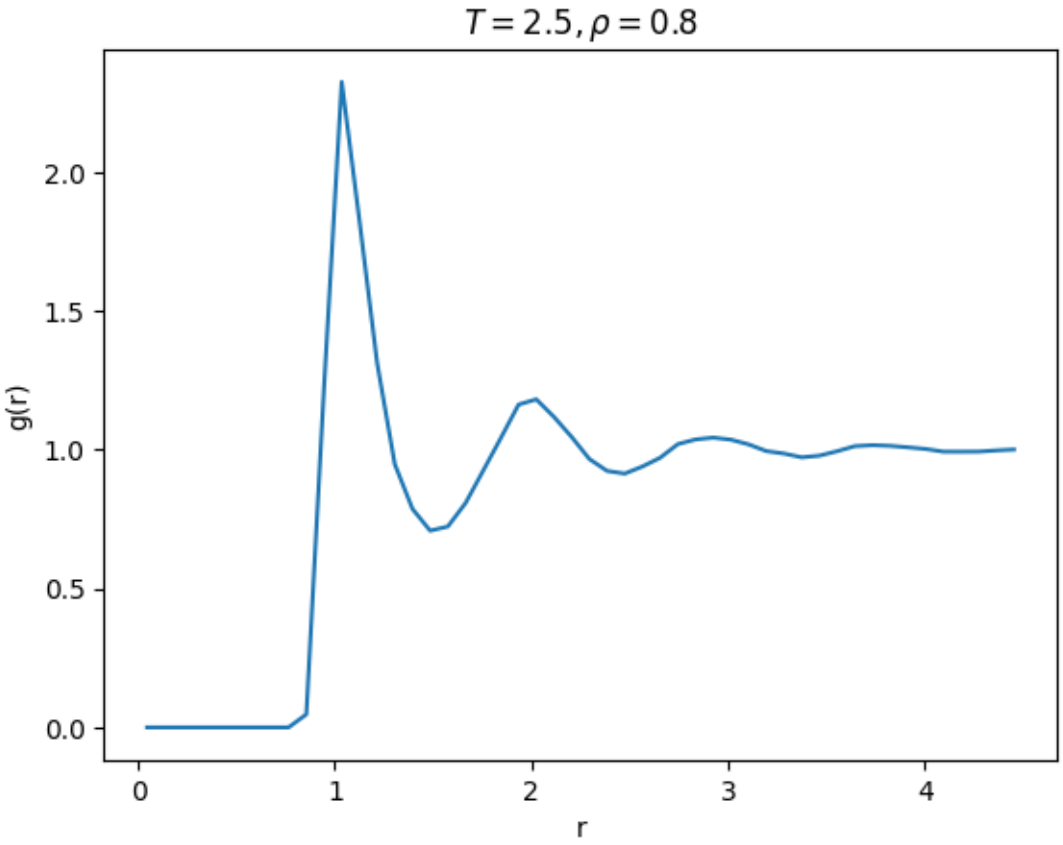


Nessa situação, a parece ter estabilizado pelo passo 5000, juntamente com a pressão, mas a temperatura não pareceu oscilar em torno de uma média bem definida.

2 3 - Explorando o diagrama de fases

Simule um sistema com N partículas e mostre que, ajustando a densidade e a energia cinética das partículas (temperatura), pode-se reproduzir o comportamento de fluidos e sólidos. Identifique pelo menos dois conjuntos de parâmetros que levem à uma fase fluida e sólida. Mostre claramente resultados que suportem suas conclusões. Em especial, sugiro que calculem e analisem a função de distribuição radial, ( ).

[58]:
```
liquid_phase = pd.read_csv(os.path.join('lammps', 'build',"questao03","liquid.
↪data"), sep=" ", names=["Index", "r", "g(r)", "N(r)" ]) # dt =0.001
plt.plot(liquid_phase['r'], liquid_phase['g(r)'])
plt.title(r"$T=2.5 ,  \rho = 0.8$");
plt.ylabel("g(r)")
plt.xlabel("r");
```



A distribuição radial acima é típica de um sistema na fase líquida que devido ao caos molecular não apresenta qualquer estrutura. Em outras palavras, a perda da ordem a medida que r aumenta.

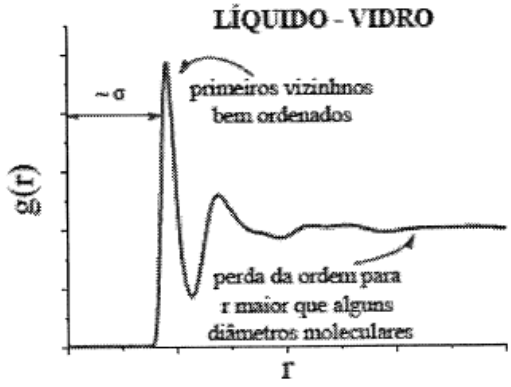
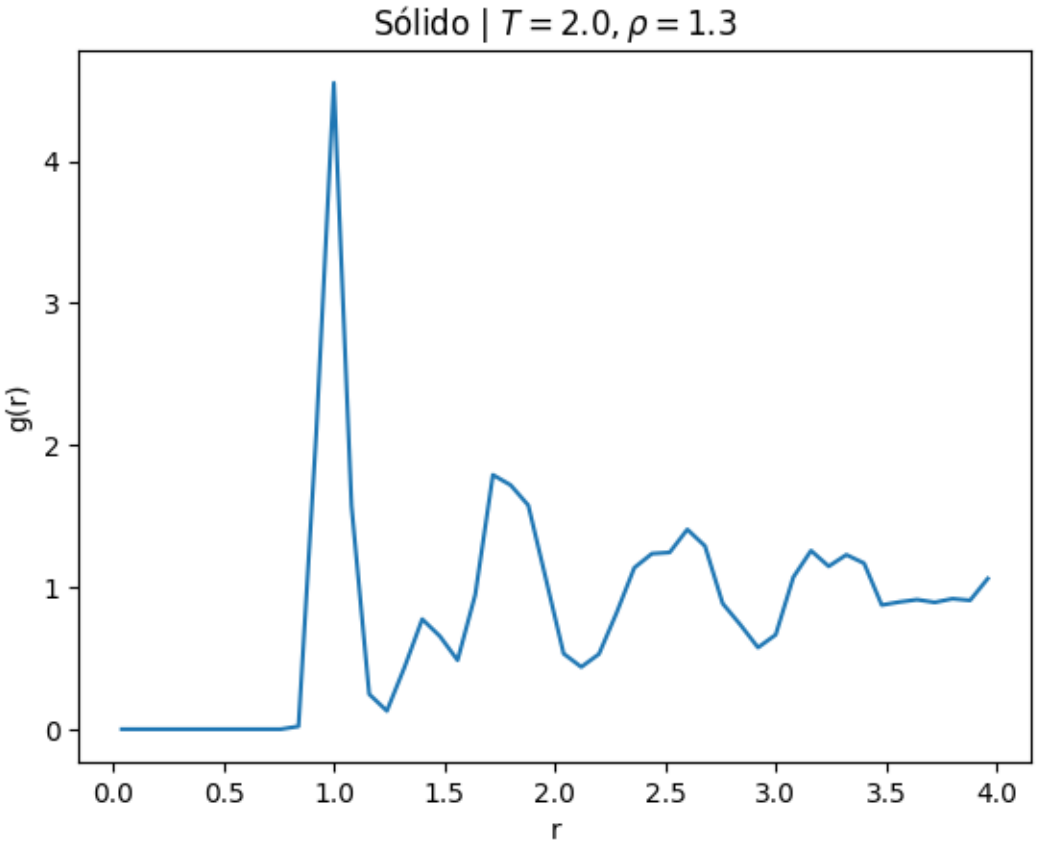


Figura 3.2: Função de distribuição radial típica de um líquido ou vidro.

fonte: ABC da Simulação Computacional

[59]:
```
solid_phase = pd.read_csv(os.path.join('lammps', 'build', "questao03" , "solid.
↪data"), sep=" ", names=["Index", "r", "g(r)", "N(r)" ]) # dt =0.001
plt.plot(solid_phase['r'], solid_phase['g(r)'])
plt.title(r"Sólido | $T=2.0, \rho = 1.3$");
plt.ylabel("g(r)")
plt.xlabel("r");
```



A função de distribuição radial acima sugere a presença da fase sólida haja vista os picos com certa periodicidade que refletem a estrutura cristalina.

3 4 - Constante de difusão num fluido

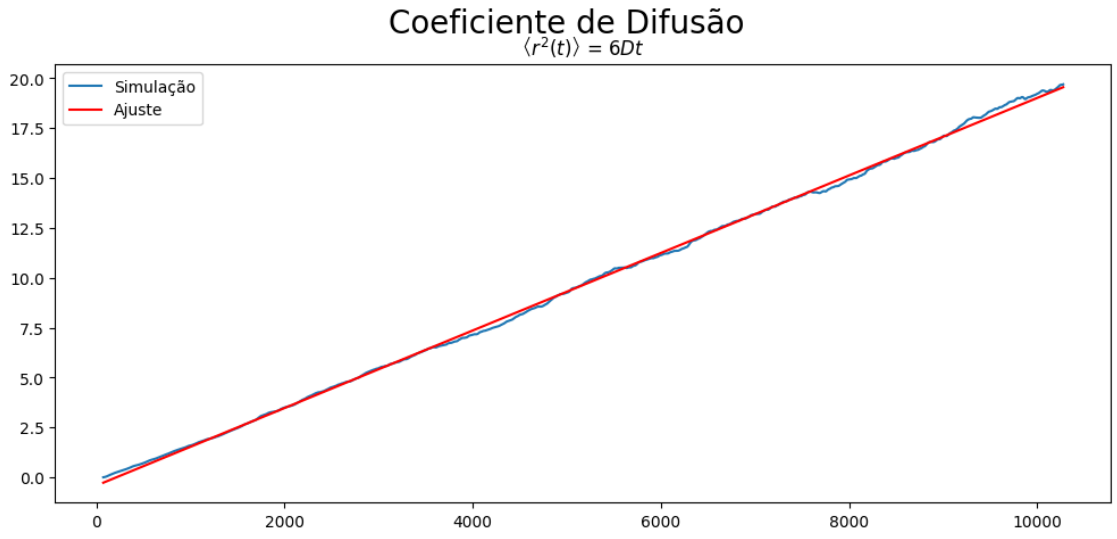
[60]:
```
sim_6 = pd.read_csv(os.path.join('lammps', 'build', "questao04" , "questao04.
↪csv"))
```

[61]:
```
x_2 = sim_6['c_MSD[1]']*sim_6['c_MSD[1]']
y_2 = sim_6['c_MSD[2]']*sim_6['c_MSD[2]']
z_2 = sim_6['c_MSD[3]']*sim_6['c_MSD[3]']
r_2 = (x_2 + y_2 + z_2)**0.5
```

[62]:
```
from scipy.stats import linregress
# Ajuste linear
y = r_2
x = sim_6['Step']

result = linregress(x, y)
coef_difusao = result.slope / 6
```

[63]:
```
plt.figure(figsize=(12,5))
plt.suptitle("Coeficiente de Difusão", fontsize=20)
plt.title(r"$\left \langle r^2(t) \right \rangle$ = $6Dt$")
plt.plot(sim_6['Step'], r_2, label='Simulação')
plt.plot(x, result.intercept + result.slope*x, 'r', label='Ajuste')
plt.legend();
print(f"Coeficiente de Difusão={coef_difusao}")
```

```
Coeficiente de Difusão=0.0003238100076500176
```



4 5 - Equação de Estado para o fluido

Considerando um sistema mantido à volume fixo na fase fluida, obtenha e discuta a equação de estado que relaciona pressão e volume à temperatura. Como o resultado obtido se compara ao esperado para um gás ideal?

- *LONGE DE PASSAR PELA ORIGEM*
- $P, V \times T$

Não consegui =)