Termodinâmica Estatística I

Construção das isotermas de gases de van der Waals e de Dieterici

Rafael Soares Andrade 96921
Humberto Sousa Martins 93724

Objetivo
O projeto tem como objetivo a construção dos gráficos das figuras 26.2 e 26.3 do livro-texto para o gás de van der Waals e para o gás de Dieterici. Para ambos os casos deve se identificar as temperaturas críticas e as respectivas linhas de coexistência de fase.

Procedimento e Dados

O projeto foi desenvolvido na linguagem de programação Python versão 3.6, a partir da ferramenta de edição e compilação *"Google Colaboratory"*. Para a construção dos gráficos, foi utilizada a ferramenta de construção de gráficos *Matplotlib*. Foram usadas as bibliotecas *numpy* para manejo de matrizes e operações vetoriais e biblioteca *scipy* para ajuste de curvas.

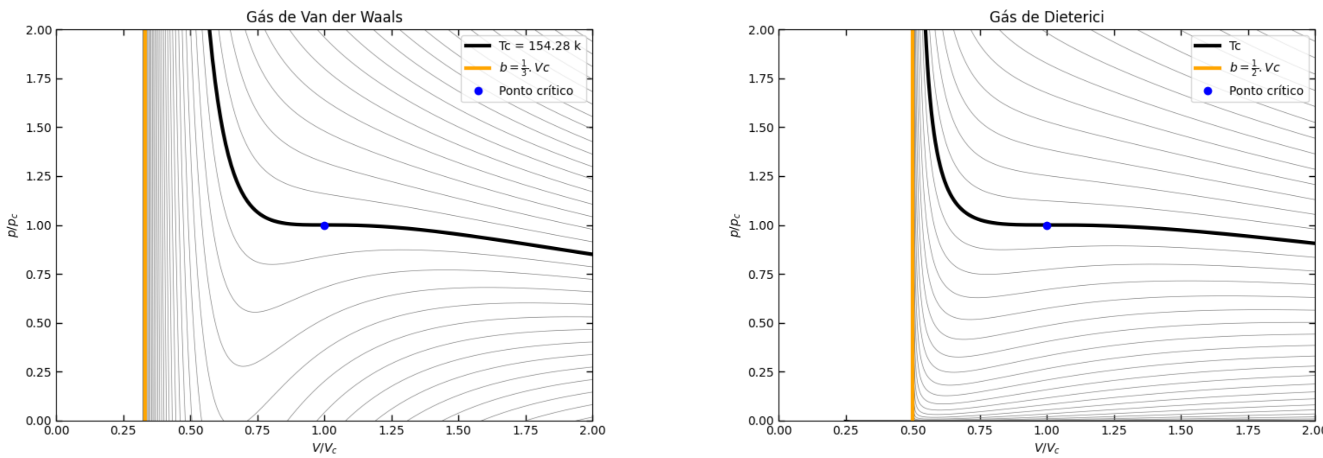
As equações de $p(V,T)$ foram definidas em coordenadas reduzidas de acordo com as equações 1 e 2. Neste trabalho, por ter sido desenvolvido a partir das equações reduzidas, denotaremos as variáveis reduzidas por seus nomes e símbolos habituais, dispensando acentos. Para construir o gráfico p vs V, definimos as equações de estado para ambos os gases, tanto para o gás de van der Waals quanto para o gás de Dieterici. O programa inicializa as variáveis de volume e temperatura e gera o conjunto da variável $p(V,T)$.

As variáveis reduzidas foram definidas como:

$$p_{VDW} = \frac{8T}{3V-1} - \frac{3}{V^2} \tag{1}$$

$$p_{Dieterici} = \frac{Te^{(2-\frac{2}{TV})}}{2V-1} \tag{2}$$

Com os conjuntos $p(V,T)$ definidos, os gráficos com os resultados são construídos, como mostrados na figura 1



(a) Gás de van der Waals. O valor de T_c demonstrado na figura é o valor para o O_2

(b) Gás de Dieterici

Figura 1: Isotermas dos gases de van der Waals e Dieterici.

A seguir, as variáveis são redefinidas para uma temperatura específica. Para realizar a integração numérica necessária, foi utilizado o método trapézio do pacote scipy, também foi necessário inverter a ordem de integração, de modo que a integral se dá da esquerda à direita no gráfico V vs p, o que levou a valores positivos de ΔG conforme a p aumenta . Com a integral resolvida, o programa encontra de maneira numérica os pontos extremos X e Y, utilizando o método *argrelextrema* do pacote scipy. Para encontrar o ponto B, foi utilizado o artifício de verificar os pontos de menor distância entre duas retas. No gráfico da energia de Gibbs vs pressão, o ponto de menor distância entre a reta que vem da pressão mínima até o ponto Y e a reta que vai do ponto X até a pressão máxima será o ponto B. Os gráficos são construídos de acordo com as figuras 2 e 3

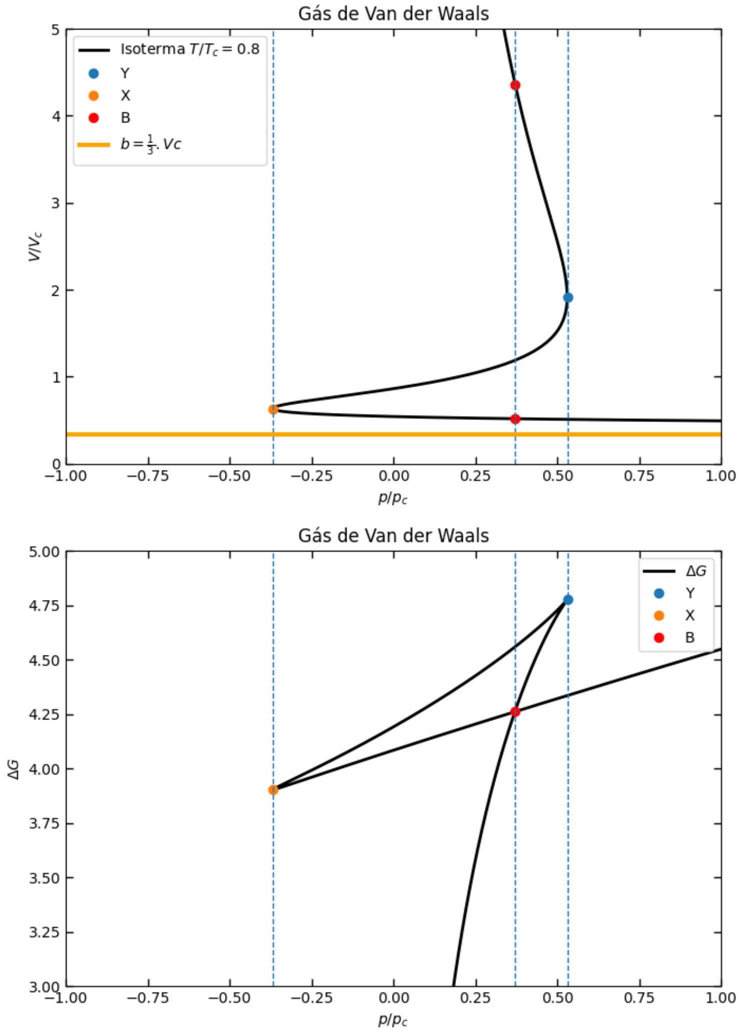


Figura 2: Comportamento do volume V e função de Gibbs do gás de van der Waals em função da pressão

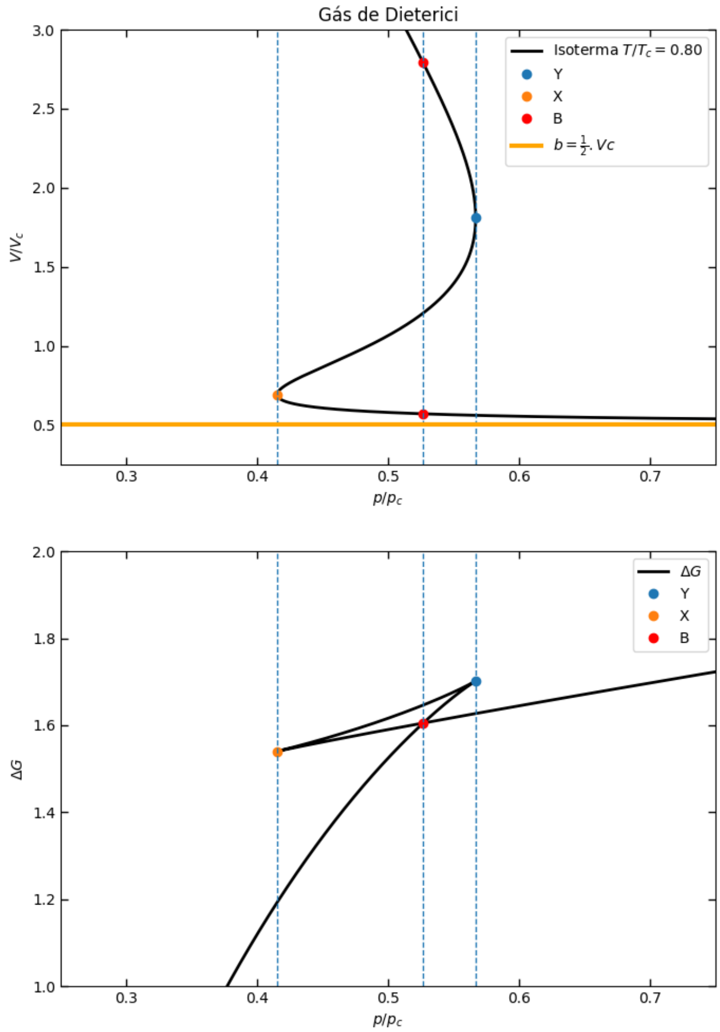
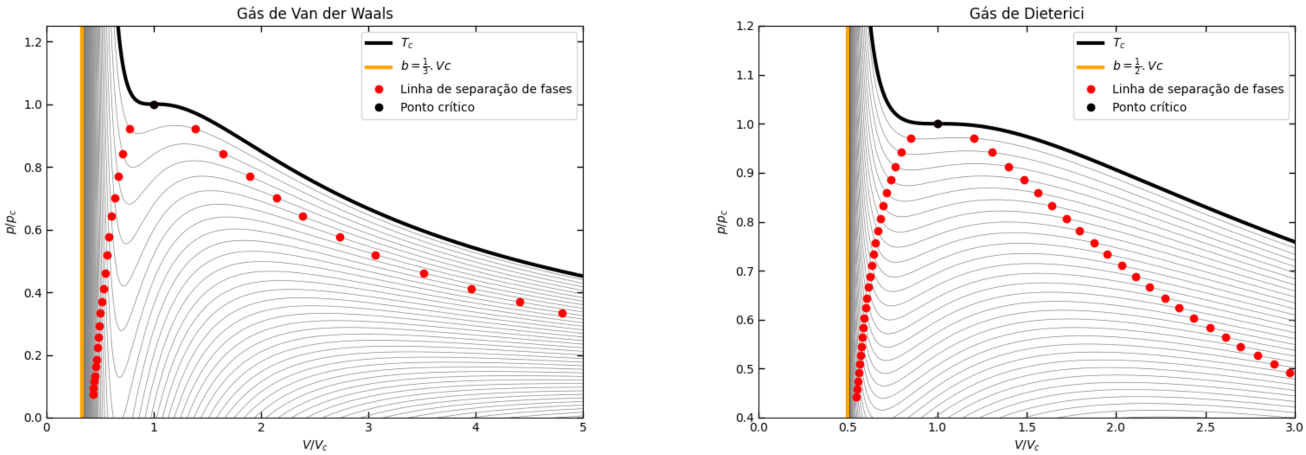


Figura 3: Comportamento do volume V e função de Gibbs do gás de Dieterici em função da pressão

A partir dos resultados obtidos, podemos construir os gráficos de pressão por volume, tal qual na figura 1, mas incluindo a linha de mudança de fase, demarcada pelos pontos B, obtidos anteriormente. A figura 4 mostra estes gráficos.



(a) Gás de van der Waals

(b) Gás de Dieterici

Figura 4: Isotermas dos gases de van der Waals e Dieterici com linha de mudança de fases. As isotermas superiores a T_c foram excluídas para melhor visualização do gráfico.

Utilizando funções semelhantes às usadas anteriormente, definimos as variáveis necessárias e aplicamos as mesmas operações, construindo assim os gráficos de V vs p e G vs p para cinco temperaturas diferentes, como mostrados nas figuras 5 e 6, tanto para o gás de van der Waals quanto para o gás de Dieterici.

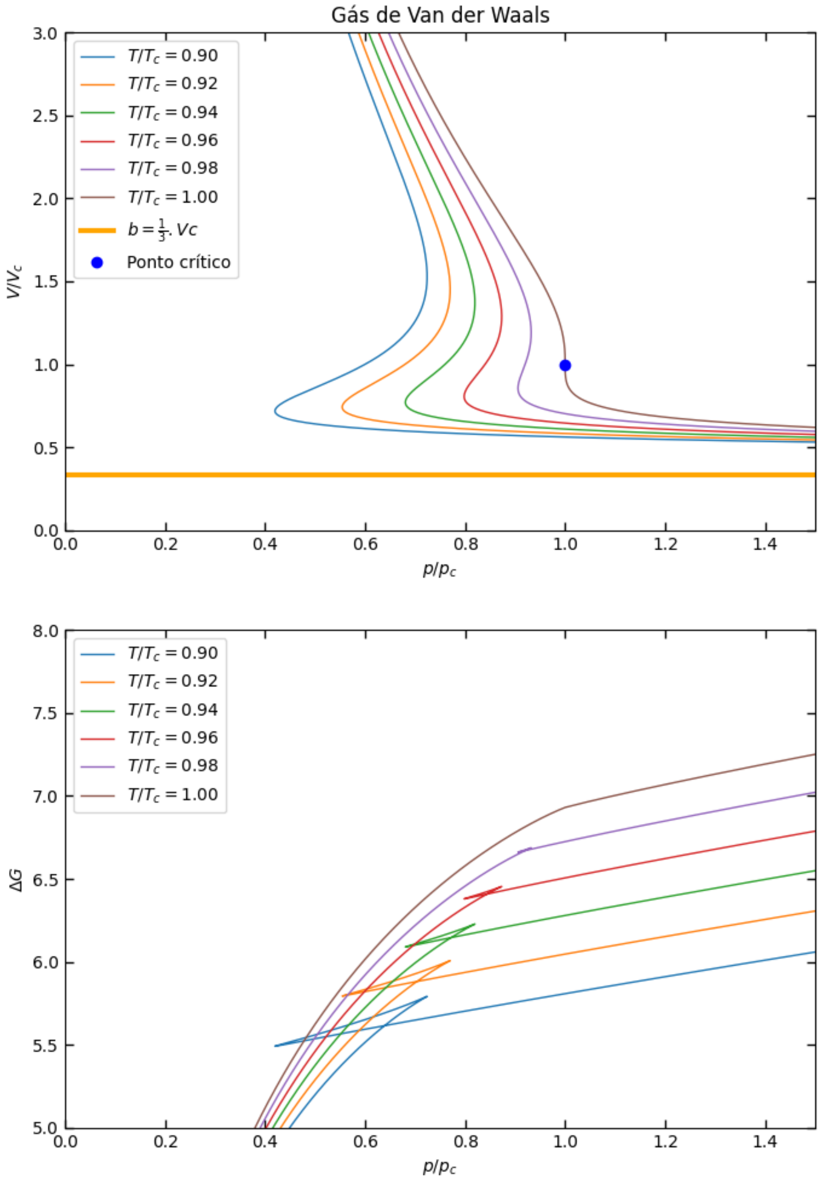


Figura 5: Comportamento do volume V e função de Gibbs do gás de van der Waals em função da pressão para 5 temperaturas

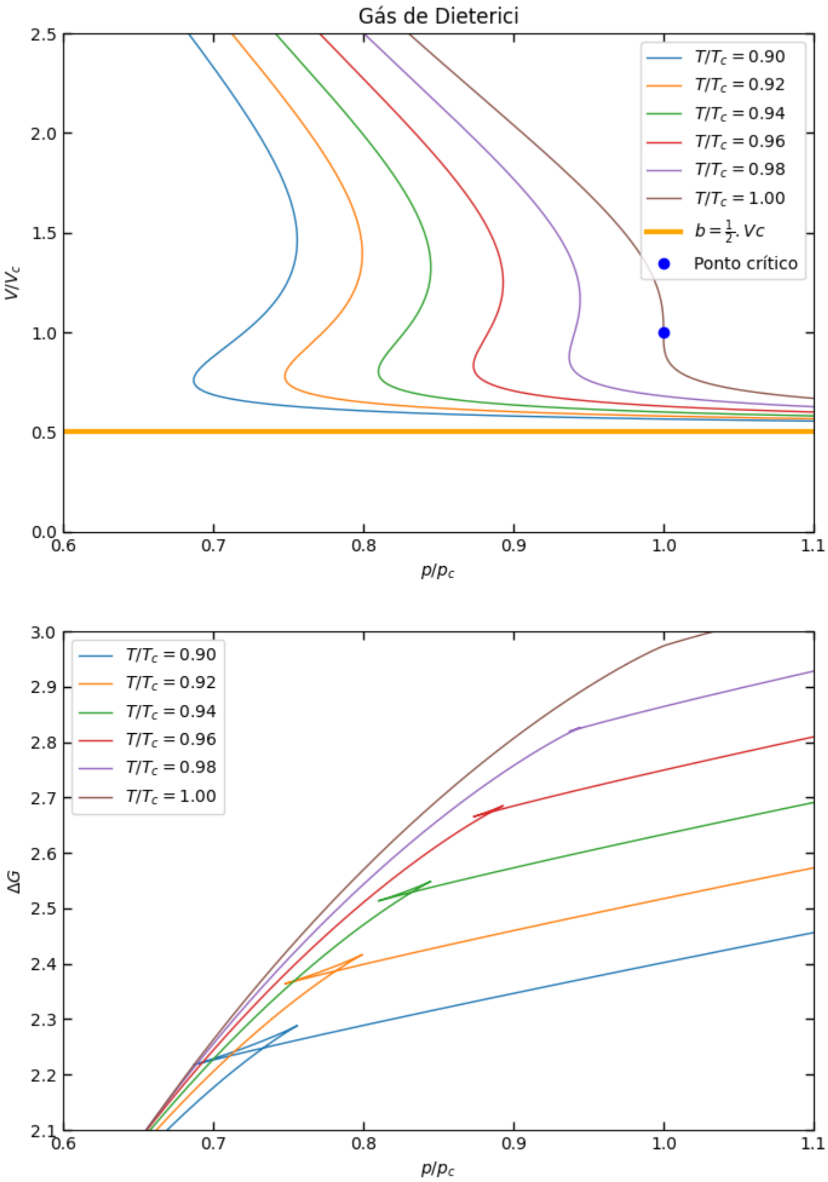


Figura 6: Comportamento do volume V e função de Gibbs do gás de Dieterici em função da pressão para 5 temperaturas

O código fonte e observações sobre sua execução seguem em anexo!

Seguem os documentos anexos.

OBS: O código fonte está formatado para ser compilado pelo programa "Jupyter Notebook" ou o Google Colaboratory, e deve ser executado em Python nas versões 3.6 ou superior. Os comentários foram escritos sem acentos e cedilha, por motivos de compatibilidade com o LaTeX

Recomendamos acesso ao código através do link:

https://colab.research.google.com/drive/1mCwXiKYV2RCo-j214VM4jXdpK9hPRJhw?usp=sharing

```
# -*- coding: utf-8 -*-
"""Termo 3.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1mCwXiKYV2RCo-
      j214VM4jXdpK9hPRJhw
"""

import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
import scipy.integrate as spit
from scipy.signal import argrelextrema

"""Constantes de van der Waals (fonte https://pt.wikipedia.org/wiki/Equa%C3%
   A7%C3%A3o_de_Van_der_Waals)

| Gas | a (litro  atm/mol ) | b (litro/mol) |
|:---:|:-------------------:|:-------------:|
| H2  | 0.2444              | 0.02661       |
| He  | 0.03412             | 0.02370       |
| N2  | 1.390               | 0.03913       |
| O2  | 1.360               | 0.03183       |
| CO  | 1.485               | 0.03985       |
| NO  | 1.340               | 0.02789       |
| CO2 | 3.592               | 0.04267       |
| H2O | 5.464               | 0.03049       |


R = 0,0820574587     L    atm    K^ 1     mol^ 1
"""

# Determinar a Temperatura Critica (Tc) para o gas da VDW
# Para o Gas O2
a = 1.360
b = 0.03183
R = 0.0820574587
Tc = 8*a/(27*R*b)
Pc = a/(27*b**2)
Vc = 3*b

# Define a equacao de estado reduzida para o gas de Van der Waals
def pVDW(V,T):
  p = (8*T/(3*V-1))-3/(V**2)
  return p
vpVDW = np.vectorize(pVDW) # Vetoriza a funcao para ser usada no pacote
```

numpy

```
# Inicializa as variaveis V e T e constroi o conjunto da variavel p(V,T)
V = np.linspace(0.25,3,20000)
T = np.linspace(0,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpVDW(V,t))

# Plota o resultado
fig,ax  = plt.subplots(figsize=(8,6),dpi=100)
ax.set(ylim=(0,2),xlim=(0,2))
ax.set(xlabel=r'$V/V_c$',ylabel=r'$p/p_c$',title='G s de Van der Waals')
for i in range(len(res)):
  ax.plot(V,res[i],c='gray',lw=0.5)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
  ')
ax.plot(V,vpVDW(V,1.),c='k',lw=3,label=f'Tc = {Tc:.2f} k')
ax.axvline(x=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax.plot(1.,1.,'o',c='b',label=f'Ponto cr tico')
ax.legend(loc='upper right')
plt.savefig('fig1 - isotermas',dpi=100)

#Plota o resultado com eixos espelhados (IGNORAR NO ARTIGO)

fig,ax  = plt.subplots(figsize=(8,6),dpi=100)
ax.set(ylim=(0,3),xlim=(0,2))
ax.set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Van der Waals')
for i in range(len(res)):
  ax.plot(res[i],V,c='gray',lw=0.5)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
  ')
ax.plot(vpVDW(V,1.),V,c='k',lw=3,label=f'Tc = {Tc:.2f} k')
ax.axhline(y=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax.plot(1.,1.,'o',c='b',label=f'Ponto cr tico')
ax.legend(loc='upper right')

# Redefine as variaveis e escolhemos um T especifico
V = np.linspace(0.25,50,30000)
T = np.linspace(0,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpVDW(V,t))
t = 25
# Inverte a ordem das listas, necessario para integracao numerica
rev_V = np.flip(V)
rev_p = np.flip(res[t])
print(T[t])
# Realiza a integracao numerica
resI=[]
for i,v in enumerate(rev_V):
  resI.append(spit.trapz(rev_V[0:i],rev_p[0:i]))
# Encontra os pontos extremos X e Y
imax = argrelextrema(rev_p, np.greater)
imin = argrelextrema(rev_p, np.less)
# Encontra os pontos B
idx=[]
```

```
erro_min = 1e5
for i,g in enumerate(resI[0:imax[0][0]]):
  for k,gg in enumerate(resI[imin[0][0]:]):
    erro = (gg-g)**2 + (-rev_p[i]+rev_p[k+imin[0][0]])**2
    if erro < erro_min:
      erro_min=erro
      idx.append((i,k+imin[0][0],erro_min,abs(gg-g),abs(-rev_p[i]+rev_p[k+
         imin[0][0]])))
idx
bidx = idx[-1]
print(bidx)

# Plota os graficos de V por p e de G por p
fig,ax  = plt.subplots(2,1,figsize=(8,12),dpi=100)
i,k,_,_,_ = bidx
ax[0].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
   'on')
ax[0].set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Van der Waals')
ax[0].plot(rev_p,rev_V,c='k',lw=2,label=fr'Isoterma $T/T_c = {T[t]:.2f}$')
#ax[0].plot(rev_p,rev_V,'ok',ms = 0.5,label=fr'Isoterma $T/T_c = {T[t]}$')
ax[0].plot(rev_p[imax[0][0]],rev_V[imax[0][0]],'o',label=f'Y')
ax[0].plot(rev_p[imin[0][0]],rev_V[imin[0][0]],'o',label=f'X')
ax[0].plot(rev_p[i],rev_V[i],'or',label=f'B')
ax[0].plot(rev_p[i],rev_V[k],'or')
ax[0].axvline(x=rev_p[imax[0][0]],ls='--',lw=1)
ax[0].axvline(x=rev_p[imin[0][0]],ls='--',lw=1)
ax[0].axvline(x=rev_p[i],ls='--',lw=1)
ax[0].axhline(y=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax[0].legend()

ax[1].set(ylim=(0,1),xlim=(0.5,1))
ax[1].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
   'on')
ax[1].plot(rev_p,resI,c='k',lw=2,label=fr'$\Delta G$')
#ax[1].plot(rev_p,resI,'ok',ms=0.5,label=fr'$\Delta G$')
ax[1].plot(rev_p[imax[0][0]],resI[imax[0][0]],'o',label=f'Y')
ax[1].plot(rev_p[imin[0][0]],resI[imin[0][0]],'o',label=f'X')
ax[1].plot(rev_p[i],resI[i],'or',label='B')
ax[1].axvline(x=rev_p[imax[0][0]],ls='--',lw=1)
ax[1].axvline(x=rev_p[imin[0][0]],ls='--',lw=1)
ax[1].axvline(x=rev_p[i],ls='--',lw=1)
ax[1].set(ylabel=r'$\Delta G$',xlabel=r'$p/p_c$')
ax[1].legend()
# Pode ser necessario ajuste manual dos limites do grafico para  tima
   apresentacao
xmin = 0
xmax = rev_p[imax[0][0]]+1
yminV = 0
ymaxV = rev_V[i]*1.20
yminG = 5
ymaxG = resI[imax[0][0]]+1
ax[0].set(ylim=(yminV,ymaxV),xlim=(xmin,xmax))
ax[1].set(ylim=(yminG,ymaxG),xlim=(xmin,xmax))
plt.savefig(f'fig2 - G por P T = {int(T[t]*100)}',dpi=100)

# Nesta parte e repetido o c digo anterior, porem calculando para uma faixa
    de valores de T
```

```
V = np.linspace(0.25,50,30000)
T = np.linspace(0,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpVDW(V,t))


tt = range(25,50)
Bidx=[]
for t in tt:
  rev_V = np.flip(V)
  rev_p = np.flip(res[t])
  resI=[]
  for i,v in enumerate(rev_V):
    resI.append(spit.trapz(rev_V[0:i],rev_p[0:i]))
  imax = argrelextrema(rev_p, np.greater)
  imin = argrelextrema(rev_p, np.less)
  idx=[]
  tabela=[]
  erro_min = 1e5
  for i,g in enumerate(resI[0:imax[0][0]]):
    for k,gg in enumerate(resI[imin[0][0]:]):
      erro = (gg-g)**2 + (-rev_p[i]+rev_p[k+imin[0][0]])**2
      if erro < erro_min:
        erro_min=erro
        idx.append((i,k+imin[0][0],erro_min,abs(gg-g),abs(-rev_p[i]+rev_p[k+
           imin[0][0]])))
  bidx = idx[-1]
  Bidx.append((t,(bidx[0],bidx[1])))
print(Bidx)

tabela=[]
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)

for t,(i,k) in Bidx:
  tabela.append((T[t],rev_p[t][i],rev_V[k],rev_V[i]))

tabela

# Plotagem no grafico invertido (IGNORAR NO ARTIGO)
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)
fig,ax  = plt.subplots(1,1,figsize=(8,6),dpi=100)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
   ')
ax.set(ylim=(0,10),xlim=(-1,2))
ax.set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Van der Waals')
for k in range(20,50):
  ax.plot(rev_p[k],rev_V,c='grey',lw=0.5)
ax.plot(rev_p[50],rev_V,c='k',lw=3,label='$T_c$')
ax.axhline(y=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax.plot(1,1,'o',c='b',label=f'Ponto cr tico')
ax.legend(loc='upper right')
for t,(i,k) in Bidx:
  ax.plot(rev_p[t][i],rev_V[i],'ro',label=f'B')
  ax.plot(rev_p[t][i],rev_V[k],'bo',label=f'B')
```

```
# Plotamos os resultados obtidos anteriormene num grafico p por V tal qual o
    primeiro, destacando a linha de mudanca de fase
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)
fig,ax = plt.subplots(1,1,figsize=(8,6),dpi=100)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
    ')
ax.set(xlim=(0,5),ylim=(0,1.25))
ax.set(xlabel=r'$V/V_c$',ylabel=r'$p/p_c$',title='G s de Van der Waals')
for k in range(50):
  ax.plot(rev_V,rev_p[k],c='grey',lw=0.5)
ax.plot(rev_V,rev_p[50],c='k',lw=3,label='$T_c$')
ax.axvline(x=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax.plot(1,1,'ro',label = 'Linha de separa  o de fases')
ax.plot(1,1,'o',c='k',label=f'Ponto cr tico')
for t,(i,k) in Bidx:
  ax.plot(rev_V[i],rev_p[t][i],'ro')
  ax.plot(rev_V[k],rev_p[t][i],'ro')
ax.legend(loc='upper right')
plt.savefig('fig3 - Linha Fases ',dpi=100)

# Redefinimos as variaveis tal qual anteriormente
V = np.linspace(0.25,20,10000)
T = np.linspace(0,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpVDW(V,t))
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)
print(np.shape(res))
print(np.shape(rev_p))

# Agora faremos a mesma operacao definida anteriormente, porem generalizando
    para todos os T , definimos uma matriz 'a' que guarda o G para cada T
a = np.zeros(shape=(51,10000))
print(np.shape(a))
for k in range(51):
  for i,v in enumerate(rev_V):
    a[k][i] = spit.trapz(rev_V[0:i],rev_p[k][0:i])
print(np.shape(a))

# Semelhante ao grafico anterior, porem aqui escolhe varios valores de T
   para apresentacao

fig,ax = plt.subplots(2,1,figsize=(8,12),dpi=100)
ax[0].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
    'on')
ax[0].set(ylim=(0,3),xlim=(0,1.5))
ax[0].set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Van der Waals')
for k in range(45,51,1):
  ax[0].plot(rev_p[k],rev_V,lw=1,label=f'$T/T_c = {T[k]:.2f}$')
#ax[0].plot(rev_p[50],rev_V,c='k',lw=3,label='$T_c$')
ax[0].axhline(y=1/3, label=r'$b = \frac{1}{3} . Vc$', c='orange',lw=3)
ax[0].plot(1,1,'o',c='b',label=f'Ponto cr tico')
ax[0].legend(loc='upper left')
```

```
ax[1].set(ylim=(5,8),xlim=(0,1.5))
ax[1].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
    'on')
for k in range(45,51,1):
  ax[1].plot(rev_p[k],a[k][:],lw=1,label=f'$T/T_c = {T[k]:.2f}$')
ax[1].set(ylabel=r'$\Delta G$',xlabel=r'$p/p_c$')
ax[1].legend()
plt.savefig('fig4 - Varias G por p',dpi=100)

"""GAS DE DIETERICI"""

# Determinar a Temperatura Critica (Tc) para o gas de Dieterici
# Para o Gas O2
a = 1.360
b = 0.03183
R = 0.0820574587
Tc = 8*a/(27*R*b)
Pc = a/(27*b**2)
Vc = 3*b

# Define a equacao de estado para o gas de Dieterici
def pDie(V,T):
  return T*np.exp(2 - 2/(T*V))/(2*V-1)
vpDie = np.vectorize(pDie)

# Inicializa as variaveis V e T e constr i o conjunto da variavel p(V,T)
V = np.linspace(0.25,5,10000)
T = np.linspace(0.12,2.12,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpDie(V,t))

# Plota o resultado
fig,ax  = plt.subplots(figsize=(8,6),dpi=100)
ax.set(ylim=(0,2),xlim=(0,2))
ax.set(xlabel=r'$V/V_c$',ylabel=r'$p/p_c$',title='G s de Dieterici')
for i in range(len(res)):
  ax.plot(V,res[i],c='gray',lw=0.5)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
    ')
ax.plot(V,vpDie(V,1.),c='k',lw=3,label=f'Tc')
ax.axvline(x=1/2, label=r'$b = \frac{1}{2} . Vc$', c='orange',lw=3)
ax.plot(1.,1.,'o',c='b',label=f'Ponto cr tico')
ax.legend(loc='upper right')
plt.savefig('fig1DIE - isotermas',dpi=100)

# Redefine as variaveis e escolhemos um T especifico
V = np.linspace(0.25,10,10000)
T = np.linspace(0.5,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpDie(V,t))
t = 45

# Inverte a ordem das listas , necessario para integracao numerica
rev_V = np.flip(V)
rev_p = np.flip(res[t])
```

```
print(T[t])
# Realiza a integracao nuerica
resI=[]
for i,v in enumerate(rev_V):
  resI.append(spit.trapz(rev_V[0:i],rev_p[0:i]))
# Encontra os pontos extremos X e Y
imax = argrelextrema(rev_p, np.greater)
imin = argrelextrema(rev_p, np.less)
# Encontra os pontos B
idx=[]
erro_min = 1e10
for i,g in enumerate(resI[0:imax[0][0]]):
  for k,gg in enumerate(resI[imin[0][0]:]):
    erro = (gg-g)**2 + (-rev_p[i]+rev_p[k+imin[0][0]])**2
    if erro < erro_min:
      erro_min=erro
      idx.append((i,k+imin[0][0],erro_min,abs(gg-g),abs(-rev_p[i]+rev_p[k+
         imin[0][0]])))
idx
bidx = idx[-1]
print(bidx)

# Plota os graficos de V por p e de G por p
fig,ax  = plt.subplots(2,1,figsize=(8,12),dpi=100)
i,k,_,_,_ = bidx
ax[0].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
   'on')
ax[0].set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Dieterici')
ax[0].plot(rev_p,rev_V,c='k',lw=2,label=fr'Isoterma $T/T_c = {T[t]:.2f}$')
#ax[0].plot(rev_p,rev_V,'ok',ms = 0.5,label=fr'Isoterma $T/T_c = {T[t]}$')
ax[0].plot(rev_p[imax[0][0]],rev_V[imax[0][0]],'o',label=f'Y')
ax[0].plot(rev_p[imin[0][0]],rev_V[imin[0][0]],'o',label=f'X')
ax[0].plot(rev_p[i],rev_V[i],'or',label=f'B')
ax[0].plot(rev_p[i],rev_V[k],'or')
ax[0].axvline(x=rev_p[imax[0][0]],ls='--',lw=1)
ax[0].axvline(x=rev_p[imin[0][0]],ls='--',lw=1)
ax[0].axvline(x=rev_p[i],ls='--',lw=1)
ax[0].axhline(y=1/2, label=r'$b = \frac{1}{2} . Vc$', c='orange',lw=3)
ax[0].legend()

ax[1].set(ylim=(0,1),xlim=(0.5,1))
ax[1].tick_params(direction='in',length=5, width = 1 , right = 'on' , top =
   'on')
ax[1].plot(rev_p,resI,c='k',lw=2,label=fr'$\Delta G$')
#ax[1].plot(rev_p,resI,'ok',ms=0.5,label=fr'$\Delta G$')
ax[1].plot(rev_p[imax[0][0]],resI[imax[0][0]],'o',label=f'Y')
ax[1].plot(rev_p[imin[0][0]],resI[imin[0][0]],'o',label=f'X')
ax[1].plot(rev_p[i],resI[i],'or',label='B')
ax[1].axvline(x=rev_p[imax[0][0]],ls='--',lw=1)
ax[1].axvline(x=rev_p[imin[0][0]],ls='--',lw=1)
ax[1].axvline(x=rev_p[i],ls='--',lw=1)
ax[1].set(ylabel=r'$\Delta G$',xlabel=r'$p/p_c$')
ax[1].legend()
# Pode ser necessario ajuste manual dos limites do grafico para  tima
   apresentacao
xmin = 0.8
xmax = 0.9
```

```
yminV = 0.25
ymaxV = 2
yminG = 2.50
ymaxG = 2.75
ax[0].set(ylim=(yminV,ymaxV),xlim=(xmin,xmax))
ax[1].set(ylim=(yminG,ymaxG),xlim=(xmin,xmax))
plt.savefig(f'fig2DIE␣-␣G␣por␣P␣T␣=␣{int(T[t]*100)}',dpi=100)

# Nesta parte e repetido o c digo anterior, porem calculando para uma faixa
    de valores de T
tt = range(25,50)
Bidx=[]
for t in tt:
  rev_V = np.flip(V)
  rev_p = np.flip(res[t])
  resI=[]
  for i,v in enumerate(rev_V):
    resI.append(spit.trapz(rev_V[0:i],rev_p[0:i]))
  imax = argrelextrema(rev_p, np.greater)
  imin = argrelextrema(rev_p, np.less)
  idx=[]
  idxf=[]
  erro_min = 1e5
  for i,g in enumerate(resI[0:imax[0][0]]):
    for k,gg in enumerate(resI[imin[0][0]:]):
      erro = (gg-g)**2 + (-rev_p[i]+rev_p[k+imin[0][0]])**2
      if erro < erro_min:
        erro_min=erro
        idx.append((i,k+imin[0][0],erro_min,abs(gg-g),abs(-rev_p[i]+rev_p[k+
            imin[0][0]])))
  bidx = idx[-1]
  Bidx.append((t,(bidx[0],bidx[1])))
print(Bidx)

# Plotamos os resultados obtidos anteriormene num grafico p por V tal qual o
    primeiro, destacando a linha de mudanca de fase
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)
fig,ax  = plt.subplots(1,1,figsize=(8,6),dpi=100)
ax.tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on
    ')
ax.set(xlim=(0,3),ylim=(0.4,1.2))
ax.set(xlabel=r'$V/V_c$',ylabel=r'$p/p_c$',title='G s␣de␣Dieterici')
for k in range(50):
  ax.plot(rev_V,rev_p[k],c='grey',lw=0.5)
ax.plot(rev_V,rev_p[50],c='k',lw=3,label='$T_c$')
ax.axvline(x=1/2, label=r'$b␣=␣\frac{1}{2}␣.␣Vc$', c='orange',lw=3)
ax.plot(1,1,'ro',label = 'Linha␣de␣separa  o␣de␣fases')
ax.plot(1,1,'o',c='k',label=f'Ponto␣cr tico')
for t,(i,k) in Bidx:
  ax.plot(rev_V[i],rev_p[t][i],'ro')
  ax.plot(rev_V[k],rev_p[t][i],'ro')
ax.legend(loc='upper␣right')
plt.savefig('fig3DIE␣-␣Linha␣Fases␣',dpi=100)

# Redefinimos as variaveis tal qual anteriormente
V = np.linspace(0.25,10,10000)
```

```python
T = np.linspace(0.5,1,51,endpoint=True)
res=[]
for i,t in enumerate(T):
  res.append(vpDie(V,t))
rev_V = np.flip(V)
rev_p = np.flip(res,axis=-1)
print(np.shape(res))
print(np.shape(rev_p))

# Agora faremos a mesma operacao definida anteriormente, porem generalizando para todos os T , definimos uma matriz 'a' que guarda o G para cada T
a = np.zeros(shape=(51,10000))
print(np.shape(a))
for k in range(51):
  for i,v in enumerate(rev_V):
    a[k][i] = spit.trapz(rev_V[0:i],rev_p[k][0:i])
print(np.shape(a))

# Semelhante ao grafico anterior, porem aqui escolhe varios valores de T para apresentacao

fig,ax  = plt.subplots(2,1,figsize=(8,12),dpi=100)
ax[0].tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on')
ax[0].set(ylim=(0,3),xlim=(0.3,1.1))
ax[0].set(ylabel=r'$V/V_c$',xlabel=r'$p/p_c$',title='G s de Dieterici')
for k in range(30,51,4):
  ax[0].plot(rev_p[k],rev_V,lw=1,label=f'$T/T_c = {T[k]:.2f}$')
#ax[0].plot(rev_p[50],rev_V,c='k',lw=3,label='$T_c$')
ax[0].axhline(y=1/2, label=r'$b = \frac{1}{2} . Vc$', c='orange',lw=3)
ax[0].plot(1,1,'o',c='b',label=f'Ponto cr tico')
ax[0].legend()

ax[1].set(ylim=(1.4,3.1),xlim=(0.3,1.1))
ax[1].tick_params(direction='in',length=5, width = 1 , right = 'on' , top = 'on')
for k in range(30,51,4):
  ax[1].plot(rev_p[k],a[k][:],lw=1,label=f'$T/T_c = {T[k]:.2f}$')
ax[1].set(ylabel=r'$\Delta G$',xlabel=r'$p/p_c$')
ax[1].legend()
plt.savefig('fig4DIE - Varias G por p',dpi=100)
```