FIS 444 - Termodinâmica 2 - 2º Trabalho

Humberto Sousa Martins - 93724

19/07/2022

Definição da Função Polilogarítmica

A função polilogarítmica, definida na equação (1), será de uso recorrente neste trabalho. Logo, sua correta implementação computacional foi de primeira necessidade.

$$Li_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n} \tag{1}$$

A implementação através da relação de recorrência resulta em uma série convergente para z no intervalo [0,1] e $n > 1$. Estamos especialmente interessados nos valores da função polilogarítmica com $n = 3/2$ e $n = 5/2$, pois estas aparecerão com frequência nos cálculos a seguir. Os resultados destas funções para z no intervalo [0,1] estão representadas na figura 1. Entretanto, apesar da implementação ter funcionado corretamente, conforme demonstrado na figura 1, seu tempo de processamento foi desnecessariamente alto. Logo, foi preferido usar a implementação do pacote *mpmath*, que usa funções computacionalmente otimizadas para o cálculo da função polilogarítmica. Uma comparação entre tempos de processamento mostrou que a função definida manualmente levou 233 s para gerar o gráfico da figura 1, para Li com $n = 3/2$ e $n = 5/2$ e com 10000 pontos no intervalo [0,1], enquanto a implementação do pacote *mpmath* levou apenas 0,515 s.

A comparação dos erros computacionais mostrou que a implementação manual está dentro de uma faixa aceitável de erro, sendo que durante todo intervalo entre [0,1) o erro está na faixa de 10^{-16}, apenas no caso de $z = 1$ na função com $n = 3/2$ ocorre uma divergência maior da ordem de 2.0×10^{-2}

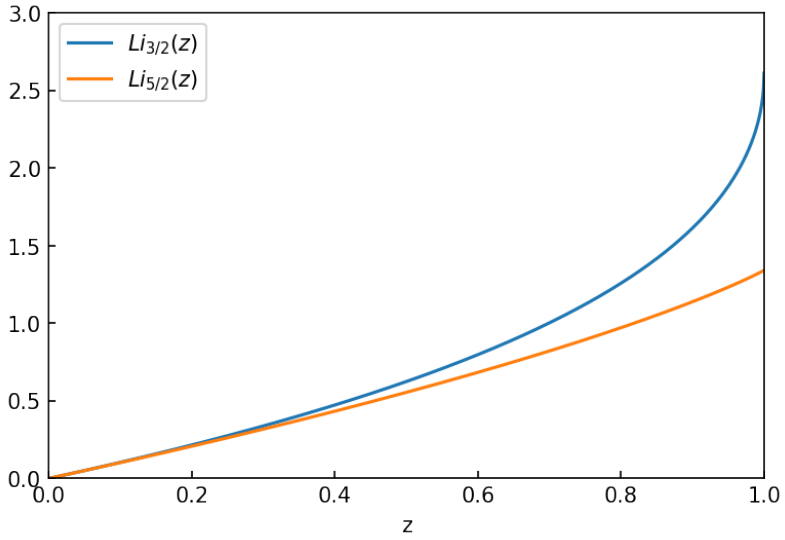

Figura 1: Gráfico dos valores da função Polilogarítmica para n = 3/2 e n=5/2, obtidos a partir da implementação manual da equação 1.

Fugacidade $\mu(z)$

A fugacidade é definida por $z = e^{\beta\mu}$ e é muitas vezes usada como parâmetro livre nas equações que demonstraremos à seguir. Podemos relacionar a fugacidade à temperatura crítica T_c de bósons através da equação 2, que é bem definida para $0 \leq z < 1$ ou para $T/T_c \geq 1$. Para valores de $T/T_c < 1$ consideramos que $n \approx 1$.

Apesar de não podermos inverter diretamente a função, através de uma inversão de eixos podemos ver o comportamento de z variando com T/T_c. A figura 2 mostra o gráfico correspondente.

$$\frac{T}{T_c} = \left[\frac{\zeta(3/2)}{Li_{3/2}(z)}\right]^{2/3} \tag{2}$$

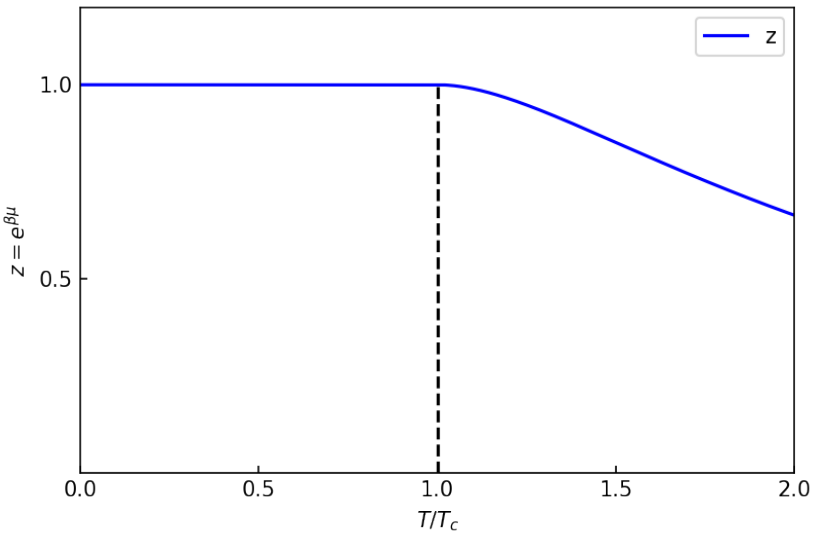

Figura 2: Gráfico dos valores de z em função de T/T_c, .

Energia Interna $U(T)$

Para definir a energia interna, será necessário separar o problema em $T/T_c < 1$ e $T/T_c > 1$, conforme demonstrado no exemplo 30.8. O resultado é equação 3. Na parte $T/T_c < 1$ usamos T/T_c como variável livre, enquanto na parte $T/T_c > 1$ usamos z como variável independente, usando a equação 2 para relacionar T/T_c em função de z. O resultado é mostrado na figura 3. Em $T = T_c$ as partes da equação se igualam.

$$\frac{U}{Nk_bT_c} = \begin{cases} \frac{3}{2}\frac{\zeta(5/2)}{\zeta(3/2)}\left(\frac{T}{T_c}\right)^{5/2} & \text{se } T/T_c \leq 1 \\ \frac{3}{2}\left(\frac{T}{T_c}\right)\frac{Li_{5/2}(z)}{Li_{3/2}(z)} & \text{se } T/T_c \geq 1 \end{cases} \tag{3}$$

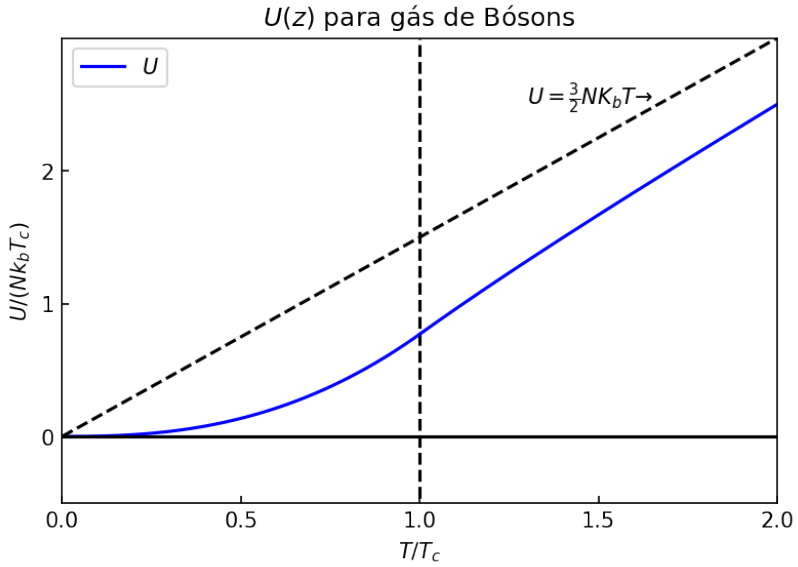

Figura 3: Gráfico de U em função de T/T_c. A linha pontilhada transversal mostra a energia interna esperada de um gás ideal, enquanto a vertical marca o ponto $T = T_c$.

Ampliando a área de visualização, percebemos que a energia interna U tende à assíntota no limite de T/T_c grande, conforme mostrado na figura 4.

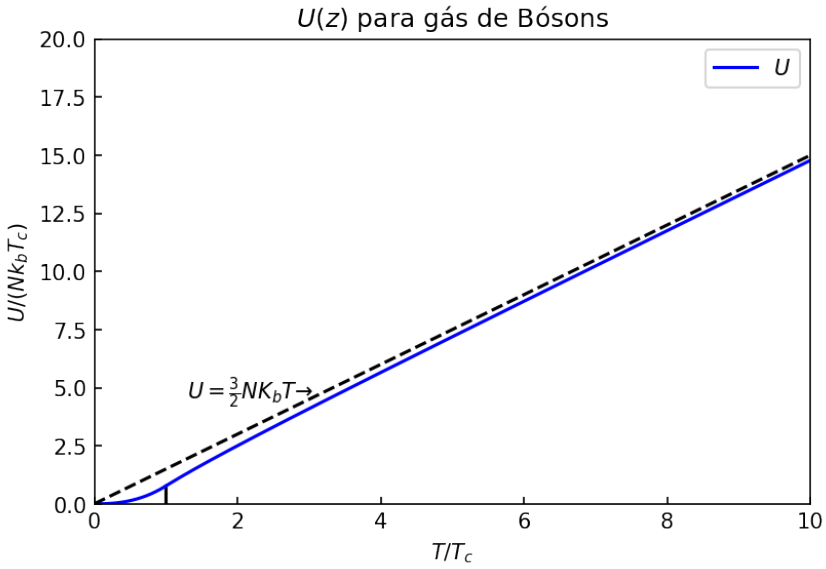

Figura 4: Gráfico mostrando a convergência de U em função de T/T_c com a assíntota da energia interna esperada de um gás ideal para altas temperaturas.

Capacidade térmica C_v

Para derivar a capacidade térmica do gás de bósons, partimos da equação 3 e derivamos em relação à T a volume constante, separando novamente em duas partes para T maior ou menor que T_c.

$$C_v = \left(\frac{\partial U}{\partial T}\right)_V \tag{4}$$

Para $T/T_c < 1$ a conta é bastante direta e resulta em:

$$C_v = \frac{3}{2}Nk_bT_c\frac{\zeta(5/2)}{\zeta(3/2)}\frac{5}{2}\left(\frac{T}{T_c}\right)^{3/2}\left(\frac{1}{T_c}\right) \text{ para } T/T_c \leq 1 \tag{5}$$

$$\frac{C_v}{Nk_b} = \frac{15}{4}\frac{\zeta(5/2)}{\zeta(3/2)}\left(\frac{T}{T_c}\right)^{3/2} \text{ para } T/T_c \leq 1 \tag{6}$$

Para $T/T_c > 1$ a conta é bastante mais longa, para simplificar a notação, quando escrito Li_n entenda como $Li_n(z)$, pois toda função é parametrizada em z. Usaremos a propriedade de derivada de funções polilogarítmicas:

$$\frac{\partial Li_n(z)}{\partial z} = \frac{\partial}{\partial z}\sum_{k=1}^{\infty}\frac{z^k}{k^n} = \sum_{k=1}^{\infty}\frac{z^{k-1}}{k^{n-1}} = \frac{Li_{n-1}(z)}{z} \tag{7}$$

Para encontrarmos a dependência de T em z, fazemos a derivada total da equação 2:

$$\frac{dT}{T_c} = \frac{-2}{3z}\zeta(3/2)^{2/3} * (Li_{3/2})^{-5/2} * Li_{1/2} * dz \tag{8}$$

$$\frac{dz}{dT} = \frac{-3z}{2T_c}\zeta(3/2)^{-2/3} * (Li_{3/2})^{5/2} * (Li_{1/2})^{-1} \tag{9}$$

Partindo então da segunda parte da equação 3 e substituindo T/T_c conforme a equação 2 temos:

$$\frac{U}{Nk_bT_c} = \frac{3}{2}\zeta(3/2)^{2/3}Li_{5/2} * (Li_{3/2})^{-5/3} \tag{10}$$

Aplicando a derivada parcial em T e a regra da cadeia nos Li obtemos:

$$\frac{C_v}{Nk_bT_c} = \frac{3}{2}\zeta(3/2)^{2/3}\frac{1}{z}\left[(Li_{3/2})^{-2/3} - \frac{5}{3}Li_{5/2} * (Li_{3/2})^{-8/3} * Li_{1/2}\right]\frac{dz}{dT} \tag{11}$$

Substituindo dz/dT obtido na equação 9, muitos termos se cancelam, sobrando somente:

$$\frac{C_v}{Nk_b} = \frac{3}{2}\left[\frac{5Li_{5/2}}{2Li_{3/2}} - \frac{3Li_{3/2}}{2Li_{1/2}}\right] \text{ para } T/T_c \geq 1 \tag{12}$$

O gráfico da função obtida combinando as duas partes (equações 6 e 12) concorda com o mostrado no livro-texto, e é apresentado na figura 5:

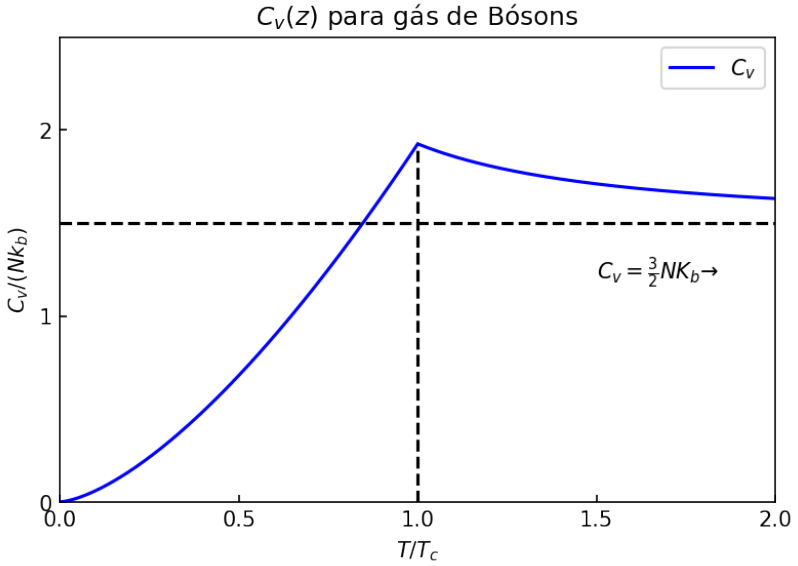

Figura 5: Gráfico de C_v em função de T/T_c, a linha pontilhada horizontal representa o valor esperado para um gás ideal.

Potencial químico μ para bósons e férmions

O potencial químico para bósons e férmions pode ser derivado da definição de $z = e^{\beta\mu}$, aplicando o ln dos dois lados obtemos a equação 13, basta então determinar a forma de $T(z)$ para férmios e bósons.

$$\ln(z) = \beta\mu \quad \text{ou} \quad \mu(z) = k_b T(z)\ln(z) \tag{13}$$

Para férmions, partimos das relações:

$$\varepsilon_f = \frac{\hbar^2}{2m}\left(\frac{6\pi^2 n}{2S+1}\right)^{2/3}, \quad \lambda_{th} = \frac{h}{\sqrt{2\pi m k_b T}} \quad \text{e} \quad T_f = \frac{\varepsilon_f}{k_b} \tag{14}$$

Podemos relacionar essas equações de maneira que:

$$\left(\frac{T_f}{T}\right)^{3/2} = \frac{\lambda_{th}^3}{(2S+1)}\frac{3\sqrt{\pi}n}{4} \tag{15}$$

Isolando o termo em T obtemos:

$$T(z) = T_f\left[\frac{-4}{3\sqrt{\pi}Li_{3/2}(-z)}\right]^{2/3} \text{ para } \quad 0 \leq z \leq \infty \tag{16}$$

Usando o resultado de 16 em 13, obtemos o seguinte gráfico para o potencial químico de férmions:

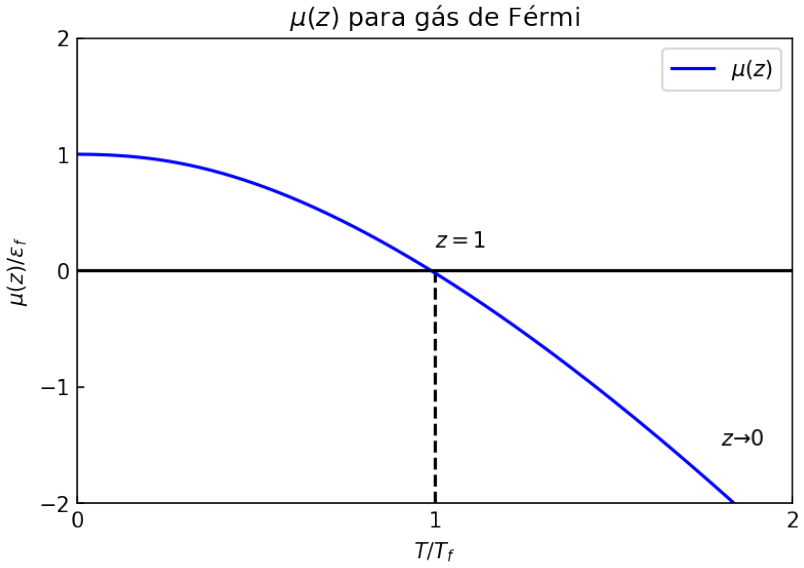

Figura 6: Gráfico do potencial químico normalizado pela energia de férmi μ/ϵ_f em função da temperatura normalizada pela temperatura de férmi T/T_f para gás de férmi.

Para bósons, basta partirmos da equação 2 que relaciona T com T_c, nos limitando ao intervalo $0 \leq z \leq 1$ e $T/T_c \geq 1$, obtendo o seguinte gráfico:

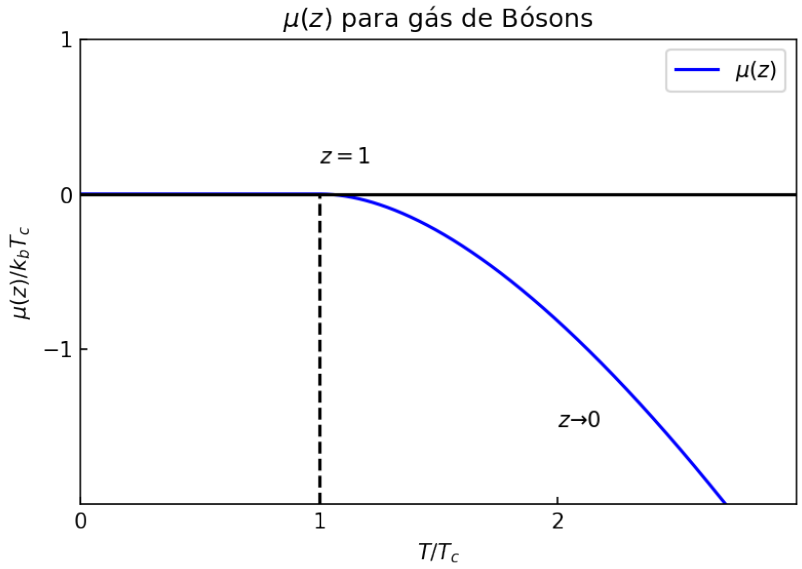

Figura 7: Gráfico do potencial químico normalizado pela constante de boltzman e temperatura crítica μ/k_bT_c em função da temperatura normalizada pela temperatura de crítica T/T_c para gás de bósons.

Energia interna U pelo somatório

Para calcular a energia interna sem dispôr da aproximação para integral no intervalo $T/T_c < 1$, partimos da definição de U:

$$U =< \epsilon >= \sum_{k=1}^{\infty} n_k \epsilon_k \tag{17}$$

Temos também as relações e n_k definido para bósons pela equação 30.8 do livro-texto:

$$N = \sum_{k=1}^{\infty} n_k \quad , \quad \epsilon_k = \frac{\hbar^2 k^2}{2m} \quad \text{e} \quad n_k = (e^{\beta(\epsilon_k - \mu)} - 1)^{-1} \tag{18}$$

Simplificamos a notação dos somatórios para maior clareza visual, lembrando que estamos somando de 1 a infinito em todos os casos. Usando as normalizações relevantes mostradas na seção 3 obtemos:

$$\frac{U}{N k_b T_c} = \frac{\sum n_k \epsilon_k}{\sum n_k k_b T_c} \tag{19}$$

No intervalo de interesse, abaixo da temperatura crítica, temos que $\mu = 0$. Definimos também unidades adimensionais de T e ϵ que vão facilitar as contas:

$$T^* = T/T_c \quad \text{e} \quad \epsilon^* = \frac{\hbar^2}{2m k_b T_c} \tag{20}$$

Logo a equação 19 resulta em:

$$\frac{U}{N k_b T_c} = \frac{\sum (e^{\epsilon_k / k_b T_c} - 1)^{-1} \epsilon_k}{\sum (e^{\epsilon_k / k_b T_c} - 1)^{-1} k_b T_c} = \frac{\sum (e^{\epsilon^* k^2 / T^*} - 1)^{-1} \epsilon^* k^2}{\sum (e^{\epsilon^* k^2 / T^*} - 1)^{-1}} \tag{21}$$

Usando $\epsilon^* = 1$, implementamos o somatório em k de 1 a infinito com um somatório (implementado manualmente) com k de 1 a 1000, pois observamos que os termos normalmente convergem rapidamente mesmo para k menores que 10, para $0 \leq T^* \leq 1$, obtemos o seguinte gráfico:

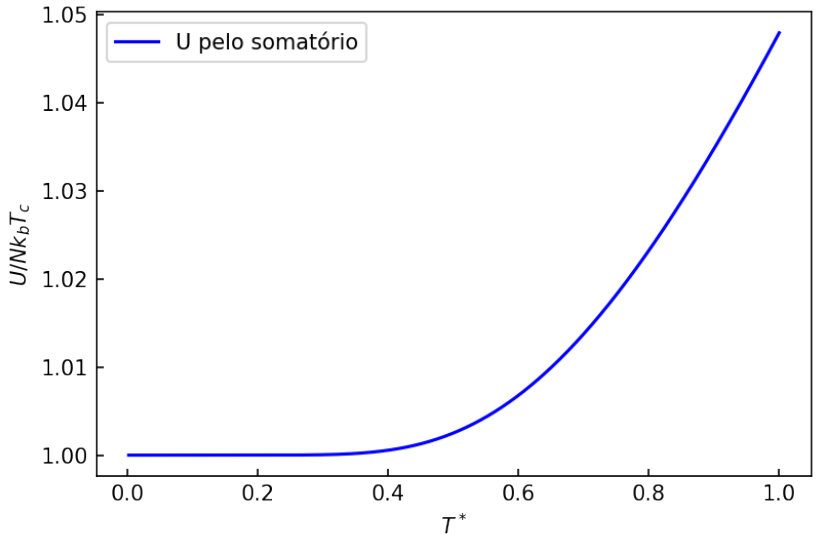

Figura 8: Gráfico da energia interna normalizada pelo numero de partículas, constante de Boltzmann e temperatura crítica U/Nk_bT_c em função da temperatura normalizada pela temperatura de crítica $T^* = T/T_c$ para gás de bósons.

Observamos que a forma geral da curva do gráfico da figura 8 corresponde à esperada, porém devido à escolha de $\epsilon^* = 1$ a escala está errada em comparação com o obtido pela integral na seção 3, onde $U(T^*)$ variava entre 0 e 0,77 no intervalo T^* de 0 a 1. Foi realizada uma busca manual de valores numéricos de ϵ^* que aproximassem do intervalo esperado, porém sem sucesso.

A solução encontrada para uma montagem gráfica que permitisse a comparação entre as duas curvas, embora longe de ideal, foi reajustar manualmente a escala de modo a corresponder à obtida na seção 3. Para tal, os valores da função obtida foram deslocados em 1 unidade para baixo e multiplicados por 16, de modo que

$$U(T^*)_{ajustado} = (U(T^*) - 1) * 16 \tag{22}$$

Montando o gráfico das duas soluções juntas, obtemos finalmente:

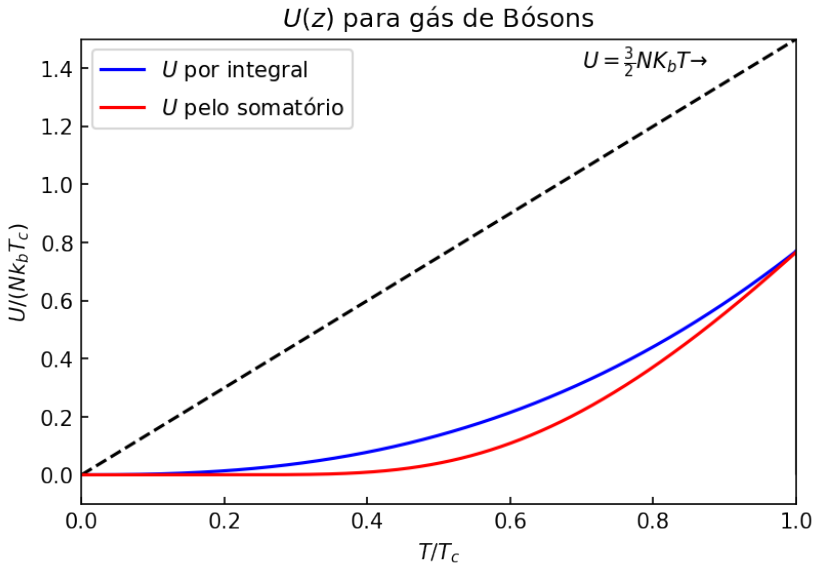

Figura 9: Gráfico da energia interna normalizada pelo numero de partículas, constante de Boltzmann e temperatura crítica U/Nk_bT_c em função da temperatura normalizada pela temperatura de crítica $T^* = T/T_c$ para gás de bósons. A linha azul é a calculada através da integral e a vermelha pelo somatório.

Anexo: código fonte

Segue em anexo o código fonte utilizado, também disponível para fácil visualização e execução online na plataforma Google Colaboratory através do link: https://colab.research.google.com/drive/18tR57DNFsmtiUQVZ$_B a7kOoOvaVLIoEI?usp = sharing$

Bibliotecas e funções básicas

```
# Importa as bibliotecas relevantes
from tqdm.notebook import trange, tqdm
import numpy as np
import matplotlib.pyplot as plt
import math
import mpmath
```

```
# Define uma função básica de plot, que pode ser acrescentada ou modificada

def plot(x,y,label,xlabel,ylabel,color):
  fig,ax = plt.subplots(1,figsize=(6,4), dpi=150)
  ax.plot(x,y,label=label,color=color)
  ax.set(xlabel=xlabel,ylabel=ylabel)
  ax.legend()
  ax.tick_params(direction='in')
  return fig,ax
```

Definição da função Polilogarítmica

```
# Começamos definindo a função polilogaritmica
# Argumentos: iter é quantas vezes vamos iterar a função
# A parte comentada serve para debug da função

def polylogh(n,z,iter=10000, erro = 1e-6, verbose = 0, historico = 0):
  termo=soma=0
  hist = []
  for k in (range(1,iter)):
    termo = z**k / k**n
    soma += termo
  #  hist.append((termo,soma))
  #   if verbose ==1:
  #     if termo/soma < erro:
  #       print(f"A função para n = {n} e z = {z} convergiu em {soma:.3f} em {len(h
  #       break
  #     if termo/soma > 1e5:
  #       print(f"A função diverge rapidamente")
  #       break

  # if historico == 1:
  #   return soma, hist
  else:
    return soma

# Define a função polilogarítmica do pacote mpmath, tomando o valor numérico da par
def polylogp(n,z):
  return mpmath.fp.polylog(n,z).real

# Vetoriza ambas as funções para fácil uso com arrays numpy, a função vetorizada re
# a definida pelo pacote tem um sufixo p (de pacote)
vpolylogh = np.vectorize(polylogh)
vpolylogp = np.vectorize(polylogp)

# Calcula os resultados de Li para n=3/2 e n=5/2 em 0<z<1
espaco = np.linspace(0,1,1000)
res3_2 = vpolylogp(1.5,espaco)
res5_2 = vpolylogp(2.5,espaco)

# Plota os resultados
ax,fig=plt.subplots(1,1,figsize=(6,4),dpi=150)
plt.plot(espaco, res3_2, label=r"$Li_{3/2}(z)$")
plt.plot(espaco, res5_2,label=r"$Li_{5/2}(z)$")
plt.legend(loc='upper left')
plt.xlabel("z")
plt.tick_params(direction='in')
plt.xlim(0,1)
plt.ylim(0,3)
```

Gráfico de z(T)

```
# usamos a função 30.62 para calcular numericamente os valores de T(z), e invertemos
# A função T faz o papel de T/Tc

def T(z):
  if z>=1:
    return 0
  else:
    return (2.612375348685488/vpolylogp(3/2,z))**(2/3)
T=np.vectorize(T)

# Calcula para o espaço de z entre 0 e 1 e plota]

zspace=np.linspace(0.5,1,1000)
T_res = T(zspace)
fig,ax = plot(T_res,zspace,"z",r"$T/T_c$",r"$z=e^{\beta\mu}$",'b');
# ax.scatter(T_res,zspace)
ax.set(xlim=(0.1,2),ylim=(0,1.2))
ax.set(xticks=[0,0.5,1,1.5,2],yticks=[0.5,1])
ax.vlines(1,0,1,'k','--')
```

$U(z)$ de bósons

```
# A função T faz o papel de T/Tc
def T(z):
  return (2.612375348685488/vpolylogp(3/2,z))**(2/3)
T=np.vectorize(T)

def Uz(z):
  # em T>Tc , z<1
  return (3/2)*T(z)*(vpolylogp(5/2,z)/vpolylogp(3/2,z))
Uz = np.vectorize(Uz)

def Ut(Tspace):
  # em T<Tc , z>1
  return (0.77)*Tspace**(5/2)
Ut = np.vectorize(Ut)

# Calculamos para z entre 0 e 1
zspace = np.linspace(0.01,1,10000)
Tspace = np.linspace(0,1,10000)
T_res = T(zspace)
U_res = Uz(zspace)
x_space = np.linspace(0,10,100)

# SEM ZOOM
# Plotamos a função obtida
fig,ax = plot(Tspace,Ut(Tspace),"$U$",r"$T/T_c$",r"$U/(Nk_bT_c)$","b");
ax.plot(x_space,(3/2)*x_space,'--k')
ax.plot(T_res,U_res,'b')
```

```
ax.annotate(r"$U=\frac{3}{2}NK_bT \to$",(1.3,4.5),xycoords='data')
# ax.annotate(r"$z\to0$",(2,-1.5),xycoords='data')
ax.set(xlim=(0,10),ylim=(0,20))
# ax.set(xticks=[0,.5,1,1.5,2],yticks=[0,1,2])
# ax.axhline(0,color='k')
ax.vlines(1,0,0.77,'k')
ax.set(title="$U(z)$ para gás de Bósons");


# COM ZOOM
# Plotamos a função obtida
fig,ax = plot(Tspace,Ut(Tspace),"$U$",r"$T/T_c$",r"$U/(Nk_bT_c)$","b");
ax.plot(x_space,(3/2)*x_space,'--k')
ax.plot(T_res,U_res,'b')

# ax.annotate(r"$U=\frac{3}{2}NK_bT \to$",(1.3,4.5),xycoords='data')
# ax.annotate(r"$z\to0$",(2,-1.5),xycoords='data')
ax.set(xlim=(0,2),ylim=(0,2))
# ax.set(xticks=[0,.5,1,1.5,2],yticks=[0,1,2])
# ax.axhline(0,color='k')
ax.vlines(1,0,0.77,'k')
ax.set(title="$U(z)$ para gás de Bósons");
```

$C_v(z)$ de bósons

```
# A função T faz o papel de T/Tc
def T(z):
  return (2.612375348685488/vpolylogp(3/2,z))**(2/3)
T=np.vectorize(T)

def Cz(z):
  # em T>Tc , z<1
  return (3/2)*((5/2)*(vpolylogp(5/2,z)/vpolylogp(3/2,z))-(3/2)*((vpolylogp(3/2,z)/vp
Cz = np.vectorize(Cz)

def Ct(Tspace):
  # em T<Tc , z>1
  return (0.77)*(5/2)*(Tspace**(3/2))
Ct = np.vectorize(Ct)

# Calculamos para z entre 0 e 1
zspace = np.linspace(.6,0.99999,10000)
Tspace = np.linspace(0,1,10000)
T_res = T(zspace)
C_res = Cz(zspace)

# Plotamos a função obtida
fig,ax = plot(Tspace,Ct(Tspace),"$C_v$",r"$T/T_c$",r"$C_v/(Nk_b)$","b");

ax.plot(T_res,C_res,'b')
```

```
ax.annotate(r"$C_v=\frac{3}{2}NK_b \to$",(1.5,1.2),xycoords='data')
#ax.annotate(r"$z\to0$",(2,-1.5),xycoords='data')
ax.set(xlim=(0,2),ylim=(0,2.5))
ax.set(xticks=[0,.5,1,1.5,2],yticks=[0,1,2])
ax.axhline(3/2,color='k',ls='--')
ax.vlines(1,0,1.925,'k','--')
ax.set(title="$C_v(z)$ para gás de Bósons");
```

#$\mu(z)$ de Férmions

```
# Definimos o potencial químico para Férmions e Bósons em função de z
# A função T faz o papel de T*kb/Ef
def T(z):
  return (-(4/(3*np.sqrt(np.pi))*(1/vpolylogp(3/2,-z))))**(2/3)
T = np.vectorize(T)

# A função muF faz o papal de mu/Ef
def muF(z):
  return T(z)*np.log(z)
muF = np.vectorize(muF)


# Calculamos para z entre 0 e infinito, aqui simulado numéricamente como um espaço
zspace = np.logspace(-1,100,10000)
T_res = T(zspace)
muF_res = muF(zspace)

# Plotamos a função obtida
fig,ax = plot(T_res,muF_res,"$\mu(z)$",r"$ T/ T_f$","$\mu(z)/\epsilon_f$","b");
ax.set(xlim=(0,2),ylim=(-2,2))
ax.set(xticks=[0,1,2],yticks=[-2,-1,0,1,2])
ax.axhline(0,color='k')
ax.vlines(1,-2,0,'k','--')
ax.set(title="$\mu(z)$ para gás de Férmi")
ax.annotate("$z=1$",(1,0.2),xycoords='data')
ax.annotate(r"$z\to0$",(1.8,-1.5),xycoords='data')
```

$\mu(z)$ de Bósons

```
# Agora definimos o mu e T/Tc para Bósons
# A função T faz o papel de T/Tc
def T(z):
  if z>=1:
    return 0.0
  else:
    return (2.612375348685488/vpolylogp(3/2,z))**(2/3)
T=np.vectorize(T)

# A função muB faz o papel de mu/(kb*Tc)
def muB(z):
  return T(z)*np.log(z)
```

```
muB = np.vectorize(muB)

# Calculamos para z entre 0 e 1
zspace = np.linspace(0.4,1,10000)
T_res = T(zspace)
muB_res = muB(zspace)

# Plotamos a função obtida
fig,ax = plot(T_res,muB_res,"$\mu(z)$",r"$T/T_c$","$\mu(z)/k_bT_c$","b");
ax.annotate("$z=1$",(1,0.2),xycoords='data')
ax.annotate(r"$z\to0$",(2,-1.5),xycoords='data')
ax.set(xlim=(0.5,3),ylim=(-2,1))
ax.set(xticks=[0,1,2],yticks=[-1,0,1])
ax.axhline(0,color='k')
ax.vlines(1,-2,0,'k','--')
ax.set(title="$\mu(z)$ para gás de Bósons");
```

$U(z)$ pelo somatório

```
# Definimos T* como tspace entre 0 e 1
tspace = np.linspace(0,1,1000)

resposta =[]
num=0
den=0
e0=1
# fazemos o somatório para cada T
for t in tqdm(tspace):
  for k in range(1,1000):
    # Separamos as partes do numerador e denominador, e vamos acumulando o somatóri
    num += (np.exp(e0*k**2/t)-1)**(-1)*e0*k**2
    den += (np.exp(e0*k**2/t)-1)**(-1)
  # Para a resposta, apenas dividimos o numerador pelo denominador
  resp=(num/den)
  # realizamos um ajuste manual para corresponder à escala esperada e guardamos a r
  resp = (resp-1)*16
  resposta.append(resp)
# Plotamos a resposta
plot(tspace,resposta,"U pelo somatório","$T^*$","$U / N k_b T_c$","b")

# Plotamos as funções  obtidas
fig,ax = plot(Tspace,Ut(Tspace),"$U$ por integral",r"$T/T_c$",r"$U/(Nk_bT_c)$","b");
ax.plot(x_space,(3/2)*x_space,'--k')
ax.plot(T_res,U_res,'b')
ax.plot(tspace,resposta,'r',label='$U$ pelo somatório')

ax.annotate(r"$U=\frac{3}{2}NK_bT \to$",(0.7,1.4),xycoords='data')
# ax.annotate(r"$z\to0$",(2,-1.5),xycoords='data')
ax.set(xlim=(0,1),ylim=(-0.1,1.5))
# ax.set(xticks=[0,.5,1,1.5,2],yticks=[0,1,2])
# ax.axhline(0,color='k')
```

```
ax.vlines(1,0,0.77,'k')
ax.set(title="$U(z)$ para gás de Bósons");
ax.legend()
```