Constantes Físicas

In [1]:
```
from scipy.constants import physical_constants as scc
```

In [2]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [3]:
```
# Uso de scipy.constants (physical_constants)

me = scc["electron mass"]
mp = scc["proton mass"]
mn = scc["neutron mass"]

print('me=',me)
print('mp=',mp)
print('mn=',mn)
```

```
me= (9.10938356e-31, 'kg', 1.1e-38)
mp= (1.672621898e-27, 'kg', 2.1e-35)
mn= (1.674927471e-27, 'kg', 2.1e-35)
```

In [4]:
```
# Constantes útiles para Termodinámica

kb = scc['Boltzmann constant']
Na = scc['Avogadro constant']
R = scc["molar gas constant"]
print('kb=',kb)
print('Na=',Na)
print('R=',R)
```

```
kb= (1.38064852e-23, 'J K^-1', 7.9e-30)
Na= (6.022140857e+23, 'mol^-1', 74000000000000000.0)
R= (8.3144598, 'J mol^-1 K^-1', 4.8e-06)
```

In [ ]:
```
```

Masas en electron Volt

In [5]:
```
eV = scc["electron volt"]
print('eV=',eV)
```

```
eV= (1.6021766208e-19, 'J', 9.8e-28)
```

In [6]:
```
meeV = scc["electron mass energy equivalent in MeV"]
mpeV = scc["proton mass energy equivalent in MeV"]
mneV = scc["neutron mass energy equivalent in MeV"]

print('me (MeV)=',meeV)
print('mp (MeV)=',mpeV)
print('mn (MeV)=',mneV)
```

```
me (MeV)= (0.5109989461, 'MeV', 3.1e-09)
mp (MeV)= (938.2720813, 'MeV', 5.8e-06)
mn (MeV)= (939.5654133, 'MeV', 5.8e-06)
```

In [7]:
```
kbeV = scc['Boltzmann constant in eV/K']
print('kbVe=',kbeV)
```

```
kbVe= (8.6173303e-05, 'eV K^-1', 5e-11)
```

In [ ]:
```
```

Velocidades medias

In [8]:
```
import sympy as sy
sy.init_printing(use_unicode=False, wrap_line=False, no_global=True)

x,y = sy.symbols("x,y")
alpha = sy.Symbol("alpha",positive=True)
v = sy.Symbol("v",positive=True)
T = sy.Symbol("T",positive=True)
m = sy.Symbol("m",positive=True)
k = sy.Symbol("k",positive=True)
E = sy.Symbol("E",positive=True)
```

In [9]:
```
# Integrales de la forma x^n exp(-alpha x^2)

def I(n):
    In = sy.integrate(x**n * sy.exp(-alpha * x**2), (x,0,sy.oo))
    return In
```

In [10]:
```
print(I(0))
```

```
sqrt(pi)/(2*sqrt(alpha))
```

In [11]:
```
print(sy.latex(I(0)))
```

```
\frac{\sqrt{\pi}}{2 \sqrt{\alpha}}
```

$$\frac{\sqrt{\pi}}{2\sqrt{\alpha}}$$

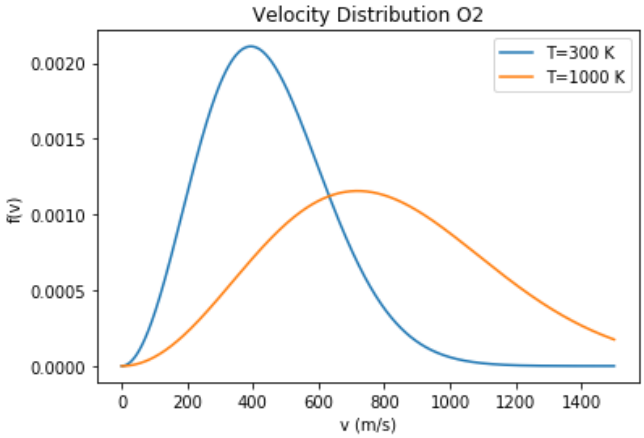
In [12]:
```
# Distribución Maxwell-Boltzmann

def fv(m,v,T):
    k = kb[0]
    ff = 4 * sy.pi * v**2 * (m /( k*T*2*sy.pi) )**(3/2) * sy.exp( - (m*v**2)
    return ff
```

In [13]:
```
# Chequeo de normalización

sy.integrate( fv(m,v,T), (v,0,sy.oo))
```

Out[13]: 1.0

In [14]:
```
# Comparación de distribución para diferentes Temperaturas

npts=1000
xmin=0
xmax=1.50e3
xv = np.linspace(xmin,xmax,npts)
yv300 = np.zeros(npts)
yv600 = np.zeros(npts)

# Oxigen
ZO2 = 8
MO2  = mp[0]*2*ZO2*2

for i in range(npts):
    yv300[i] = fv(MO2,xv[i],300)
    yv600[i] = fv(MO2,xv[i],1000)
```

In [15]:
```
plt.plot(xv,yv300,label='T=300 K')
plt.plot(xv,yv600,label='T=1000 K')

# Estética
plt.title("Velocity Distribution O2")
plt.xlabel("v (m/s)")
plt.ylabel("f(v)")
plt.legend()
plt.show()
```

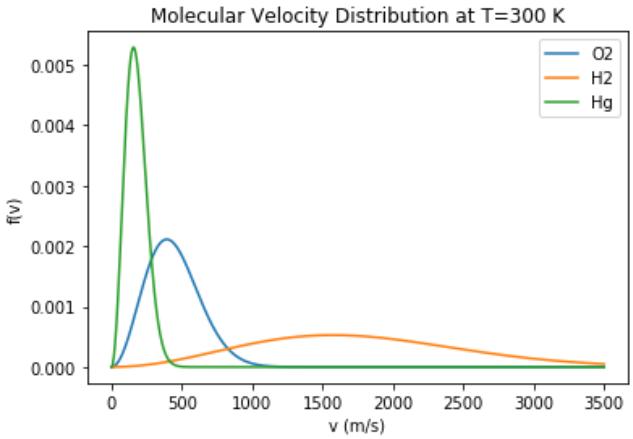


In [16]:
```
# Velocidad media a T=300 K

v300 = sy.integrate( v*fv(MO2,v,300), (v,0,sy.oo))
print('<v> (300 K)=:',v300.evalf())
```

```
<v> (300 K)=: 443.913519784452
```

In [17]:
```
# Comparison other molecules (O2, H2, Hg)

npts=1000
xmin=0
xmax=3.50e3
xv = np.linspace(xmin,xmax,npts)
yvO2 = np.zeros(npts)
yvH2 = np.zeros(npts)
yvHg = np.zeros(npts)

# Oxigen
ZO2 = 8
ZH2 = 1

MO2  = mp[0]*2*ZO2*2
MH2  = mp[0]*1*ZH2*2
MHg = mp[0]*200.59

for i in range(npts):
    yvO2[i] = fv(MO2,xv[i],300)
    yvH2[i] = fv(MH2,xv[i],300)
    yvHg[i] = fv(MHg,xv[i],300)
```

In [18]:
```
plt.plot(xv,yvO2,label='O2')
plt.plot(xv,yvH2,label='H2')
plt.plot(xv,yvHg,label='Hg')

# Estética
plt.title("Molecular Velocity Distribution at T=300 K")
plt.xlabel("v (m/s)")
plt.ylabel("f(v)")
plt.legend()
plt.show()
```



In [19]:
```
# Velocidad media H2 a 300 K

vH2300 = sy.integrate( v*fv(MH2,v,300), (v,0,sy.oo))
print('<v> (300 K)=:',vH2300.evalf())
```

```
<v> (300 K)=: 1775.65407913781
```

In [20]:
```
# Velocidad media Hg a 300 K

vHg300 = sy.integrate( v*fv(MHg,v,300), (v,0,sy.oo))
print('<v> (300 K)=:',vHg300.evalf())
```

```
<v> (300 K)=: 177.304076989847
```

Velocidad de máxima probabilidad

In [21]:
```
# Defino distribución de velocidades, pero con k genérico
# (no está la línea k=kb)

def ffv(m,v,T):
    # k = kb[0]   (Cuidado !!!)
    ff = 4 * sy.pi * v**2 * (m /( k*T*2*sy.pi) )**(3/2) * sy.exp( - (m*v**2)
    return ff

vmed = sy.integrate( v*ffv(m,v,T), (v,0,sy.oo))
vmed
```

Out[21]: $$\frac{2.82842712474619 T^{0.5} k^{0.5}}{\pi^{0.5} m^{0.5}}$$

In [22]:
```
# 2.828 ??
np.sqrt( 8 )
```

Out[22]: 2.8284271247461903

In [23]:
```
# velocidad media

def vmed(m,T):
    return sy.sqrt(8 * kb[0] * T/( sy.pi * m))
```

In [24]:
```
# Derivamos y sacamos vmax

def vmax(m,T):
    vm = sy.sqrt(2 * kb[0] * T / m)
    return vm
```

In [25]:
```
# Velocidad de máxima probabilidad del O2 a T=300 K
vmax(MO2,300).evalf()
```

Out[25]: 393.408113805487

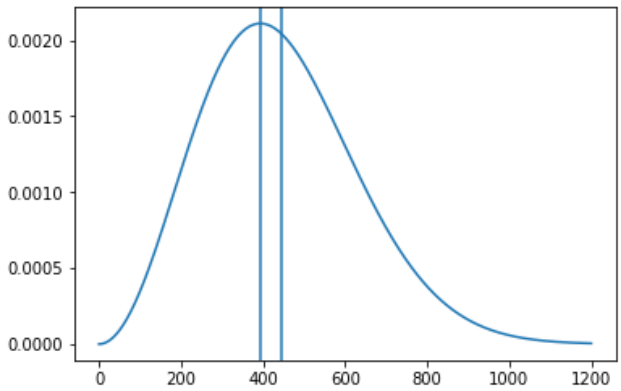
In [26]:
```
# Relación entre velocidades máxima y media

vmed(m,T)/vmax(m,T)
```

Out[26]: $$\frac{2.0}{\sqrt{\pi}}$$

In [27]:
```
2/np.sqrt(np.pi)
```

Out[27]: 1.1283791670955126

In [28]:
```
# Comparison vmax vs <v>:

npts=1000
xmin=0
xmax=1200
xv = np.linspace(xmin,xmax,npts)
yvO2 = np.zeros(npts)

for i in range(npts):
    yvO2[i] = fv(MO2,xv[i],300)

plt.plot(xv,yvO2,label='T=300 K')
plt.axvline(393,0,1)
plt.axvline(443,0,1)
plt.show()
```



Velocidad cuadrática media

In [29]:
```
# Calculo <v^2> con k general para obtener forma analítica

v2m  = sy.integrate( v**2 * ffv(m,v,T), (v,0,sy.oo))
sy.sqrt(v2m )
```

Out[29]: $$\frac{1.45647531512197\sqrt[4]{2}\sqrt{T}\sqrt{k}}{\sqrt{m}}$$

In [30]:
```
# 1.456 * 2^(1/4) ??

1.45647531512197 * np.sqrt(np.sqrt(2))
```

Out[30]: 1.732050807568877

In [31]:
```
np.sqrt(3)
```

Out[31]: 1.7320508075688772

In [32]:
```
# Velocidad cuadrática media

def v2med(m,T):
    return sy.sqrt(3 * kb[0] * T/( m))
```

In [33]:
```
v2med(m,T)/vmax(m,T)
```

Out[33]: 1.22474487139159

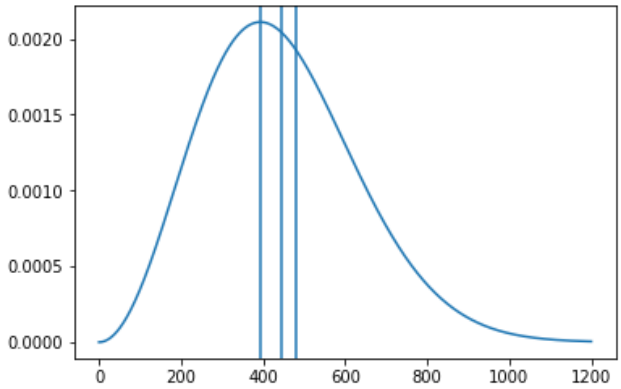
In [34]:
```
# 1.2247 ??

1.22474487139159**2
```

Out[34]: 1.5000000000000024

In [35]:
```
# sqrt<v^2> para O2 a T=300 K

v2300  = sy.integrate( v**2 * fv(MO2,v,300), (v,0,sy.oo))
sy.sqrt(v2300)
```

Out[35]: 481.824569747109

In [36]:
```
# Comparison vmax  <v> y vrms :

npts=1000
xmin=0
xmax=1200
xv = np.linspace(xmin,xmax,npts)
yvO2 = np.zeros(npts)

for i in range(npts):
    yvO2[i] = fv(MO2,xv[i],300)

plt.plot(xv,yvO2,label='T=300 K')
plt.axvline(393,0,1)
plt.axvline(443,0,1)
plt.axvline(482,0,1)
plt.show()
```



In [37]:
```
# chequeo con analytical ratio vrms/vmax

482/393
```

Out[37]: 1.2264631043256997

In [38]:
```
# chequeo con analytical ratio vmed/vmax

443/393
```

Out[38]: 1.1272264631043256

Presión

In [39]:
```
# Cálculo de Presión para N/V y T determinados

# p = N/V k T

p = 3e25 * kb[0] * 300
```

In [40]:
```
# en atmósferas
atm = scc["standard atmosphere"][0]
Patm = p/atm
Patm
```

Out[40]: 1.2263347327905254

In [41]:
```
print("p=",p," (Pa)  = ",Patm," (atm)")
```

```
p= 124258.36679999999  (Pa)  =  1.2263347327905254  (atm)
```

Probabilidad en un rango de velocidades

In [42]:
```
# 2C Probabilidad entre vmed y vmax

npar = sy.integrate( fv(MO2,v,273.15), (v,393.4,443.9)).evalf()
npar
```

Out[42]: 0.108580808826817

In [43]:
```
# Cantidad de moléculas con esas velocidades en 1 Kmol

npar * Na[0] * 1000
```

Out[43]: $6.53888925122079 \cdot 10^{25}$

In [ ]:
```
```

Distribución en Energía

In [44]:
```
def fe(E,T):
    ye = 2/( sy.sqrt(sy.pi)) *  sy.sqrt(E) * 1/(k * T)**(3/2) * sy.exp(- E/(
    return ye
```

In [45]:
```
Enorm = sy.integrate( fe(E,T), (E,0,sy.oo))
print('Enorm=:',Enorm)
```

```
Enorm=: 1
```

In [46]:
```
Emed = sy.integrate(E * fe(E,T), (E,0,sy.oo))
print('<E>=:',Emed)
```

```
<E>=: 3*T*k/2
```

Número de choques

In [47]:
```
#N2:
ZN2 = 7
MN2  = mp[0]*2*ZN2*2
MN2
```

Out[47]: $4.6833413144000004e-26$

In [48]:
```
vN2 = np.sqrt( 8 * kb[0] * 300 / (np.pi * MN2) )
vN2
```

Out[48]: 474.5635143972332

In [49]:
```
rho = 1*atm/(kb[0] * 300)
rho
```

Out[49]: $2.4463141422843813e+25$

In [50]:
```
# en molec/cm^3:

rho/(10**6)
```

Out[50]: $2.4463141422843814e+19$

In [51]:
```
# nro de choques:
nchoq = rho/4 * vN2
nchoq
```

Out[51]: $2.902328591705323e+27$

In [52]:
```
# en molec/(cm^2 sec):

2.5 * 10**19 / 4 * 47600
```

Out[52]: $2.975e+23$

Problema 3

In [53]:
```
# V = 4/3 Pi r^3
V = 4/3 * np.pi * 0.1**3
V
```

Out[53]: 0.004188790204786391

In [54]:
```
# No hace falta calcular las presiones, sólo P/DP
# De todos modos, para recordar como convertir mmHg a atm:

P = 10/760
P*atm
```

Out[54]: 1333.2236842105262

In [55]:
```
# vmed
#H20:
MH20  = mp[0]*(2+16)
vH20= np.sqrt( 8 * kb[0] * 300 / (np.pi * MH20) )
vH20
```

Out[55]: 591.8846930459364

In [56]:
```
t = 4 * V * 5 /( vH2O * 1e-4)
t
```

Out[56]: 1.4154075123079073

Ejercicio 13

In [57]:
```
# N0 = RT/(W (h1-h2) ) Log (n2/n1)

# V = 4/3 Pi r^3
V = 4/3 * np.pi * (1 * 0.212e-6)**3
V
```

Out[57]: $3.9911329236350936e-20$

In [58]:
```
W = 1.194e3*V
W
```

Out[58]: $4.7654127108203016e-17$

In [85]:
```
# Na = RT / ( m g h) log(n2/n1)

g = 9.8

N0 = R[0]*(20+273.15) / ( W * g * 6e-5) * np.log(49/14)
N0
```

Out[85]: $1.0897201929979927e+23$

Ejercicio 10

In [60]:
```
def fv10(m,v,T):
    ff =  v**3 * sy.exp( - (m*v**2)/(2*k*T) )
    return ff
```

In [61]:
```
Norm10  = sy.integrate(fv10(m,v,T), (v,0,sy.oo))
Norm10
```

Out[61]: $\frac{2T^2k^2}{m^2}$

In [62]:
```
# chequeo de normalización

def fvN10(m,v,T):
    ff = 1/2 * (m/(T*k))**2 * v**3 * sy.exp( - (m*v**2)/(2*k*T) )
    return ff
```

In [63]:
```
Norm10n  = sy.integrate(fvN10(m,v,T), (v,0,sy.oo))
Norm10n
```

Out[63]: 1.0

In [64]:
```
# velocidad cuadrática media

v210m = sy.integrate(v**2 * fvN10(m,v,T), (v,0,sy.oo))
sy.sqrt(v210m)
```

Out[64]:
$$\frac{2.0\sqrt{T}\sqrt{k}}{\sqrt{m}}$$

In [65]:
```
# Lo podemos hacer utilizando las integrales I(n)
```

In [66]:
```
I(3)
```

Out[66]:
$$\frac{1}{2\alpha^2}$$

In [67]:
```
# 2 alpha^2 I(3) debería estar normalizado:

2 * alpha**2 * I(3)
```

Out[67]:
$$1$$

In [68]:
```
I(5)
```

Out[68]:
$$\frac{1}{\alpha^3}$$

In [69]:
```
2 * alpha**2 * I(5)
```

Out[69]:
$$\frac{2}{\alpha}$$

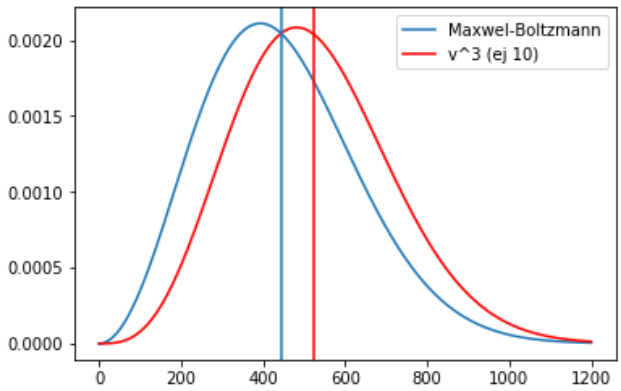
In [70]:
```
# alpha = m/( 2 k T)

sy.sqrt( 2 / alpha)
```

Out[70]:
$$\frac{\sqrt{2}}{\sqrt{\alpha}}$$

In [71]:
```
# velocidad media

def fvNk10(m,v,T):
    k = kb[0]
    ff = 1/2 * (m/(T*k))**2 * v**3 * sy.exp( - (m*v**2)/(2*k*T) )
    return ff

v10med = sy.integrate(v * fvNk10(MO2,v,300), (v,0,sy.oo))
v10med.evalf()
```

Out[71]:
$$522.973294719008$$

In [72]:
```
# Gráfico y valores medios:

npts=1000
xmin=0
xmax=1200
xv = np.linspace(xmin,xmax,npts)
yv02 = np.zeros(npts)
yv10 = np.zeros(npts)

for i in range(npts):
    yv10[i] = fvNk10(MO2,xv[i],300)
    yvO2[i] = fv(MO2,xv[i],300)

plt.plot(xv,yvO2,label='Maxwel-Boltzmann')
plt.plot(xv,yv10,label='v^3 (ej 10)',color='red')
plt.axvline(443,0,1)
plt.axvline(523,0,1,color='red')
plt.legend()
plt.show()
```



Ejercicio 4

In [73]:
```
# Separamos la parte de p y de x

def fEpNN(m,p,T):
    yy = sy.exp(-p**2 /( 2* m * k * T) )
    return yy
```

In [74]:
```
# pongo v en lugar de p, porque ya la usé para presión
Normp = sy.integrate(fEpNN(m,v,T) , (v,0,sy.oo))
Normp
```

Out[74]:
$$\frac{\sqrt{2}\sqrt{\pi}\sqrt{T}\sqrt{k}\sqrt{m}}{2}$$

In [75]:
```
# Distribución normalizada:
def fEp(m,p,T):
    yy = sy.sqrt( 2/( sy.pi * m * k * T)) * sy.exp(-p**2 /( 2* m * k * T) )
    return yy
```

In [76]:
```
Normp = sy.integrate(fEp(m,v,T) , (v,0,sy.oo))
Normp
```

Out[76]: 1

In [77]:
```
# Parte x

b = sy.Symbol("b",positive=True)

def fExNN(b,x,T):
    yy = sy.exp(-b * x**4 /( k * T) )
    return yy
```

In [78]:
```
Normx = sy.integrate(fExNN(b,x,T) , (x,-sy.oo,sy.oo))
Normx
```

Out[78]:
$$\frac{\sqrt[4]{T}\sqrt[4]{k}\Gamma\left(\frac{1}{4}\right)}{2\sqrt[4]{b}}$$

In [79]:
```
# Distribución normalizada:
def fEx(b,x,T):
    yy = 2 * (b/(k * T) )**(1/4) / sy.gamma(1/4) * sy.exp(-b * x**4 /( k * T
    return yy
```

In [80]:
```
Normx = sy.integrate(fEx(b,x,T) , (x,-sy.oo,sy.oo))
Normx
```

Out[80]:
$$0.275815662830209\Gamma\left(\frac{1}{4}\right)$$

In [81]:
```
Normx.evalf()
```

Out[81]:
$$1.0$$

In [82]:
```
Epmed = sy.integrate(v**2 /( 2* m ) * fEp(m,v,T) , (v,0,sy.oo))
Epmed
```

Out[82]:
$$\frac{Tk}{2}$$

In [83]:
```
Exmed = sy.integrate(b * x**4 * fEx(b,x,T) , (x,-sy.oo,sy.oo))
Exmed
```

Out[83]:
$$0.0689539157075523Tk\Gamma\left(\frac{1}{4}\right)$$

In [84]:
```
Exmed.evalf()
```

Out[84]:
$$0.25Tk$$

In [ ]:
```

```