Chem 30324, Spring 2017, Homework 10

Due April 27, 2017

Computational chemistry.

Today properties of a molecule are more often than not calculated rather than inferred. Quantitative molecular quantum mechanical calculations require highly specialized numerical solvers like Gaussian (https://www.gaussian.com (https://www.gaussian.com)). Following are instructions for using Gaussian with the Webmo graphical interface (https://www.webmo.net/demoserver/cgi-bin/webmo/login.cgi (https://www.webmo.net/demoserver/cgi-bin/webmo/login.cgi)).

Now, let's set up your calculation (you may do this with a partner if you choose):

1. Log into the Webmo server https://www.webmo.net/demoserver/cgi-bin/webmo/login.cgi (https://www.webmo.net/demoserver/cgi-bin/webmo/login.cgi) using "guest" as your username and password.
2. Select New Job-Creat New Job.
3. Use the available tools to sketch a molecule.
4. Use the right arrow at the bottom to proceed to the Computational Engines.
5. Select the "B3LYP" functional and the appropriate basis set.
6. Select the right arrow to run the calculation.
7. From the job manager window choose the completed calculation to view the results.

The molecule you are to study depends on your last name. Choose according to the list:

- A-G: **CO**
- H-R: **BN**
- S-Z: **BeO**

For your convenience, here are the total energies (in Hartree, 27.212 eV/Hartree) of the constituent atoms, calculated using the B3LYP DFT treatment of v_{ee} and the "Routine" basis set:

Atom	Energy	Atom	Energy
B	–24.65435	N	–54.559498
Be	–14.65446	O	–75.031179
C	–37.68086	F	–99.681600

1. Construct a potential energy surface for your molecule. Using covalent radii, guess an approximate equilbrium bond length, and use the Webmo editor to draw the molecule with that length. Specify the "Molecular Energy" option to Gaussian and the "Routine" basis set for better accuracy. Calculate and plot out total molecular energy vs. bond distance in increments of 0.05 Å about your guessed minimum, including enough points to encompass the actual minimum. (You will find it convenient to subtract off the individual atom energies from the molecular total energy and to convert to more convenient units, like eV or kJ/mol.) By fitting the few points nearest the minimum, determine the equilibrium bond length. How does your result compare to literature?

Literature Bond length CO 1.128 Å, BN 1.325 Å, BeO 1.331 Å.

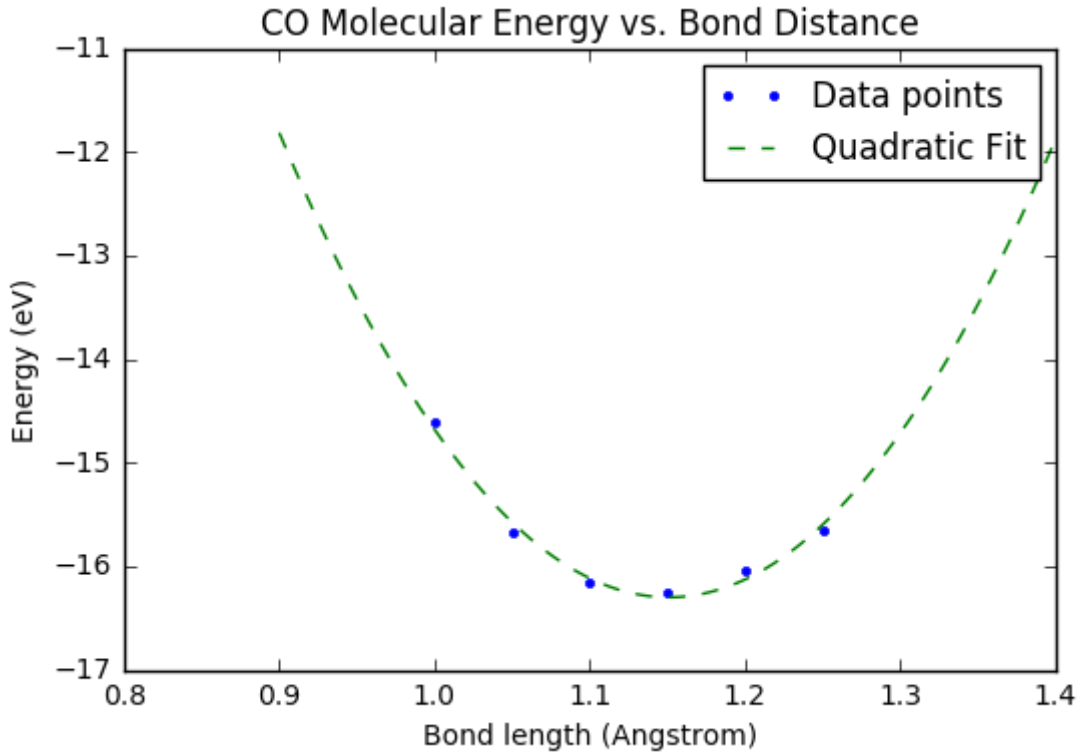
CO molecule

In [4]:
```
"CO molecule"
import numpy as np
import matplotlib.pyplot as plt

"quadratic fit"
E_C = -37.68086 # Ha
E_O = -75.031179 # Ha
R_CO = [1.00, 1.05, 1.10, 1.15, 1.2, 1.25] # distance, angstrom
E_total_CO = [-113.249199,-113.287858,-113.305895,-113.309135,-113.30190
2,-113.287408] # Ha, total energy of the molecule
E_CO = [] # eV, subtract off the individual atom energies from the molec
ular total energy and convert to eV
for i in E_total_CO:
    E_CO.append((i-E_C-E_O)*27.212)
fit = np.polyfit(R_CO, E_CO, 2) # quadratic fit
print(fit)
```

```
[  71.30418671 -164.11063691   78.12636826]
```

In [5]:
```
"plot energy vs. bond length"
x = np.linspace(0.9, 1.4, 100)
z = 71.30418671*x**2 - 164.1106369*x + 78.12636826 # quadratic fit
E_min_CO = min(z) # minimum energy
print('The energy minimum is {:.5f} eV.'.format(E_min_CO))
plt.plot(R_CO, E_CO, '.', label='Data points')
plt.plot(x, z, '--',label='Quadratic Fit')
plt.xlabel('Bond length (Angstrom)')
plt.ylabel('Energy (eV)')
plt.title('CO Molecular Energy vs. Bond Distance')
plt.legend()
plt.show()

"find equilbrium bond length"
import sympy as sp
x = sp.symbols('x')
z = 71.30418671*x**2 - 164.1106369*x + 78.12636826 # quadratic fit
l = sp.solve(sp.diff(z,x),x)
print('The equilibrium bond length is {:.4f} angstroms.'.format(l[0])) #
 equilibrium bond length
```

```
The energy minimum is -16.30090 eV.
```




```
The equilibrium bond length is 1.1508 angstroms.
```

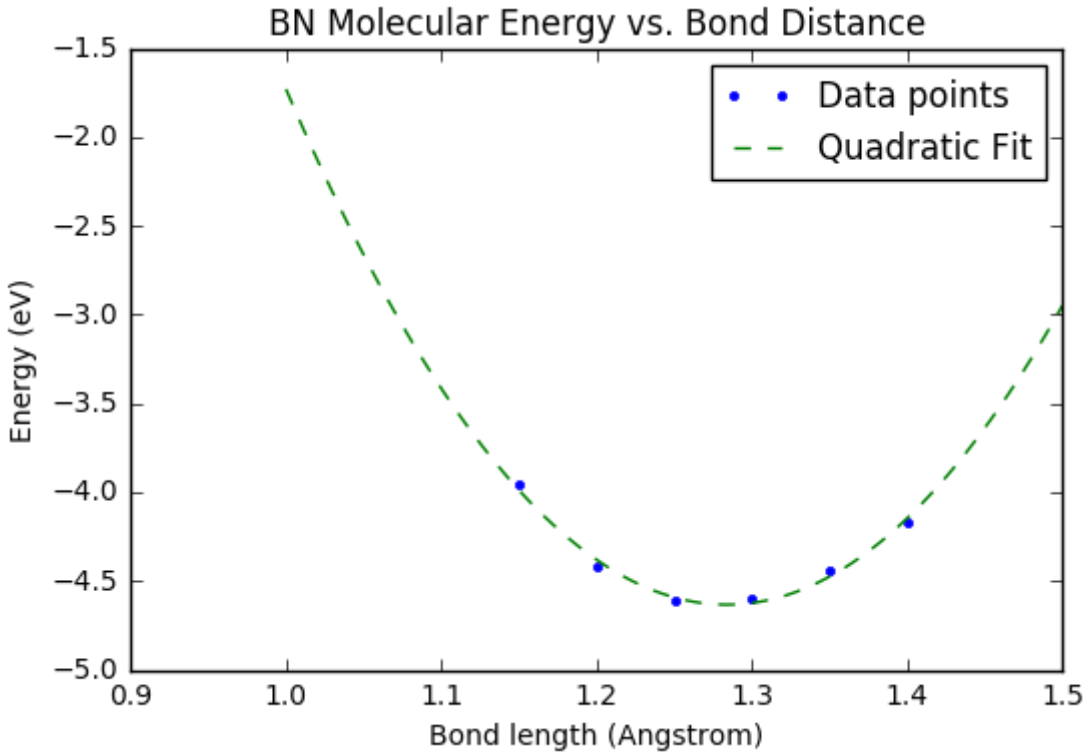
BN molecule

In [6]:
```
"BN molecule"
import numpy as np
import matplotlib.pyplot as plt

"quadratic fit"
E_B = -24.65435 # Ha
E_N = -54.559498 # Ha
R_BN = [1.15, 1.2, 1.25, 1.3, 1.35, 1.4] # distance, angstrom
E_total_BN = [-79.359357,-79.376368,-79.383355,-79.382896,-79.377003,-7
9.367236] # Ha, total energy of the molecule
E_BN = [] # eV, subtract off the individual atom energies from the molec
ular total energy and convert to eV
for i in E_total_BN:
    E_BN.append((i-E_B-E_N)*27.212)
fit = np.polyfit(R_BN, E_BN, 2) # quadratic fit
print(fit)
```

```
[ 36.03840657 -92.53300264  54.76376165]
```

In [7]:
```
"plot energy vs. bond length"
x = np.linspace(1.0, 1.5, 100)
z = 36.03840657*x**2 - 92.53300264*x + 54.76376165 # quadratic fit
E_min_BN = min(z) # minimum energy
print('The energy minimum is {:.5f} eV.'.format(E_min_BN))
plt.plot(R_BN, E_BN, '.', label='Data points')
plt.plot(x, z, '--',label='Quadratic Fit')
plt.xlabel('Bond length (Angstrom)')
plt.ylabel('Energy (eV)')
plt.title('BN Molecular Energy vs. Bond Distance')
plt.legend()
plt.show()

"find equilbrium bond length"
import sympy as sp
x = sp.symbols('x')
z = 36.03840657*x**2 - 92.53300264*x + 54.76376165 # quadratic fit
l = sp.solve(sp.diff(z,x),x)
print('The equilibrium bond length is {:.4f} angstroms.'.format(l[0])) #
 equilibrium bond length
```

```
The energy minimum is -4.63365 eV.
```




```
The equilibrium bond length is 1.2838 angstroms.
```

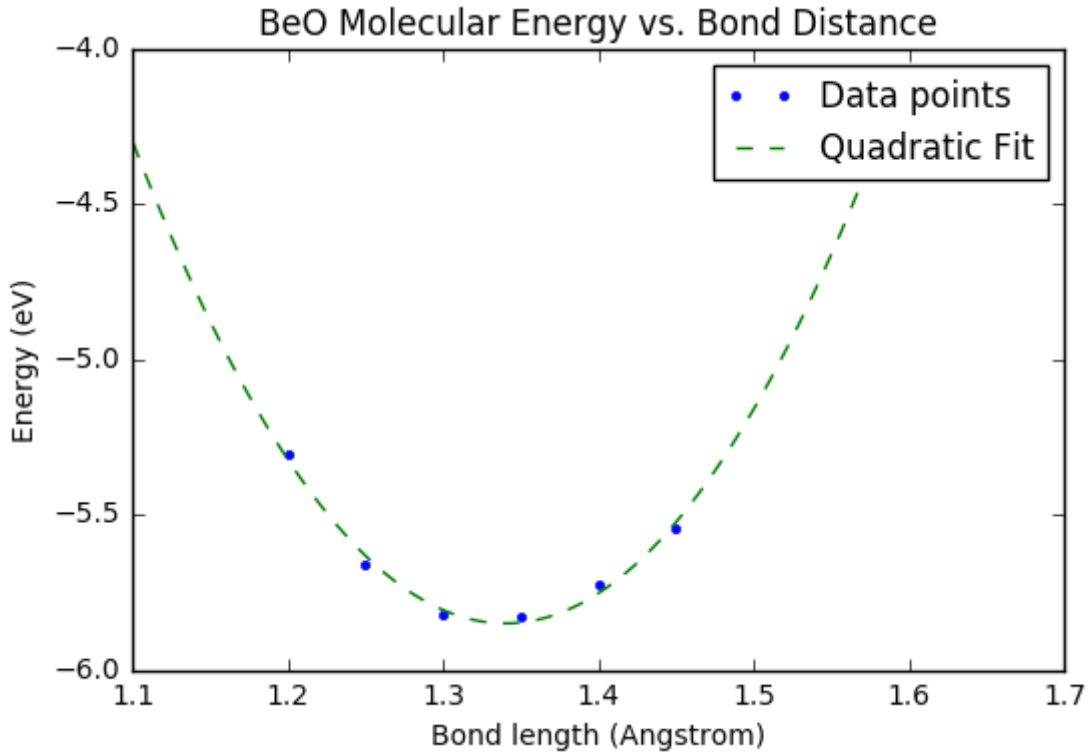
BeO molecule

In [8]:
```
"BeO molecule"
import numpy as np
import matplotlib.pyplot as plt

"quadratic fit"
E_Be = -14.65446 # Ha
E_O = -75.031179 # Ha
R_BeO = [1.2, 1.25, 1.3, 1.35, 1.4, 1.45] # distance, angstrom
E_total_BeO = [-89.880569,-89.893740,-89.899599,-89.899934,-89.896149,-8
9.889335] # Ha, total energy of the molecule
E_BeO = [] # eV, subtract off the individual atom energies from the mole
cular total energy and convert to eV
for i in E_total_BeO:
    E_BeO.append((i-E_Be-E_O)*27.212)
fit = np.polyfit(R_BeO, E_BeO, 2) # quadratic fit
print(fit)
```

```
[ 26.92063723 -72.13881957  42.4763745 ]
```

In [10]:
```
"plot energy vs. bond length"
x = np.linspace(1.1, 1.6, 100)
z = 26.92063723*x**2 - 72.13881957*x + 42.4763745 # quadratic fit
E_min_BeO = min(z) # minimum energy
print('The energy minimum is {:.5f} eV.'.format(E_min_BeO))
plt.plot(R_BeO, E_BeO, '.', label='Data points')
plt.plot(x, z, '--',label='Quadratic Fit')
plt.xlabel('Bond length (Angstrom)')
plt.ylabel('Energy (eV)')
plt.title('BeO Molecular Energy vs. Bond Distance')
plt.legend()
plt.show()

"find equilbrium bond length"
import sympy as sp
x = sp.symbols('x')
z = 26.92063723*x**2 - 72.13881957*x + 42.4763745 # quadratic fit
l = sp.solve(sp.diff(z,x),x)
print('The equilibrium bond length is {:.4f} angstroms.'.format(l[0])) #
 equilibrium bond length
```

```
The energy minimum is -5.85078 eV.
```




```
The equilibrium bond length is 1.3398 angstroms.
```

2. Use the quadratic fit from Question 1 to determine the harmonic vibrational frequency of your molecule, in cm^{-1}. Recall that the force constant is the second derivative of the energy at the minimum, and that the frequency (in wavenumbers) is related to the force constant according to

$$\tilde{\nu} = \frac{1}{2\pi c}\sqrt{\frac{k}{\mu}}$$

CO

In [13]:
```
"calculate harmonic vibrational frequency"
J = 1.6022e-19 # 1 eV = 1.6022e-19 J
A = 1e-10 # 1 angstrom = 1e-10 m
c = 2.99792e8 # m/s
m_C = 12.0107
m_O = 15.9994
mu_CO = m_C*m_O/(m_C+m_O)*1.6605e-27 # kg, reduced mass
k_CO = 2*71.30418671*J/A**2 # J/m**2
nu_CO = 1/(2*np.pi*c)*np.sqrt(k_CO/mu_CO)/100 # cm^-1, wavenumber
print('The harmonic vibrational frequency is {:.2f}
cm^-1.'.format(nu_CO))
```

```
The harmonic vibrational frequency is 2377.57 cm^-1.
```

BN

In [14]:
```
"calculate harmonic vibrational frequency"
J = 1.6022e-19 # 1 eV = 1.6022e-19 J
A = 1e-10 # 1 angstrom = 1e-10 m
c = 2.99792e8 # m/s
m_B = 10.811
m_N = 14.0067
mu_BN = m_B*m_N/(m_B+m_N)*1.6605e-27 # kg, reduced mass
k_BN = 2*36.0384*J/A**2 # J/m**2
nu_BN = 1/(2*np.pi*c)*np.sqrt(k_BN/mu_BN)/100 # cm^-1, wavenumber
print('The harmonic vibrational frequency is {:.2f}
cm^-1.'.format(nu_BN))
```

```
The harmonic vibrational frequency is 1792.32 cm^-1.
```

BeO

In [15]:
```
"calculate harmonic vibrational frequency"
J = 1.6022e-19 # 1 eV = 1.6022e-19 J
A = 1e-10 # 1 angstrom = 1e-10 m
c = 2.99792e8 # m/s
m_Be = 9.01218
m_O = 15.9994
mu_BeO = m_Be*m_O/(m_Be+m_O)*1.6605e-27 # kg, reduced mass
k_BeO = 2*26.92063723*J/A**2 # J/m**2
nu_BeO = 1/(2*np.pi*c)*np.sqrt(k_BeO/mu_BeO)/100 # cm^-1, wavenumber
print('The harmonic vibrational frequency is {:.2f} cm^-1.'.format(nu_Be
O))
```

```
The harmonic vibrational frequency is 1593.68 cm^-1.
```

3. Use your results to determine the zero-point-corrected bond energy of your molecule. How does this model compare with the experimental value?

Experimental value: CO 1072 kJ/mol, BN 385 kJ/mol, BeO 445 kJ/mol.

CO

In [18]:
```
"determine the zero-point-corrected bond energy"
h = 6.62607e-34 # J*s
NA = 6.02214e23
E0_CO = 0.5*h*nu_CO*100*c # J, zero point energy for harmonic oscillator
E_Bond_CO = (E_min_CO*J + E0_CO)*NA/1000 # kJ/mol, zero-point-corrected
 bond energy
print('The zero-point-corrected bond energy is {:.4f} kJ/mol.'.format(-E
_Bond_CO))
```

```
The zero-point-corrected bond energy is 1558.5998 kJ/mol.
```

BN

In [16]:
```
"determine the zero-point-corrected bond energy"
h = 6.62607e-34 # J*s
NA = 6.02214e23
E0_BN = 0.5*h*nu_BN*100*c # J, zero point energy for harmonic oscillator
E_Bond_BN = (E_min_BN*J + E0_BN)*NA/1000 # kJ/mol, zero-point-corrected
 bond energy
print('The zero-point-corrected bond energy is {:.4f} kJ/mol.'.format(-E
_Bond_BN))
```

```
The zero-point-corrected bond energy is 436.3648 kJ/mol.
```

BeO

In [17]:
```
"determine the zero-point-corrected bond energy"
h = 6.62607e-34 # J*s
NA = 6.02214e23
E0_BeO = 0.5*h*nu_BeO*100*c # J, zero point energy for harmonic oscillator
E_Bond_BeO = (E_min_BeO*J + E0_BeO)*NA/1000 # kJ/mol, zero-point-corrected bond energy
print('The zero-point-corrected bond energy is {:.4f} kJ/mol.'.format(-E_Bond_BeO))
```

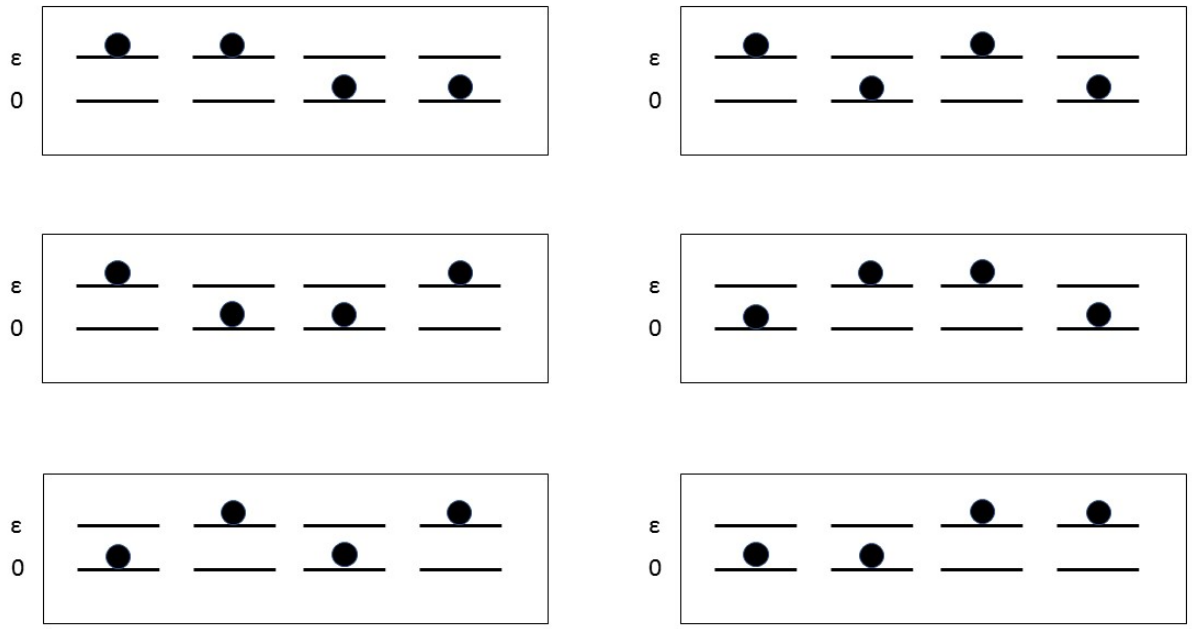
```
The zero-point-corrected bond energy is 554.9907 kJ/mol.
```

The two-state system.

Consider a closed system containing N objects, each of which can be in one of two energy states, of energy either 0 or ε. The total internal energy U of the box is the sum of the energies of the individual objects.

4. Write down all the possible microstates for a box in which $N = 4$ and the internal energy $U = 2\varepsilon$.

$\Omega = \binom{4}{2} = 6$



5. What does the postulate of *equal a priori probabilities* say about the relative likelihood of occurance of any one of these microstates?

Given an isolated system in equilibrium, it is found with equal probability in each of it's accessible microstates

6. What is the entropy of the box? (Thank you, Ludwig Boltzmann.)

$\Omega = \binom{4}{2} = 6$

$S = k_B ln\Omega = 2.474 \times 10^{-23} J/k$

7. Suppose two identical such boxes are brought into thermal contact. Calculate the change in internal energy ΔU and in entropy ΔS associated with this process.

$U_{initial} = 2\varepsilon + 2\varepsilon = 4\varepsilon = U_{final}$, so, $\Delta U = 0$.

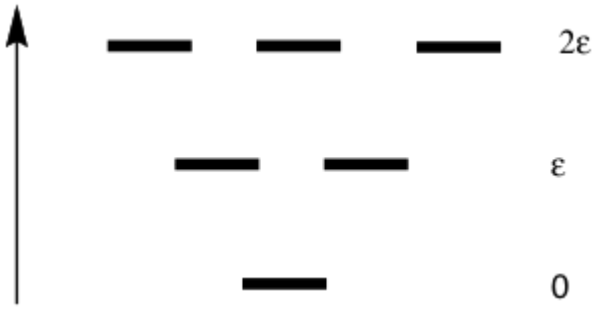
$\Omega_{initial} = 6, S_{initial} = k_B ln6 + k_B ln6.$

$\Omega_{final} = \binom{8}{4} = 70, S_{final} = k_B ln\Omega_{final} = k_B ln70.$

So, $\Delta S = k_B ln70 - 2k_B ln6 = 9.18 \times 10^{-24} J/k.$

The canonical ensemble.

The energy spectrum of some molecule is described by the diagram below.

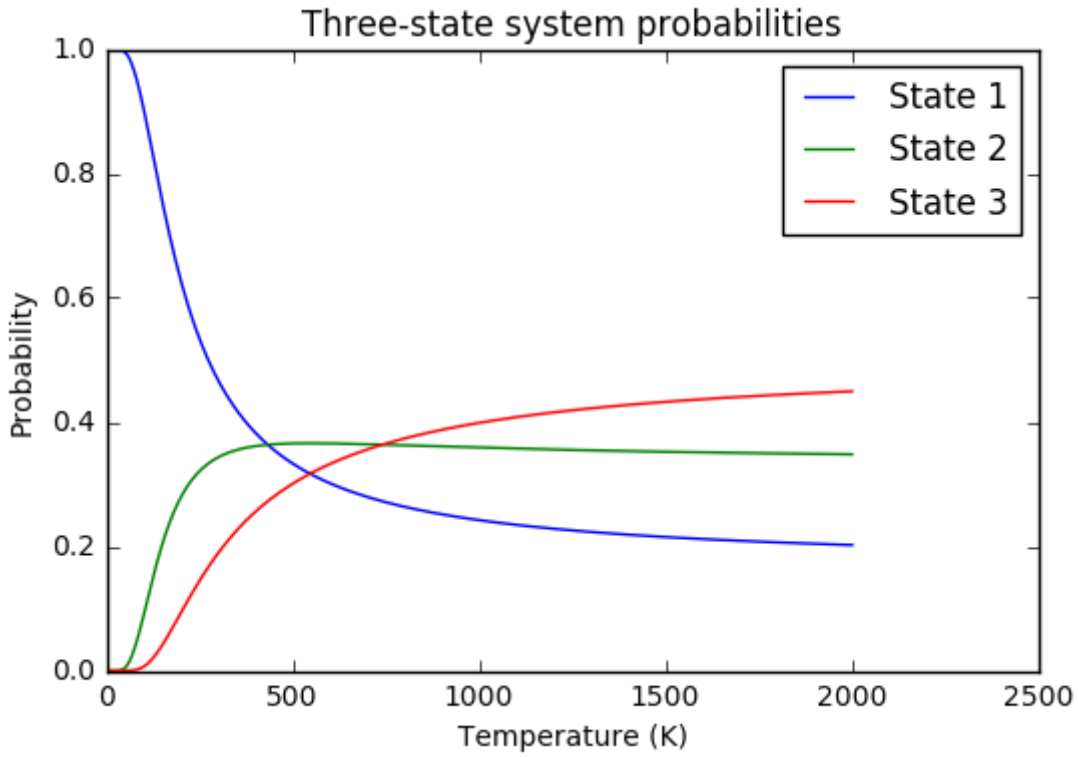


8. Write the partition function q for the molecular at thermal equilibrium at a temperature $\beta = 1/k_B T$.

$q = 1 + 2e^{-\beta\varepsilon} + 3e^{-2\beta\varepsilon}$

9. Plot the probability for the molecule to be in each of the three energy states vs. temperature. Be sure to indicate the probabilities in the limits of $T \to 0$ and $T \to \infty$.

In [21]:
```
import numpy as np
import matplotlib.pyplot as plt

k = 8.61734e-5   # eV /K
theta = 300.  # epsilon/kB


def q(T):
   return 1. + 2.*np.exp(-theta/T) + 3.*np.exp(-2.*theta/T)

def P1(T):
   return 1/q(T)

def P2(T):
   return 2.*np.exp(-theta/T)/q(T)

def P3(T):
   return 3.*np.exp(-2.*theta/T)/q(T)

T = np.linspace(1,2001,500)

plt.plot(T,P1(T),label='State 1')
plt.plot(T,P2(T),label='State 2')
plt.plot(T,P3(T),label='State 3')
plt.xlabel('Temperature (K)')
plt.ylabel('Probability')
plt.legend()
plt.title('Three-state system probabilities')
plt.show()
```

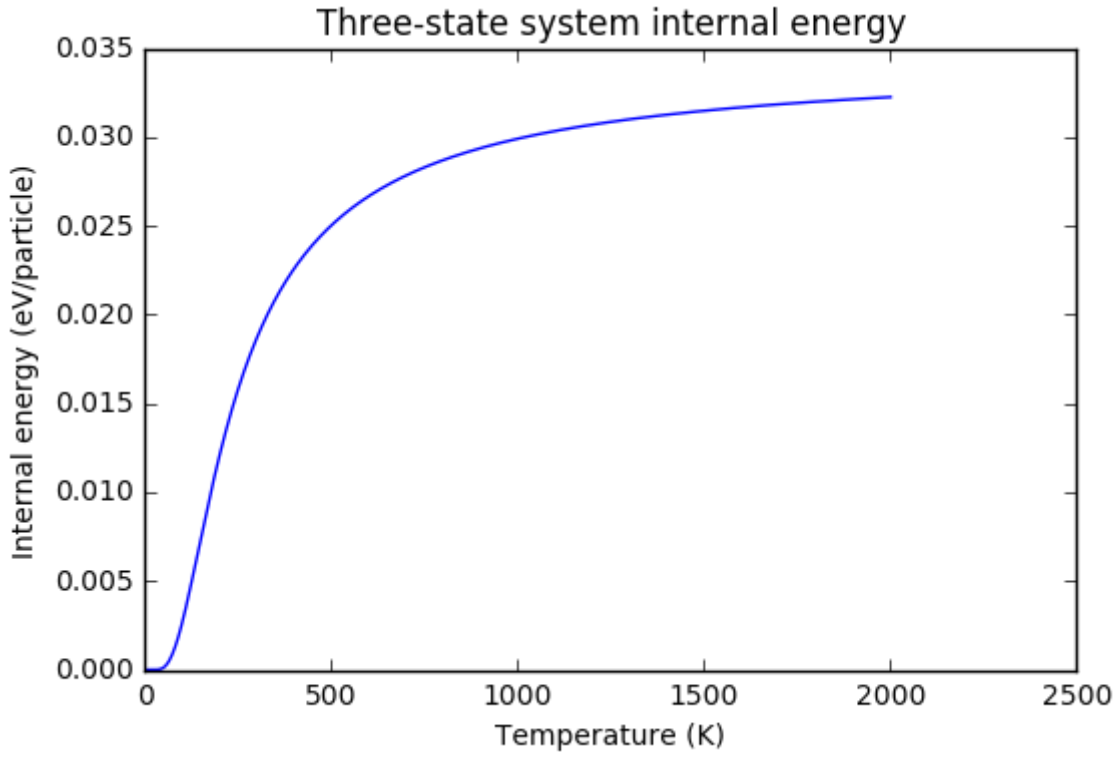


Probabilities	$T \to 0$	$T \to \infty$
$P(0) = \frac{1}{q}$	1	$\frac{1}{6}$
$P(\varepsilon) = \frac{2e^{-\beta\varepsilon}}{q}$	0	$\frac{2}{6}$
$P(2\varepsilon) = \frac{3e^{-2\beta\varepsilon}}{q}$	0	$\frac{3}{6}$

10. Derive an expression for the energy U per molecule by summing over the possible microstates weighted by their probabilities. Plot the average energy vs. temperature.

$$U = 0P(0) + \varepsilon P(\varepsilon) + 2\varepsilon P(2\varepsilon) = \varepsilon\frac{2e^{-\beta\varepsilon}}{q} + 2\varepsilon\frac{3e^{-2\beta\varepsilon}}{q} = \frac{2\varepsilon e^{-\beta\varepsilon} + 6\varepsilon e^{-2\beta\varepsilon}}{q}$$

In [23]:
```
def U(T):
    epsilon = theta*k
    return (2.*epsilon * np.exp(-theta/T) + 6.*epsilon *
np.exp(-2.*theta/T))/ q(T)
plt.show()
plt.plot(T,U(T))
plt.xlabel('Temperature (K)')
plt.ylabel('Internal energy (eV/particle)')
plt.title('Three-state system internal energy')
```



Out[23]: <matplotlib.text.Text at 0x2b8be0a8f9b0>

11. Derive an expression for the energy U per molecule by taking the appropriate derivative of the partition function from problem 8 (*Hint:* it is easier to work with the expressions in term of β than in T.) Does your result agree with that from problem 10?

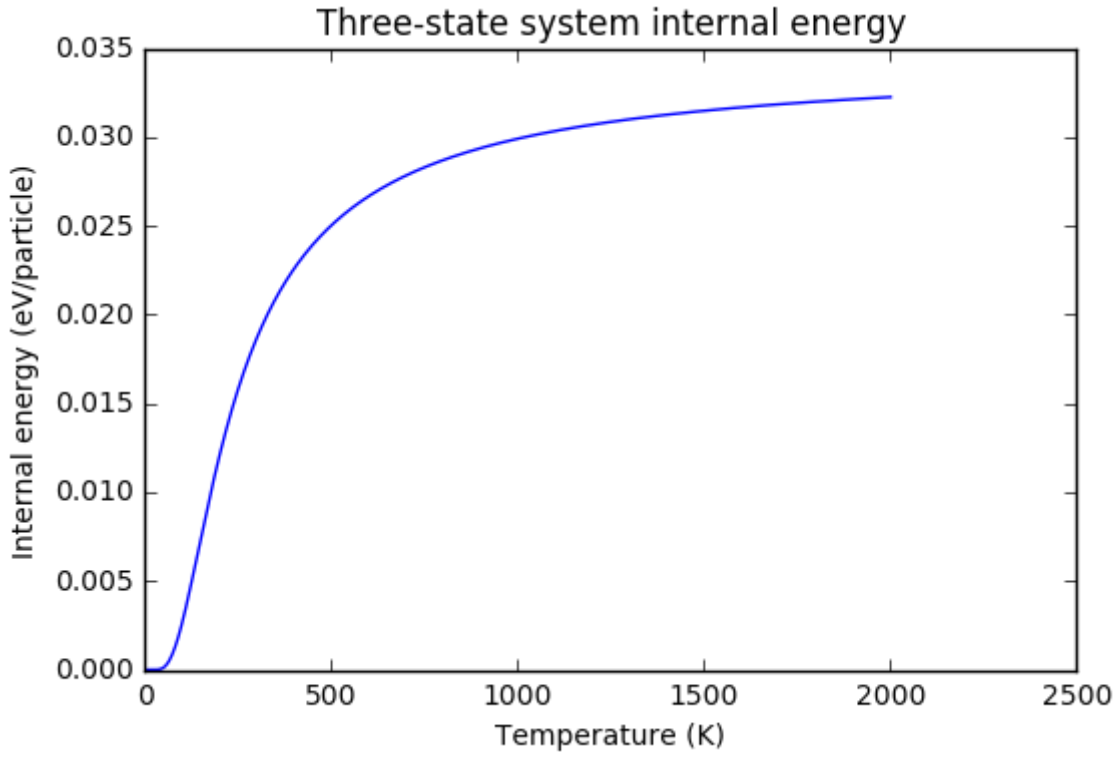
$\left = -\left(\frac{dlnq}{d\beta}\right) = -\frac{1}{q}\left(\frac{dq}{d\beta}\right) = -\frac{1}{q}\frac{d(1 + 2e^{-\beta\varepsilon} + 3e^{-2\beta\varepsilon})}{d\beta} = \frac{2\varepsilon e^{-\beta\varepsilon} + 6\varepsilon e^{-2\beta\varepsilon}}{q}$

Yes, the result is the same as problem 10.

12. Derive an expression for the Helmholtz energy A per molecule from the partition function. Plot A vs. temperature, assuming $\varepsilon/k_B = 300$ K.

$$A = U - TS = U - T(\frac{U}{T} + k_B lnq) = -k_B Tlnq = -k_B Tln(1 + 2e^{-\beta\varepsilon} + 3e^{-2\beta\varepsilon}) = -k_B Tln(1 + 2e^{-300/T} + 3$$

In [24]:
```
def A(T):
   return -k*T*np.log(q(T))
plt.show()
plt.plot(T,A(T))
plt.xlabel('Temperature (K)')
plt.ylabel('Free energy (eV/particle)')
plt.title('Three-state system Helmholtz free energy')
```

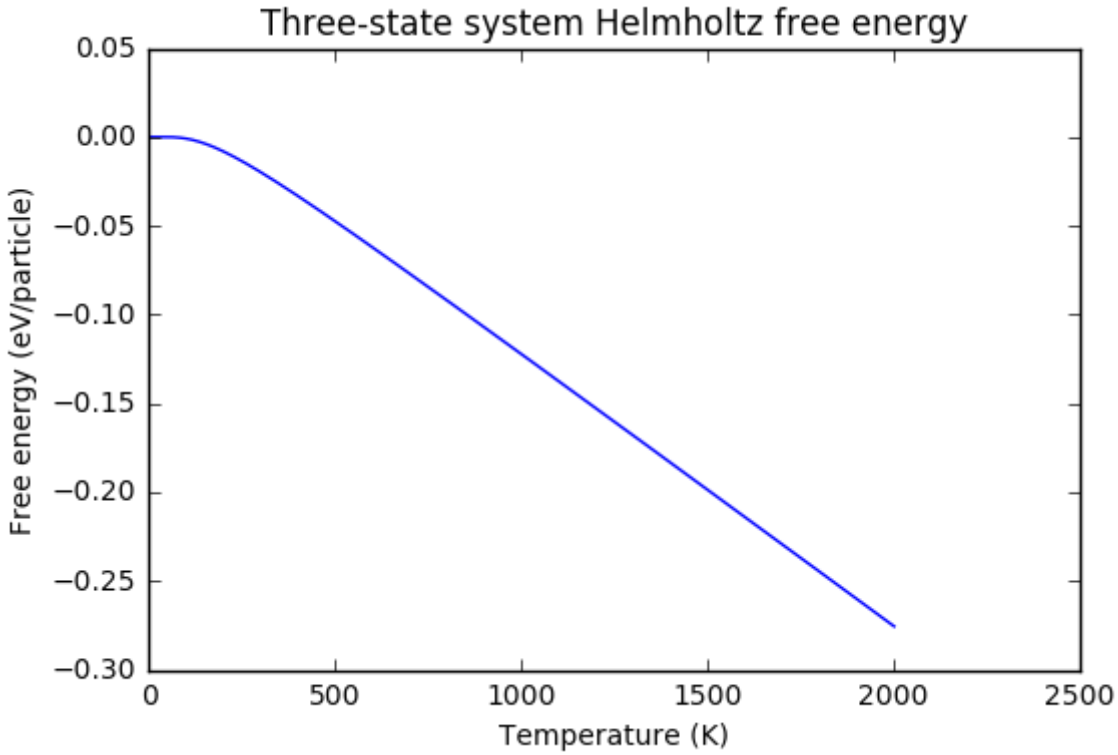


Out[24]: <matplotlib.text.Text at 0x2b8be0aeee48>

13. Derive an expression for the entropy S per molecules and plot vs. temperature, again assuming $\varepsilon/k_B = 300$ K.

$$S = \frac{U-A}{T} = \frac{U}{T} + k_B lnq = \frac{2\varepsilon e^{-\beta\varepsilon}+6\varepsilon e^{-2\beta\varepsilon}}{(1+2e^{-\beta\varepsilon}+3e^{-2\beta\varepsilon})T} + k_B ln(1 + 2e^{-\beta\varepsilon} + 3e^{-2\beta\varepsilon})$$

In [25]:
```
def S(T):
   return (U(T) - A(T))/T
plt.show()
plt.plot(T,S(T))
plt.xlabel('Temperature (K)')
plt.ylabel('Entropy (eV/T)')
plt.title('Three-state system entropy')
```



Out[25]: <matplotlib.text.Text at 0x2b8be0b55828>

14. In class we took the First Law as a postulate and demonstrated the Second Law. Look at your results for Problems 9 and 13. Can you use them to rationalize the Third Law? Explain your answer.

Yes. The entropy at absolute zero is equal to zero. From problem 9, we can get when T = 0 K, the molecule has the probability of 1 to be in the 0 energy state. $S(T = 0K) = k_B ln(1) = 0$. From number 13, we can also get $S \rightarrow 0$ when $T \rightarrow 0$.

In [ ]:
```
```