HW1

Adriaan Riet

September 15, 2015

First, we need to generate a spline of the data, and remove the points that don't exist from the table. I will be using the Gibbs convention.

```
In [1]: %matplotlib inline
        import scipy.interpolate as sci
        import scipy.optimize as sco
        import scipy.constants as c
        import numpy as np
        from mpl_toolkits.mplot3d import Axes3D
```

```
In [2]: x = np.nan
        gamma = np.array([[73.7,72.5,71.2,68.4,65.3,62.2,59.0],
                          [71.0,69.5,68.1,65.8,63.4,60.5,57.4],
                          [65.0,64.3,63.5,62.2,40.4,58.2,55.5],
                          [62.0,59.9,58.6,56.8,55.2,53.9,52.3],
                          [52.2,51.1,50.3,48.8,47.5,46.6,46.0],
                          [45.9,44.0,42.4,40.9,40.0,39.5,39.2],
                          [35.9,34.8,33.8,32.5,31.4,31.1,31.0],
                          [28.2,x   ,26.6,x   ,23.2,21.4,19.7]],np.float64)

        #Data for this problem set was taken from Vargaftik, Volkov and Voljack, International tables
        #of the surface tension of water,
        #Journal of Physical Chemistry Reference Data, Vol 12, No 3. 1983
        #www.nist.gov/data/PDFfiles/jpcrd231.pdf
        surfaceTensionH2O = np.array([75.64,74.95,74.23,73.5,72.75,71.99,71.20,\
                                      70.41,69.6,68.78,67.94,67.1,66.24,65.36,64.47,63.58,62.67,61.75,6
        TsforST = np.arange(0.,101,5)
        TsforST[0]=0.01
        stH2O = sci.interp1d(TsforST,surfaceTensionH2O)


        conc = np.array([0.007,0.021,0.050,0.104,0.246,0.489,1.006,11.04],np.float64)
        T = np.array([0,10,20,40,60,80,100])
        molNumUnits = "millimole/m^2"
        weights = np.ones(gamma.shape)
        mask = np.isnan(gamma)
        weights[mask]=0
        #gamma[mask]=20
```

Now, we need a relation between γ and everything else. We know that

$$\left|\frac{\mathrm{d}\gamma}{\mathrm{d}T}\right|_{\mu} = -\bar{S}$$

$$\left|\frac{\mathrm{d}\gamma}{\mathrm{d}\mu}\right|_T = -\Gamma$$

with these two relations, we need only to relate concentration to chemical potential.

$$\mu = \mu^\circ + RT\ln(a)$$
$$a(C) = \alpha C$$

We will assume that the concentration is small enough for $\alpha = 1$ in order to make this problem tractable.

$$\therefore a(C) = C \Rightarrow \mu = \mu^\circ + RT\ln(C)$$

and

$$\mathrm{d}\mu = R\ln(C)\mathrm{d}T + \frac{RT}{C}\mathrm{d}C$$

Limiting ourselves to isotherms eliminates the first term in the equation, while constant concentration eliminates the second

```
In [3]: T2 = np.tile(T,(np.size(conc),1))[~mask].ravel()
        c2 = np.tile(conc,(np.size(T),1)).T[~mask].ravel()
        g2 = gamma[~mask].ravel()
        mu2 = c.R*c.C2K(T2) *np.log(c2)
        weights = weights[~mask]
        gammaF=sci.bisplrep(c.C2K(T2),mu2,g2,w=weights.ravel(),quiet=1)
        def gF(x,y,g=gammaF,dx=0,dy=0):
            return sci.bisplev(c.C2K(x),y,gammaF,dx=dx,dy=dy)
```

```
Warning:     The required storage space exceeds the available storage space.
    Probably causes: nxest or nyest too small or s is too small. (fp>s)
        kx,ky=3,3 nx,ny=9,9 m=54 fp=320.604436 s=43.607695
```

```
In [4]: Sbar = - gF(20,c.R*293*np.log(0.05),dx=1)
        Gamma = - gF(20,c.R*293*np.log(0.05),dy=1)
        print("Excess Entropy","= {:.2f} mN/(m K)\r\nExcess Mole Number= {:.2e} {}".format(Sbar,Gamma,m
        def myFunc(x):
            return gF(20,x,dy=1)
        num = sco.minimize(myFunc,-300,method='Powell')
```

```
Excess Entropy = 0.27 mN/(m K)
Excess Mole Number= 2.47e-03 millimole/m^2
```

b) Discuss the meaning of the number for Γ computed in (a) in terms of molecular distribution functions (density of n-butyric acid with z) as might be determined by an MD Simulation. Calculate a reasonable Γ_{Max} for $T = 20°C$

Γ is the excess surface energy per change in chemical potential of the system.

$$\left|\frac{\mathrm{d}\gamma}{\mathrm{d}\mu}\right|_T = -\Gamma$$

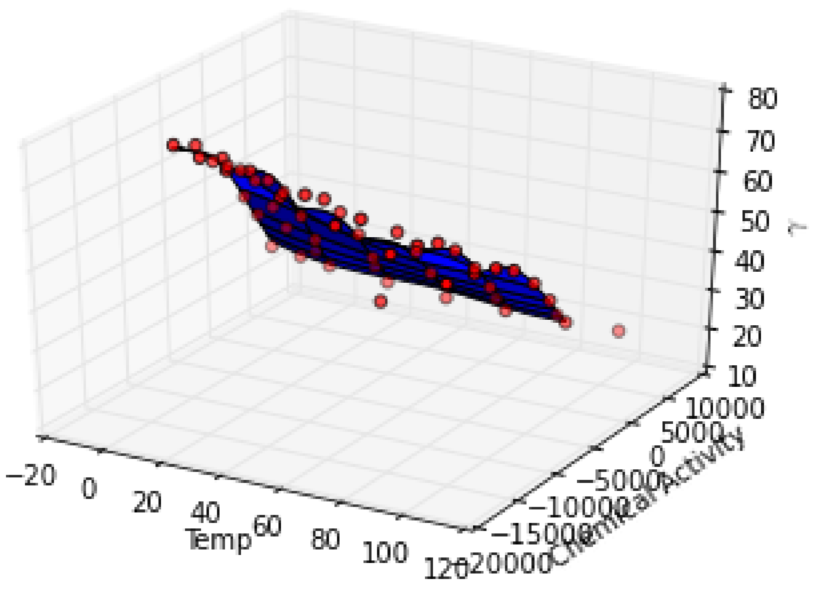
so an increase in the concentration at the surface will tend to reduce surface energy, which means that it is energetically favorable for the system to concentrate n-butyric acid at the surface. this means that the interface would be lined with n-butyric acid.

```
In [5]: print("The maximum excess mole number is {:.2e} {} at 293 Kelvin".format(-myFunc(num.x.item()),m
```

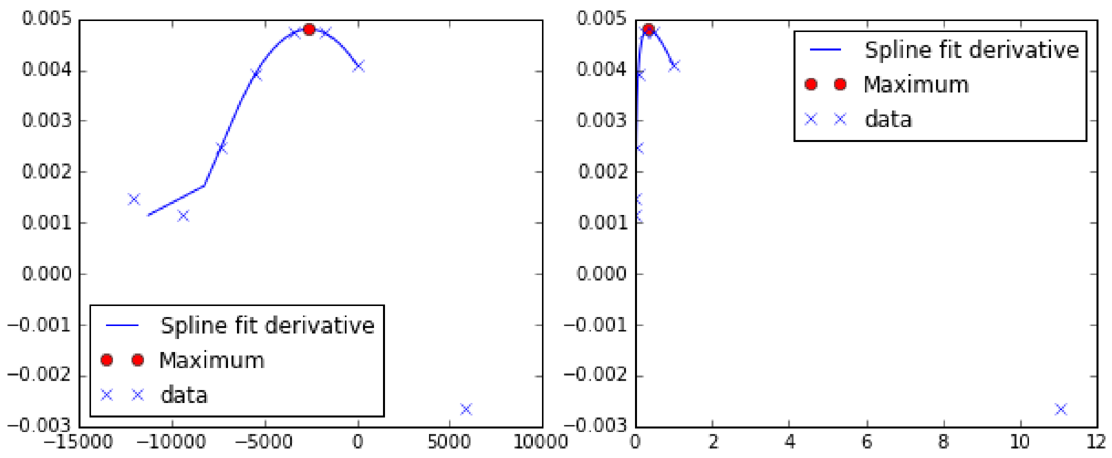
```
The maximum excess mole number is 4.79e-03 millimole/m^2 at 293 Kelvin
```

```
In [6]: Ts = np.linspace(0,100)
        concs = np.linspace(0.007,1)
        mus = c.R*c.C2K(Ts)*np.log(concs)
        X,Y = np.meshgrid(Ts,mus)
        zs = np.array([gF(Ts,mus) for Ts,mus in zip(np.ravel(X),np.ravel(Y))])
        Z = zs.reshape(X.shape)
        fig = plt.figure()
        ax = fig.add_subplot(111,projection='3d')
        ax.plot_surface(X,Y,Z)
        ax.scatter(T2,mu2,g2,c='red')
        ax.set_xlabel('Temp')
        ax.set_ylabel('Chemical Activity')
        ax.set_zlabel('$\gamma$')
```

```
Out[6]: <matplotlib.text.Text at 0x7f0a59982208>
```



```
In [7]: fig=plt.figure(figsize=(10,4))
        fig.add_subplot(121)
        plt.plot(mus,-gF(20,mus,dy=1),label='Spline fit derivative')
        if num.success:
            plt.plot(num.x,-gF(20,num.x,dy=1),'ro',label='Maximum')
            plt.plot(c.R*c.C2K(20)*np.log(conc),-gF(20,c.R*c.C2K(20)*np.log(conc),dy=1),'bx',label='data
            plt.legend(loc="best")
        fig.add_subplot(122)
        plt.plot(concs,-gF(20,mus,dy=1),label='Spline fit derivative')
        if num.success:
            plt.plot(np.exp(num.x/(c.R*c.C2K(20))),-gF(20,num.x,dy=1),'ro',label='Maximum')
            plt.plot(conc,-gF(20,c.R*c.C2K(20)*np.log(conc),dy=1),'bx',label='data')
            plt.legend(loc="best")
```



The thermodynamic variables that can be computed from this table include the following excess properties: Temperature, entropy, enthalpy, chemical potential, excess mole number.

$$\left|\frac{\mathrm{d}\gamma}{\mathrm{d}T}\right|_{\mu} = -\bar{S}$$

$$\left|\frac{\mathrm{d}\gamma}{\mathrm{d}\mu}\right|_{T} = -\Gamma$$

$$\gamma - T\left|\frac{\mathrm{d}\gamma}{\mathrm{d}T}\right| = \bar{H}$$

$$\mu = \mu^{\circ} + RT\ln(a)$$

Taking $\mu^{\circ} = 0$ (excess chemical potential) allows us to calculate the previous derivatives numerically. Doing a spline fit of the table results in the ability to take derivatives, which is important for our purposes. The error will be the smallest for chemical potential, and will be larger for the derivative based properties. The equation for enthalpy, while true, uses both the chemical potential and its derivative, so it should be used with a lot of caution.

The limitations of the Langmuir isotherm include that it cannot accurately predict behavior around the critical micellar concentration, or above that concentration. It also cannot predict when the excess mol number will be negative, which can happen in reality.

```
In [8]: def excessEntropy(T,conc):
            mu = c.R*c.C2K(T)*np.log(conc)
            return -gF(T,mu,dx=1)
        def excessMolNum(T,conc):
            mu = c.R*c.C2K(T)*np.log(conc)
            return -gF(T,mu,dx=1)
        def excessChemicalPotential(T,conc):
            return c.R*c.C2K(T)*np.log(conc)
        def excessEnthalpy(T,conc):
            mu = c.R*c.C2K(T)*np.log(conc)
            return gF(T,mu)- T * gF(T,mu,dx=1)
```

$$\frac{\pi}{kT\Gamma} = 1 + B\Gamma + C\Gamma^2 + \dots$$

$$\frac{\pi}{kT} = \Gamma + B\Gamma^2 + C\Gamma^3 + \dots$$

In order to do a least-squares regression, we define the sum of squares function S

$$S(B,C,\ldots) = \sum_{i=1}^{N}(e_i)^2 = \sum_{i=1}^{N}(\frac{\pi_i}{kT} - \Gamma_i - B\Gamma_i^2 - C\Gamma_i^3 - \ldots)^2$$

$$\frac{\partial S}{\partial B} = \sum_{i=1}^{N} 2(\frac{\pi_i}{kT} - \Gamma_i - B\Gamma_i^2 - C\Gamma_i^3 - \ldots)(-\Gamma^2) = 0$$

$$\frac{\partial S}{\partial C} = \sum_{i=1}^{N} 2(\frac{\pi_i}{kT} - \Gamma_i - B\Gamma_i^2 - C\Gamma_i^3 - \ldots)(-\Gamma^3) = 0$$

And so forth. This can be rewritten in the form:

$$\sum_{i=1}^{N} \frac{\pi_i}{kT} = \sum_{i=1}^{N} \Gamma_i^2 + \sum_{i=1}^{N} B\Gamma_i^3 + \sum_{i=1}^{N} C\Gamma_i^4 + \ldots$$

$$\sum_{i=1}^{N} \frac{\pi_i}{kT} = \sum_{i=1}^{N} \Gamma_i^3 + \sum_{i=1}^{N} B\Gamma_i^4 + \sum_{i=1}^{N} C\Gamma_i^5 + \ldots$$

Or in matrix form:

$$Y = Ax$$

$$Y = \begin{pmatrix} \sum_{i=1}^{N} \frac{\pi_i}{kT}\Gamma_i^2 \\ \sum_{i=1}^{N} \frac{\pi_i}{kT}\Gamma_i^3 \\ \sum_{i=1}^{N} \frac{\pi_i}{kT}\Gamma_i^4 \\ \ldots \end{pmatrix}$$

$$A = \begin{pmatrix} B \\ C \\ D \\ \ldots \end{pmatrix}$$

$$x = \begin{pmatrix} \sum_{i=1}^{N} \Gamma_i^2 & \sum_{i=1}^{N} \Gamma_i^3 & \sum_{i=1}^{N} \Gamma_i^4 & \ldots \end{pmatrix}$$

I will also assume a langmuir isotherm

$$\Gamma = \frac{bC\Gamma_{\max}}{1 + bC}$$

```
In [9]: def y(Temp,deg=2):
            if(Temp in T):
                c2 = np.array([0.007,0.021,0.050,0.104,0.246,0.489,1.006,11.04],np.float64)
                gF2 = sci.interp1d(c2,gamma[:,np.argwhere(T==Temp)].ravel(),kind='linear')
                yOut = np.zeros(deg)
                for n in range(deg):
                    Gamma = -gF(Temp,c.R*c.C2K(Temp)*np.log(c2),dy=1)
                    yOut[n] = np.sum((stH2O(Temp) - gamma[:,np.argwhere(T==Temp)].ravel())\
                                   /(c.k*c.C2K(Temp))*(Gamma**(n+2)))
                return yOut
            else:
                return conc*0
        def x(Temp,deg=2):
            c2 = np.array([0.007,0.021,0.050,0.104,0.246,0.489,1.006,11.04],np.float64)
            if(Temp in T):
                aOut = np.zeros((deg,deg))
                Gamma = -gF(Temp,c.R*c.C2K(Temp)*np.log(c2),dy=1)
                for n in range(deg):
```

```
                for m in range(deg):
                    aOut[n,m] = np.sum(Gamma**(n+m+2))
            return aOut
        else:
            return 0
    answer = np.linalg.solve(x(20),y(20))
    print(answer)
    Gamma = -gF(20,c.R*c.C2K(20)*np.log(np.array([0.007,0.021,0.050,0.104,0.246,0.489,1.006,11.04],
```

```
[  7.72178173e+21  -3.47840251e+23]
```

This work was done with help from Lichao Liu and Rui, as well as a healthy contribution from J. Adin Mann, PhD.