Lab 3: Solution Calorimetry

Max Huggins – UCA Department of Chemistry

February 9, 2021

Abstract

The proton-transfer enthalpies of glycine were studies in this experiment. A solution calorimeter was used to determine the change in temperature for three reactions of glycine. In each, from these two enthalpy values were found for the two stage proton transfer reaction shown below.

$$H_2Gly^+ \longrightarrow H^+ + HGly^{+-} \tag{1}$$

$$HGly^{+-} \longrightarrow H^+ + Gly^- \tag{2}$$

The values for each reaction, ΔH_1 and ΔH_2 (respectively) were determined to be (0.57 and 55.7)kJ. Where these values have an (86.6 and 22.4)% error.

1 Introduction and Objective

This experiment uses a solution calorimeter to measure changes in temperature of a reaction in order to measure the heat produced by the reaction. Prior to this, however a standardization must be made such that the heat capacity of the calorimeter could be determined. This is done following the procedure outlined in the manual [1] for the calorimeter. There, an equation is given that relates the change in temperature of a standard solution with the heat capacity of the calorimeter. The calculations are shown in the python script in the appendix.
After standardization the relationships shown in the original source [2] are used for the determining of the enthalpy values. I interpreted the original source to be providing three methods of determining the enthalpy of the reactions. One uses the experimental approach with the calorimeter, another used the King's equations and a thermodynamic relationship, and the last uses that same thermodynamic relationship along with stoichiometric values. I used the King's equations to determine the expected values for the enthalpies. Meaning the percent error I have included is the error of the experimental value with respect to the King's equation values.

2 Procedure

2.1 Overview

As mentioned prior, the procedure for this lab was taken from [2]. However, the procedure was mainly just measuring the correct values for glycine and adding it to a dewer.

2.2 Details

Solutions of the HCl and NaOH were provided and the only measuring done was to measure 100mL of the solution to be added to the dewer. After this was done, 20mmols of the solid glycine were measured and the data collected for the experiment was started. A logger pro was used to acquire data and after data collection was completed, a new solution was added to the calorimeter and this was continued until all the reactions had been measured.

3 Results and Analysis

The data was imported into a python script and all analysis was done there. Figure 1 shows plots for the temperatures over time. From these, all the values required to complete the enthalpy calculation are found and the output of the code shows the results.

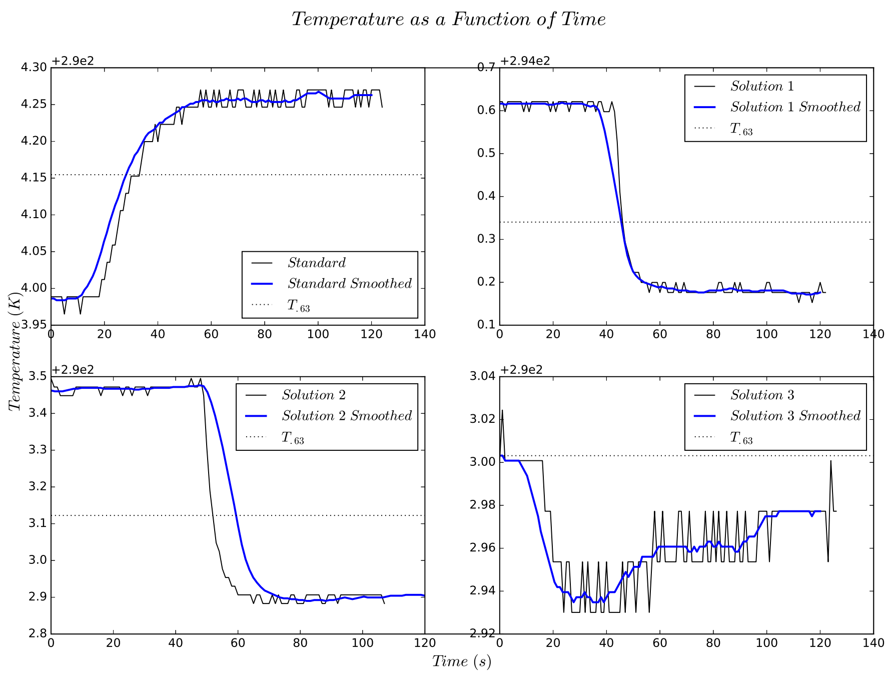


Figure 1: Plots of the temperature over time for each solution.

I have added a line for the 63% complete mark for the temperature as well as a smoothed curve to achieve a more linear top and bottom value.

4 Conclusion

The determination of the enthalpy was done two ways here. The first used a calorimeter with experimental data. The second used known equations and was used as a comparison for the previous. All in all the data acquired was not favorable. It can be seen that the third solution has very little resemblance to the others. The changes in temperature are so small that the resolution of the analog to digital converter for the sensor can clearly be seen. Additionally, the value acquired for the second reaction's enthalpy doesn't even come close to the predicted value. The manual for the calorimeter has inconsistent notation and caused quite a lot of confusion for the reader and the source paper has very little explanation on the theory behind the relationships acquired. After much consideration I was not able to sort through the paper such as to produce a coherent result.

5 Appendix

Here I have included the script written for this lab. I have also included the output of the code in the form of a photo from the kernel.

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 18 22:41:20 2020

@author: maxhu
collaborator grant
"""

#==============================================================================#
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
plt.style.use('classic')
#==============================================================================#
SpecificHeatofHCl = .99894
massofHCl = 100
SmoothFactor = 10
width = 2
MassofTRIS = .499
MMofGlycine = 75.07
CalorietoJConversion = 4.184
T = 298.15
R = 8.314
#==============================================================================#
def DataExtractor(x,y,path):
    lines = np.loadtxt(path)
    for line in lines:
        x.append(line[0])
        y.append(line[1])

def LinearRegression(x,y):
    m, b, r_value, p_value, std_err = stats.linregress(
            x,y)
    return m*x + b

def TemperatureChange(TempData, Solution):
    ChangeInTemp = round(np.max(TempData) - np.min(TempData), 2)
    if TempData[0] < TempData[len(TempData)-1-SmoothFactor]:
        T_63 = TempData[len(TempData) - SmoothFactor] - (1 - .63) * abs(ChangeInTemp)
        plt.axhline(T_63, alpha=.7, linestyle='dotted', color='black', label='$T_{.63}$')
        plt.legend(loc='best')
        print('The change in temperature of {} is {}K and T_63 is {}K'.format(Solution, ChangeInTemp, round(T_63,2)))
    elif TempData[0] > TempData[len(TempData)-SmoothFactor]:
        T_63 = (1 - .63) * abs(ChangeInTemp) + TempData[len(TempData)-SmoothFactor]
        plt.axhline(T_63, alpha=.9, linestyle='dotted', color='black', label='$T_{.63}$')
        plt.legend(loc='best')
        print('The change in temperature of {} is {}K and T_63 is {}K'.format(Solution, ChangeInTemp, round(T_63,2)))
    return ChangeInTemp

def Smooth(y, N):
    cumsum = np.cumsum(np.insert(y, 0, 0))
    y_smooth = (cumsum[N:] - cumsum[:-N]) / float(N)
    return y_smooth

def Enthalpy(TempData, MassofGlycine, Solution):
    ChangeInTemp = round(np.min(TempData) - np.max(TempData), 2)
    Q_n = -HeatCapacityofCalorimeter * ChangeInTemp
    H = Q_n / (MassofGlycine / MMofGlycine)
    if Solution == 'NULL':
        pass
    else:
        print('The enthalpy of {} is {}kJ'.format(Solution, round(H * 1e-3,2)))
    return H

def ErrorInEnthalpy(Enthalpy, Reaction, ExpectedEnthalpy):
    Error = abs(((ExpectedEnthalpy - Enthalpy) / ExpectedEnthalpy)*100)
    print("The %err for {}'s enthalpy is {}%".format(Reaction, round(Error,1)))
#==========================================================================#
StandardTime = []
StandardTemp = []
#=============================================#
Solution1Time = []
Solution1Temp = []
```

```python
#==================================================#
Solution2Time = []
Solution2Temp = []
#==================================================#
Solution3Time = []
Solution3Temp = []
#================================================================================#
DataExtractor(StandardTime,StandardTemp,'Cleaned_Data\standard.txt')
StandardTime = np.array(StandardTime)
StandardTemp = np.array(StandardTemp) + 273.15
#==================================================#
DataExtractor(Solution1Time,Solution1Temp,'Cleaned_Data\sol_1.txt')
Solution1Time = np.array(Solution1Time)
Solution1Temp = np.array(Solution1Temp) + 273.15
#==================================================#
DataExtractor(Solution2Time,Solution2Temp,'Cleaned_Data\sol_2.txt')
Solution2Time = np.array(Solution2Time)
Solution2Temp = np.array(Solution2Temp) + 273.15
#==================================================#
DataExtractor(Solution3Time,Solution3Temp,'Cleaned_Data\sol_3.txt')
Solution3Time = np.array(Solution3Time)
Solution3Temp = np.array(Solution3Temp) + 273.15
#================================================================================#
size = 20
size_config = .8
fig = plt.figure(1,figsize=(15,10))
fig.suptitle('$Temperature\ as\ a\ Function\ of\ Time$', fontsize=size)
plt.tick_params(labelcolor='none', top='off', bottom='off', left='off', right='off')
plt.ticklabel_format(style='plain', axis='both', scilimits=(0,0))
plt.xlabel('$Time\ (s)$', fontsize=size_config*size)
plt.ylabel('$Temperature\ (K)$', fontsize=size_config*size)
#==================================================#
fig.add_subplot(221)
plt.plot(StandardTime,StandardTemp, color='black', label='$Standard$')
xDataSmoothed = np.linspace(0,120,len(Smooth(StandardTemp, SmoothFactor)))
plt.plot(xDataSmoothed,Smooth(StandardTemp, SmoothFactor),
         label='$Standard\ Smoothed$', linewidth=width)
plt.legend(loc='best')
TemperatureChange(Smooth(StandardTemp, SmoothFactor), 'standard')

fig.add_subplot(222)
plt.plot(Solution1Time,Solution1Temp, color='black', label='$Solution\ 1$')
xDataSmoothed = np.linspace(0,120,len(Smooth(Solution1Temp, SmoothFactor)))
plt.plot(xDataSmoothed,Smooth(Solution1Temp, SmoothFactor),
         label='$Solution\ 1\ Smoothed$', linewidth=width)
plt.legend(loc='best')
TemperatureChange(Smooth(Solution1Temp, SmoothFactor), 'solution 1')

fig.add_subplot(223)
plt.plot(Solution2Time,Solution2Temp, color='black', label='$Solution\ 2$')
xDataSmoothed = np.linspace(0,120,len(Smooth(Solution2Temp, SmoothFactor)))
plt.plot(xDataSmoothed,Smooth(Solution2Temp, SmoothFactor),
         label='$Solution\ 2\ Smoothed$', linewidth=width)
```

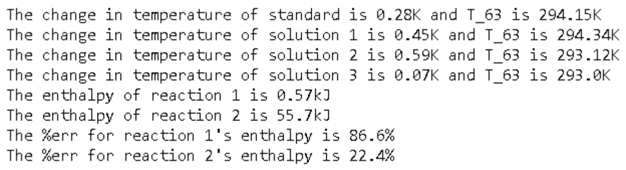
```
plt.legend(loc='best')
TemperatureChange(Smooth(Solution2Temp, SmoothFactor), 'solution 2')

fig.add_subplot(224)
plt.plot(Solution3Time,Solution3Temp, color='black', label='$Solution\ 3$')
xDataSmoothed = np.linspace(0,120,len(Smooth(Solution3Temp, SmoothFactor)))
plt.plot(xDataSmoothed,Smooth(Solution3Temp, SmoothFactor),
          label='$Solution\ 3\ Smoothed$', linewidth=width)
plt.legend(loc='best')
TemperatureChange(Smooth(Solution3Temp, SmoothFactor), 'solution 3')
#===================================================================#
plt.savefig('Solution_Calorimetry.eps')
#==============================================================================================#
StandardTemp = Smooth(StandardTemp, SmoothFactor)
ChangeInTempofStandard = np.max(StandardTemp) - np.min(StandardTemp)
T_63 = StandardTemp[len(StandardTemp) - SmoothFactor] - (1 - .63) * abs(
    ChangeInTempofStandard) - 273.15
HeatCapacityofHCl = massofHCl * SpecificHeatofHCl
HeatofStandard = MassofTRIS * (58.738 + 0.3433 * (25 - T_63))
TotalHeatCapacity = HeatofStandard / ChangeInTempofStandard
HeatCapacityofCalorimeter = CalorietoJConversion * (TotalHeatCapacity -
    HeatCapacityofHCl)

EnthalpyforReaction1 = Enthalpy(Solution1Temp, 1.5035, 'reaction 1')
EnthalpyforReaction1
EnthalpyforReaction2 = Enthalpy(Solution3Temp, 1.5048, 'NULL') - Enthalpy(
    Solution2Temp, 1.5016, 'NULL') + 13465*CalorietoJConversion
print('The enthalpy of {} is {}kJ'.format('reaction 2', round(
    EnthalpyforReaction2 * 1e-3,2)))

ExpectedEnthalpyReaction1 = 5477.04 * R - 16.64 * R * T
ExpectedEnthalpyReaction2 = 7290.75 * R - 6.09 * R * T

ErrorInEnthalpy(EnthalpyforReaction1, 'reaction 1', ExpectedEnthalpyReaction1)
ErrorInEnthalpy(EnthalpyforReaction2, 'reaction 2', ExpectedEnthalpyReaction2)
```



```
The change in temperature of standard is 0.28K and T_63 is 294.15K
The change in temperature of solution 1 is 0.45K and T_63 is 294.34K
The change in temperature of solution 2 is 0.59K and T_63 is 293.12K
The change in temperature of solution 3 is 0.07K and T_63 is 293.0K
The enthalpy of reaction 1 is 0.57kJ
The enthalpy of reaction 2 is 55.7kJ
The %err for reaction 1's enthalpy is 86.6%
The %err for reaction 2's enthalpy is 22.4%
```

Figure 2: Output of the above code.

References

[1] Parr Instrument Company. *1455 SOLUTION CALORIMETER Operating Instruction Manual*. Parr Instrument Company, 211 Fifty-Third Street Moline, Illinois 61265 USA, 1 edition, 6 1997.

[2] RW Ramette. Solution calorimetry in the advanced laboratory: A study of glycine proton-transfer enthalpies. *Journal of Chemical Education*, 61(1):76, 1984.