**Title**
Deep UV Second Harmonic Generation Studies of Thiocyanate at Hydrophobe/Water Interfaces



**Author**
McCaffrey, Debra Lynn

**Publication Date**
2016

Peer reviewed|Thesis/dissertation

# Deep UV Second Harmonic Generation Studies of Thiocyanate at Hydrophobe/Water Interfaces

By

Debra Lynn McCaffrey

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

In

Chemistry

In the

Graduate Division

Of the

University of California, Berkeley

Committee in charge:

Professor Richard J. Saykally, Chair
Professor Phillip L. Geissler
Professor Feng Wang

Fall 2016

Abstract

Deep UV Second Harmonic Generation Studies of Thiocyanate at Hydrophobe/Water Interfaces

By

Debra Lynn McCaffrey

Doctor of Philosophy in Chemistry

University of California, Berkeley

Professor Richard J. Saykally, Chair

Our understanding of the air/water interface and ion adsorption to this interface has developed rapidly in the last decade. While tremendous progress has been made in this area, it is essential for the field to branch out into additional interfaces. This dissertation describes the work I've done to extend the field to additional hydrophobic interfaces.

Chapter 1 gives historical context for the work done on the air/water interface. Macroscopic measurements, such as surface tension, suggested that ions should be depleted from the air/water interface. Microscopic measurements, such as molecular dynamics simulations and second harmonic generation spectroscopy, show that some ions are enhanced at the air/water interface and that the degree of enhancement follows the Hofmeister series. The mechanism that drives adsorption is a delicate balance between factors such as solvent repartitioning, electrostatics, capillary waves, and configurational entropy.

Chapter 2 presents nonlinear optical spectroscopy theory and outlines the experimental apparatus. It also details the sample preparation and the data analysis procedure. The processed data are fit to a Langmuir model, derived therein.

Chapter 3 details studies on hydrocarbon/water interfaces. Hydrocarbons provide a condensed phase analogue to the air/water interface. While the data are preliminary and need to be refined, the general trend is that thin layers of hydrocarbons have little effect on the free energy of adsorption of thiocyanate.

Chapter 4 details studies on the graphene/water interface. While graphene has a high charge carrier mobility, it seems that graphene behaves like a hydrophobe in regards to thiocyanate adsorption. Experimental studies showed that the free energy of adsorption to the graphene/water interface is similar to that of the air/water interface. Molecular dynamics simulations reveal that the adsorption mechanism is drastically different, however.

Chapter 5 evolved out of a concern to include a surface potential term in the Langmuir model. This created nested models that were compared with several model comparison metrics. The better model seems to depend on the dataset. The method of calculating errors for the fit parameters was also examined, but further work still needs to be done on finding the most accurate method.

Chapter 6 presents some broad conclusions and directions for future study. It seems that thin, uncharged monolayers inserted into an interface have little effect on the free energy of adsorption, although the mechanism can change. This chapter also proposes a new study to help elucidate the effect of surface charge. Surfactants can mix with alkanes to form an ordered monolayer on water. Varying the concentration of the surfactant can vary the surface charge.

# TABLE OF CONTENTS

**Acknowledgements**

Thanks to Rich, of course, for guiding me through this process and putting up with all times I went straight for the wallet. I had a fun time making science and music with you.

Thanks to my parents. You're the ones who got me to Berkeley, figuratively and literally. I couldn't have started this without you.

Thanks to all my friends at Open Door UMC for supporting me throughout this process. Thanks also to Vic. I wouldn't have stuck it out to the end if it weren't for you. I couldn't have ended this without all of you.

# Chapter 1 – Introduction

This dissertation work examines the nature of aqueous electrolyte interfaces by the combination of nonlinear optical laser spectroscopy and molecular dynamics modeling. Here we present an overview of the underlying theory and context.

## 1.1 Surface Tension and the Gibbs Adsorption Equation

There are many chemists who have contributed to the field to the point where every student knows their names, but none so familiar as Josiah Willard Gibbs. Among his many contributions, his work on surface and interfacial thermodynamics is of particular interest. Gibbs developed an equation that relates the surface excess of a solute to the surface tension[1]

$$\Gamma = -\frac{d\gamma}{d\mu}, \tag{1.1}$$

where $\Gamma$ is the surface excess of the solute, $\mu$ is the chemical potential of the solute, and $\gamma$ is the surface tension of the solution. Equation (1.1) is called the Gibbs adsorption equation. Since chemical potential increases with concentration, if an increase in concentration causes an increase in surface tension, then the solute is predicted to be depleted at the surface, and vice versa.

Jones and Ray performed a series of experiments[2–5] to measure the surface tension of several salts with painstaking precision. A number of surface tension curves had minima at millimolar concentrations (a separate phenomenon called the Jones-Ray effect), but otherwise, the slopes were all positive, indicating the ions should not be present at the surface. Since then, many more surface tension studies have been performed and the only simple ionic solutes that produce negative slopes are inorganic acids.[6]

## 1.2 Onsager-Samaras Theory and the Method of Image Charges

Onsager and Samaras strove to describe this behavior analytically.[7] The derivation started with the method of image charges from electrostatics,[8,9] incorporated screening and the Gibbs adsorption equation, and produced a limiting law for the surface tension

$$\sigma = \sigma_0 + const.* c \log\left(\frac{const.}{c}\right), \tag{1.2}$$

where $\sigma$ is the surface tension of the solution, $\sigma_0$ is the surface tension of the pure solvent, $c$ is concentration, and the constants depend on the system of interest.

The important factor in the derivation is the method of image charges. In electrostatics, there is a theorem, called the uniqueness theorem, which states that the solution to Laplace's equation in a volume with a specific set of boundary conditions is unique.[8] In practice, this means a complex scenario can be replaced with a simpler scenario that has the same boundary conditions. The potentials in both scenarios will be the same and the simpler scenario can be used to derive the electrostatic potential.

Imagine two dielectric media with static dielectric constants $\epsilon_1$ and $\epsilon_2$. Embed a point charge with charge $q$ in medium 1 a distance $d$ from the interface. The scenario is illustrated in Figure 1.1.

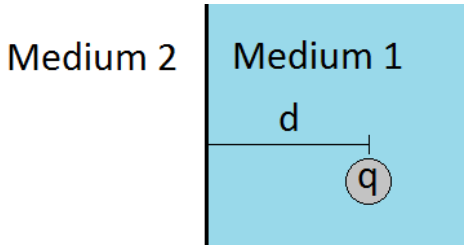



Figure 1.1: A charge $q$ embedded in a medium with static dielectric constant $\epsilon_1$ a distance $d$ from another medium with dielectric constant $\epsilon_2$.

The boundary condition is that the components of the electric field must approach equality at the interface between the media, noting that the perpendicular components are also multiplied by the respective static dielectric constants.[9] A scenario with the same condition is two charges, $q$ and $q_{image}$, a distance $2d$ apart. Solving Laplace's equation for the boundary condition yields an expression for $q_{image}$:

$$q_{image} = -\frac{\epsilon_2 - \epsilon_1}{\epsilon_2 + \epsilon_1} q. \tag{1.3}$$

For ions at the air/water interface (at 293K), medium 1 is water and medium 2 is air, so $\epsilon_1 = 80$, $\epsilon_2 = 1$, and $q_{image} \approx q$. Therefore, ions are repelled electrostatically from the interface.

## 1.3 The Microscopic Picture Emerges

One thing to note is that surface tension is a macroscopic measurement, devoid of microscopic information. However, there were still some inklings that the macroscopic picture did not capture the whole story.

For one, the slopes of the concentration isotherms for simple salts are positive, but the magnitude of the slopes differ greatly. For example, sulfate salts tend to have steep slopes, while iodide salts tend to have shallow slopes.[6] Looking back to the Gibbs adsorption equation, this suggest that some salts have a greater surface depletion than others. Surprisingly, the magnitudes don't correlate with charge; the halides all have vastly different slopes.[6] Clearly, a molecular detail is missing from the description.

The second indication came from atmospheric chemistry. Both bromide and chloride participate in tropospheric ozone reactions, but the rate constants for bromide reactions are much larger,[10] despite the fact that there are roughly 650 moles of chloride for every mole of bromide in sea water (the source of halides in the atmosphere). These reactions are known to occur at the surface of sea salt aerosol particles. One of the possible explanations was that bromide was present at the aerosol surface in greater concentrations than chloride, despite chloride having a much greater bulk concentration.

This explanation gained traction due to theoretical work by Jungwirth and Tobias.[11–13] They performed molecular dynamics (MD) simulations of sodium halides in a water slab.[11] The number densities of the species in the direction perpendicular to the interface showed that iodide had a large preference for the first interfacial liquid layer, bromide had some preference, chloride had no preference, and fluoride was absent. The cations then formed a double layer with the anions. Interestingly, for the ions that preferred the interface, the subsequent interfacial layers showed depletions such that the integrated surface region showed a net depletion, consistent with the surface tension data. The authors postulated that the difference in preference was due to ion size and polarizability.[12]

The simulations were confirmed indirectly through vibrational sum frequency generation (VSFG) experiments that examined the structure of interfacial water and how ions perturbed it.[14] Direct confirmation came through second harmonic generation (SHG) experiments. Petersen and Saykally showed that azide was enhanced at the air/water interface with a free energy of adsorption of -9.9 ± 0.3 kJ/mol.[15] Petersen et al. also showed that NaI and KI were enhanced in the dilute concentration range, with free energies of -6.1 ± 0.2 and -6.3 ± 0.2, respectively.[16] Photoelectron spectroscopy provided another direct confirmation.[17] Numerous VSFG[18,19] and SHG[20–27] studies followed. Two patterns emerged: One, the degree of enhancement follows the Hofmeister series.[28,29] Two, the countercations do not affect surface adsorption.[16,20,30]

## 1.4 The Picture Comes into Focus

Nearly a decade of work has led to a number of refinements in the story of selective ion adsorption to the air/water interface. Further studies agree that countercations do not affect surface adsorption.[31,32] Polarizability no longer seems to be the explanation for the degree of enhancement, as postulated by Jungwirth and Tobias.[12] In fact, the explanation for enhancement and the explanation for the Hofmeister series are now considered one and the same. It seems that the mechanism for the Hofmeister series is different for anions than for cations,[6,33] but a complete mechanism is more complex, involving all interactions in the interfacial environment.[34]

### *1.4.1 New techniques*

Several new techniques have emerged to expand our knowledge even further. The first is phase-sensitive sum frequency generation (PS-SFG). As the name suggests, the technique is able to directly measure the imaginary component of the spectra generated from SFG, which is responsible for resonance effects and gives orientation information.[35] In the vibrational region, this technique has been used to determine how water orientation – namely, in which direction the water hydrogens point – changes in response to added ions.[36] This has allowed researchers to indirectly probe ions not at the interface (such as sulfates and carbonates) and ions that have no easily accessed transitions (such as monatomic cations). PS-SFG has also led to a better understanding of the neat air/water interface as well. Combining isotopic dilution with the orientational information of PS-SFG has allowed the water spectrum to be assigned.[37,38]

The second set of new techniques are nonlinear scattering techniques[39], including second harmonic scattering[40,41] (SHS) and sum frequency scattering[42] (SFS). These techniques allow the interfaces of centrosymmetric particles, such as oil nanodroplets,[43,44] in bulk to be probed. It has also been applied to liquid microjets to collect scattering signal from the air/water interface.[45] This exciting technique has the potential to probe systems unavailable to SHG and SFG experiments.

### *1.4.2 A thermodynamic mechanism for adsorption*

The advantage of SHG over SFG is that SHG can provide a quantitative result, namely the free energy of adsorption. Further temperature dependent measurements can then elucidate the enthalpy and entropy changes of the adsorption process. Otten et al. did just this for thiocyanate at the air/water interface.[46] They found that a negative enthalpy change drives the ion adsorption, while a negative entropy change impedes it. MD simulations suggested the underlying mechanism: First, when the ion moves from the bulk to the interface, weakly interacting water molecules are displaced from both the surface and the ion solvent shell into the bulk solution, where they form stronger water-water bonds, leading to a negative enthalpy change. Second, the presence of an ion at the interface dampens its capillary wave fluctuations, leading to a negative entropy change.

## 1.5 Present Work

The present work expands upon the SHG studies of Petersen, Onorato, and Otten. Chapter 2 presents an introduction to SHG theory and details the experimental procedure for collecting signal from aqueous solution samples. Chapter 3 explores several hydrocarbon/water interfaces. While those final results are not easily interpreted, there are suggestions for future work to resolve the interpretation. Chapter 4 explores the graphene/water interface and compares its properties with those of air/water. Through experiments and MD simulations, we discovered that the free energy of adsorption remains similar, but the mechanism of adsorption is qualitatively different. Chapter 5 examines an extension of the model used to describe the SHG signal vs concentration isotherm. It also explores methods for improving the fit to the model and the errors reported. Chapter 6 makes some general conclusions regarding thin uncharged monolayers at interfaces and suggests some future directions for the project.

## 1.6 References


(1) Chattoraj, D. K.; Birdi, K. S. *Adsorption and the Gibbs Surface Excess*; Springer US: Boston, MA, 1984.

(2) Jones, G.; Ray, W. A. The Surface Tension of Solutions of Electrolytes as a Function of the Concentration. I. A Differential Method for Measuring Relative Surface Tension. *J. Am. Chem. Soc.* **1937**, *59* (1), 187–198.

(3) Jones, G.; Ray, W. A. The Surface Tension of Solutions of Electrolytes as a Function of the Concentration II*. *J. Am. Chem. Soc.* **1941**, *63* (1), 288–294.

(4) Jones, G.; Ray, W. A. The Surface Tension of Solutions of Electrolytes as a Function of the Concentration. III. Sodium Chloride. *J. Am. Chem. Soc.* **1941**, *63* (12), 3262–3263.

(5) Jones, G.; Ray, W. A. The Surface Tension of Solutions of Electrolytes as a Function of the Concentration. IV. Magnesium Sulfate. *J. Am. Chem. Soc.* **1942**, *64* (12), 2744–2745.
(6) Pegram, L. M.; Record, M. T. Hofmeister Salt Effects on Surface Tension Arise from Partitioning of Anions and Cations between Bulk Water and the Air−Water Interface. *J. Phys. Chem. B* **2007**, *111* (19), 5411–5417.
(7) Onsager, L.; Samaras, N. N. T. The Surface Tension of Debye-Huckel Electrolytes. *J. Chem. Phys.* **1934**, *2* (8), 528–536.
(8) Griffiths, D. J. *Introduction to Electrodynamics*, Fourth edition.; Pearson: Boston, 2013.
(9) Jackson, J. D. *Classical Electrodynamics*, 3rd ed.; Wiley: New York, 1999.
(10) Finlayson-Pitts, B. J.; Hemminger, J. C. Physical Chemistry of Airborne Sea Salt Particles and Their Components. *J. Phys. Chem. A* **2000**, *104* (49), 11463–11477.
(11) Jungwirth, P.; Tobias, D. J. Molecular Structure of Salt Solutions: A New View of the Interface with Implications for Heterogeneous Atmospheric Chemistry. *J. Phys. Chem. B* **2001**, *105* (43), 10468–10472.
(12) Jungwirth, P.; Tobias, D. J. Ions at the Air/Water Interface. *J Phys Chem B* **2002**, *106* (25), 6361–6373.
(13) Jungwirth, P.; Tobias, D. J. Specific Ion Effects at the Air/Water Interface. *Chem Rev* **2006**, *106* (4), 1259–1281.
(14) Liu, D.; Ma, G.; Levering, L. M.; Allen, H. C. Vibrational Spectroscopy of Aqueous Sodium Halide Solutions and Air−Liquid Interfaces: Observation of Increased Interfacial Depth. *J. Phys. Chem. B* **2004**, *108* (7), 2252–2260.
(15) Raymond, E. A.; Richmond, G. L. Probing the Molecular Structure and Bonding of the Surface of Aqueous Salt Solutions. *J. Phys. Chem. B* **2004**, *108* (16), 5051–5059.
(16) Petersen, P. B.; Saykally, R. J. Confirmation of Enhanced Anion Concentration at the Liquid Water Surface. *Chem. Phys. Lett.* **2004**, *397* (1–3), 51–55.
(17) Petersen, P. B.; Johnson, J. C.; Knutsen, K. P.; Saykally, R. J. Direct Experimental Validation of the Jones–Ray Effect. *Chem. Phys. Lett.* **2004**, *397* (1–3), 46–50.
(18) Ghosal, S.; Hemminger, J. C.; Bluhm, H.; Mun, B. S.; Hebensteit, E. L. D.; Ketteler, G.; Ogletree, D. F.; Requejo, F. G.; Salmeron, M. Electron Spectroscopy of Aqueous Solution Interfaces Reveals Surface Enhancement of Halides. *Science* **2005**, *307* (5709), 563–566.
(19) Gopalakrishnan, S.; Liu, D.; Allen, H. C.; Kuo, M.; Shultz, M. J. Vibrational Spectroscopic Studies of Aqueous Interfaces: Salts, Acids, Bases, and Nanodrops. *Chem. Rev.* **2006**, *106* (4), 1155–1175.
(20) Jubb, A. M.; Hua, W.; Allen, H. C. Environmental Chemistry at Vapor/Water Interfaces: Insights from Vibrational Sum Frequency Generation Spectroscopy. *Annu Rev Phys Chem* **2012**, *63* (1), 107–130.
(21) Petersen, P. B.; Saykally, R. J. Evidence for an Enhanced Hydronium Concentration at the Liquid Water Surface. *J Phys Chem B* **2005**, *109* (16), 7976–7980.
(22) Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. Enhanced Concentration of Polarizable Anions at the Liquid Water Surface: SHG Spectroscopy and MD Simulations of Sodium Thiocyanide. *J Phys Chem B* **2005**, *109* (21), 10915–10921.
(23) Petersen, P. B.; Saykally, R. J. Adsorption of Ions to the Surface of Dilute Electrolyte Solutions: The Jones−Ray Effect Revisited. *J Am Chem Soc* **2005**, *127* (44), 15446–15452.

(24) Petersen, P. B.; Saykally, R. J. Probing the Interfacial Structure of Aqueous Electrolytes with Femtosecond Second Harmonic Generation Spectroscopy. *J Phys Chem B* **2006**, *110* (29), 14060–14073.
(25) Petersen, P. B.; Saykally, R. J. ON THE NATURE OF IONS AT THE LIQUID WATER SURFACE. *Annu. Rev. Phys. Chem.* **2006**, *57*, 333–364.
(26) Petersen, P. B.; Saykally, R. J. Is the Liquid Water Surface Basic or Acidic? Macroscopic vs. Molecular-Scale Investigations. *Chem. Phys. Lett.* **2008**, *458* (4–6), 255–261.
(27) Otten, D. E.; Petersen, P. B.; Saykally, R. J. Observation of Nitrate Ions at the Air/Water Interface by UV-Second Harmonic Generation. *Chem. Phys. Lett.* **2007**, *449* (4–6), 261–265.
(28) Otten, D. E.; Onorato, R.; Michaels, R.; Goodknight, J.; Saykally, R. J. Strong Surface Adsorption of Aqueous Sodium Nitrite as an Ion Pair. *Chem. Phys. Lett.* **2012**, *519–520* (0), 45–48.
(29) Gurau, M. C.; Lim, S.-M.; Castellana, E. T.; Albertorio, F.; Kataoka, S.; Cremer, P. S. On the Mechanism of the Hofmeister Effect. *J. Am. Chem. Soc.* **2004**, *126* (34), 10522–10523.
(30) Zhang, Y.; Cremer, P. S. Chemistry of Hofmeister Anions and Osmolytes. *Annu. Rev. Phys. Chem.* **2010**, *61* (1), 63–83.
(31) Weber, R.; Winter, B.; Schmidt, P. M.; Widdra, W.; Hertel, I. V.; Dittmar, M.; Faubel, M. Photoemission from Aqueous Alkali-Metal−Iodide Salt Solutions Using EUV Synchrotron Radiation. *J. Phys. Chem. B* **2004**, *108* (15), 4729–4736.
(32) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.
(33) Rizzuto, A. M.; Irgen-Gioro, S.; Eftekhari-Bafrooei, A.; Saykally, R. J. Broadband Deep UV Spectra of Interfacial Aqueous Iodide. *J. Phys. Chem. Lett.* **2016**, *7* (19), 3882–3885.
(34) Nihonyanagi, S.; Yamaguchi, S.; Tahara, T. Counterion Effect on Interfacial Water at Charged Interfaces and Its Relevance to the Hofmeister Series. *J. Am. Chem. Soc.* **2014**, *136* (17), 6155–6158.
(35) Jungwirth, P.; Cremer, P. S. Beyond Hofmeister. *Nat. Chem.* **2014**, *6* (4), 261–263.
(36) Hua, W.; Jubb, A. M.; Allen, H. C. Electric Field Reversal of Na2SO4, (NH4)2SO4, and Na2CO3 Relative to CaCl2 and NaCl at the Air/Aqueous Interface Revealed by Heterodyne Detected Phase-Sensitive Sum Frequency. *J Phys Chem Lett* **2011**, *2* (20), 2515–2520.
(37) Verreault, D.; Hua, W.; Allen, H. C. From Conventional to Phase-Sensitive Vibrational Sum Frequency Generation Spectroscopy: Probing Water Organization at Aqueous Interfaces. *J. Phys. Chem. Lett.* **2012**, *3* (20), 3012–3028.
(38) Tian, C.-S.; Shen, Y. R. Isotopic Dilution Study of the Water/Vapor Interface by Phase-Sensitive Sum-Frequency Vibrational Spectroscopy. *J. Am. Chem. Soc.* **2009**, *131* (8), 2790–2791.
(39) Nihonyanagi, S.; Ishiyama, T.; Lee, T.; Yamaguchi, S.; Bonn, M.; Morita, A.; Tahara, T. Unified Molecular View of the Air/Water Interface Based on Experimental and Theoretical $\chi^{(2)}$ Spectra of an Isotopically Diluted Water Surface. *J. Am. Chem. Soc.* **2011**, *133* (42), 16875–16880.
(40) Roke, S.; Gonella, G. Nonlinear Light Scattering and Spectroscopy of Particles and Droplets in Liquids. *Annu Rev Phys Chem* **2012**, *63* (1), 353–378.

(41) Wang, H.; Yan, E. C. Y.; Borguet, E.; Eisenthal, K. B. Second Harmonic Generation from the Surface of Centrosymmetric Particles in Bulk Solution. *Chem. Phys. Lett.* **1996**, *259* (1–2), 15–20.
(42) Liu, Y.; Dadap, J. I.; Zimdars, D.; Eisenthal, K. B. Study of Interfacial Charge-Transfer Complex on $TiO_2$ Particles in Aqueous Suspension by Second-Harmonic Generation. *J. Phys. Chem. B* **1999**, *103* (13), 2480–2486.
(43) Roke, S.; Roeterdink, W. G.; Wijnhoven, J. E. G. J.; Petukhov, A. V.; Kleyn, A. W.; Bonn, M. Vibrational Sum Frequency Scattering from a Submicron Suspension. *Phys. Rev. Lett.* **2003**, *91* (25).
(44) Vácha, R.; Rick, S. W.; Jungwirth, P.; de Beer, A. G. F.; de Aguiar, H. B.; Samson, J.-S.; Roke, S. The Orientation and Charge of Water at the Hydrophobic Oil Droplet–Water Interface. *J. Am. Chem. Soc.* **2011**, null.
(45) de Aguiar, H. B.; Samson, J.-S.; Roke, S. Probing Nanoscopic Droplet Interfaces in Aqueous Solution with Vibrational Sum-Frequency Scattering: A Study of the Effects of Path Length, Droplet Density and Pulse Energy. *Chem. Phys. Lett.* **2011**, *512* (1–3), 76–80.
(46) Smolentsev, N.; Chen, Y.; Jena, K. C.; Brown, M. A.; Roke, S. Sum Frequency and Second Harmonic Generation from the Surface of a Liquid Microjet. *J. Chem. Phys.* **2014**, *141* (18), 18C524.
(47) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.

# Chapter 2 – Apparatus and Methodology

This chapter describes the spectroscopic technique used, the essential theory of second harmonic generation (SHG), and the general experimental setup used to collect data. It also discusses the measurement processing and data analysis.

## 2.1 Nonlinear optical processes and SHG spectroscopy

Many people are familiar with UV/vis and FTIR spectroscopies, which are both linear absorption spectroscopies. An electric field (light), $E$, interacts with matter to produce a new electric field, the polarization, $P$:

$$P \propto P_0 + \chi^{(1)}E. \tag{2.1}$$

This equation can be expanded into higher order terms:

$$P \propto P_0 + \chi^{(1)}E + \chi^{(2)}EE + \chi^{(3)}EEE + \cdots. \tag{2.2}$$

With a strong enough electric field (fast and ultrafast pulses, in general),[1–3] these higher order susceptibilities, $\chi^{(n)}$, can be observed.

Looking at the second order term and writing the electric field as $E = E_1 \cos(\omega t)$, we see that:

$$\begin{aligned} P^{(2)} &\propto \chi^{(2)}E^2 \\ &\propto \chi^{(2)}(E_1 \cos(\omega t))^2 \\ &\propto \frac{\chi^{(2)}}{2}{E_1}^2(1 + \cos(2\omega t)). \end{aligned} \tag{2.3}$$

The second order polarization has a frequency that is double the input electric field frequency, called the second harmonic.

$P$ and $E$ are both first rank tensors, while $\chi^{(2)}$is a third rank tensor. Imagine that $\chi^{(2)}_{ijk}$ has inversion symmetry, meaning $\chi^{(2)}_{ijk} = \chi^{(2)}_{-i-j-k}$. Then, inverting the coordinates yields:

$$\begin{aligned} P^{(2)}_i &\propto \chi^{(2)}_{ijk}E_jE_k \\ P^{(2)}_{-i} &\propto \chi^{(2)}_{-i-j-k}E_{-j}E_{-k} \\ -P^{(2)}_i &\propto \chi^{(2)}_{ijk}(-E_j)(-E_k) \\ &\propto \chi^{(2)}_{ijk}E_jE_k. \end{aligned} \tag{2.4}$$

Therefore, $P^{(2)}_{-i} = -P^{(2)}_i = 0$. Since $E$ is non-zero, $\chi^{(2)}_{ijk}$ is necessarily zero for centrosymmetric environments in the dipole approximation,[4] meaning that the second order (and any even order) process is necessarily surface specific.

Since intensities are easier to measure than electric fields, the polarization equation is often rewritten with intensities, making use of the fact that $I \propto E^2$,

$$I_{2\omega} \propto \left|\chi^{(2)}\right|^2 I_\omega^2$$
$$\frac{I_{2\omega}}{I_\omega^2} \propto \left|\chi^{(2)}\right|^2, \tag{2.5}$$

where $\chi^{(2)}$ is complex in general.

One thing to note is that $\chi^{(2)}$is a macroscopic rotational average. It corresponds to the microscopic property, $\beta$, the hyperpolarizability, which is the second order analogue to the polarizability, $\alpha$.[4] The dependence on frequency is

$$\beta \propto \frac{1}{\omega_{trans} - \omega_{field} - i\Gamma}, \tag{2.6}$$

where $\omega_{trans}$ is the frequency of a transition (electronic, vibrational, etc.), $\omega_{field}$ is the frequency of $P$ or $E$, and $\Gamma^{-1}$ is the transition relaxation time. As $\omega_{field}$ approaches $\omega_{trans}$, $\beta$ becomes large. This means that tuning the input frequency or the second harmonic to a transition will enhance the SHG signal. For anions, there is a transition in the UV called a charge-transfer-to-solvent (CTTS) transition.[5] It is a broad, strong transition that even atomic anions like iodide possess. It allows us to probe ions at surfaces directly.

## 2.2 Spectroscopic Apparatus

A schematic of the experimental setup is shown in Figure 2.1 and a full description of the alignment procedure is in Appendix 1. Femtosecond pulses were generated with a MaiTai oscillator (Spectra Physics, 800 nm, 100 fs, 80 MHz), amplified with a Spitfire amplifier (Spectra Physics, ~805 nm, ~110 fs, 1 kHz), and passed through a TOPAS (Light Conversion, ~9 μJ, p-polarization) to generate 386 nm light. All work described in this document probed the thiocyanate ion, which has a CTTS transition that is resonant with the second harmonic at 193nm.[6,7] Because the CTTS transition is strong in general and thiocyanate has a high surface concentration, thiocyanate generates a strong SHG signal. A dielectric coated mirror (Edmund Optics 84-621 360-440 nm) was used to separate the 386 nm light from the output of the TOPAS. The beam was focused onto the sample with a 100 mm fused silica lens and the reflected light was collected with a 100 mm $CaF_2$ lens. The spot size was about ~100 μm in diameter. Since the fundamental beam and the SHG beam (193 nm) are collinear, a Pellin-Broca prism was used to separate the wavelengths and send the SHG signal (p-polarization) through a monochromator (Acton, SpectraPro 2150i) to a solar-blind PMT (Hamamatsu, R7154PHA). The fundamental beam is sampled and collected with a photodiode (PD) before the sample.

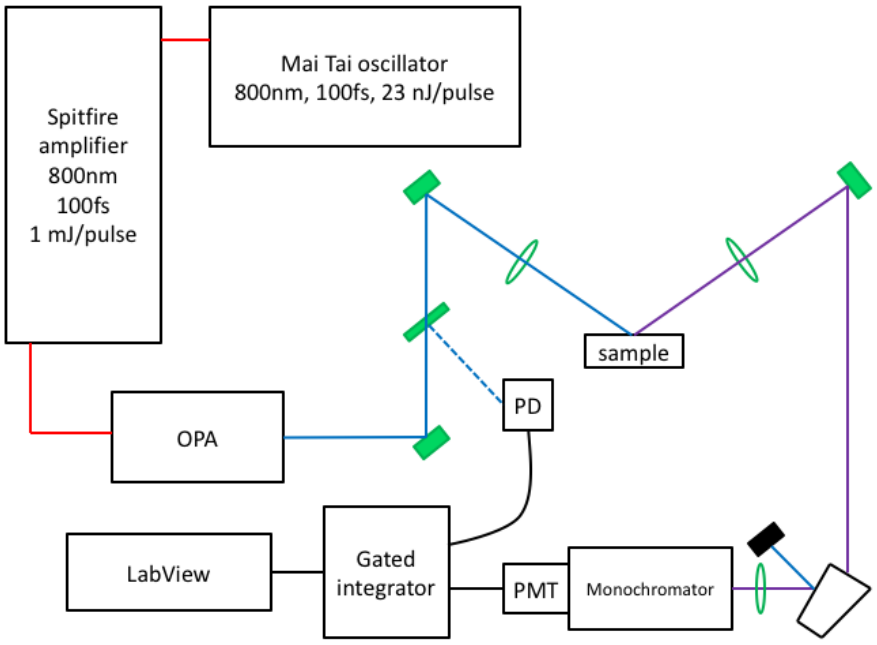



Figure 2.1: A schematic of the experimental setup. An oscillator creates 100 fs pulses at 800 nm that are amplified and converted to 386 nm light by the OPA. Before the sample, the beam is sampled and collected by a photodiode (PD) for a reference measurement. The SHG signal is reflected off the sample, separated with a Pellin Broca prism, and collected by a PMT. The currents from the PD and PMT are collected with gated integrators and LabView software.

The current from the PMT was amplified with a wide bandwidth amplifier (Hamamatsu C6438). The amplified PMT current and the current from the photodiode were processed with gated integrators (Stanford Research Systems SR250). The two channels are called Signal and Reference, respectively. The integrators were triggered by the Spitfire so that the Reference and Signal were correlated in time using the Delay setting. The integrator Width was adjusted so that the main peak was just covered by the box signal and most of the negative signal was covered by the tail of the box signal. Signal Sensitivity was kept constant across measurements. The pulses were not averaged by the integrator, meaning the Averaging Samples setting was always set to LAST. The output from the integrators was sent to a DAQ card (NI 9215 and NI 9401) for data collection in LabView 2009 SP1. The Signal Input Offset was adjusted so that the baselines for the Reference and Signal channels were near zero. The Reference baseline tended to drift downward over time, so the baseline could be set to slightly above zero. Typical settings are summarized in Table 2.1.

| **Setting** | **Reference** | **Signal** |
|---|---|---|
| Delay | 12 ns | 17 ns |
| Width | 38 ns | 61 ns |
| Sensitivity | 0.1 V/V | 0.2 V/V |

Table 2.1: Summary of typical gated integrator settings.

## 2.3 Solution Preparation

All the studies described in this document used the thiocyanate anion, $SCN^-$. The cation has little effect on SHG signal for this anion, at least for sodium and potassium.[8,9] NaSCN tends to be easier to work with than KSCN, since it has a higher melting point. A detailed description of the procedure is included in Appendix 2. Briefly, glassware is soaked in NoChromix overnight and washed the next morning with 18.2MΩ water from a Millipore filtration system (Milli-Q Advantage A10). Solutions were made with 18.2MΩ water and NaSCN (J. T. Baker, ACS reagent ≥98%) that had been baked at 200°C overnight. Data was taken the day after solutions were made to reduce noise in the measurements.

## 2.4 Sample measurement

For each solution, at least three aliquots were measured. More aliquots were measured if available, but the aliquots from the bottom of the flask tended to have more impurities. Solution aliquots were always dispensed using a sterile pipet, with liquid taken from the bulk and never the surface. Volume of the aliquots was ~18mL. Precise volumes were not necessary because the sample stage had a height adjustment. At least one aliquot of water was taken before and after each aliquot of solution. Water aliquots were dispensed directly from the Millipore system. Each aliquot was only measured once. At higher concentrations (> 1M), thiocyanate begins to generate photoproducts.[10] To mitigate this, the input intensity was attenuated (~1-9 μJ, depending on the concentration) so that no more than one SHG photon was generated per pulse, solutions were never made at concentrations above 3M, and measurements were never for longer than one minute (60,000 points). Measurement processing also assumes that there is no more than one photon per pulse. If the signal was weak enough, the measurement time could be extended to get better statistics during processing. Clearly, a perturbing level of photoproducts were not being generated in that case. The input power was varied during the measurement using a neutral density filter wheel rotating at 1Hz.

## 2.5 Measurement Processing

The data obtained from a measurement is processed in Matlab, consisting of a time series for each channel. A typical series for a water sample is shown in Figure 2.2. With the integrator settings of Table 2.1, one Signal photon corresponds to ~0.5 on the y-axis. In practice, input intensity for solutions was attenuated until the Signal was at ~0.5. The cyclic pattern seen in the Reference series comes from the filter wheel.

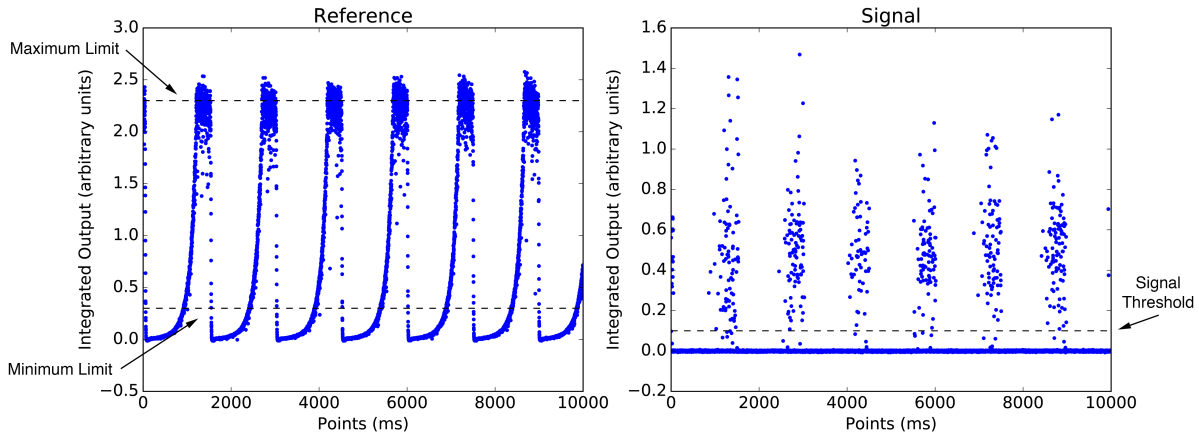



Figure 2.2: The time series data for a water sample. The x-axis is shown in points, which is equivalent to ms. The y-axis is the integrated output in arbitrary units. A neutral density filter wheel is used to modulate the Reference power at 1Hz. The dashed lines represent examples of the parameters used to process the measurements: Maximum Limit, Minimum Limit, and Signal Threshold.

To process the measurement, the Signal is binarized into "photon" and "no photon". In this way, the Signal ($I_{2\omega}$) can be treated with Poisson statistics. A threshold is set that divides the Signal data into "photon" and "no photon". The signal is binned according to the Reference value ($I_{\omega}$). Then, the average photon count per bin, $\langle k \rangle$, is[6,11]

$$I_{2\omega} \propto \langle k \rangle = -ln\left[\frac{N_{k=0}^{pulse}}{N^{pulse}}\right], \tag{2.7}$$

where $N_{k=0}^{pulse}$ is the number of "no photon" points and $N^{pulse}$ is the total number of points in the bin. If $\langle k \rangle$ is plotted vs $I_{\omega}^2$, then the slope of that line is $\left|\chi^{(2)}\right|^2$. This becomes the sample measurement.

There are several reasons why this approach is more beneficial than performing a measurement at one static intensity. One, taking an average input power and squaring it, $\langle I_{\omega} \rangle^2$, is not necessarily the same as $\langle I_{\omega}^2 \rangle$. By using an appropriate number of bins in the linear regression, this ambiguity can be avoided. Two, plotting a line allows us to verify that the interaction is indeed the second harmonic. If the Signal was coming from a stray linear process, $\langle k \rangle$ plotted vs $I_{\omega}^2$ would deviate from a straight line. Photoproduct generation also distorts the line.[10] Three, treating the Signal as binary eliminates any noise from the integration of weak signals, so long as the signal can be distinguished from the baseline. This is easily achieved by keeping the variable voltage on the PMT high.

## 2.6 Algorithm Optimization

The LabView software was optimized before the work in this dissertation was done. Previously, it calculated variance through the relation $\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$ and used this in a weighted linear regression to determine the slope. However, this expression occasionally

produced a negative variance. The relation is only valid for discrete random variables, and the points are obviously correlated in time, so the algorithm had to be corrected. At first, the online algorithm for variance was tried,[12] but this resulted in fitted lines that tended to deviate from the data, especially at low intensity. The high intensity bins are noisier in general, simply because they are sampled less, and were influencing the regression results. The algorithm was switched to using bisquare weighting, which is more robust to outliers.[13] This eliminated the need to calculate the variance.

This, unfortunately, led to the regression being more variable with respect to the choice of threshold (called Signal Threshold) as well as to the limits on the Reference values used (called Minimum Limit and Maximum Limit). These parameters are illustrated on Figure 2.2. Reference points outside the Minimum Limit and Maximum Limit are excluded. Signal above the Signal Threshold is considered a "photon" count and Signal below is considered a "no photon" count. The effects of these parameters on the slope were studied in Matlab and a procedure for finding appropriate limits was generated. For an entire 1M NaSCN dataset (4 water samples and 3 1M NaSCN samples), values were chosen for each of the parameters: 100 values for the Maximum Limit that ranged from the minimum to the maximum Reference values, 11 points for the Minimum Limit that ranged from the minimum of the Reference to 10% of the maximum of the Reference, and 11 points for the Signal Threshold that ranged from the minimum of the Signal to 10% of the maximum of the Signal. The slope was computed for all combinations of the three parameters and all points outside the 99.3% confidence interval (the definition of outliers when drawing a boxplot) were removed.

Upon examining the data sets, it was immediately apparent that the three parameters vary independently. Signal Threshold and Minimum Limit were also mostly consistent across the values of Maximum Limit. Because of this, the Signal Threshold and Minimum Limit were averaged over. Figure 2.3 shows the resulting curve of slope vs Maximum Limit. Notice that the slope approaches an asymptotic value at higher values. These observations all suggest that the parameters can be chosen independently. A procedure for doing so is to compute and display 11 values of each parameter at once while holding the other two constant. For Signal Threshold and Minimum Limit, outliers and the mean value can be identified visually. For the Maximum Limit, the asymptote can be identified visually. Matlab programs were created that display 11 values of the chosen parameter at a time while keeping the other two constant. The appropriate value for the parameter is selected by looking for the asymptotic value or the mean value among the outliers. A more detailed description of the analysis procedure, along with scripts, is given in Appendix 3. The full text of the Matlab files is included in Appendix 4.

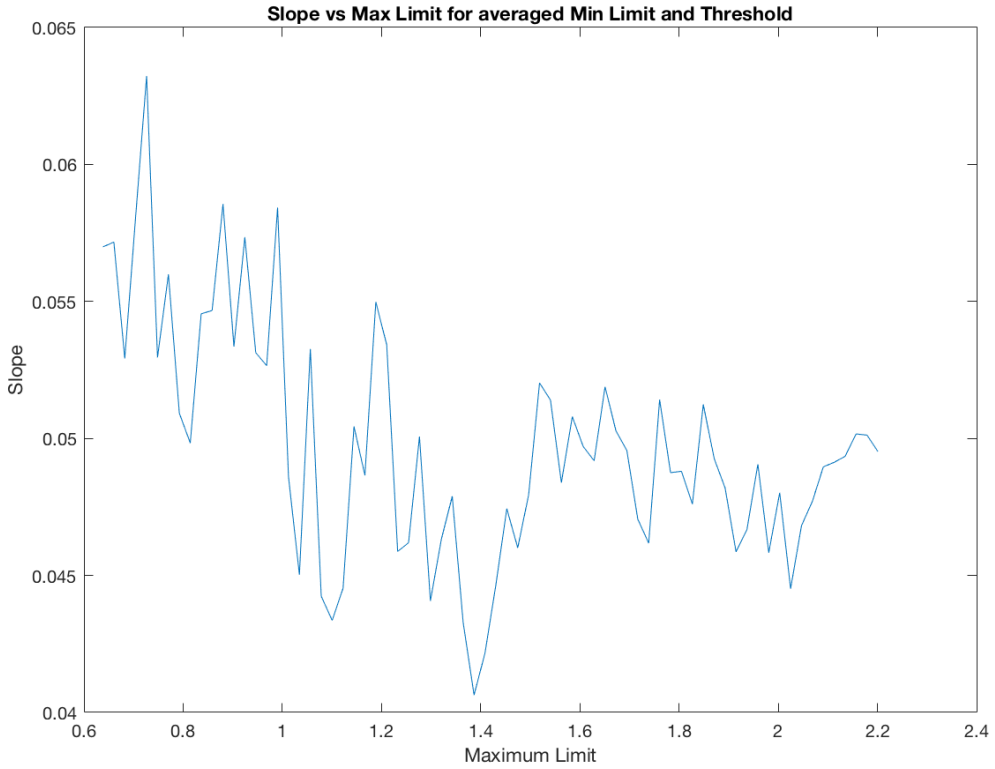



Figure 2.3. Slope vs Maximum Limit. The values of Signal Threshold and Minimum Limit have been averaged.

## 2.7 Normalization Procedure

The data set now consists of a single value (the slope, $\left|\chi^{(2)}\right|^2$) for each aliquot. The aliquots alternate between water and the solution of interest. The solution signal is normalized to the water signal by dividing the solution aliquot values by the average of the water values immediately before and immediately after the solution aliquot. For example, consider the values below:

| Sample | Slope | Normalized Value |
|---|---|---|
| Water 1 | 0.0238 | – |
| 1M NaSCN | 0.397 | 0.397 / average(0.0238, 0.0241) = 16.6 |
| Water 2 | 0.0241 | – |

Table 2.2: Example for how to normalize the sample slope values.

The final normalized value for the solution is 16.6. Different aliquots from the same flask can be considered repeated measures, so the data point for the concentration in the flask is the average of the aliquots and the error is the standard deviation.[14] Different flasks can be considered replicates, so each flask should remain a separate data point, even if two flasks have the same concentration.

An example data set is shown in Table 2.3. The individual solution samples are normalized by the water samples before and after. Then, the samples from the same flasks (indicated by letters in the name here) are averaged to produce the data point for the flask and the standard deviation is used as the error. The flasks are kept as separate data points, since they are considered replicates.

| Sample | Slope | Normalized |
|---|---|---|
| water | 0.017763 | |
| 1M NaSCN A | 0.361012 | 20.10760833 |
| water | 0.018145 | |
| 1M NaSCN A | 0.452147 | 25.17312029 |
| water | 0.017778 | |
| 1M NaSCN A | 0.302956 | 15.74983754 |
| water | 0.020693 | |
| 1M NaSCN B | 0.394102 | 19.42681093 |
| water | 0.01988 | |
| 1M NaSCN B | 0.338401 | 16.80910987 |
| water | 0.020384 | |
| 1M NaSCN B | 0.364186 | 21.61212984 |
| water | 0.013318 | |
| 1M NaSCN C | 0.378125 | 25.034759 |
| water | 0.01689 | |
| 1M NaSCN C | 0.421348 | 27.01295038 |
| water | 0.014306 | |
| 1M NaSCN C | 0.315488 | 20.57709366 |
| water | 0.016358 | |

| normalized | avg | stdev |
|---|---|---|
| 1M NaSCN A | 20.34352205 | 4.716068902 |
| 1M NaSCN B | 19.28268355 | 2.404751491 |
| 1M NaSCN C | 24.20826768 | 3.296570772 |

Table 2.3: An example of the data normalization procedure on a full data set. The top table is individual aliquot measurements and the bottom table is the averaged data points. The letters indicate that three different flasks were used, so they remain separate data points.

## 2.8 The Langmuir model

Recall the following relation from Equation (2.5):

$$\frac{I_{2\omega}}{I_{\omega}^{2}} \propto \left|\chi^{(2)}\right|^{2}. \tag{2.5}$$

In general, the effective $\chi^{(2)}$ is the sum of the $\chi^{(2)}$ of the individual species. In the case of the air/water interface, the species are water and the aqueous anion (cation signal is assumed

to be negligible since cations are not expected to approach the interface as far as the anions and they are not resonant).

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left|\chi_{water}^{(2)} + \chi_{anion}^{(2)}\right|^2 \tag{2.8}$$

On the molecular level, $\chi^{(2)} \propto N\beta^{eff}$, where $\beta^{eff}$ is the rotational average of all $\beta$. Therefore:

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left|N_{water}\beta_{water}^{eff} + N_{anion}\beta_{anion}^{eff}\right|^2. \tag{2.9}$$

Water has a non-resonant signal in the UV, so $\beta_{water}^{eff}$ has a real component only. The anion is resonant by design, so $\beta_{anion}^{eff}$ has both real and imaginary components.

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left(N_{water}\beta_{water}^{eff} + N_{anion}Re\{\beta_{anion}^{eff}\}\right)^2 + \left(N_{anion}Im\{\beta_{anion}^{eff}\}\right)^2. \tag{2.10}$$

Dividing by $N_{water}$ gives:

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left(\beta_{water}^{eff} + \frac{N_{anion}}{N_{water}}Re\{\beta_{anion}^{eff}\}\right)^2 + \left(\frac{N_{anion}}{N_{water}}Im\{\beta_{anion}^{eff}\}\right)^2. \tag{2.11}$$

Assuming that all $\beta^{eff}$ remain constant and noting that $N_{anion}/N_{water}$ is a concentration yields

$$\frac{I_{2\omega}}{I_{\omega}^2} = \left(A + B[A^-]_{surf}\right)^2 + \left(C[A^-]_{surf}\right)^2, \tag{2.12}$$

where the subscript "surf" indicates the anions at the surface.

To find an expression for $[A^-]_{surf}$, we turn to the Langmuir model of adsorption.[15] The model assumes that an anion in the bulk exchanging with a water on the surface is an equilibrium process:

$$A_{bulk}^- + W_{surf} \leftrightarrow A_{surf}^- + W_{bulk}. \tag{2.13}$$

Then the equilibrium equation is

$$K_{ads} = \frac{[A^-]_{surf}[W]_{bulk}}{[W]_{surf}[A^-]_{bulk}}. \tag{2.14}$$

Assuming that there are a maximum number of possible surface sites, $[sites]_{max}$ yields

$$K_{ads} = \frac{[A^-]_{surf}[W]_{bulk}}{([sites]_{max} - [A^-]_{surf})[A^-]_{bulk}}. \quad (2.15)$$

Rearranging gives an expression for $[A^-]_{surf}$:

$$[A^-]_{surf} = [sites]_{max} \frac{[A^-]_{bulk}}{[W]_{bulk}K_{ads}^{-1} + [A^-]_{bulk}}. \quad (2.16)$$

Substituting this expression into Equation (2.12) gives

$$\frac{I_{2\omega}}{I_{\omega}^2} = \left(A + B' \frac{[A^-]_{bulk}}{[W]_{bulk}K_{ads}^{-1} + [A^-]_{bulk}}\right)^2 + \left(C' \frac{[A^-]_{bulk}}{[W]_{bulk}K_{ads}^{-1} + [A^-]_{bulk}}\right)^2. \quad (2.17)$$

Here the max surface sites constant was absorbed by the constants $B'$ and $C'$. Changing to mole fractions for concentration and substituting the relation between $K_{ads}$ and $\Delta G_{ads}$ gives

$$\frac{I_{2\omega}}{I_{\omega}^2} = \left(A + B' \frac{X_{anion}}{(1 - X_{anion})e^{\Delta G/RT} + X_{anion}}\right)^2 + \left(C' \frac{X_{anion}}{(1 - X_{anion})e^{\Delta G/RT} + X_{anion}}\right)^2. \quad (2.18)$$

Equation (2.18) is the simple Langmuir model. It relates the bulk anion concentration to the normalized second harmonic generation (SHG) signal generated from the samples.

## 2.9 Fitting Procedure

For the oil/water data in Chapter 3, the alkane datasets were fit using Origin 6 and the toluene datasets were fit using custom Python modules and the lmfit package. For the graphene/water data in Chapter 4, the dataset was fit using custom Python modules and the lmfit package. The full text of the codes as well as usage examples are given in Appendix 5. For both software implementations, the Levenberg-Marquardt algorithm[16] was used. The X data points were bulk mole fractions of thiocyanate, the Y data points were the mean normalized SHG signal, and the Y error data points were the standard deviations of the normalized SHG signals. Replicates were treated as separate data points. The sum of squares was weighted by the standard deviation (called instrumental weighting by Origin)

$$\chi^2 = \sum_{i=1}^{n} \frac{[Y_i - f(X_i; \theta)]^2}{\sigma_i^2}, \quad (2.19)$$

where $\chi^2$ is the sum of squared errors (not the second order susceptibility), $Y_i$ is the set of Y measurements, $f(X_i; \theta)$ is the fitting equation with the parameters $\theta$, and $\sigma_i$ are the measurement standard deviations. The Levenberg-Marquardt algorithm minimizes Equation (2.19). The parameter errors were not scaled by reduced $\chi^2$. Parameters were initialized to +1 unless otherwise noted.

## 2.10 References


(1) Shen, Y. R. *The Principles of Nonlinear Optics*, Wiley classics library ed.; Wiley classics library; Wiley-Interscience: Hoboken, N.J, 2003.

(2) Shen, Y. R. *Fundamentals of Sum-Frequency Spectroscopy*; Cambridge molecular science; Cambridge University Press: Cambridge, United Kingdom; New York, NY, 2016.

(3) Boyd, R. W. *Nonlinear Optics*, 3rd ed.; Academic Press: Amsterdam; Boston, 2008.

(4) Lambert, A. G.; Davies, P. B.; Neivandt, D. J. Implementing the Theory of Sum Frequency Generation Vibrational Spectroscopy: A Tutorial Review. *Appl. Spectrosc. Rev.* **2005**, *40*, 103–145.

(5) Chen, X.; Bradforth, S. E. The Ultrafast Dynamics of Photodetachment. *Annu Rev Phys Chem* **2008**, *59* (1), 203–231.

(6) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.

(7) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.

(8) Petersen, P. B.; Saykally, R. J. Probing the Interfacial Structure of Aqueous Electrolytes with Femtosecond Second Harmonic Generation Spectroscopy. *J Phys Chem B* **2006**, *110* (29), 14060–14073.

(9) Rizzuto, A. M.; Irgen-Gioro, S.; Eftekhari-Bafrooei, A.; Saykally, R. J. Broadband Deep UV Spectra of Interfacial Aqueous Iodide. *J. Phys. Chem. Lett.* **2016**, *7* (19), 3882–3885.

(10) Otten, D. UV Second-Harmonic Studies of Concentrated Aqueous Electrolyte Interfaces. PhD. Dissertation, University of California, Berkeley, 2010.

(11) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Measurement of Bromide Ion Affinities for the Air/Water and Dodecanol/Water Interfaces at Molar Concentrations by UV Second Harmonic Generation Spectroscopy. *J Phys Chem C* **2010**, *114* (32), 13746–13751.

(12) Welford, B. P. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics* **1962**, *4* (3), 419–420.

(13) Heritier, S. *Robust Methods in Biostatistics*; J. Wiley: Chichester, U.K.; Hoboken, 2009.

(14) Motulsky, H.; Christopoulos, A. *Fitting Models to Biological Data Using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting*; Oxford University Press: Oxford ; New York, 2004.

(15) Górecki, T. Solid versus Liquid Coatings. In *Applications of solid phase microextraction*; Pawliszyn, J., Ed.; RSC chromatography monographs; Royal Society of Chemistry: Cambridge, 1999; pp 92–108.

(16) Monahan, J. F. *Numerical Methods of Statistics*, 2nd ed.; Cambridge series in statistical and probabilistic mathematics; Cambridge University Press: Cambridge ; New York, 2011.

# Chapter 3 – Thiocyanate Adsorption to the Oil/Water Interface

## 3.1 Introduction

As described in Chapter 1, the ion adsorption mechanism for the air/water interface involves a nuanced balancing of several factors (e.g. electrostatics, solvent repartitioning, capillary waves, configurational entropy) that leads to some ions being enhanced in concentration at the interface and some being expelled. It is an example of a hydrophobic interface, but it differs from other hydrophobic interfaces in one important regard: the second medium is gaseous, not condensed. One goal of studying aqueous solution/water interfaces is to understand the Hofmeister series,[1] but the series' main application – protein stabilization in solution – involves protein interfaces. To truly understand the "Hofmeister Effects," it is essential to examine how a condensed phase affects this balance.

Many studies have been done on the carbon tetrachloride/water and alkane/water interfaces.[2] They generally show that interfacial hydrogen bonding interactions become weaker, which leads to interfacial ordering. However, these studies were all homodyne SHG studies that only measured $|\chi^2|^2$, not $Im\{\chi^2\}$, so the interpretation is suspect.[3] Another study employed sum frequency scattering, along with molecular dynamics simulations, to show that the negative charge on oil droplets in water is due to asymmetric charge transfer between water molecules.[4] However, quantitative measures of interfacial ion enhancement are lacking, let alone a mechanism of adsorption.

The simplest medium to start with is a hydrocarbon. Unfortunately, it is difficult to find hydrocarbons that wet water. One possibility is hexane. Hexane exhibits two wetting transitions: one where the thickness of the wetting layer discontinuously jumps from near zero to a mesoscopic layer of hundreds of Angstroms and one where the thickness varies continuously to a macroscopically thick layer.[5] The salt concentration of the solution changed the temperature at which the transitions happened. Staying within the appropriate concentration range would allow the layer thickness to be tuned. Decane is a volatile, nonwetting alkane and hexadecane is a nonvolatile, nonwetting alkane. These alkanes can provide comparisons, since the water will only interact with the vapor layer produced by the alkanes. Another possibility is toluene. Toluene is known to spread continuously on water[6] and is able to form hydrogen bonds through its $\pi$ system.[7] Most importantly, all four candidates have static dielectric constants of ~2.[8] This keeps the dielectric scenario as similar as possible, so the only added interactions are those that stem from the condensed phase.

To parallel the air/water interface, predictions can be made for the oil/water case based on the Gibbs adsorption equation. Only one reliable paper has been published with experimentally measured interfacial tensions. Aveyard and Saleem found that LiCl, NaCl, KCl, KBr, and $Na_2SO_4$ increased the interfacial tension of the dodecane/water interface, while KI decreased it,[9] so iodide is expected to be enhanced at the interface, while chloride, bromide, and sulfate are repelled. There are a few interfacial tension measurements of the aromatic hydrocarbon/water interface, but the curves are highly nonlinear and the data included for the air/water interface disagree with the literature, making the interfacial tension measurements suspect as well.[10,11]

Theory provides a more solid prediction. Polarizable anion dielectric continuum theory predicts that ion adsorption to the oil/water interface will be greater than for the air/water interface, due to dispersion forces.[12] A study based on Schmutzer's model suggests that iodide and thiocyanate will have a greater affinity for the oil/water interface than the air/water interface.[13] Molecular dynamics simulations including ion polarizability show that fluoride is absent from, chloride and bromide have no preference for, and iodide has a strong preference for the water/decane interface.[14] The calculated interfacial tension changes in that study agree with Aveyard and Saleem,[9] supporting the reliability of the simulation results. Unfortunately, it is hard to compare the calculated surface excesses quantitatively with results for the air/water interface; the polarizable model gave air/water results that didn't come close to experimental results and the original air/water results did not calculate the same quantity.[15] Qualitatively, chloride was slightly depleted at the air/water interface, whereas it retained the bulk concentration at the water/decane interface. This suggests that the decane/water interface should be slightly more favorable. The conclusion seems to be that the oil/water interface should be more favorable than the air/water interface. Here, I describe DUV second harmonic generation (SHG) studies that aimed to quantitatively measure the affinity of thiocyanate for several hydrocarbon/water interfaces.

## 3.2 Methods and Materials

Solution preparation, the optical design, and data analysis are as described in Chapter 2. The methods detailed here are performed in addition to those described in Chapter 2.

### *3.2.1 Hydrocarbon preparation*

The hydrocarbons used were hexane (99%, Alfa Aesar), decane (99%, Alfa Aesar), hexadecane (99%, Alfa Aesar), and toluene (99.9%, Alfa Aesar). All equipment was washed in NoChromix when possible and saturated KOH in ethanol when NoChromix wasn't possible. Silica gel (Avantor) was baked for two hours at 500°C. The hydrocarbon was poured into a dark colored bottle and the baked silica poured in after until there was a ~5 mm layer of silica in the bottom. The hydrocarbon was left to purify overnight. Solutions were made so that an air/water measurement and a hydrocarbon/water measurement could be taken from the same flask.

### *3.2.2 Forming the hydrocarbon layer*

Pipets with sterile tips were used to dispense the hydrocarbons. For the alkanes, 1-2 μL were used, dispensed from an air cushion pipet (so the exact volume is not known). For the first toluene dataset, 10 μL were used, dispensed from a positive displacement pipet. For the second toluene dataset, 75 μL were used, except for 50 μL used with the 0.01 M samples, all dispensed from a positive displacement pipet. The hydrocarbon was dropped onto the water surface and left for a minute to form a surface layer. Then the measurement was taken for 1-3 minutes, depending on the amount of signal from the sample. For the alkane datasets, an alkane lens would often wander under the beam focus and distort the measurement. When this happened, the sample dish was jostled to move the lens away and the measurement restarted. For the second toluene dataset, the initial signal was weak. The measurement would continue

for ~1 minute, then the signal would jump. The measurement was immediately stopped, saved, and a new measurement was started. Once the second measurement was done, the sample stage height would be checked. If the height was not optimal, a third measurement would be taken. The last measurement was used in the data analysis.

### *3.2.3 Langmuir model*

Since a third species was added to the sample, this must be accounted for in the Langmuir model:

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left|N_{water}\beta_{water}^{eff} + N_{oil}\beta_{oil}^{eff} + N_{anion}\beta_{anion}^{eff}\right|^2. \tag{3.1}$$

The UV/vis spectrum of toluene was obtained and the extinction coefficient at 193 nm was calculated to be ~2 $M^{-1}$ $cm^{-1}$. Compare this to thiocyanate, which has an extinction coefficient of 3.5 x $10^3$ $M^{-1}$ $cm^{-1}$ at ~222 nm;[16] the coefficient at 193 nm is even greater than that.[17] While toluene is also resonant, the signal is much weaker than that from thiocyanate and can be neglected. Factoring the real and imaginary components and dividing by $N_{water}$ gives

$$\frac{I_{2\omega}}{I_{\omega}^2} \propto \left(\beta_{water}^{eff} + \frac{N_{oil}}{N_{water}}\beta_{oil}^{eff} + \frac{N_{anion}}{N_{water}}Re\{\beta_{anion}^{eff}\}\right)^2 + \left(\frac{N_{anion}}{N_{water}}Im\{\beta_{anion}^{eff}\}\right)^2. \tag{3.2}$$

Assuming that the oil terms remain constant, they can be absorbed into the constant 'A':

$$\begin{aligned}\frac{I_{2\omega}}{I_{\omega}^2} &= \left(A + B[A^-]_{surf}\right)^2 + \left(C[A^-]_{surf}\right)^2 \\ &= \left(A + B'\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2 + \left(C'\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2.\end{aligned} \tag{3.3}$$

The functional form of Equation (3.3) ends up being the same as Equation (2.18).

For the toluene/water datasets, $\Delta G_{ads}^{air}$ and $\Delta G_{ads}^{alk}$ were initialized to -7000 J/mol to avoid numerical overflows in the fitting calculations. All other parameters were initialized to +1.

## 3.3 Results

The results for the alkane datasets are summarized in Figure 3.1 and Table 3.1. All three alkane datasets had corresponding air/water curves, but for clarity, only the air/water curve for the hexadecane dataset is shown in Figure 3.1. Table 3.1 includes the full fit results for all datasets and curves. For the hexadecane dataset, $\Delta G_{ads}^{air}$ = -4 ± 1 kJ/mol and $\Delta G_{ads}^{alk}$ = -5 ± 1 kJ/mol, where the superscripts 'air' and 'alk' indicate the air/water and alkane/water interfaces, respectively. For hexane and decane, only four concentrations each were collected for preliminary analysis. This means that there were not enough degrees of freedom to calculate errors for the parameters. Assuming that the errors are the same order of magnitude as for the hexadecane interface, $\Delta G_{ads}^{air}$ = -3 kJ/mol and $\Delta G_{ads}^{alk}$ = -3 kJ/mol for hexane and $\Delta G_{ads}^{air}$ = -3 kJ/mol and $\Delta G_{ads}^{alk}$ = -3 kJ/mol for decane.

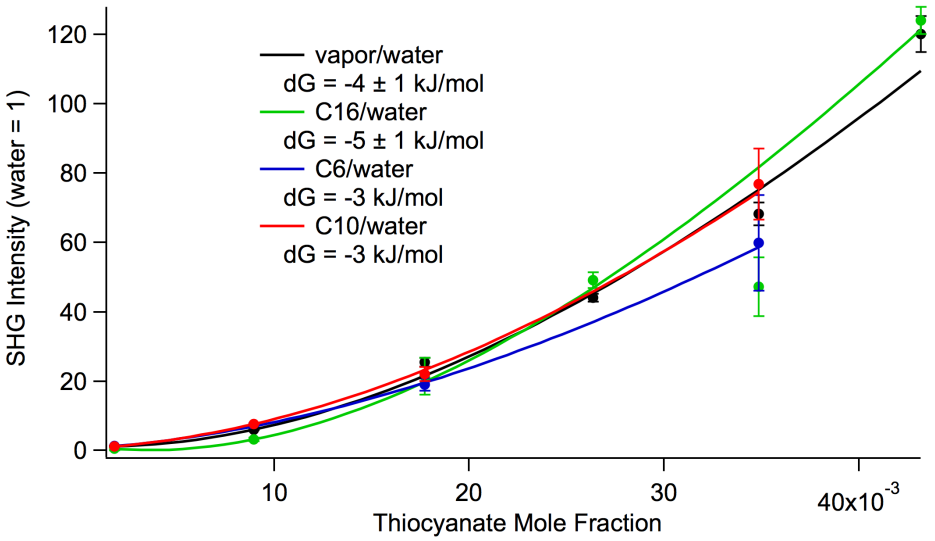



Figure 3.1: SHG signal from three alkane/water interfaces: hexadecane, hexane, and decane. All three alkane datasets have their own air/water curves, but only the curve for hexadecane is shown for clarity. The extracted free energies are presented below the corresponding curve in the legend. The hexane and decane datasets only have four concentrations each, so errors could not be calculated.

| | | Air/Water | | Alkane/Water | |
|---|---|---|---|---|---|
| **Alkane** | **Parameter** | **Value** | **Error** | **Value** | **Error** |
| Hexadecane | A | 1.17344 | 0.11083 | -1.30368 | 0.29727 |
| | B | -16.35043 | 3.94454 | 51.13381 | 15.2272 |
| | C | 53.27203 | 23.16704 | 18.11317 | 22.06878 |
| | $\Delta G$ (J/mol) | -4060.48083 | 1182.47439 | -4543.26861 | 1166.10139 |
| Hexane | A | 0.99938 | - | 0.77916 | - |
| | B | 8.47994 | - | 56.81505 | - |
| | C | 86.48984 | - | 13.32012 | - |
| | $\Delta G$ (J/mol) | -2649.693 | - | -3197.29116 | - |
| Decane | A | 0.71237 | - | 0.72714 | - |
| | B | 76.29785 | - | 61.37336 | - |
| | C | -0.00226 | - | 51.46552 | - |
| | $\Delta G$ (J/mol) | -2927.73253 | - | -2757.1415 | - |

Table 3.1: The full fit results. For each alkane, an air/water curve was collected at the same time from the same flasks. The hexane and decane datasets only have four concentrations each, so errors could not be calculated.

The first toluene dataset, using 10 μL of toluene, is summarized in Figure 3.2 and the full fit results are included in Table 3.2. The fit returned reasonable results only when the parameters 'B' and 'C' were shared. This implicitly assumes that the number of surface sites and $\beta_{anion}^{eff}$ remain constant when adding toluene. It is not an ideal assumption, but it seems reasonable given the difficulty of fitting. The free energies are $\Delta G_{ads}^{air}$ = -7.5 ± 0.7 kJ/mol and $\Delta G_{ads}^{tol}$ = -7.3 ± 0.7 kJ/mol, where the superscripts 'air' and 'tol' indicate the air/water and toluene/water interfaces, respectively.

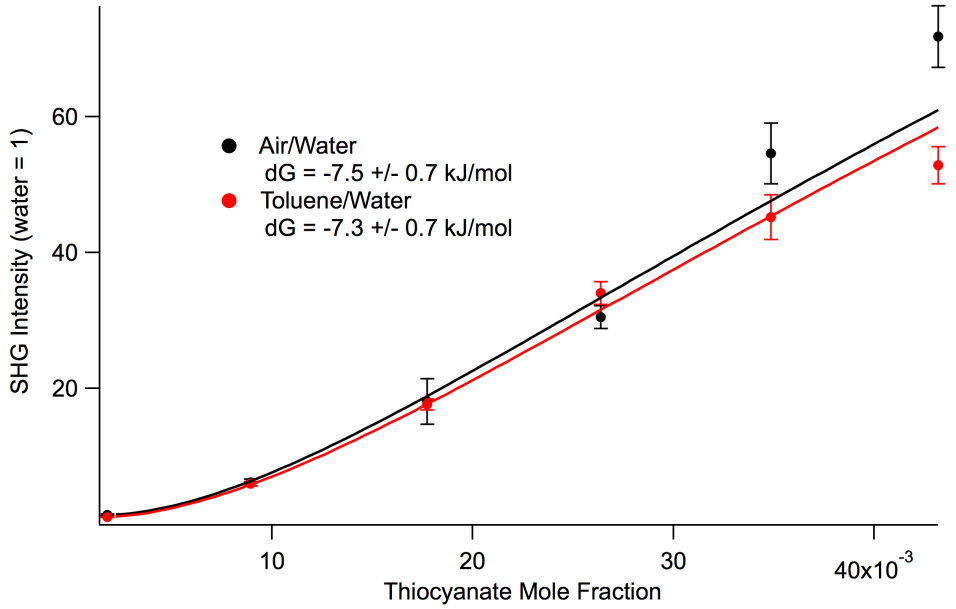



Figure 3.2: SHG signal from the toluene/water interface, where the amount of toluene used was 10 µL. The corresponding air/water curve is also presented. The extracted free energies are presented below the corresponding curve in the legend.

| Parameter | Value | Error |
|---|---|---|
| $A^{air}$ | 1.238896689 | 0.20041642 |
| $A^{tol}$ | 1.101604796 | 0.188590553 |
| B | -5.959226902 | 4.06205978 |
| C | 15.50613721 | 3.371643733 |
| $\Delta G^{air}$ (J/mol) | -7465.595017 | 691.6049937 |
| $\Delta G^{tol}$ (J/mol) | -7349.052002 | 654.8320025 |

Table 3.2: The full fit results. The super scripts 'air' and 'tol' indicate the air/water and toluene/water interfaces, respectively. The parameters 'B' and 'C' were shared between the datasets.

For the second toluene dataset, using 75 µL of toluene, the points are inconsistent above 0.020 mole fractions. These points correspond to 1.5M, 2M, 2.5M, and 3M. The 1.5M and 2.5M points seem to differ systematically from the 2M and 3M points. This is interesting, because the 2M and 2.5M points were collected on the same day, but the 1M and 3M points were on different days. Because it is difficult to tell which points have the systematic error, two fits were used: one with the full dataset and one without the 3M point. For the full dataset, the most reasonable fit had all parameters unshared. For the truncated dataset, the most reasonable fit had parameters 'B' and 'C' shared. Figure 3.3 shows the resulting curves. The air/water curve was also fit both ways, but the lines ended up overlapping completely, so only one is shown on the graph (black). For toluene, the full dataset fit is the solid red line and the truncated dataset fit is the dashed red line. The resulting free energies are $\Delta G_{ads}^{air}$ = -4.9 ± 0.4 kJ/mol and $\Delta G_{ads}^{tol}$ = -8.4 ± 0.4 kJ/mol for the full dataset and $\Delta G_{ads}^{air}$ = -4.5 ± 0.4 kJ/mol and $\Delta G_{ads}^{tol}$ = -4.2 ± 0.4 kJ/mol for the truncated dataset. The full fit results are shown in Table 3.3 and Table 3.4.

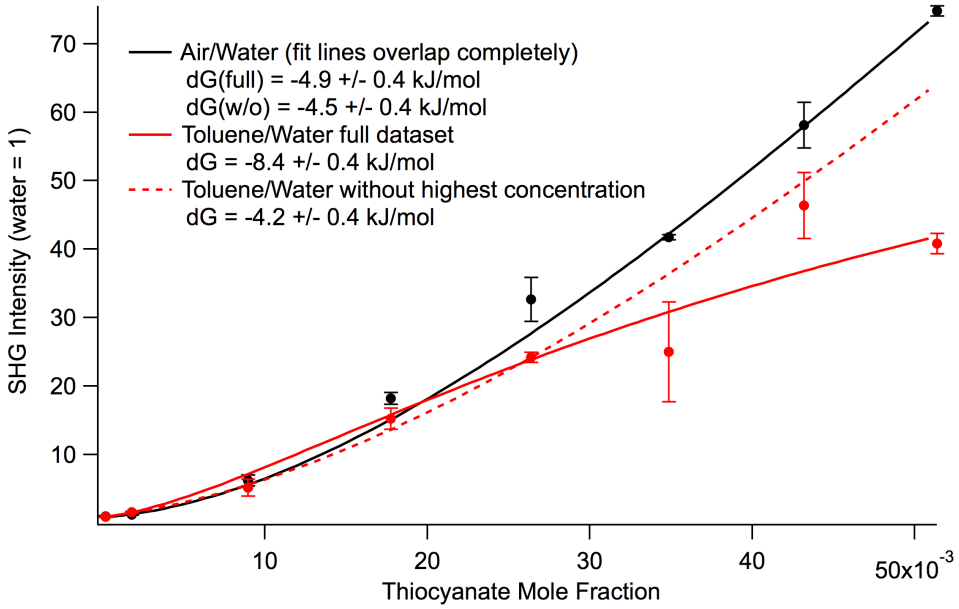



Figure 3.3: SHG signal from the toluene/water interface, where the amount of toluene used was 75 µL. The corresponding air/water curve is also presented. Two fits were performed: one with the full dataset and one with the highest concentration excluded. The extracted free energies are presented below the corresponding curve in the legend. The two air/water fits overlapped, so only one is shown (black), but both free energies are included in the legend.

| Parameter | Value | Error |
|---|---|---|
| $A^{air}$ | 0.908662481 | 0.056802157 |
| $A^{tol}$ | 0.936134364 | 0.03421958 |
| $B^{air}$ | 13.70003337 | 7.872519347 |
| $B^{tol}$ | 4.389247243 | 2.553745078 |
| $C^{air}$ | 24.47898327 | 2.954098149 |
| $C^{tol}$ | 8.360635559 | 1.246757848 |
| $\Delta G^{air}$ (J/mol) | -4922.492677 | 362.3396161 |
| $\Delta G^{tol}$ (J/mol) | -8435.72712 | 437.2161104 |

Table 3.3: The full fit results for the full dataset. The super scripts 'air' and 'tol' indicate the air/water and toluene/water interfaces, respectively. No parameters were shared.

| Parameter | Value | Error |
|---|---|---|
| $A^{air}$ | 0.836296986 | 0.04801207 |
| $A^{tol}$ | 0.948339088 | 0.028501011 |
| B | 26.73911439 | 9.417877303 |
| C | 14.5379066 | 12.8133451 |
| $\Delta G^{air}$ (J/mol) | -4539.115368 | 374.242722 |
| $\Delta G^{tol}$ (J/mol) | -4233.660634 | 387.6065922 |

Table 3.4: The full fit results for the dataset with the highest concentration excluded. The super scripts 'air' and 'tol' indicate the air/water and toluene/water interfaces, respectively. The parameters 'B' and 'C' were shared between the datasets.

## 3.4 Discussion

For reference, some previously published values for $\Delta G_{ads}$ are included in Table 3.5. All of the alkane datasets have less favorable $\Delta G_{ads}^{air}$ than the previous results, which suggests that the magnitudes of all values are probably erroneous. However, since the air/water curves and alkane/water curves are from the same solutions, the two curves can still be compared to each other. Keeping in mind that the hexane and decane datasets have no errors, it appears that the free energies differ very little between the air/water interface and the alkane/water interfaces. Since decane and hexadecane are known not to wet water, it is likely that hexane was not wetting either, perhaps evaporating entirely. Most ellipsometric and x-ray reflectivity studies use closed cells as their sample chambers and were equilibrated over days.[5,18–20]

| Dataset | $\Delta G_{ads}$ (kJ/mol) |
|---|---|
| NaSCN air/water[17] | -7.53 ± 0.13 |
| KSCN air/water[21] | -6.78 ± 0.03 |
| NaSCN dodecanol/water[22] | -6.7 ± 1.1 |
| KSCN dodecanol/water[22] | -6.3 ± 1.8 |

Table 3.5: Free energy values from previously published results for comparison.

Due to the effort that would be required to continue with the alkane interfaces, I moved on to the toluene/water interface. The first toluene dataset, at least, was consistent with previous thiocyanate datasets.[17,21] The fact that the free energies of the air/water and toluene/water interfaces again differ only slightly is also consistent with results from the dodecanol/water interface.[22] Interpretation of the second dataset is less straightforward. It does not agree quantitatively with previous results, but the curves can still be compared to each other. The full dataset implies that the toluene/water interface is more favorable, in line with predictions.[9,12–14] The truncated dataset implies that the free energies are the same, in line with the first toluene dataset and the dodecanol/water data.

This begs the question: "Is toluene actually present at the interface?" Figure 3.2 and Figure 3.3 show that the SHG signal is affected by the presence of toluene – more so than for the alkanes – so it is likely that the interface does change when the toluene is added. However, there is reason to believe that microliters of toluene would dissolve into solution, especially at high concentration.[23] With the first dataset, the lack of alignment trouble likely indicates that no toluene phase was formed. With the second dataset, given the initial lack of signal and the jump in signal after exposure to the laser beam, a toluene phase likely formed and was evaporated. Whatever toluene was left behaves like the toluene in the first dataset. It is unclear what this behavior is, however.

## 3.5 Conclusions and Future Directions

To continue with the alkane interfaces, a new sample chamber will have to be constructed. It could be a closed sample cell that allows droplets to equilibrate[5,20] or a cell in which alkane vapors are allowed to adsorb on the surface.[18,19] However, these types of cells require long equilibration times. Another possibility is a cell that avoids a meniscus through a thicker layer and appropriately placed windows.[24]

The data presented here for toluene are preliminary and could certainly use improvement. First, the volume of toluene used should be optimized to get a consistent surface. Second, the high concentration regime needs to be redone with greater precision. Third, the assumptions made when sharing parameters 'B' and 'C' are likely not particularly accurate. However, it also does not seem that the two parameters should differ drastically. The lmfit module in Python allows for parameters to have bounds, including conditions like "$B^{air} = B^{tol}$ + const." New models could be created that incorporate such bounds.

Even after the data are improved, however, the behavior of the toluene/water system is still unknown. Is it forming a microscopic layer on the solution surface? Is it dissolving partially? Is it dissolving completely? If it dissolves, is the toluene diffusing away from the interface? If the toluene is not diffusing away, does it make a thermodynamic difference if the toluene is dissolved or not? Molecular dynamics simulations can be used to elucidate some of these issues. Once these questions are answered, an interpretation can be made more confidently.

## 3.6 References


(1) Jungwirth, P.; Cremer, P. S. Beyond Hofmeister. *Nat. Chem.* **2014**, *6* (4), 261–263.

(2) McFearin, C. L.; Beaman, D. K.; Moore, F. G.; Richmond, G. L. From Franklin to Today: Toward a Molecular Level Understanding of Bonding and Adsorption at the Oil−Water Interface †. *J. Phys. Chem. C* **2009**, *113* (4), 1171–1188.

(3) Shen, Y. R. Phase-Sensitive Sum-Frequency Spectroscopy. *Annu. Rev. Phys. Chem.* **2013**, *64* (1), 129–150.

(4) Vácha, R.; Rick, S. W.; Jungwirth, P.; de Beer, A. G. F.; de Aguiar, H. B.; Samson, J.-S.; Roke, S. The Orientation and Charge of Water at the Hydrophobic Oil Droplet–Water Interface. *J. Am. Chem. Soc.* **2011**, null.

(5) Shahidzadeh, N.; Bonn, D.; Ragil, K.; Broseta, D.; Meunier, J. Sequence of Two Wetting Transitions Induced by Tuning the Hamaker Constant. *Phys. Rev. Lett.* **1998**, *80* (18), 3992.

(6) Kunieda, M.; Liang, Y.; Fukunaka, Y.; Matsuoka, T.; Takamura, K.; Loahardjo, N.; Winoto, W.; Morrow, N. R. Spreading of Multi-Component Oils on Water. *Energy Fuels* **2012**, *26* (5), 2736–2741.

(7) Gierszal, K. P.; Davis, J. G.; Hands, M. D.; Wilcox, D. S.; Slipchenko, L. V.; Ben-Amotz, D. π-Hydrogen Bonding in Liquid Water. *J. Phys. Chem. Lett.* **2011**, *2* (22), 2930–2933.

(8) Haynes, W. M.; Lide, D. R.; Bruno, T. J. *CRC Handbook of Chemistry and Physics: A Ready-Reference Book of Chemical and Physical Data.*, 97th ed.; CRC Press/Taylor & Francis, 2016.

(9) Aveyard, R.; Saleem, S. M. Interfacial Tensions at Alkane-Aqueous Electrolyte Interfaces. *J. Chem. Soc. Faraday Trans. 1 Phys. Chem. Condens. Phases* **1976**, *72* (0), 1609–1617.

(10) Singh, M.; Matsuoka, H. Effect of Ionic Sizes of Halide Anions of Potassium Salts on Surface and Interfacial Tensions of Benzene and Water Interfaces for Mutual Mixing. *Surf. Rev. Lett.* **2009**, *16* (5), 743–747.

(11) Singh, M. Effect of Potassium Halide Salts on Mutual Solubility of Water+aromatic Hydrocarbons Liquid–liquid Interface Studied with Surface and Interfacial Tensions. *J. Mol. Liq.* **2014**, *200*, 289–297.

(12) Levin, Y.; Santos, A. P. dos. Ions at Hydrophobic Interfaces. *J. Phys. Condens. Matter* **2014**, *26* (20), 203101.
(13) Slavchov, R. I.; Peshkova, T. V. Adsorption of Ions at the Interface Oil|aqueous Electrolyte and at Interfaces with Adsorbed Alcohol. *J. Colloid Interface Sci.* **2014**, *428*, 257–266.
(14) Vazdar, M.; Pluhařová, E.; Mason, P. E.; Vácha, R.; Jungwirth, P. Ions at Hydrophobic Aqueous Interfaces: Molecular Dynamics with Effective Polarization. *J. Phys. Chem. Lett.* **2012**, *3* (15), 2087–2091.
(15) Jungwirth, P.; Tobias, D. J. Molecular Structure of Salt Solutions: A New View of the Interface with Implications for Heterogeneous Atmospheric Chemistry. *J. Phys. Chem. B* **2001**, *105* (43), 10468–10472.
(16) Blandamer, M. J.; Fox, M. F. Theory and Applications of Charge-Transfer-to-Solvent Spectra. *Chem Rev* **1970**, *70* (1), 59–93.
(17) Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. Enhanced Concentration of Polarizable Anions at the Liquid Water Surface: SHG Spectroscopy and MD Simulations of Sodium Thiocyanide. *J Phys Chem B* **2005**, *109* (21), 10915–10921.
(18) Kwon, O.-S.; Jing, H.; Shin, K.; Wang, X.; Satija, S. K. Formation of *N* -Alkane Layers at the Vapor/Water Interface. *Langmuir* **2007**, *23* (24), 12249–12253.
(19) Pfohl, T.; Möhwald, H.; Riegler, H. Ellipsometric Study of the Wetting of Air/Water Interfaces with Hexane, Heptane, and Octane from Saturated Alkane Vapors. *Langmuir* **1998**, *14* (18), 5285–5291.
(20) Bertrand, E.; Dobbs, H.; Broseta, D.; Indekeu, J.; Bonn, D.; Meunier, J. First-Order and Critical Wetting of Alkanes on Water. *Phys. Rev. Lett.* **2000**, *85* (6), 1282.
(21) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.
(22) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.
(23) Pasciak, J.; Zjawiony, I. Hydrotropic properties of ammonium, potassium, and sodium thiocyanate. *Rocz. Chem.* **1970**, *44* (1), 229–234.
(24) Mitrinović, D. M.; Tikhonov, A. M.; Li, M.; Huang, Z.; Schlossman, M. L. Noncapillary-Wave Structure at the Water-Alkane Interface. *Phys. Rev. Lett.* **2000**, *85* (3), 582.

# Chapter 4 – Thiocyanate Adsorption to the Graphene/Water Interface

## 4.1 Introduction

The work described in this chapter was also motivated by the debate over the behavior of ions at interfaces of water with materials of differing static dielectric constant. While Chapter 3 described several hydrocarbon/water interfaces with similar dielectric conditions to the air/water interface, it is also interesting to consider what happens when the scenario is reversed, i.e. when $\epsilon_2 > \epsilon_{water} = 80_{293K}$. The first such material that comes to mind is a metal. In this case, it is more instructive to think about the boundary conditions: the potential is zero at the interface and far from the charge. This is the same as having a second charge with opposite sign,[1] leading to image charge attraction. Put in terms of Equation (1.3), this means $\epsilon_2 \to \infty$. However, metals are opaque to light, so a laser pulse cannot propagate through the metal. The pulse could propagate through the water, but this poses the same alignment and signal challenges that plagued the oil/water experiments.

Another potential experiment involves the deliquescence of salt crystals in humid environments. The Salmeron group performed an experiment with salt crystals on $SiO_2$ exposed to varying amounts of humidity.[2] They used scanning polarization force microscopy (SPFM) to show that all salt crystals were dissolved at 95% humidity. The technique can also measure surface potentials. The halide surface potentials in their experiment roughly followed the Hofmeister series. This is indicative of specific ion effects. This approach can be adapted for the SHG experiment by drying solutions on pieces of metal to deposit salt crystals and placing the pieces in water (but not covering them!) to generate a humid environment. Brief preliminary tests indicated that this is possible, but there are a number of difficulties in the interpretation. One, the bulk concentration of the deliquesced solution would be difficult to determine. Two, the resulting solution would be on the order of nanometers thick. It is unclear if this would be thick enough to generate a bulk region at all.

Another approach is to employ a thin, conductive layer that can be floated on top of the solution. One material that fits the bill is graphene. It is atomically thin and only absorbs 2.3% of the light per layer,[3] allowing laser pulses to propagate through it and still generate signal. An interesting material in its own right,[4] graphene has exciting potential applications which involve interfaces and ion adsorption, such as solution-gated field effect transistors for sensing,[5] porous membranes for filtering[6] and desalination,[7] supercapacitors,[8] and lithium-ion batteries.[9] In the context of specific ion effects, two properties of graphene are particularly relevant, namely high in-plane carrier mobility and hydrophobicity. The high carrier mobility makes graphene metal-like, ostensibly engendering image charge *attraction* of ions, whereas hydrophobicity makes it a condensed phase analog to the air/water interface, engendering image charge ion *repulsion*. Simulations show that water tends to be more disordered near the graphene/water interface,[10] which could affect the entropic contributions. Geiger *et al.* have studied the adsorption of ions and molecules to graphene deposited on a silica substrate[11–13] and found that graphene does not significantly alter the free energy of adsorption to the silica/water interface. However, it is unclear how the graphene itself affects the interfacial water structure in this situation.[14] Here we describe a study of surface ion adsorption by DUV-SHG spectroscopy and molecular

dynamics simulations addressing graphene suspended on the aqueous solution surface in order to explore these issues and compare properties of the resulting interface with those of air/water.

## 4.2 Methods and Materials

Solution preparation, the optical design, and data analysis are as described in Chapter 2. The methods detailed here are performed in addition to those described in Chapter 2. Experimental work was performed in collaboration with Son Nguyen under Paul Alivisatos and Horst Weller.

### 4.2.1 Graphene Preparation

All glassware was soaked in NoChromix overnight and rinsed vigorously with 18.2MΩ water. Commercial CVD graphene on copper foils (3-5 layers, one sided) was purchased from ACS Material. The foil was cut into 7x25 mm pieces, then submerged in warm acetone five times to clean up any contamination. This stock graphene was stored in a closed box to avoid new contamination. The stock graphene on copper foil was cut into 7x7 mm pieces and floated on a 30 mL aqueous solution of 10% $Na_2S_2O_8$ for ~5 hours to etch away the copper. Polyethylene O-rings (cleaned with saturated KOH in ethanol and stored in water) with inner and outer diameters of ~1.2 and ~2.5, respectively, were placed around the pieces (Figure 4.1). The O-rings stabilize the water surface so that the graphene pieces are less likely to break during transfers and are more likely to align the laser beam properly. After etching, 200 mL of 18.2MΩ water was added to dilute the etching solution, then the floating graphene sheet and the surrounding O-ring was scooped into a small cup and transferred into 250 mL water. The solution was stirred slowly to ensure the desorption of any etching product from the graphene and to equilibrate the concentration. The graphene samples were transferred into fresh water three more times and left to float overnight. The next day, the graphene samples were transferred once more into 350 mL water for the last cleaning step.

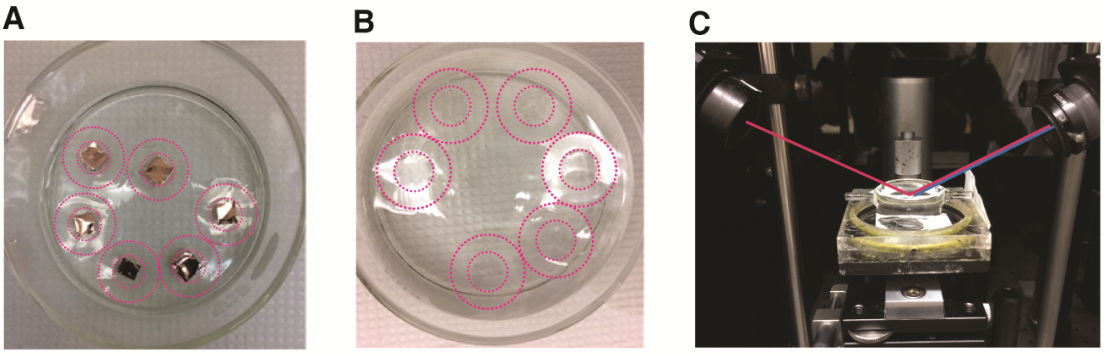



Figure 4.1: Preparing samples of graphene floating in sodium thiocyanate solution for SHG experiment. **A)** CVD 3-5 layer graphene on copper foils. The polyethylene o-rings are outlined in red for clarity. **B)** 3-5 layer graphene samples confined inside the o-rings after cleaning. **C)** Each graphene sample with its o-ring is scooped in a small cup for SHG measurement. The red and blue lines represent the fundamental and SHG beam paths, respectively. Beam colors are for clarity and do not correspond to the experimental wavelengths.

Solutions were prepared volumetrically the day before, using 18.2MΩ water and NaSCN (J. T. Baker, ACS reagent ≥98%) that had been baked at 200°C overnight. Solutions were poured into a large Petri dish and graphene samples were transferred into the dish with a glass scoop. The volume of solution in the scoop was accounted for when calculating bulk concentrations. Once all samples of graphene were in the dish, the solution was stirred to ensure a uniform bulk concentration. The samples were then removed one at a time to be measured in the SHG experiment and replaced in the same concentration solution (Figure 4.1). After all samples were measured, any samples to be reused were transferred to the new concentration.

### 4.2.2 Optical design

The laser energy was attenuated to < 1 μJ for graphene samples to prevent damage[15] and to ensure that no more than one photon was generated per pulse. Figure 4.2A depicts the experiment, where CVD graphene (3-5 layers) is suspended on top of solutions of NaSCN. The input laser pulses (100fs, 386nm) incident on the surface of generate 193nm second harmonic radiation, which is resonant with the charge-transfer-to-solvent (CTTS) transition of thiocyanate[16,17]. Figure 4.2B diagrams the interfacial structure.

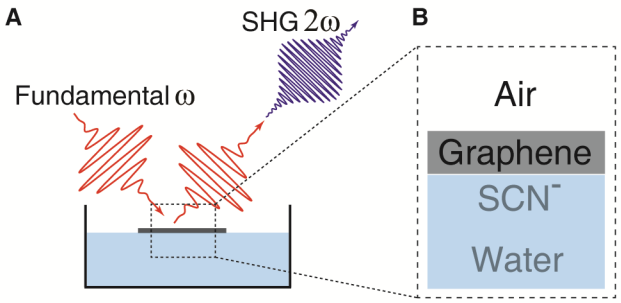



Figure 4.2: (**A**) The experimental design. Fundamental (386 nm) pulses are reflected from the graphene/water surface and SHG (193 nm) pulses are generated. The collected signal is proportional to the number of thiocyanate ions at the surface. (**B**) Structure of the interface studied in **A**.

### 4.2.3 Raman characterization of graphene

Raman spectra were acquired under ambient conditions with a WiTech alpha300R+ confocal Raman microscope equipped with a 488 nm excitation laser and a 600 lines/mm grating spectrograph operating in 180° backscattering geometry. A Zeiss 50x or 20x objective was used to focus the excitation laser light spot of on the samples. To ensure quality, all purchased foils were reexamined under a Raman microscope (Figure 4.3A, B, C). The average intensity ratios of the 2D over the G band indicate the graphene has 3 to 5 layers on average[18,19]. The graphene was also examined after etching and cleaning, but before an SHG measurement. Raman spectra show that the floating graphene is of the same quality as it was on copper (Figure 4.3D, E, F). After all SHG measurements were finished, the graphene samples were transferred first to clean water, then to a glass slide to inspect under a Raman microscope. Figure 4.3G, H, K show similar Raman spectra to the original graphene on copper (Figure 4.3A, B, C), indicating that there was no damage of graphene during the SHG experiment.

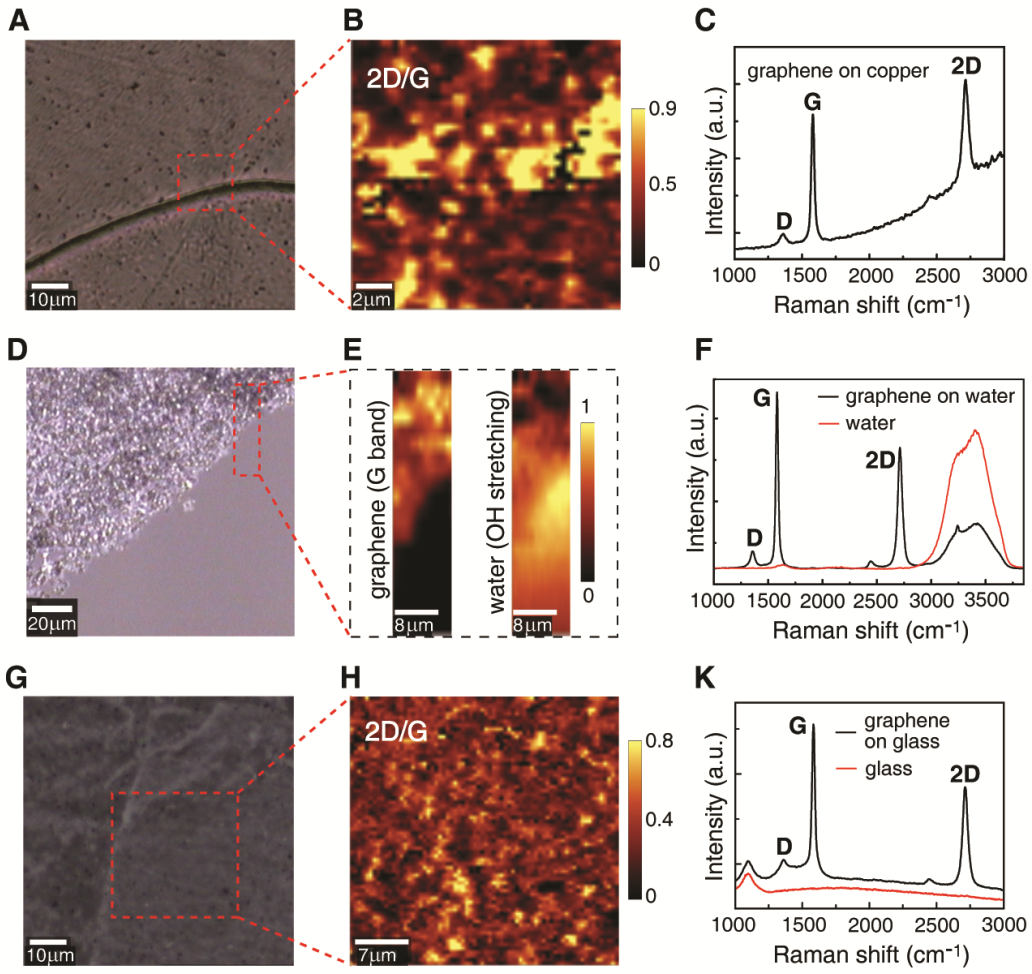



Figure 4.3: Raman characterization of the graphene samples used in the experiment before etching, after etching, and after exposure to the laser beam. **A**) Optical image of the graphene on Cu with Cu grain boundary (dark line). **B**) Spatial map of 2D/G intensity (peak height) ratios over the area marked in **A**. **C**) Average spectrum of area marked in **A** (without background subtraction). **D**) Optical image of the graphene on clean water after Cu etching. **E**) Spatial map of Raman spectra of graphene and water in the area marked in **D**. **F**) Average spectra of area with and without graphene covered on water. **G**) Optical image of graphene on glass after SHG measurement. **H**) Spatial map of 2D/G intensity ratios over the area marked in **G**. **K**) Average spectrum of area marked in **G**, and spectrum of glass substrate.

### 4.2.4 Molecular Dynamics simulation details

Simulations were performed in collaboration with Stephen Cox under Phillip Geissler. To calculate the potential of mean force (PMF), we used umbrella sampling. The system consisted of 264 SPC/E water molecules[20] placed above a 2.13×1.97 $nm^2$ graphene sheet consisting of 160 carbon atoms. Initial simulations of a larger system with 1151 water molecules above a 2.55x2.46 $nm^2$ graphene sheet with no vapor phase found only a small effect on the PMF (the adsorption free energy of a single ion was more favorable in the small system by only 0.3 $k_BT$). We therefore opted to use the smaller system size of 264 water molecules, as this permits the calculation of the energy and entropy profiles with reasonable computational resources. The plane of the graphene sheet was taken to be the $xy$-plane, with the normal direction taken to be $z$. Periodic boundary conditions, commensurate with the graphene sheet, were used with

the length of the $z$-direction set to 4.5 nm. The simulation setup could thus be described as a thin slab of liquid water (approx. 2 nm thick), with one graphene/water interface and one air/water interface. An iodide ion with charge $q_I = -0.8e$, where $e$ is the elementary unit of charge, was restrained at different heights $z_0$ above the graphene sheet with a harmonic bias potential:

$$U_{bias}(z) = \frac{k_{bias}}{2}(z - z_0)^2. \tag{4.1}$$

Here, $z$ is the instantaneous height of the iodide above the graphene sheet. A total of 23 'windows' with $z_0$ = 0.3, 0.4, …, 2.5 nm were used, with $k_{bias}$ = 836.8 kJ/mol/nm$^2$. Dynamics were propagated at a temperature of 298 K using Langevin dynamics[21,22] as implemented in the `LAMMPS` simulation package[23] (available at http://lammps.sandia.gov), with a time step of 1.0 fs and a damping constant of 1 ps. For each window, a simulation of 7 ns was performed. To reconstruct the PMF, the multistate Bennett acceptance ratio[24] (`MBAR`) method was used. At ambient conditions, it is reasonable to ignore contributions due to pressure-volume work, and contributions from kinetic energy are independent of $z$. We therefore equate the changes in enthalpy to the changes in potential energy. Potential energy profiles were measured directly from the umbrella sampling simulations by binning the samples according to the $z$-coordinate of the ion, with a bin width 0.1 nm. The autocorrelation time of the potential energy in each window was used to construct uncorrelated data sets, and the standard error for each height was computed as

$$s = \frac{\sigma}{\sqrt{n-1}}. \tag{4.2}$$

Here, $\sigma$ is the standard deviation of the potential energy at a given height, and $n$ is the number of samples. The entropy profiles were calculated by subtracting the potential of mean force from the enthalpy $T\Delta S(z) = \Delta U(z) - \Delta F(z)$, and error bars were calculated by simple propagation of errors.

Long-ranged Coulomb interactions were computed using the particle-particle particle-mesh solver[25] with an interpolation order 5, a neutralizing background charge, a $k$-space grid of 18×16×30 and a screening parameter of 2.95 nm$^{-1}$. Short range Lennard-Jones (LJ) interactions were also defined between atomic species $i$ and $j$:

$$U_{LJ}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right], \tag{4.3}$$

with parameters given in Table 4.1[20,26,27]. There were no Coulomb interactions between graphene carbon atoms and other species. Furthermore, as their equations of motion were not integrated, no interaction potential between carbon atoms was defined (although tests with a flexible graphene model were performed, see below). Similarly, as only a single iodide ion was present, no iodide-iodide LJ parameters were defined.

| Interaction | $\epsilon_{ij}$ (kJ/mol) | $\sigma_{ij}$ (nm) |
|---|---|---|
| O-O | 0.650 | 0.3166 |
| C-O | 0.392 | 0.3190 |
| $I^{-0.8}$-O | 0.521 | 0.4145 |
| $I^{-0.8}$-C | 0.708 | 0.4169 |

Table 4.1: Lennard-Jones parameters used in the simulations. The water-water parameters were taken from Reference 20, the water-carbon parameters from Reference 26 and water-iodide parameters from Reference 27. The iodide-carbon $\epsilon_{ij}$ was chosen to obtain agreement with experiment, and the obtained absorption energy of a single iodide at the graphene sheet (no waters) is in reasonable agreement with literature values obtained with density functional theory.[28,29]

### 4.2.5 Analysis of interfacial fluctuations

To analyze the fluctuations of both the air/water and graphene/water interfaces, we closely followed the methodology outlined in Reference 16. Specifically, we used the *instantaneous interface* method of Willard and Chandler[30], in which Gaussian mass distributions are assigned to each water oxygen atom. At each point in space, the coarse-grained density field is defined as the sum of all such Gaussian mass distributions, and the interface is taken to be the 2-dimensional manifold where the coarse grained density field is equal to half its bulk value (16 $nm^{-3}$). The Gaussian mass distribution had a width 0.3 nm, and was truncated and shifted at 0.9 nm. In practice, the coarse grained density field is evaluated on a grid, with spacings 0.1014, 0.1036 and 0.0500 nm in the $x$, $y$ and $z$ directions, respectively.

### 4.2.6 Computing the adsorption free energy from the PMFs

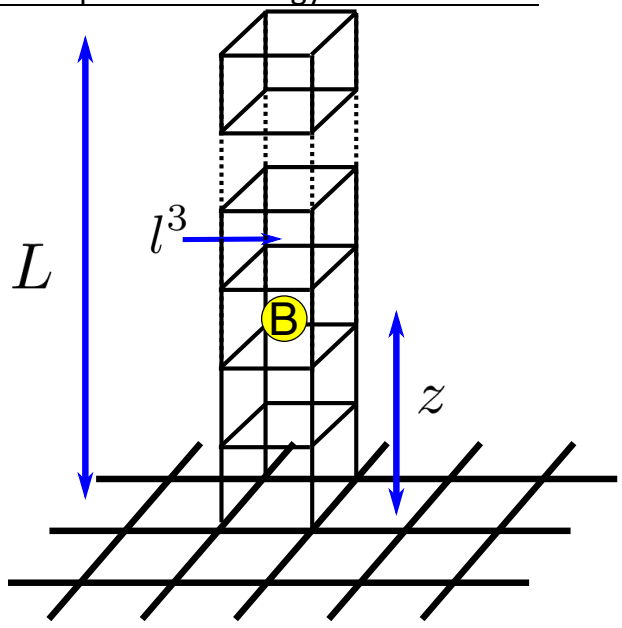



Figure 4.4: Schematic of the model used to calculate the adsorption free energy. We imagine that a single solute 'B' is constrained to a column above the surface. The dimensions of the column are $L \times l \times l$ and we imagine that it has been divided into small cubes of volume $l^3$. The probability that the solute is adsorbed to the surface is given by Equation (4.4).

In our simulations, we obtain a PMF, $\Delta F(z)$, for the ion above the graphene surface. In order to compare to experiment, we need to obtain an adsorption free energy, $\Delta G_{ads}$. To do

this, we imagine dividing space into small cubic cells of volume $l^3$. Now consider a single solute 'B' constrained to an $L \times l \times l$ column, where $L$ is a macroscopic distance. This is shown schematically in Figure 4.4. Let us define the following:

- $q_B$, the internal partition function of a cell in bulk containing a solute molecule.
- $q_{B,s}$, the internal partition function of a cell at the surface containing a solute molecule.
- $q_0$, the internal partition function of a cell in bulk with no solute molecule.
- $q_{0,s}$, the internal partition function of a cell at the surface with no solute molecule.

The surface binding probability of a solute that is constrained to reside in the column is:

$$P_{ads} = \frac{q_{B,s}q_0}{q_B q_{0,s}} \frac{l}{L}. \tag{4.4}$$

We can also compute this probability from simulation:

$$\begin{aligned} P_{ads} &= \frac{1}{L}\int_0^{z^*} dz' e^{-\beta \Delta F(z')} \\ &= \frac{K}{L}, \end{aligned} \tag{4.5}$$

$$K \equiv \int_0^{z^*} dz' e^{-\beta \Delta F(z')}. \tag{4.6}$$

Here, $z^*$ is a microscopic distance from the interface below which we consider the solute adsorbed, and $\beta = 1/k_B T$ is the inverse temperature. It immediately follows that:

$$K = \frac{q_{B,s}q_0}{q_B q_{0,s}} l. \tag{4.7}$$

Let us denote the probability that a solute occupies a particular site by $\phi_B$. If there are a total of $N_B$ solutes, and the total volume of the system is $V$, which we have also divided into small cubes of volume $l^3$, we can write:

$$\begin{aligned} \frac{\phi_B}{1-\phi_B} &= \frac{Q_{occ}}{Q_{unocc}} \\ &= \frac{q_{B,s} q_B^{N_B-1} q_0^{\left(\frac{V}{l^3}\right)-N_B+1} \left(\frac{V}{l^3}\right)^{N_B-1} \frac{1}{(N_B-1)!}}{q_{0,s} q_B^{N_B} q_0^{\left(\frac{V}{l^3}\right)-N_B} \left(\frac{V}{l^3}\right)^{N_B} \frac{1}{(N_B)!}} \\ &= \frac{q_{B,s}q_0}{q_B q_{0,s}} \left(\frac{N_B l^3}{V}\right) \\ &= K l^2 \rho_B. \end{aligned} \tag{4.8}$$

Here, $Q_{occ}$ and $Q_{unocc}$ are the total partition functions for the system when a particular site is occupied and unoccupied by a solute, respectively, and $\rho_B$ is the bulk concentration of solute molecules. The equilibrium constant for the Langmuir model used to interpret the experimental data is:

$$K_{ads} = \frac{\sigma_B \rho_A}{\sigma_A \rho_B}. \tag{4.9}$$

Here, $\sigma_X$ is the surface density of species 'X', and the subscript 'A' indicates quantities pertaining to the solvent. If the maximum surface density is $\sigma_{max}$, and all surface sites are occupied by either solvent or solute molecules we can write:

$$\begin{aligned} K_{ads} &= \frac{\sigma_B \rho_A}{(\sigma_{max} - \sigma_B)\rho_B} \\ &= \frac{\phi_B}{(1-\phi_B)} \frac{\rho_A}{\rho_B} \\ &= K l^2 \rho_A. \end{aligned} \tag{4.10}$$

The adsorption free energy is therefore calculated from the simulation PMF as follows:

$$\Delta G_{ads} = -k_B T ln[(\rho_A l^2) K]. \tag{4.11}$$

For a liquid-vapor interface, there is a certain degree of ambiguity in choosing the size of an adsorption site $l$. If we assume that $l$ is the same for adsorption to the air/water and graphene/water interfaces, then the difference in free energies

$$\begin{aligned} \Delta\Delta G_{ads} &\equiv \Delta G_{ads}^{gra} - \Delta G_{ads}^{vap} \\ &= -k_B T ln \frac{K^{gra}}{K^{vap}} \end{aligned} \tag{4.12}$$

will be independent of the choice of $l$. The superscripts 'gra' and 'vap' indicate quantities calculated for the graphene and air interfaces, respectively. A negative value of $\Delta\Delta G_{ads}$ corresponds to more favorable adsorption at graphene than at air. This provides a direct way to compare the simulation and experimental results.

## 4.3 Results

Figure 4.5 shows the actual SHG signal collected (normalized to the nonresonant SHG signal of water) versus bulk concentration of thiocyanate. The data were fit to the same simple Langmuir model used in Reference 16, as described by Equation (2.18). The only difference is the nonresonant graphene term. Assuming that the same number of carbon atoms are probed on average, this is constant and can be included in 'A.'

$$\frac{I_{2\omega}}{I_{\omega}^{2}} \propto \left(\beta_{water}^{eff} + \frac{N_{gra}}{N_{water}}\beta_{gra}^{eff} + \frac{N_{SCN^-}}{N_{water}} Re\{\beta_{SCN^-}^{eff}\}\right)^2 + \left(\frac{N_{SCN^-}}{N_{water}} Im\{\beta_{SCN^-}^{eff}\}\right)^2$$
$$= \left(A + B[SCN^-]_{surf}\right)^2 + \left(C[SCN^-]_{surf}\right)^2$$
$$= \left(A + B'\frac{X_{SCN^-}}{(1 - X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2 + \left(C'\frac{X_{SCN^-}}{(1 - X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2 \quad (4.13)$$

The functional form of Equation (4.13) ends up being the same as Equation (2.18).

The parameter values are summarized in Table 4.2. In particular, note the free energy of ion adsorption: $\Delta G_{ads}^{gra}$ = -8.8 ± 0.4 kJ/mol. While statistically different from the air/water interface – $\Delta G_{ads}^{vap}$ = -6.78 ± 0.03 kJ/mol) – this corresponds to a difference of < 1 $k_B T$ and is not thermodynamically significant. The model used does not account for any surface potential caused by the electrical double layer, which explains the deviation of the fit from the data at higher concentrations.

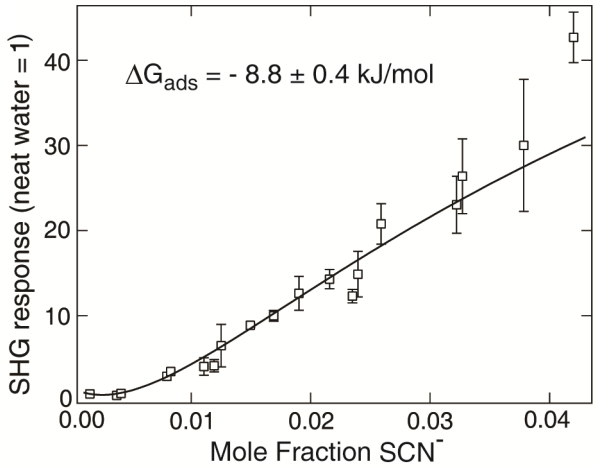



Figure 4.5: SHG signal (normalized to the nonresonant SHG signal of water) versus bulk concentration of thiocyanate. The data were fit to a Langmuir model (Equation (2.18)) and the free energy of adsorption was extracted.

| | |
|---|---|
| A | 1.21 ± 0.07 |
| B | -7.4 ± 0.5 |
| C | 7 ± 1 |
| $\Delta G$ | -8.8 ± 0.4 kJ/mol |

Table 4.2: Values for the fitting parameters obtained from the Levenberg-Marquardt algorithm. Parameter errors are the square roots of the corresponding diagonal elements in the variance-covariance matrix.

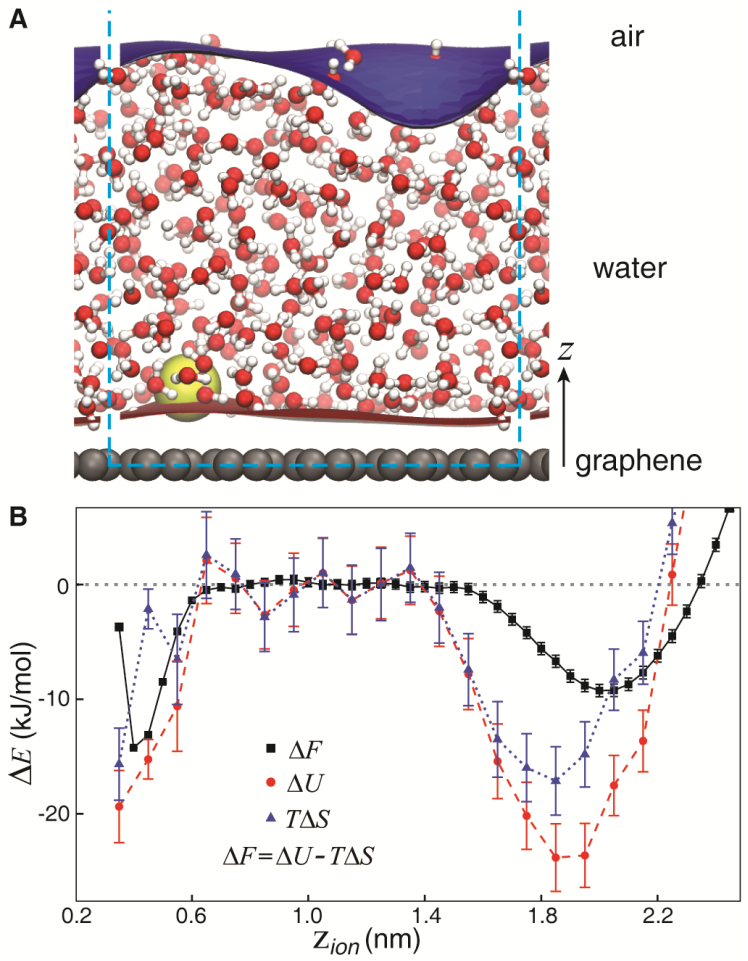



Figure 4.6: Simulation results for iodide interacting with both graphene/water and air/water interfaces. **(A)** A representative snapshot of the simulation box. The graphene is at $z$=0.0 nm in grey. The iodide is in yellow. The instantaneous interfaces for the air/water interface and graphene/water interface are shown in blue and red, respectively. The periodic boundaries are indicated by blue dashed lines. The carbons and waters are not rendered with a space-filling representation, meaning any gaps are not realistic. This was done to make sure the graphene/water instantaneous interface is visible. **(B)** The potential of mean force (black), total potential energy (red), and entropy (blue) curves for the ion vs distance from the graphene sheet. Graphene is centered at $z_{ion}$=0.0nm. Distances less than $z_{ion}$~0.4nm are effectively disallowed by steric repulsion. Total potential energy is nearly identical to enthalpy at ambient conditions.

To elucidate the molecular details underlying the small change in $\Delta G_{ads}$, we performed MD simulations of ion adsorption to the graphene/water interface. Previous studies have shown that iodide and thiocyanate exhibit very similar behavior,[16] hence our previous halide model was used. The simulation box contained a graphene/water interface on one end (with graphene centered at $z$=0.0nm) and an air/water interface on the other (Figure 4.6A). The potential of mean force (PMF) for an iodide ion above the graphene was constructed using umbrella sampling, in which the height of the ion was biased with a harmonic potential (Figure 4.6B, black curve). As discussed in the methods, computing $\Delta G_{ads}$ directly from the PMF requires the size of an adsorption for an ion at the air/water interface to be defined. This is avoided by comparing the difference $\Delta\Delta G_{ads}$ (Equation (4.12)) of values for the graphene/water and air/water interfaces. Discussed in detail below, there is some flexibility in choosing the model parameters. We have found that our results are surprisingly robust to

varying the flexibility of graphene, but that $\Delta\Delta G_{ads}$ does depend on the choice of interaction strength between the ion and the graphene. We have chosen this interaction to reproduce the experimental free energy differences, which yields an adsorption energy for iodide at graphene in vacuum (no waters) in reasonable agreement with density functional theory.[28,29] Despite the similarity in the adsorption free energies at the two interfaces, there are qualitative differences in the PMFs for the two interfaces, namely the graphene/water free energy minimum is much deeper and narrower than that for air/water ($z$~2nm).

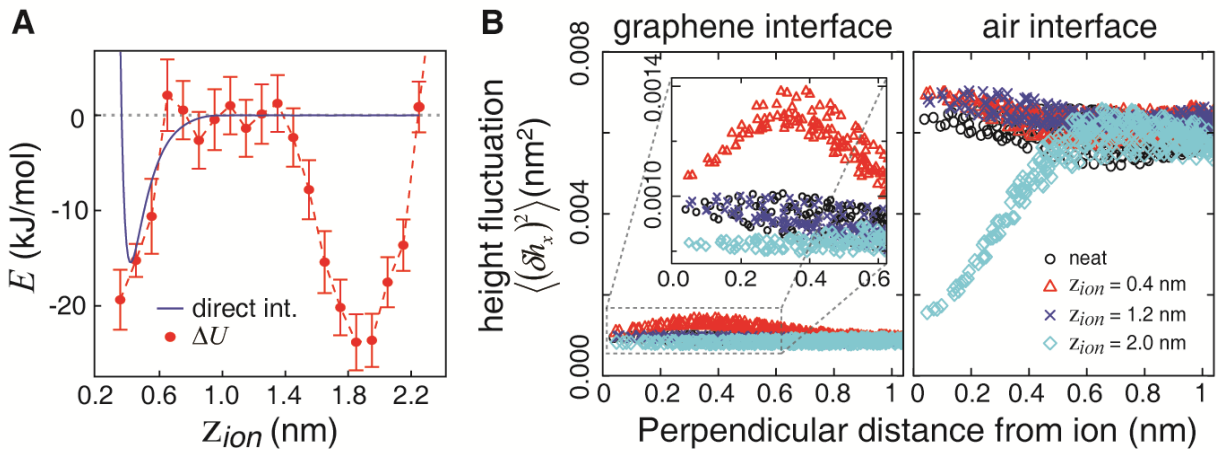



Figure 4.7: Examining potential energy (enthalpy) and height fluctuations (entropy). **(A)** The total potential energy (red) of the iodide in solution and the direct interaction energy (blue) of a single iodide at the graphene sheet (no waters). Graphene is centered at $z_{ion}$=0.0nm. The total potential energy has contributions from ion-graphene interactions, water-ion interactions, water-water interactions, and Coulomb forces. **(B)** The variance of the height fluctuations of the instantaneous interface[30] for the graphene interface (left) and the air interface (right) with the ion positioned at the graphene interface (0.4 nm, red triangle), in bulk (1.2 nm, blue x), and at the air interface (2.0 nm, cyan diamond). Neat simulations without the ion are shown with black circles for comparison. The main plots are shown with the same $y$ axis for direct comparison; the smaller scale of the inset highlights changes of the comparably placid graphene/water interface due to the ion's proximity.

The enthalpy was calculated directly from the total potential energy of the simulations (Figure 4.6B, red curve) and the entropy was calculated by subtracting the enthalpy from the free energy (Figure 4.6B, blue curve) – see methods. The air/water interface has a more favorable enthalpy change than does the graphene/water interface, but this is offset by an unfavorable entropy change, whereas the graphene/water interface exhibits an entropy contribution near zero (distances less than $z_{ion}$~0.4nm are effectively disallowed by steric repulsion). To clarify the enthalpy contributions, Figure 4.7A compares the total potential energy to the direct interaction of iodide with graphene in vacuum (i.e. with no waters in the simulation box). The total potential energy has contributions from ion-graphene interactions, water-ion interactions, water-water interactions, and Coulomb forces. Comparison of the two curves reveals that the potential energy at the graphene/water interface is primarily due to the direct interaction, and not to the solvent repartitioning energy, as for the air/water interface, which has no equivalent direct interaction. Figure 4.8 depicts this, displaying spatial maps of water-water interactions (row A) and ion-water interactions (row B), as first described in Reference 16. Note that graphene interactions are not included in this calculation. Notice in Figure 4.8A (especially the middle panel) that the water-water interactions are less disrupted at the graphene interface than at the air interface. This leads to a less favorable enthalpy change

when these interfacial waters are repartitioned back to the bulk solution. To better understand the entropy contributions, Figure 4.7B shows the height fluctuations of the two interfaces relative to the instantaneous interface,[30] an important contribution to the entropy at the air/water interface.[16] The graphene sheet itself severely dampens these fluctuations (Figure 4.7B left panel, cyan curve) and the iodide actually *enhances* the fluctuations when it approaches the interface (Figure 4.7B left panel, red curve). In contrast, at the air/water interface, large fluctuations are dampened when the ion approaches the interface (Figure 4.7B right panel).

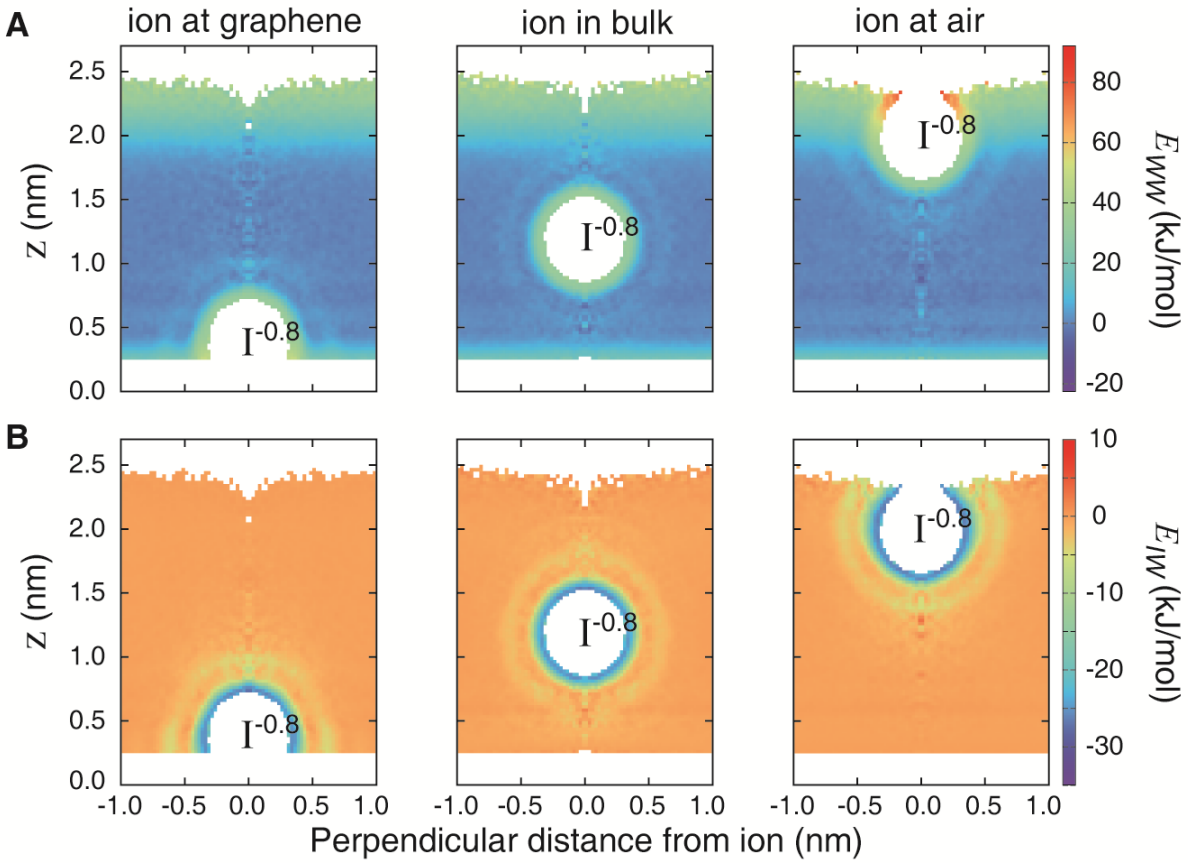



Figure 4.8: Spatial maps of water interactions. Graphene is centered at $z$=0.0nm. Note that interactions with graphene are not included in this calculation. **(A)** The average interaction a water experiences with all other waters for the ion positioned at the graphene interface (left), in the bulk (middle), and at the air interface (right). The zero of energy for all maps corresponds to bulk values for easy comparison. The depression at the air-water interface is an artifact of the cylindrical averaging. **(B)** The average interaction a water experiences with the ion for the ion positioned at the graphene interface (left), in the bulk (middle), and at the air interface (right).

## 4.4 Discussion

Given the electronic properties of graphene and the qualitative differences in the molecular details underlying ion adsorption to graphene and air, it is surprising that the experimental free energies differ only by -2.0 ± 0.4 kJ/mol. This corresponds to < 1 $k_BT$, so this difference, while *statistically* significant, is not *thermodynamically* significant. Free energies of adsorption at the silica/water(cyclohexane) interface for chloride and magnesium (hexanol) are in the tens of

kJ/mol.[11–13] If graphene were indeed behaving like a metal, one might expect a result on the same order of magnitude, due to strong image charge attraction between graphene and the ion. Because the free energies are so similar, it is likely that graphene is behaving primarily as a hydrophobe. Nevertheless, the mechanism of adsorption to the graphene/water interface is qualitatively different than for air/water. The air/water interface exhibits a large enthalpic contribution dominated by favorable solvent repartitioning and an unfavorable entropic contribution from the dampening of capillary waves. The graphene/water interface exhibits a smaller enthalpic contribution dominated by the direct interaction of the iodide and graphene and a much reduced entropic contribution. These differences in adsorption enthalpy and entropy cancel such that the free energies of the interfaces remain very similar. This distinction challenges the common notion that hydrophobic interfaces present a solvation environment much like the air/water interface.[31] We show that the similarity of adsorption affinities actually reflects a subtle cancellation in differences in adsorption enthalpy and entropy. The underlying mechanistic differences we have identified are not specific to graphene; they may apply as well for ion solvation at other substrates with significant hydrophobic character, such as biological macromolecules.

### 4.4.1 Using the instantaneous interface to compute the PMFs

In addition to computing the PMF relative to the graphene sheet, it is also possible to use the height relative to the instantaneous interface,[32] as shown in Figure 4.9. This analysis reveals that at the air/water interface, the ion prefers to reside slightly in the vapor phase, whereas the opposite is true at the graphene/water interface. We also find $\Delta\Delta G_{ads}$ = -1.7 kJ/mol when these PMFs are used, in good agreement with the value computed with the PMF relative to the graphene sheet ($\Delta\Delta G_{ads}$ = -1.9 kJ/mol).

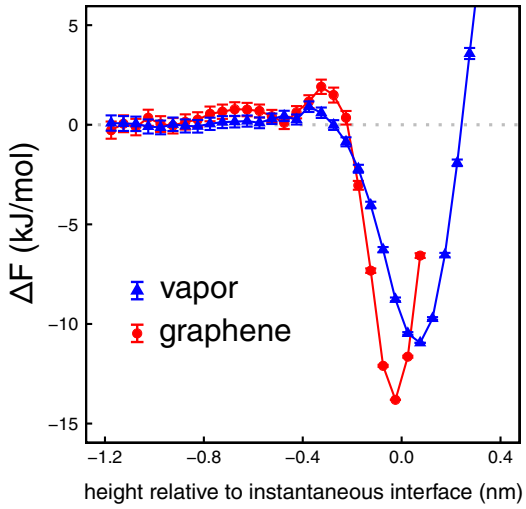



Figure 4.9: PMFs computed relative to the instantaneous interface. The air/water interface is in blue and the graphene/water interface is in red. A height of 0.0 corresponds to the appropriate instantaneous interface. The change in axis convention is to account for the fact that multiple instantaneous interfaces exist. Negative values for the height correspond to the liquid phase.

### 4.4.2 The effect of graphene flexibility

The simulation results presented so far have been obtained with a rigid model of graphene. It is possible that allowing the graphene to vibrate could affect water's interfacial fluctuations and consequently, the propensity of ions to adsorb to the interface. We have therefore performed the simulations using an optimized Tersoff model for graphene,[33] which has been shown to predict the harmonic and anharmonic interactions for graphene reasonably well.[34] Figure 4.10 shows the PMF calculated with $\epsilon_{C-I^{-0.8}}$ = 0.315 kJ/mol, both relative to the average height of the graphene surface and the instantaneous interface. While there are some very slight changes in the PMF when computed relative to the average graphene height, these have negligible effect on the adsorption free energies, with $\Delta\Delta G_{ads}$ = +5.1 and +4.9 kJ/mol for the rigid and flexible models, respectively. (The PMFs relative to the instantaneous interface give $\Delta\Delta G_{ads}$ = +5.0 and +4.7 kJ/mol for rigid and flexible graphene models, respectively.)

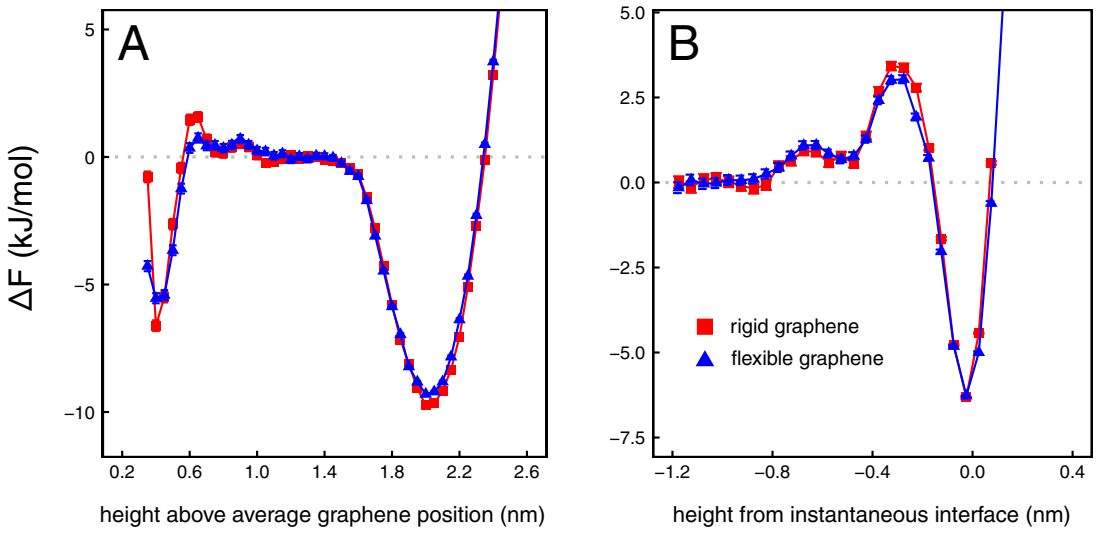



Figure 4.10: PMF for an iodide above rigid and flexible graphene models, with $\epsilon_{C-I^{-0.8}}$ = 0.315 kJ/mol**.** The introduction of graphene flexibility has little effect. The air/water interface is in blue and the graphene/water interface is in red. **(A)** The PMF relative to the average graphene position. Graphene is centered at 0.0 nm. **(B)** The height relative to the instantaneous interface. Graphene is centered at 0.0 nm and negative values for the height correspond to the liquid phase. The change in axis convention is to account for the fact that multiple instantaneous interfaces exist.

### 4.4.3 Sensitivity of the potential of mean force to direct interactions between the iodide and graphene

In the discussion so far, we have presented results using the Lennard-Jones parameters given in Table 4.1. To test the sensitivity of our simulation results to the direct interaction between the iodide and the graphene, we also calculated the PMFs using different values of $\epsilon_{C-I^{-0.8}}$. First, we applied the standard Lorentz- Berthelot mixing rules, which gave $\epsilon_{C-I^{-0.8}}$ = 0.315 kJ/mol. Second, we used a much stronger interaction with $\epsilon_{C-I^{-0.8}}$ = 1.046 kJ/mol. Third, we used an interaction that reproduced experimental results. The adsorption energies for a single iodide at the graphene sheet (no waters) using the different $\epsilon_{C-I^{-0.8}}$ values are presented in Table 4.3, along with differences in adsorption free energies between the graphene/water and air/water interfaces (see Equation (4.12)). Adsorption energies in the literature for iodide adsorption at graphene are scarce. Using density functional theory (DFT), Zhu and Yang[28] report

a value of -27.0 kJ/mol for iodide at the center of a $C_{54}H_{18}$ single layer sheet (19 aromatic rings). Shi *et al.*[29] report DFT adsorption energies of -68.2, -32.2, and -25.1 kJ/mol for $F^-$, $Cl^-$ and $Br^-$, respectively, suggesting that the adsorption energy of $I^-$ is weaker than -25 kJ/mol. While it is clear from Table 4.3 that $\Delta\Delta G_{ads}$ is sensitive to $\epsilon_{C-I^{-0.8}}$, it is reassuring that a value that gives reasonable agreement with experiment also gives a sensible adsorption energy. Furthermore, given the insensitivity of the simulation results to graphene's flexibility, this suggests that the experimental observation of similar adsorption free energies to the graphene and air interfaces is due to contributions from the direct graphene-solute interaction that offsets a weaker contribution from solvent repartitioning interactions relative to the vapor interface.

| $\epsilon_{C-I^{-0.8}}$ | Adsorption energy (kJ/mol) | $\Delta\Delta G_{ads}$ (kJ/mol) |
|---|---|---|
| 0.315 | -6.9 | +5.1 |
| 0.708 | -15.4 | -1.9 |
| 1.046* | -22.8 | -8.9 |

Table 4.3: Adsorption energy of a single iodide to the graphene sheet (no waters) for different values of $\epsilon_{C-I^{-0.8}}$, and differences in adsorption free energies between the graphene/water and air/water interfaces ($\Delta\Delta G_{ads}$). *These simulations were performed with the flexible graphene model.

## 4.5 Conclusions and Future Directions

In summary, SHG signal vs thiocyanate concentration was collected at the graphene/water interface and fit to a Langmuir model in order to extract the free energy of adsorption, which is only -2.0 ± 0.4 kJ/mol larger than for the air/water interface. This corresponds to < 1 $k_BT$, so this difference, while *statistically* significant, is not *thermodynamically* significant. If the high carrier mobility of graphene led to metal-like behavior, one would expect that the free energy would instead be much larger be in the tens of kJ/mol, like at the silica/water interface.[11–13] Molecular dynamics simulations indicate that despite the similar adsorption free energies, qualitative differences in the adsorption mechanism exist. First, the direct interaction of the ion with graphene dominates the favorable adsorption enthalpy, while the solvent repartitioning of weakly interacting waters, which dominates the enthalpy for the air/water case, is negligible. Second, the entropic penalty due to pinning of capillary waves at air/water is negligible for the much more rigid graphene interface.

While these results are certainly interesting and informative, there is still much to be done on the experimental side. To verify that the signal collected is actually on a resonance, and to help make the fit more accurate, data are collected at multiple wavelengths.[17,35,36] This has not yet been done for the graphene/water interface, so it would be prudent to do so. Also, the adsorption mechanism suggested by the molecular dynamics simulations needs to be verified experimentally through temperature dependent experiments like those described in Reference 16.

## 4.6 References

(1) Griffiths, D. J. *Introduction to Electrodynamics*, Fourth edition.; Pearson: Boston, 2013.

(2) Arima, K.; Jiang, P.; Lin, D.-S.; Verdaguer, A.; Salmeron, M. Ion Segregation and Deliquescence of Alkali Halide Nanocrystals on $SiO_2$. *J. Phys. Chem. A* **2009**, *113* (35), 9715–9720.
(3) Kuzmenko, A. B.; van Heumen, E.; Carbone, F.; van der Marel, D. Universal Optical Conductance of Graphite. *Phys. Rev. Lett.* **2008**, *100* (11), 117401.
(4) Novoselov, K. S.; Fal'ko, V. I.; Colombo, L.; Gellert, P. R.; Schwab, M. G.; Kim, K. A Roadmap for Graphene. *Nature* **2012**, *490* (7419), 192–200.
(5) Dankerl, M.; Hauf, M. V.; Lippert, A.; Hess, L. H.; Birner, S.; Sharp, I. D.; Mahmood, A.; Mallet, P.; Veuillen, J.-Y.; Stutzmann, M.; et al. Graphene Solution-Gated Field-Effect Transistor Array for Sensing Applications. *Adv. Funct. Mater.* **2010**, *20* (18), 3117–3124.
(6) Joshi, R. K.; Carbone, P.; Wang, F.-C.; Kravets, V. G.; Su, Y.; Grigorieva, I. V.; Wu, H. A.; Geim, A. K.; Nair, R. R. Precise and Ultrafast Molecular Sieving through Graphene Oxide Membranes. *Science* **2014**, *343* (6172), 752–754.
(7) Surwade, S. P.; Smirnov, S. N.; Vlassiouk, I. V.; Unocic, R. R.; Veith, G. M.; Dai, S.; Mahurin, S. M. Water Desalination Using Nanoporous Single-Layer Graphene. *Nat. Nanotechnol.* **2015**, *10* (5), 459–464.
(8) Stoller, M. D.; Park, S.; Zhu, Y.; An, J.; Ruoff, R. S. Graphene-Based Ultracapacitors. *Nano Lett.* **2008**, *8* (10), 3498–3502.
(9) Yoo, E.; Kim, J.; Hosono, E.; Zhou, H.; Kudo, T.; Honma, I. Large Reversible Li Storage of Graphene Nanosheet Families for Use in Rechargeable Lithium Ion Batteries. *Nano Lett.* **2008**, *8* (8), 2277–2282.
(10) D'Urso, L.; Satriano, C.; Forte, G.; Compagnini, G.; Puglisi, O. Water Structure and Charge Transfer Phenomena at the Liquid-Graphene Interface. *Phys. Chem. Chem. Phys.* **2012**, *14* (42), 14605–14610.
(11) Achtyl, J. L.; Vlassiouk, I. V.; Fulvio, P. F.; Mahurin, S. M.; Dai, S.; Geiger, F. M. Free Energy Relationships in the Electrical Double Layer over Single-Layer Graphene. *J. Am. Chem. Soc.* **2013**, *135* (3), 979–981.
(12) Achtyl, J. L.; Vlassiouk, I. V.; Surwade, S. P.; Fulvio, P. F.; Dai, S.; Geiger, F. M. Interaction of Magnesium Ions with Pristine Single-Layer and Defected Graphene/Water Interfaces Studied by Second Harmonic Generation. *J. Phys. Chem. B* **2014**, *118* (28), 7739–7749.
(13) Achtyl, J. L.; Vlassiouk, I. V.; Dai, S.; Geiger, F. Interactions of Organic Solvents at Graphene/α-$Al_2O_3$ and Graphene Oxide/α-$Al_2O_3$ Interfaces Studied by Sum Frequency Generation. *J. Phys. Chem. C* **2014**, *118* (31), 17745–17755.
(14) Shih, C.-J.; Strano, M. S.; Blankschtein, D. Wetting Translucency of Graphene. *Nat. Mater.* **2013**, *12* (10), 866–869.
(15) Currie, M.; Caldwell, J. D.; Bezares, F. J.; Robinson, J.; Anderson, T.; Chun, H.; Tadjer, M. Quantifying Pulsed Laser Induced Damage to Graphene. *Appl. Phys. Lett.* **2011**, *99* (21), 211909.
(16) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.
(17) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.

(18) Malard, L. M.; Pimenta, M. A.; Dresselhaus, G.; Dresselhaus, M. S. Raman Spectroscopy in Graphene. *Phys. Rep.* **2009**, *473* (5–6), 51–87.
(19) Hao, Y.; Wang, Y.; Wang, L.; Ni, Z.; Wang, Z.; Wang, R.; Koo, C. K.; Shen, Z.; Thong, J. T. L. Probing Layer Number and Stacking Order of Few-Layer Graphene by Raman Spectroscopy. *Small* **2010**, *6* (2), 195–200.
(20) Berendsen, H. J. C.; Grigera, J. R.; Straatsma, T. P. The Missing Term in Effective Pair Potentials. *J. Phys. Chem.* **1987**, *91* (24), 6269–6271.
(21) Schneider, T.; Stoll, E. Molecular-Dynamics Study of a Three-Dimensional One-Component Model for Distortive Phase Transitions. *Phys. Rev. B* **1978**, *17* (3), 1302.
(22) Dünweg, B.; Paul, W. Brownian Dynamics Simulations without Gaussian Random Numbers. *Int. J. Mod. Phys. C* **1991**, *2* (3), 817–827.
(23) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117* (1), 1–19.
(24) Shirts, M. R.; Chodera, J. D. Statistically Optimal Analysis of Samples from Multiple Equilibrium States. *J. Chem. Phys.* **2008**, *129* (12), 124105.
(25) Hockney, R. W.; Eastwood, J. W. *Computer Simulation Using Particles*, Special student ed.; Adam Hilger: Bristol and New York, 1988.
(26) Werder, T.; Walther, J. H.; Jaffe, R. L.; Halicioglu, T.; Koumoutsakos, P. On the Water−carbon Interaction for Use in Molecular Dynamics Simulations of Graphite and Carbon Nanotubes. *J. Phys. Chem. B* **2003**, *107* (6), 1345–1352.
(27) Dang, L. X. Computational Study of Ion Binding to the Liquid Interface of Water. *J. Phys. Chem. B* **2002**, *106* (40), 10388–10394.
(28) Zhu, C.; Yang, G. Insights from the Adsorption of Halide Ions on Graphene Materials. *ChemPhysChem* **2016**, *17* (16), 2482–2488.
(29) Shi, G.; Ding, Y.; Fang, H. Unexpectedly Strong Anion-π Interactions on the Graphene Flakes. *J. Comput. Chem.* **2012**, *33* (14), 1328–1337.
(30) Willard, A. P.; Chandler, D. Instantaneous Liquid Interfaces. *J. Phys. Chem. B* **2010**, *114* (5), 1954–1958.
(31) Willard, A. P.; Chandler, D. The Molecular Structure of the Interface between Water and a Hydrophobic Substrate Is Liquid-Vapor like. *J. Chem. Phys.* **2014**, *141* (18), 18C519.
(32) Stern, A. C.; Baer, M. D.; Mundy, C. J.; Tobias, D. J. Thermodynamics of Iodide Adsorption at the Instantaneous Air-Water Interface. *J. Chem. Phys.* **2013**, *138* (11), 114709.
(33) Lindsay, L.; Broido, D. A. Optimized Tersoff and Brenner Empirical Potential Parameters for Lattice Dynamics and Phonon Thermal Transport in Carbon Nanotubes and Graphene. *Phys. Rev. B* **2010**, *81* (20), 205441.
(34) Ma, M.; Tocci, G.; Michaelides, A.; Aeppli, G. Fast Diffusion of Water Nanodroplets on Graphene. *Nat. Mater.* **2015**, *15* (1), 66–71.
(35) Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. Enhanced Concentration of Polarizable Anions at the Liquid Water Surface: SHG Spectroscopy and MD Simulations of Sodium Thiocyanide. *J Phys Chem B* **2005**, *109* (21), 10915–10921.
(36) Petersen, P. B.; Saykally, R. J. Evidence for an Enhanced Hydronium Concentration at the Liquid Water Surface. *J Phys Chem B* **2005**, *109* (16), 7976–7980.

# Chapter 5 – Model Comparison and Error Analysis

## 5.1 Introduction

By inserting the Langmuir equilibrium expression into the second harmonic generation expression to derive Equation (2.18), the assumption is made that the input laser pulses are the only electric fields present to generate the polarization. However, with ion adsorption, a double layer can form[1] that can also contribute an electric field. A solution-gated field effect transistor made with graphene and $Na_2PO_3$ solution even showed a high capacitance.[2] Knowing this, it seems pertinent to account for the surface potential in the Langmuir model.

## 5.2 The Langmuir Model with a Surface Potential Term

If the double layer contributes a third (static) electric field, $E_0$, a third order polarization term must be considered:

$$P^{(3)} \propto \chi^{(3)} E_\omega^2 E_0. \tag{5.1}$$

Since $E_0$ is a static field, the frequency of $P^{(3)}$ is also $2\omega$ and $P^{(3)}$ is indistinguishable from $P^{(2)}$. Then the total polarization at $2\omega$ will be:

$$\begin{aligned} P_{2\omega} &= P^{(2)} + P^{(3)} \\ &\propto \chi^{(2)} E_\omega^2 + \chi^{(3)} E_\omega^2 E_0. \end{aligned} \tag{5.2}$$

The static electric field can be integrated from the interface to positive infinity (water phase) to yield[3]

$$P_{2\omega} \propto \chi^{(2)} E_\omega^2 + \chi^{(3)} E_\omega^2 \Phi_0, \tag{5.3}$$

where $\Phi_0$ is the interfacial potential. Equation (5.3) is the $\chi^{(3)}$ equation and is used in the Eisenthal $\chi^{(3)}$ technique (sometimes called electric field induced second harmonic generation, or EFISH).[4] Generally, this technique is used for nonresonant measurements, where the main contribution is from water molecules aligning to the field.[3–5] This means the terms are all real valued. The resonant case becomes more complicated. Converting to intensities and dividing by the input field gives

$$\frac{I_{2\omega}}{I_\omega^2} \propto \left| \chi^{(2)} + \chi^{(3)} \Phi_0 \right|^2. \tag{5.4}$$

Expanding into real and imaginary terms yields

$$\frac{I_{2\omega}}{I_\omega^2} \propto \left( Re\{\chi^{(2)}\} + \Phi_0 Re\{\chi^{(3)}\} \right)^2 + \left( Im\{\chi^{(2)}\} + \Phi_0 Im\{\chi^{(3)}\} \right)^2. \tag{5.5}$$

On the molecular level, $\chi^{(2)} \propto N\beta^{eff}$ and $\chi^{(3)} \propto N\gamma^{eff}$, where $\beta$ is the molecular hyperpolarizability, $\gamma$ is the third order molecular polarizability, and $\beta^{eff}$ and $\gamma^{eff}$ are

rotational averages. In the interest of being succinct, the derivation will be for the air/water interface. For other interfaces, the equation will be functionally the same, using the same reasoning as in Chapter 3 and Chapter 4. Then, the species that contribute to the signal are water (real only) and the anion, $A^-$ (real and imaginary):

$$\frac{I_{2\omega}}{I_\omega^2} \propto \left(N_{water}\beta_{water}^{eff} + N_{A^-}Re\{\beta_{A^-}^{eff}\} + \Phi_0(N_{water}\gamma_{water}^{eff} + N_{A^-}Re\{\gamma_{A^-}^{eff}\})\right)^2 + \left(N_{A^-}Im\{\beta_{A^-}^{eff}\} + \Phi_0 N_{A^-}Im\{\gamma_{A^-}^{eff}\}\right)^2. \quad (5.6)$$

Again, dividing by $N_{water}$ gives

$$\frac{I_{2\omega}}{I_\omega^2} \propto \left(\beta_{water}^{eff} + \frac{N_{A^-}}{N_{water}}Re\{\beta_{A^-}^{eff}\} + \Phi_0(\gamma_{water}^{eff} + \frac{N_{A^-}}{N_{water}}Re\{\gamma_{A^-}^{eff}\})\right)^2 + \left(\frac{N_{A^-}}{N_{water}}Im\{\beta_{A^-}^{eff}\} + \Phi_0\frac{N_{A^-}}{N_{water}}Im\{\gamma_{A^-}^{eff}\}\right)^2. \quad (5.7)$$

Assuming $\beta^{eff}$ and $\gamma^{eff}$ are constant and converting to molar concentration gives

$$\frac{I_{2\omega}}{I_\omega^2} \propto \left(A + B[A^-]_{surf} + \Phi_0\left(A' + D[A^-]_{surf}\right)\right)^2 + \left(C[A^-]_{surf} + \Phi_0 E[A^-]_{surf}\right)^2, \quad (5.8)$$

where the subscript "surf" indicates the anions at the surface. Constant names were chosen to parallel the simple Langmuir model.

The next step is to find an expression for $\Phi_0$. Jena et al. performed a nonresonant $\chi^{(3)}$ study using NaCl and examined how the interfacial potential behaved for different concentration regions.[5] They found that, for $[Cl^-]_{bulk} > 0.13\text{M}$, the interfacial potential can be described by a Stern model

$$\Phi_0 = \frac{4\pi\sigma_0 d}{\epsilon}, \quad (5.9)$$

where $\sigma_0$ is the surface charge, $d$ is the distance between the surface and anions, and $\epsilon$ is the dielectric constant. Considering that chloride lies towards the kosmotropic end of the Hofmeister series, it is safe to assume that thiocyanate will reach an equivalent surface concentration (and interfacial potential) at a lower bulk concentration. The only dataset that reaches a significantly lower concentration than 0.13M is the temperature dependent dataset. Assuming that $\sigma_0$ is proportional to $[A^-]_{surf}$ and the other quantities remain constant yields

$$\begin{aligned}\frac{I_{2\omega}}{I_\omega^2} &\propto \left(A + B[A^-]_{surf} + A'[A^-]_{surf} + D[A^-]_{surf}^2\right)^2 + \left(C[A^-]_{surf} + E[A^-]_{surf}^2\right)^2 \\ &\propto \left(A + B[A^-]_{surf} + D[A^-]_{surf}^2\right)^2 + \left(C[A^-]_{surf} + E[A^-]_{surf}^2\right)^2. \end{aligned} \quad (5.10)$$

Substituting Equation (2.16) into Equation (5.10) and converting to mole fractions gives our working Langmuir model with a surface potential term.

$$\frac{I_{2\omega}}{I_{\omega}^2} = \left( A' + B' \frac{X_{A^-}}{(1 - X_{A^-}) e^{\frac{\Delta G}{RT}} + X_{A^-}} + D' \left( \frac{X_{A^-}}{(1 - X_{A^-}) e^{\frac{\Delta G}{RT}} + X_{A^-}} \right)^2 \right)^2 + \left( C' \frac{X_{A^-}}{(1 - X_{A^-}) e^{\frac{\Delta G}{RT}} + X_{A^-}} + E' \left( \frac{X_{A^-}}{(1 - X_{A^-}) e^{\frac{\Delta G}{RT}} + X_{A^-}} \right)^2 \right)^2 \tag{5.11}$$

During the fitting process, the initial value for $\Delta G_{ads}$ had to be adjusted for several datasets to avoid numerical overflows and erroneous local minima. These changes are noted in Table 5.1. All other parameters were initialized to +1.

| | **$\Delta G_{ads}$ initial value (J/mol)** | |
|---|---|---|
| **Dataset** | **SL** | **LSP** |
| KSCN temperature dependence[8] | -7000 | -7000 |
| NaSCN dodecanol/water[9] | -1 | -1000 |
| NaSCN < 4M dodecanol/water[9] | -5000 | -1 |
| KSCN < 4M dodecanol/water[9] | -7000 | -7000 |

Table 5.1: Initial values used for some datasets. All other initial values were +1. SL: simple Langmuir model. LSP: Langmuir with surface potential.

### *5.2.1 Model comparison*

There are several metrics for comparing models, all originating from different paradigms of statistics. The frequentist metric is the F-test[6], calculated from Equation (5.12).

$$F = \frac{(\chi^2_{null} - \chi^2_{alt})/(df_{null} - df_{alt})}{\chi^2_{alt}/df_{alt}} \tag{5.12}$$

The null hypothesis is that the simpler model (Equation (2.18)) is correct. The alternative hypothesis is that the more complex model is correct (Equation (5.11)). Then the F statistic can be converted to a p-value and interpreted as normal. Note that the models must be applied to the same datasets and the models must be nested, i.e. the complex model must be a modification of the simple model, such as adding the potential term to the Langmuir model.

The Akaike Information Criterion (AIC) is a metric from Kullback-Leibler information theory.[7] This criterion starts from the assumption that there is a true model ($f(x)$, not known) and calculates how much information is lost by approximating the true model with another model ($g(x|\theta)$). If the estimated parameters are $\hat{\theta}$ and the data are $y$, then the AIC is calculated as follows:

$$AIC = -2\log\left(\mathcal{L}\left(\hat{\theta}\middle|y\right)\right) + 2K, \tag{5.13}$$

where $\mathcal{L}\left(\hat{\theta}\middle|y\right)$ is the likelihood of the estimated parameters given the data, and $K$ is the number of parameters that were allowed to vary in the fitting procedure ($A, B, C, D, E$, and $\Delta G$, but not $R$ and $T$). In this way, the criterion accounts for the likelihood of the fit while also penalizing models with many parameters; it takes into account the bias-variance tradeoff. For least squares, this becomes

$$AIC = n\log(\chi^2/n) + 2K, \tag{5.14}$$

where $n$ is the number of data points and $\chi^2$ is the sum of squared errors.

Equations (5.13) and (5.14) are generally only accurate for $n/K > 40$.[7] For small datasets, the AIC needs to be corrected. Unfortunately, the correction depends on the type of model and is not easily derived. The standard correction is for a fixed-effects linear model with normal errors and constant residual variances, given by Equation (5.15):

$$AIC_c = AIC + 2K\frac{K+1}{n-K-1} \tag{5.15}$$

One thing to note about the AIC is that it is relative to the unknown true model. The AIC can determine which of two models is better, but cannot determine if both models are bad.

The Bayesian metric is the Bayesian Information Criterion (BIC) and is given by Equation (5.16):[7]

$$BIC = -2\log\left(\mathcal{L}\left(\hat{\theta}\middle|y\right)\right) + K\log(n) \tag{5.16}$$

For least squares, this becomes

$$BIC = n\log(\chi^2/n) + K\log(n). \tag{5.17}$$

The derivation is purely from a Bayesian standpoint, which presents two problems:[7] One, it is based on the true model being one of the models tested. Two, the penalty for parameter number is not derived from the bias-variance tradeoff, so it is unclear if it will actually address the tradeoff. The BIC also does not have corrections for small sample sizes.

For all of the information criteria, the difference, $\Delta_i$, can be used to interpret the results:

$$\Delta_i = IC_i - IC_{min}, \tag{5.18}$$

where $IC_{min}$ is the model with the minimum value and the other $i$ models are compared against it. The differences can be interpreted according to Table 5.2:[7]

| $\Delta_i$ | Level of empirical support for Model $i$ |
|---|---|
| (0, 2] | Substantial support |
| (2, 7] | Some support |
| (7, 10] | Little support |
| > 10 | No support |

Table 5.2: The interpretation of information criteria differences, where Model $i$ is defined by Equation (5.19). Table adapted from Reference 7.

### *5.2.2 Comparisons*

For clarity, Equation (2.18) will be abbreviated as SL, for the simple Langmuir, and Equation (5.11) will be abbreviated as LSP, for Langmuir with surface potential.

The F-test was interpreted with a significance level of 0.05. The information criteria differences were always calculated as $IC_{SL} - IC_{LSP}$. This means that Model $i$ is SL for positive differences and LSP for negative differences. This works because there are only two models in the comparison. The absolute values of the differences were interpreted according to Table 5.2. To look at the correlations between variables, all values above 0.1 were extracted and the fraction of these values that were 0.9 or greater was calculated for each model. A lower fraction was considered support for that model.

| Dataset | Method | Result |
|---|---|---|
| NaSCN graphene/water | F-test | evidence for LSP |
| | AIC | no support for SL over LSP |
| | BIC | little support for SL over LSP |
| | AICc | some support for SL over LSP |
| | Correlations | support for SL |
| | Conclusion | inconclusive* |
| KSCN temperature dependence[8] | F-test | no evidence for LSP |
| | AIC | no support for SL over LSP |
| | BIC | some support for LSP over SL |
| | AICc | no support for LSP over SL |
| | Correlations | support for SL |
| | Conclusion | SL is better |
| NaSCN dodecanol/water[9] | F-test | evidence for LSP |
| | AIC | some support for SL over LSP |
| | BIC | some support for LSP over SL |
| | AICc | substantial support for LSP over SL |
| | Correlations | support for LSP |
| | Conclusion | LSP is better |

| | | |
|---|---|---|
| NaSCN < 4M dodecanol/water[9] | F-test | evidence for LSP |
| | AIC | some support for SL over LSP |
| | BIC | some support for LSP over SL |
| | AICc | some support for LSP over SL |
| | Correlations | support for SL |
| | Conclusion | inconclusive* |
| KSCN dodecanol/water[9] | F-test | no evidence for LSP |
| | AIC | some support for LSP over SL |
| | BIC | little support for LSP over SL |
| | AICc | little support for LSP over SL |
| | Correlations | support for LSP |
| | Conclusion | SL is better |
| KSCN < 4M dodecanol/water[9] | F-test | no evidence for LSP |
| | AIC | some support for LSP over SL |
| | BIC | little support for LSP over SL |
| | AICc | no support for LSP over SL |
| | Correlations | support for SL |
| | Conclusion | SL is better |
| NaSCN air/water[10] | F-test | evidence for LSP |
| | AIC | no support for SL over LSP |
| | BIC | no support for SL over LSP |
| | AICc | no support for SL over LSP |
| | Correlations | support for SL |
| | Conclusion | inconclusive** |

Table 5.3: A summary of the model comparisons for all the datasets considered, along with the final conclusions about the appropriate model. For the datasets from Reference 9, in the original analysis, the fits did not converge for the full datasets and were truncated. SL: simple Langmuir model. LSP: Langmuir with surface potential. *It's likely that neither model works well for these datasets. **LSP had trouble converging on sensible fits, which made the comparisons suspect.

Table 5.3 summarizes the results from the model comparisons for the graphene/water dataset described in Chapter 4, as well as several other datasets from previous publications.[8–10] One set is the temperature dependence of KSCN at the air/water interface by Otten et al. Four sets come from Onorato et al.: both NaSCN and KSCN at the dodecanol/water interface and the two datasets truncated to concentrations less than 4M. The reason given by the authors is that the full datasets did not converge to sensible fits; the KSCN dataset didn't converge at all. This was also done here for completeness. The final set is NaSCN at the air/water interface from Petersen et al. The $\Delta G_{ads}$ values have also been collected in Table 5.4. This is the only parameter that retains physical meaning, so the others are not discussed further. The full results of the model comparisons and the fit values are given in Appendix 6.

It is worth noting some discrepancies. First, the value for $\Delta G_{ads}$ for the NaSCN air/water dataset is more negative than the other datasets, including the analogous air/water isotherm

from the temperature dependent dataset. This is likely due to the accumulation of photoproducts during the measurement process.[11] Subsequent experiments were modified to avoid this. Second, the current study had no trouble with the full dodecanol/water datasets converging; the truncated datasets were more troublesome in this regard. Also, the values obtained for $\Delta G_{ads}$ in this study (shown in Table 5.4) were systematically less negative than those reported in Reference 9. This is likely due to differences in the fitting equation and/or algorithms. However, this cannot be determined with certainty because the exact fitting procedure is undocumented.

For the graphene/water dataset, the simple Langmuir performed better than the Langmuir with surface potential, but this is likely because neither model does a good job of fitting the data; hence, the inconclusive results for this dataset. This is likely because of the quality of the data. Obtaining better quality data (by reducing error bars, extending the concentration range, or collecting at more wavelengths) would likely give a more conclusive comparison. The NaSCN < 4M dodecanol/water[9] dataset also gave inconclusive results for similar reasons. The result for the NaSCN air/water dataset was inconclusive because the Langmuir with surface potential did not behave well. The support is large for the Langmuir with surface potential over the simple Langmuir, but the difficulty of using the Langmuir with surface potential indicates that the results may be suspect. The simple Langmuir was better for the temperature dataset, which is likely because the low concentration region is not well described by the Stern model. The simple Langmuir was also better for the KSCN dodecanol/water datasets, however the same explanation does not apply.

One interesting thing to note is that the Langmuir with surface potential performed better for the full NaSCN dodecanol/water[10] dataset. The authors of the original paper noted that there was a "salient kink" in the dataset that caused upward deviations from the fit at higher concentrations, the same as with the graphene/water dataset. Perhaps this kink is indicative of the surface potential becoming nonnegligible. Using the Langmuir model with surface potential does bring $\Delta G_{ads}$ for the full NaSCN dodecanol/water dataset into agreement with the truncated dataset.

| Datasets | Model | Reported values (kJ/mol) | Values from fit and covariance (kJ/mol) | Values from Jackknife, $\widehat{\boldsymbol{\theta}}_{(.)} \pm \widehat{\boldsymbol{var}}$ (kJ/mol) | Bias corrected, $\widehat{\boldsymbol{\theta}}_{\boldsymbol{corr}}$ (kJ/mol) |
|---|---|---|---|---|---|
| NaSCN graphene/water | SL | -8.8 ± 0.4 | -8.8 ± 0.4 | -9 ± 2 | -11 |
| | LSP | | -5.0 ± 1.7 | -5 ± 3 | -5 |
| KSCN 274K air/water[8] | SL | -7.06 ± 0.09 | -7.16 ± 0.09 | -7.2 ± 0.2 | -7.1 |
| | LSP | | -9.3 ± 0.3 | -9.4 ± 1.8 | -6.6 |
| KSCN 283K air/water[8] | SL | -6.97 ± 0.04 | -7.09 ± 0.03 | -7.09 ± 0.14 | -7.15 |
| | LSP | | -9.0 ± 0.2 | -9 ± 5 | -9 |
| KSCN 293K air/water[8] | SL | -6.78 ± 0.03 | -6.92 ± 0.03 | -6.9 ± 0.2 | -6.9 |
| | LSP | | -9.6 ± 0.4 | -9.6 ± 0.4 | -9.1 |

| | | | | | |
|---|---|---|---|---|---|
| KSCN 303K air/water[8] | SL | -6.5 ± 0.2 | -6.68 ± 0.15 | -6.68 ± 0.9 | -6.73 |
| | LSP | | -8.2 ± 1.9 | -8.2 ± 1.3 | -8.1 |
| KSCN 313K air/water[8] | SL | -6.46 ± 0.09 | -6.66 ± 0.08 | -6.7 ± 0.7 | -6.0 |
| | LSP | | -10.3 ± 0.5 | -10 ± 3 | -6 |
| NaSCN dodecanol/water[9] | SL | -4.5 ± 0.9 | -3.2 ± 1.4 | -3.2 ± 1.3 | -3.9 |
| | LSP | | -6.7 ± 1.5 | N/A | N/A |
| NaSCN < 4M dodecanol/water[9] | SL | -6.7 ± 1.1 | -6.2 ± 1.3 | -6.2 ± 1.5 | -7.7 |
| | LSP | | -5 ± 6 | -5 ± 3 | -9 |
| KSCN dodecanol/water[9] | SL | N/A | -3.5 ± 0.6 | -3.5 ± 0.3 | -3.2 |
| | LSP | | -7 ± 3 | -7 ± 1 | -8 |
| KSCN < 4M dodecanol/water[9] | SL | -6.3 ± 1.8 | -6 ± 3 | -5.5 ± 1.4 | -6 |
| | LSP | | -7 ± 10 | N/A | N/A |
| NaSCN air/water[10] | SL | -7.53 ± 0.13 | -7.6 ± 0.4 | -7.6 ± 3 | -7.6 |
| | LSP | | -7 ± 3 | -7 ± 3 | -5 |

Table 5.4: The collected $\Delta G_{ads}$ values for the datasets. All units are in kJ/mol. SL: simple Langmuir model. LSP: Langmuir with surface potential. The reported values are included for comparison. The three calculated values are from the least squares fit, the jackknife calculation, and the bias-corrected jackknife calculation. For the NaSCN dodecanol/water dataset and the KSCN < 4M dodecanol/water dataset, the LSP subfits had trouble converging and the subsequent calculations were not reasonable.

## 5.3 Error Analysis

In Chapter 4, we found that the difference in free energy between the graphene/water interface and the air/water interface was statistically significant, but not thermodynamically significant. In this case, getting accurate error bars on the parameters was not crucial. It is easy to imagine a situation where the difference *is* thermodynamically significant, such as if graphene was indeed behaving like a metal. In that case, it is important to know the parameter errors accurately. Given the difficulties that arose during the model comparison calculations, it is also worth discussing the errors generated from the nonlinear fitting process.

### *5.3.1 Nonlinear regression and the covariance matrix*

Nonlinear regression programs generally employ the least squares approach. There are four main assumptions in this approach:[6]

- There is no error in X, only in Y.
- The error in Y follows a known distribution.
- In unweighted nonlinear regression, Y has uniform variance. In weighted nonlinear regression, the variance is described by a known relation, such as $1/\sigma^2$.
- The Y values are independent.

The parameter errors are calculated from the covariance matrix, which is the inverse of the Hessian matrix.[12] The Hessian is approximated with the Jacobian matrix: $\mathcal{H} = J^T J$, setting the second order derivative term to zero. This approximation is valid when there are many data points, little scatter, and the points define the curve well. Otherwise, the error tends to be

underestimated when calculated in this way.[6] In the literature, reported errors are usually calculated in this way.

### *5.3.2 The jackknife estimator*

One tool in the statistician's belt is the jackknife estimator. It is a nonparametric method that allows the calculation of variance when a sum of squares is not easily defined.[13] If the errors generated from the covariance matrix can be thought of as a lower bound on the true error, then the jackknife can be thought of as an upper bound. More importantly, the jackknife can remove bias (i.e. the difference between the estimated parameter and the true parameter) of the order $1/n$, where $n$ is the total number of data points.[14]

The algorithm works by systematically removing points from the dataset and estimating the parameter with this subset. Let $\hat{\theta}_{(-i)}$ be the parameter estimates with point $i$ removed, then the jackknife estimate of the parameters, $\hat{\theta}_{(.)}$ , is the average of these estimates:

$$\hat{\theta}_{(.)} = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta}_{(-i)} \,. \tag{5.19}$$

The bias corrected estimate, $\hat{\theta}_{corr}$, is

$$\hat{\theta}_{corr} = n\hat{\theta} - (n-1)\hat{\theta}_{(.)}, \tag{5.20}$$

where $\hat{\theta}$ are the estimates for the full dataset.

The variance, $\widehat{var}$, is calculated as follows:

$$\widehat{var} = \frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_{(-i)} - \hat{\theta}_{(.)}\right)^2. \tag{5.21}$$

For the NaSCN air/water dataset, the estimated parameter values for the full dataset were used as the initial values for the subsets. For the other datasets, the initial values for the subsets were the same as used for the full datasets. If any subsets generated erroneous fits, the fits were recalculated using the estimated parameter values for the full dataset as initial values.

The values calculated for $\Delta G_{ads}$ are summarized in Table 5.4. For the NaSCN dodecanol/water dataset and the KSCN < 4M dodecanol/water dataset, the LSP subfits had trouble converging and the subsequent calculations were not reasonable. Considering the size of the error for the KSCN < 4M dodecanol/water dataset, this is understandable. One curiosity is that most of the dodecanol/water results have smaller errors for the jackknife calculation. The reason for this is not apparent and further analysis will be needed to ascertain it. Another curiosity is that using the Langmuir model with surface potential for the temperature dependent dataset yields $\Delta G_{ads}$ that are not monotonic with temperature, no matter which

result is examined. However, the model comparison conclusively supported the simple Langmuir for this dataset.

The jackknife estimates, $\hat{\theta}_{(.)}$, agree with the values from the fitting procedure, which supports the validity of the jackknife calculation. In general, the bias-corrected results also agree, which proves the validity of the fitting procedure. Some notable exceptions are the SL model for the graphene/water dataset and both models for the NaSCN < 4M dodecanol/water dataset. This further supports the conclusion that neither model was able to fit these datasets very well.

## 5.4 Conclusions and Future Directions

Two models for ion adsorption were compared: the simple Langmuir model (Equation (2.18)) and the Langmuir model with surface potential (Equation (5.11)). The results generally indicate that the simple Langmuir is sufficient, but there are situations where the Langmuir model with surface potential is more appropriate. Datasets that have "salient kinks," like the dodecanol/water datasets[9] or the graphene/water dataset, are potentially described better by the Langmuir model with surface potential. However, without significant improvements in the data, the error analysis indicates that the model will overfit the data in most cases.

A possibility for calculating more accurate parameter errors would be to use the F-test for model comparison as above, except that the alternate hypothesis is a different set of parameter values.[6] The critical F value for the desired significance level is easily looked up. Then parameter values are varied to find the contour of values that make up the significance level boundary. The subsequent interval is then the maximum and minimum values of the parameter on this contour. The dodecanol/water datasets seem like good candidates for this, as does the graphene/water dataset. See Reference 6 for a more in-depth explanation of the algorithm.

A possibility for improving the fit, when necessary, would be to treat the model with a full maximum likelihood treatment. Under the assumptions of nonlinear fitting, maximizing the likelihood function is the same as minimizing the sum of squared errors.[12] The bias-corrected values for the SL model of the graphene/water dataset and both models for the NaSCN < 4M dodecanol/water dataset indicate that the least squares algorithm did not find the most likely parameter estimates. Maximizing the likelihood function directly might bypass this. The algorithm seems straightforward: maximize the joint probability of the data points, given the parameter values. The problem will be to derive a probability distribution function from the Langmuir model.

## 5.5 References


(1) Jungwirth, P.; Tobias, D. J. Specific Ion Effects at the Air/Water Interface. *Chem Rev* **2005**, *106* (4), 1259–1281.

(2) Hess, L. H.; Hauf, M. V.; Seifert, M.; Speck, F.; Seyller, T.; Stutzmann, M.; Sharp, I. D.; Garrido, J. A. High-Transconductance Graphene Solution-Gated Field Effect Transistors. *Appl. Phys. Lett.* **2011**, *99* (3), 33503.

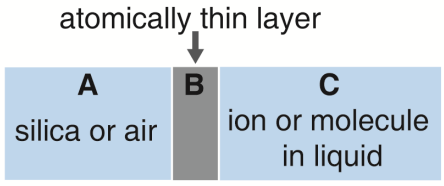
(3) Ong, S.; Zhao, X.; Eisenthal, K. B. Polarization of Water Molecules at a Charged Interface: Second Harmonic Studies of the Silica/Water Interface. *Chem. Phys. Lett.* **1992**, *191* (3–4), 327–335.
(4) Hayes, P. L.; Malin, J. N.; Jordan, D. S.; Geiger, F. M. Get Charged up: Nonlinear Optical Voltammetry for Quantifying the Thermodynamics and Electrostatics of Metal Cations at Aqueous/Oxide Interfaces. *Chem. Phys. Lett.* **2010**, *499* (4–6), 183–192.
(5) Jena, K. C.; Covert, P. A.; Hore, D. K. The Effect of Salt on the Water Structure at a Charged Solid Surface: Differentiating Second- and Third-Order Nonlinear Contributions. *J. Phys. Chem. Lett.* **2011**, *2* (9), 1056–1061.
(6) Motulsky, H.; Christopoulos, A. *Fitting Models to Biological Data Using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting*; Oxford University Press: Oxford ; New York, 2004.
(7) Burnham, K. P.; Anderson, D. R.; Burnham, K. P. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, 2nd ed.; Springer: New York, 2002.
(8) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.
(9) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.
(10) Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. Enhanced Concentration of Polarizable Anions at the Liquid Water Surface: SHG Spectroscopy and MD Simulations of Sodium Thiocyanide. *J Phys Chem B* **2005**, *109* (21), 10915–10921.
(11) Otten, D. UV Second-Harmonic Studies of Concentrated Aqueous Electrolyte Interfaces. PhD. Dissertation, University of California, Berkeley, 2010.
(12) Monahan, J. F. *Numerical Methods of Statistics*, 2nd ed.; Cambridge series in statistical and probabilistic mathematics; Cambridge University Press: Cambridge ; New York, 2011.
(13) Efron, B. *The Jackknife, the Bootstrap, and Other Resampling Plans*; Society for Industrial and Applied Mathematics: Philadelphia, Pa., 1982.
(14) Manly, B. F. J. *Randomization, Bootstrap, and Monte Carlo Methods in Biology*, 3rd ed.; Chapman & Hall/ CRC: Boca Raton, FL, 2007.

# Chapter 6 – Conclusions and Future Directions

Chapters 3-5 each provided some suggestions for future work directly related to the projects, but there are a few directions that overlap between the projects which warrant further discussion.

## 6.1 Thin, uncharged monolayers

In Chapter 4, we saw that the free energy of adsorption of thiocyanate to the graphene/water interface was similar to that for the air/water interface, masking an adsorption mechanism that differed significantly. This result is similar to a previous study from our group involving dodecanol.[1] When a monolayer of dodecanol was added to the surface of NaSCN and KSCN solutions, the extracted free energies of adsorption were also similar to those for the air/water interface. Furthermore, the Geiger group has done several studies of the silica/graphene/liquid interface where the liquids are aqueous NaCl,[2] aqueous $MgCl_2$,[3] and hexanol in cyclohexane.[4] In all three studies, the free energy of adsorption did not change much when the graphene was added. Considering the range of values for adsorption free energy, the differences when adding graphene in the above systems are small in comparison. The different results are tabulated in Figure 6.1. What is curious is that the systems still show mechanistic changes, like for the graphene/water interface. Dodecanol can participate in the hydrogen bonding network,[5,6] affecting the enthalpy. Graphene screened silica in the NaCl study,[2] affecting enthalpy as well. As seen for the graphene/water interface, the changes balance out so that the free energy of adsorption remains similar. We postulate that adding an atomically thin, uncharged layer to an interface does not significantly affect equilibrium adsorption behavior, regardless of the mechanism. If toluene actually forms a monolayer when dropped onto the surface, then the results of Chapter 3 also support this. The enthalpy and entropy would also be affected by $\pi$-hydrogen bonding.[7]




| A | B | C | $\Delta G_{ads}$ (kJ/mol) |
|---|---|---|---|
| air | graphene[a] | $SCN^-_{(aq)}$ | - 6.9 ± 0.4 |
| | docecanol[b] | | - 6.7 ± 1.1 |
| | no material[c] | | - 6.78 ± 0.03 |
| silica | graphene[d] | $Cl^-_{(aq)}$ | - 33.4 ± 0.3 |
| | no material[d] | | - 31 ± 1 |
| silica | graphene[e] | $Mg^{2+}_{(aq)}$ | - 30.4 ~ - 31.1 |
| | no material[e] | | - 30 ~ - 30.5 |
| silica | graphene[f] | hexanol in cyclohexane | - 20.0 ± 0.3 |
| | no material[f] | | - 20.9 ± 0.3 |

Figure 6.1: A summary of systems that show little change in the free energy of adsorption with and without an inserted atomically thin layer. a) Chapter 4. b) Reference 1. c) Reference 8. d) Reference 2. e) Reference 3. See the reference for a detailed explanation of $\Delta G_{ads}$. f) Reference 4.

This requires a reinterpretation of experiments that use substrates to support the monolayers, such as windows or prisms (as in attenuated total reflection experiments). For example, Tian and Shen measured the adsorption of chloride to the octadecyltrichlorosilane/water interface and found $\Delta G_{ads}$ = -39.5 kJ/mol (no error reported).[9] Considering that chloride is not enhanced or depleted at the air/water interface,[10] it is surprising that a hydrophobic interface would cause such a large increase. Vazdar et al. predicted that the surface excess for chloride at a hydrophobic interface would only be 0.06 ions/nm$^2$.[11] However, the monolayer of octadecyltrichlorosilane was supported by a fused silica window. Achtyl *et al.* demonstrated that a layer of graphene was not sufficient to completely screen the silica surface charge.[2] It is possible that the silica is dictating the interfacial behavior, not the octadecyltrichlorosilane, but this is hard to say with certainty, since Tian and Shen did not measure the bare silica/water interface and the free energy cannot be compared to other experiments.

More work will have to be done to verify this hypothesis. For one thing, the toluene results need to be refined. More interfaces will also have to be studied. It would be interesting to repeat the studies with silica, but changing phase A to air, or to a different window material, such as $CaF_2$.

## 6.2 Surface potential

Related to alkane monolayers, there is a phenomenon called surface freezing that is known to happen with liquid alkanes.[12] At a temperature a few degrees above the bulk freezing temperature, the surface layer will actually form an ordered, crystalline monolayer. A similar transition can be induced in alkane lenses on water by adding a surfactant; the surfactant creates a mixed monolayer with the alkane, forming an ordered monolayer.[13–18] The most common surfactants used are alkyltrimethylammonium bromides ($C_n$TAB). Surface freezing has been demonstrated for $C_{16}$TAB with $C_{11}$-$C_{20}$ (where $C_n$ is an alkane),[13–16] $C_{14}$TAB with $C_{14}$ and $C_{16}$,[16] $C_{12}$TAB with $C_{16}$,[18] and $C_{16}$TAB, $C_{14}$TAB, and $C_{12}$TAB with dodecanol.[17]

These surfactants are charged, so varying their concentration in the monolayer varies the surface potential. In light of the results in Chapter 5, it seems prudent to study the surface potential involved in the adsorption process more thoroughly. By choosing an appropriate surfactant/alkane combination, the surface potential can be varied systematically. Matsubara et al. reviewed several combinations and provide a helpful summary.[19] These experiments would also compliment work done by the Netz group to examine how surface charge affects the ordering of the Hofmeister series.[20–22]

## 6.3 References


(1) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.

(2) Achtyl, J. L.; Vlassiouk, I. V.; Fulvio, P. F.; Mahurin, S. M.; Dai, S.; Geiger, F. M. Free Energy Relationships in the Electrical Double Layer over Single-Layer Graphene. *J. Am. Chem. Soc.* **2013**, *135* (3), 979–981.

(3) Achtyl, J. L.; Vlassiouk, I. V.; Surwade, S. P.; Fulvio, P. F.; Dai, S.; Geiger, F. M. Interaction of Magnesium Ions with Pristine Single-Layer and Defected Graphene/Water Interfaces Studied by Second Harmonic Generation. *J. Phys. Chem. B* **2014**, *118* (28), 7739–7749.
(4) Achtyl, J. L.; Vlassiouk, I. V.; Dai, S.; Geiger, F. Interactions of Organic Solvents at Graphene/α-$Al_2O_3$ and Graphene Oxide/α-$Al_2O_3$ Interfaces Studied by Sum Frequency Generation. *J. Phys. Chem. C* **2014**, *118* (31), 17745–17755.
(5) Legrand, J. F.; Renault, A.; Konovalov, O.; Chevigny, E.; Als-Nielsen, J.; Grübel, G.; Berge, B. X-Ray Grazing Incidence Studies of the 2D Crystallization of Monolayers of 1-Alcohols at the Air-Water Interface. *Thin Solid Films* **1994**, *248* (1), 95–99.
(6) Can, S. Z.; Mago, D. D.; Esenturk, O.; Walker, R. A. Balancing Hydrophobic and Hydrophilic Forces at the Water/Vapor Interface: Surface Structure of Soluble Alcohol Monolayers †. *J. Phys. Chem. C* **2007**, *111* (25), 8739–8748.
(7) Gierszal, K. P.; Davis, J. G.; Hands, M. D.; Wilcox, D. S.; Slipchenko, L. V.; Ben-Amotz, D. π-Hydrogen Bonding in Liquid Water. *J. Phys. Chem. Lett.* **2011**, *2* (22), 2930–2933.
(8) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.
(9) Tian, C. S.; Shen, Y. R. Structure and Charging of Hydrophobic Material/Water Interfaces Studied by Phase-Sensitive Sum-Frequency Vibrational Spectroscopy. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15148–15153.
(10) Jungwirth, P.; Tobias, D. J. Ions at the Air/Water Interface. *J Phys Chem B* **2002**, *106* (25), 6361–6373.
(11) Vazdar, M.; Pluhařová, E.; Mason, P. E.; Vácha, R.; Jungwirth, P. Ions at Hydrophobic Aqueous Interfaces: Molecular Dynamics with Effective Polarization. *J. Phys. Chem. Lett.* **2012**, *3* (15), 2087–2091.
(12) Ocko, B. M.; Wu, X. Z.; Sirota, E. B.; Sinha, S. K.; Gang, O.; Deutsch, M. Surface Freezing in Chain Molecules: Normal Alkanes. *Phys. Rev. E* **1997**, *55* (3), 3164.
(13) Wilkinson, K. M.; Qunfang, L.; Bain, C. D. Freezing Transitions in Mixed Surfactant/Alkane Monolayers at the Air–solution Interface. *Soft Matter* **2006**, *2* (1), 66.
(14) Sloutskin, E.; Sapir, Z.; Bain, C.; Lei, Q.; Wilkinson, K.; Tamam, L.; Deutsch, M.; Ocko, B. Wetting, Mixing, and Phase Transitions in Langmuir-Gibbs Films. *Phys. Rev. Lett.* **2007**, *99* (13), 136102.
(15) Lei, Q.; Bain, C. D. Surfactant-Induced Surface Freezing at the Alkane-Water Interface. *Phys. Rev. Lett.* **2004**, *92* (17).
(16) Ohtomi, E.; Takiue, T.; Aratono, M.; Matsubara, H. Freezing Transition of Wetting Film of Tetradecane on Tetradecyltrimethylammonium Bromide Solutions. *Colloid Polym. Sci.* **2010**, *288* (12–13), 1333–1339.
(17) Casson, B. D.; Bain, C. D. Phase Transitions in Mixed Monolayers of Cationic Surfactants and Dodecanol at the Air/Water Interface. *J. Phys. Chem. B* **1999**, *103* (22), 4678–4686.
(18) Matsubara, H.; Ohtomi, E.; Aratono, M.; Bain, C. D. Wetting and Freezing of Hexadecane on an Aqueous Surfactant Solution: Triple Point in a 2-D Film. *J. Phys. Chem. B* **2008**, *112* (37), 11664–11668.

(19) Matsubara, H.; Ushijima, B.; Law, B. M.; Takiue, T.; Aratono, M. Line Tension of Alkane Lenses on Aqueous Surfactant Solutions at Phase Transitions of Coexisting Interfaces. *Adv. Colloid Interface Sci.* **2014**, *206*, 186–194.
(20) Schwierz, N.; Horinek, D.; Netz, R. R. Reversed Anionic Hofmeister Series: The Interplay of Surface Charge and Surface Polarity. *Langmuir* **2010**, *26* (10), 7370–7379.
(21) Schwierz, N.; Horinek, D.; Netz, R. R. Anionic and Cationic Hofmeister Effects on Hydrophobic and Hydrophilic Surfaces. *Langmuir* **2013**, *29* (8), 2602–2614.
(22) Schwierz, N.; Horinek, D.; Netz, R. R. Specific Ion Binding to Carboxylic Surface Groups and the pH Dependence of the Hofmeister Series. *Langmuir* **2014**, 141226104330008.

# Appendix 1 – Alignment

See the figures at the end for the alignment diagram and for abbreviations.

There is a box that surrounds L1-M6. This is to block air currents around the table. These currents can affect measurements, so make sure it is in place when collecting data. There are also remnants of a purge box around M6, PB, and L3. I use these to block stray beams. I also put a piece of cardboard over the top of the panels above L3 to catch spills from my samples as I put them in the holder. Boxes are represented by dashed lines in the diagram.

**Full Alignment**

The beam shape isn't the best, so choose a bright point in the beam and always align to that. Don't try to align the whole shape. I don't usually have to adjust any of the stands, either. If you feel like you have to adjust stands, the alignment is likely off somewhere else. Try to do the alignment as quickly as possible, too, because the water does evaporate noticeably.

1. Slide the box out of the way (it can be pushed straight back a few inches).
2. Remove filters, M3-M6, L1, L2, and PB. The beam sampler can be rotated out of the way. A collar is helpful for M3.
3. Align M1 and M2. The line of the experiment after M2 is marked on the table. The height is 6".
4. Set the iris.
5. Replace M3. Since this one just goes straight up, it's difficult to be certain of the alignment. There is a notecard taped to the underside of the shelf with a target on it that can be used.
6. Take the sample dish off the magnetic base. Replace M4. M4 should direct the beam to a ~60 degree angle. If the post hasn't been adjusted, then the card taped to the table on the far right should achieve the desired angle. Now the trick is to align the beam horizontally. I have a target on a post (made of a 1.5", 1", and 12" post) that I use to do this. I've marked a screw hole to the left of the sample holder where the height works out. Unfortunately, the target doesn't screw tightly, so you have to judge the straightness by eye. If you are having a lot of trouble, adjust M3 a bit.
7. Replace the sample dish and put a water sample in it. Adjust the height so that your chosen beam spot is in the center of the dish and on the water surface. Dipping a notecard vertically into the water can help with this.
8. At this point, the water level is set. If anything happens to change it, the height needs to be realigned. If M6 is in, you can align the height there.
9. Replace M5. There is a target below this mirror to aim for. Use the post target to double check horizontal alignment going to M5. There is another screw hole to the right of the sample that is marked.
10. Replace M6. **Be careful! This mirror can end up pointed at your face.** At this point, I find it helpful to remove the panels. The new line for the beam is marked again. The new height is 4". If this mirror is hard to align, try adjusting M5 a little bit. It is possible to see the beam through the alignment card, so I usually use the card facing me.

11. Replace L1. It is mounted in a lens tube with an iris on it. In my experience, the beam can still be used for alignment after M6, even though it's diverging. Between that and the iris, you can align the lens. Watch out for the back reflection on this one. There is no anti-reflective coating on the lens, so there's a large back reflection.
12. Replace L2. This is also in a lens tube. Align the lens and then check the collimation. If the collimation needs adjusting, turn the tube in the mount. This is much easier than adjusting the post.
13. Move the beam sampler back into place and rotate to direct it onto the photodiode.
14. At this point, you are ready to put the Pellin Broca prism in. Since this has to be done daily, anyway, see the Daily Alignment section.
15. Final checks:
    a. Make sure the focus of L1 is near the center of the stage. Turn the tube in the mount to adjust the focus, if needed.
    b. Take the sample holder off and check the alignment on the far right card. It probably won't be exactly on target, but remember where it is. Outline it on the card, if you want.

**Daily Alignment**

1. Move the box back and remove the sample holder.
2. Align to the iris and the card on the table to the far right.
3. Replace the sample holder and put in a water sample.
4. Check the height after M6. Adjust M1 and M2 more if needed.
5. If the prism is not in place yet:
    a. Make sure that the PMT is off.
    b. Remove L3 if it is there.
    c. Make sure the front and right panels are in place.
    d. Replace PB. Adjust it so that the fundamental is normal. If the prism was removed while the SHG was normal, turn it in the base until the fundamental is at ~10mm from center when the alignment card is 4" away. Then use the knob on the rotation stage to make the fundamental normal.
    e. Replace L3. This is also in a beam tube with an iris, so you can use the iris to assist in alignment.
    f. Turn the fundamental past ~10mm on the card 4" away.
    g. Turn the PMT on and start the LabView acquisition program.
    h. Switch to a thiocyanate sample if available.
    i. Turn the knob back towards normal while watching the signal in LabView. Adjust the knob for maximum signal.
6. Adjust the boxcar for the signal and reference. I adjust the delay so that the signal is slightly after the box. I try to adjust the width so that the main part of the box gets the positive signal and the box shoulder gets the negative signal. Reference sensitivity is always 0.1V/V and signal sensitivity is always 0.2V/V. Check my lab notebooks for typical values. Adjust the baselines to be near zero. The reference baseline tends to drift downward throughout the day.
7. 7. I adjust the power supply on the table to 0.97V.

**Notes on the electronics**

Turn on order:

1. Power supply below table, boxcar, DAQ card
2. Photodiode, power supply on table, amplifier box, oscilloscope

Reverse for shut down

Power supply below table setup:
This power supply runs the custom box for the amplifier. The PMT also used to be connected to the box, but we don't use that connection anymore. There is a special wire that connects the two. It has a DB9 connector on one end that plugs into the box. The other end has five wires that need to be plugged into the power supply. Left to right, the wires are red, black (either), uncovered, black, white. The voltage between red and uncovered needs to be 15.5V. The voltage between white and uncovered needs to be -5V. There is a cheat sheet on top of the power supply.

Power supply on table setup:
This power supply runs the PMT. It also has a special wire. One end is a DB9 connector that plugs into the PMT and the other end has three wires that connect to the power supply. From left to right, the wires are yellow, black, red, none, none. Adjust the 20V channel to 15V. The 6V channel is a variable voltage. It can be set between 0.2-1.2V. This spec is for the PMT, not the power supply. The power supply can do 6V, but you should not send that to power the PMT. I find that 0.97V works best for this experiment. There is also a cheat sheet on the power supply.

**Parts:**

M1: Edmund Optics 84-621 360-440nm Ultrafast dielectric. (needed to separate TOPAS output)
M2: CVI PAUV-PM-1025-C
M3: CVI Al PW1-1037-C (discontinued, their aluminum mirrors should work)
M4: CVI Al PW1-1037-C
L1: f = 100mm at 400nm, fused silica (don't know much else, but it works)
L2: CVI PLCX-25.4-46.4-CFUV, f = 100mm
M5: CVI DUVA-PM-1025-UV (these wear out quickly)
M6: CVI DUVA-PM-1025-UV
PB: Thorlabs ADBV-10 $CaF_2$
L3: CVI PLCX-25.4-46.4-CFUV, f = 150mm
Grating: Richardson Gratings 53*-120R 1200gr/mm 150nm blaze
PMT: Hamamatsu R7154 PHA
In general, anti-reflective coatings kill the signal, so I don't use them on my optics.

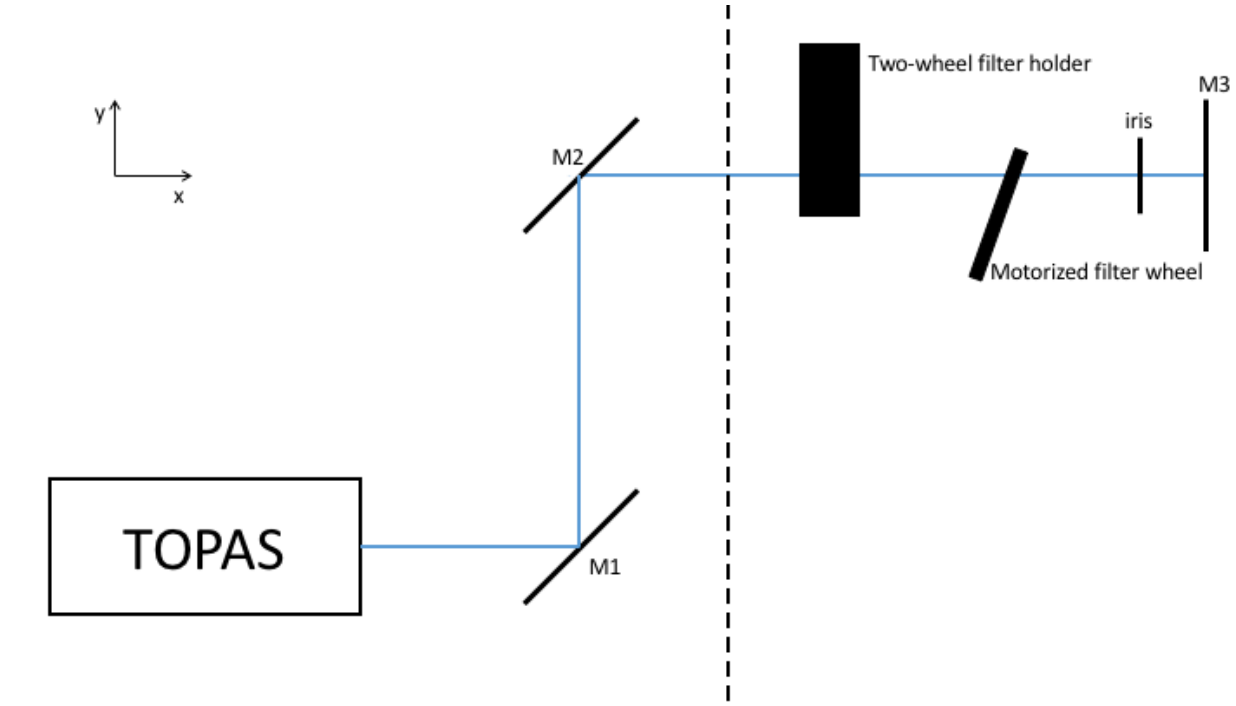



*Figure A2.1: The first part of the optical setup. This view is in the plane of the laser table.*

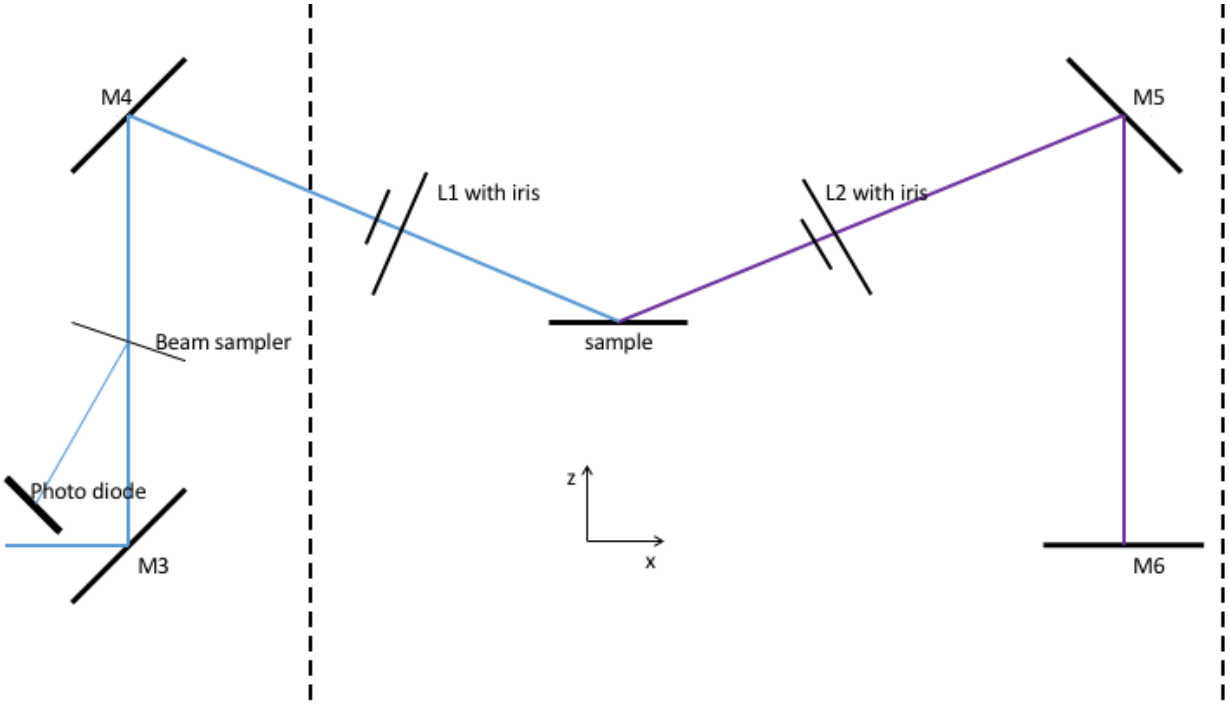



*Figure A2.2: This view is perpendicular to the table and parallel to the fume hood. M3 directs the beam up by 90 degrees.*

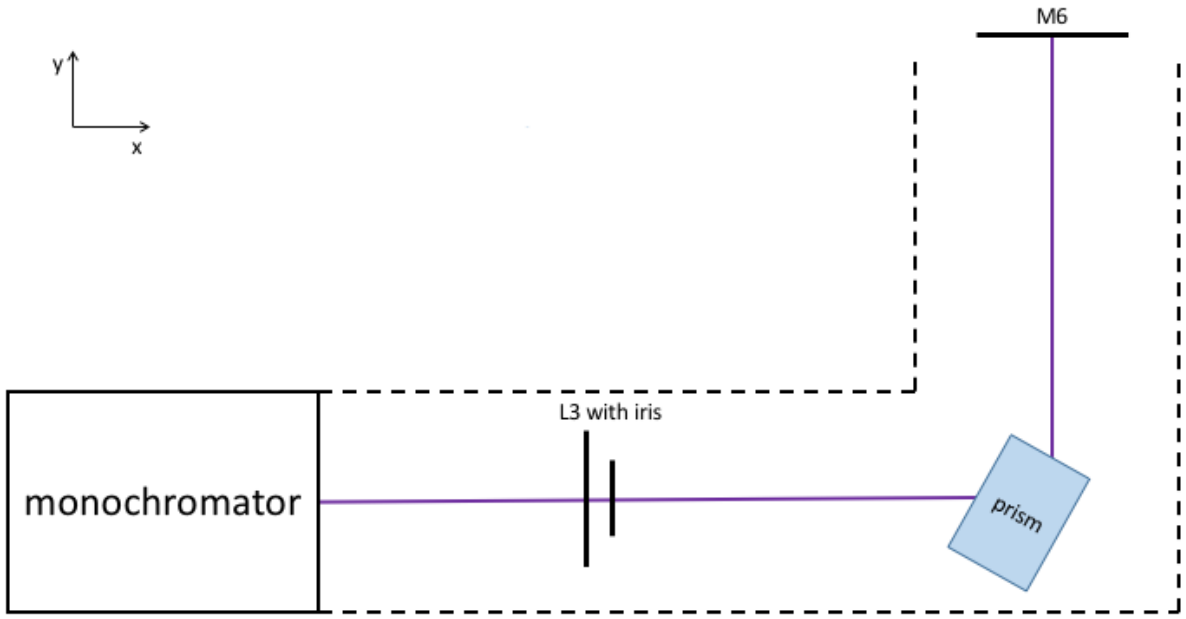



*Figure A2.3: This view is in the plane of the table again. M6 directs the beam towards the hood, essentially acting as a periscope.*

# Appendix 2 – Sample Preparation

**For solutions**

**Day 0 (can be left out)**

1. Put a 100x50mm petri dish (or several if a lot of salt is needed) in NoChromix. Let soak overnight.

**Day 1**

If Day 0 left out:

1. Put a 100x50mm petri dish (or several if a lot of salt is needed) in piranha (~1 part $H_2O_2$: 3 parts $H_2SO_4$). Let soak for ~1 hour.

Both:

2. Rinse dish vigorously with Millipore water.
3. Place dish in oven to dry (any setting 4 or above). Takes 10-30 mins.
4. Remove dish from oven.
5. As soon as you are able to handle it, pour NaSCN solid into the dish. Weigh if desired. You just need enough for the solutions, plus ~5g more for water loss. As long as there is at least that much, mass doesn't matter.
6. Put dish back in oven (I use the top rack so nothing falls in it). Bake overnight at maximum setting.
7. Put in NoChromix:
    a. 1 **amber** flask per concentration (no stopper yet)
    b. 1 flask to hold Millipore water (200mL is best)
    c. spatulas (a 50mL graduated cylinder works for containing them)
    d. 1 sample dish/lid pair
    e. 1 4" glass disk per salt dish (used as a lid)
    f. a dish to dry the disks in (125x65mm works well)

**Day 2**

1. Put 1 stopper per flask in NoChromix for ~30 mins. The stoppers are hollow with thin glass on top, and porous on the bottom. We don't want to etch them too much.
2. Rinse all pieces with Millipore water. Stopper the flasks once they are rinsed.
3. Dry in oven at maximum (~10 mins):
    a. Spatulas in GC. Use a small dish to cover it (there is a spout for air to escape).
    b. Put the disk in the larger dish and prop up with the other small dish (dries faster and prevents the disk from getting stuck to the dish).
4. If possible (i.e. nobody else is using the oven), turn the oven down to 4 and wait ~20mins. Glassware will be easier to handle at this temperature.
5. Take out of the oven:
    a. Salt
    b. Disk in dish
    c. Spatulas in GC

    d. Cover the salt dish with the disk, with the side that was facing the drying dish now touching the salt dish.
6. Let cool for 25 mins. May take longer if oven was on maximum. You want the salt to be cool enough to get a proper mass measurement, but you want it to be warm enough that it doesn't stick to surfaces. This means it's starting to accumulate water, which could affect the mass measurement.
7. NaSCN likes to clump and is hard to unclump. The easiest way to get it massed and into a flask is to use a weight boat, add Millipore water to the boat, and pour into the flask.
    a. Use the clean water flask and disposable pipets to add Millipore water.
8. NaSCN and KSCN dissolving are both endothermic and slowish. I fill the flask a bit more (not to neck) and put it in the bottom cabinet to the left of the sink while I clean up from measuring the salt.
9. Once the salt is fully dissolved, I fill the flask to the line. You can use the Millipore spout directly, but I'm not very skilled at that. I pipet water from the flask again to fill to the line.
10. SCN doesn't like to mix, so you have to invert the flask 10 times to make sure it is fully mixed. It will not spontaneously mix between now and the experiment. There are visible swirls that appear when the solution is not fully mixed.
11. Vent the flask quickly and store it in the cabinet under the hood overnight. Empirically, I get better results if I let the solution sit overnight.
12. Put in NoChromix:
    a. 1 sample dish/lid pair per solution
    b. 1 sample dish/lid pair for water
    c. extra sample dishes and lids, as many as possible
    d. A 170x90mm dish works well for this. Then this dish is also clean for storage the next morning.
    e. A 190x100mm dish to cover the 170x90mm dish. I usually only fill the bottom with 1-2cm of NoChromix. The sides don't matter as much as the surface that will face the clean dishes.

**Day 3 – Data day**

1. Rinse all the glassware with Millipore water. Use the large petri dishes for covered storage.
2. When taking aliquots for measurement:
    a. Take water aliquots straight from the Millipore system. Rinse three times, then get the aliquot. I find it helps to dry off the bottom of the dish before taking it over to the sample holder.
    b. Use a disposable pipet to take aliquots out of flasks. Draw up to -1mL on the pipet. Rinse the dish three times with ~1mL each. Dispense the rest in the dish. Draw again to 0mL and dispense. This puts ~18mL in the dish.
    c. Always keep dishes that are not in use covered.

**For hydrocarbons and alcohols**

Add these steps to the appropriate solution days:

**Day 1**

1. Put in NoChromix:
    a. Amber wide mouth bottle (if a new bottle is needed; sometimes I just pour new silica in an old bottle)
    b. Two small petri dishes (bottom/lid pair works)
2. Put in saturated KOH in ethanol:
    a. Casserole dish
    b. Lid for the amber bottle

**Day 2**

1. Rinse the items with Millipore water. I suggest rinsing all the items in base first, cleaning up from the base, then rinsing items in the acid.
2. Put the casserole, bottle and one petri dish in the oven to dry.
3. Shake as much water off the bottle lid as possible, then place in hood to dry the rest of the way. Put it face up and cover as much as possible with the other petri dish while leaving space to vent.
    a. The Teflon pad in the lid will warp in heat, so it can't be put in the oven.
4. Take the casserole out and fill it about half way up with silica. That should be plenty. Cover with foil, leaving the spout uncovered.
5. The silica needs to be baked at 500C, which requires a furnace. The upper class labs have one. Talk to Dante Valdez (306 Latimer, dvaldez@berkeley.edu) to get access and to find out when you can use it. For using this furnace:
    a. Take the casserole with silica and a clay triangle up to the lab.
    b. Place the clay triangle on the bottom of the furnace and put the casserole on the triangle. I find that the casserole fits best along the diagonal with the handle pointing out. Try to arrange it so that it doesn't touch the walls. **Definitely** do not touch the thermocouple in the back corner.
    c. I like to stay in the room while the temperature heats up, just to make sure nothing goes wrong. I also check on it periodically while it's heating.
6. Bake the silica for at least 2 hours.
    a. Cooling takes at least 2 hours, so make sure to account for this!
    b. When cooling, I just crack the door a little bit.
7. Before the casserole is finished cooling, take the bottle out to cool. Cover with the petri dish.
8. One everything is cool, pour the liquid into the bottle, then pour silica.
    a. Silica makes a mess.
    b. I usually pour in about 1/4 - 1/3 of the liquid volume of silica.
9. The silica degasses for a bit, so I usually like to leave the bottle in the hood with the cap on loosely until it stops degassing.
10. Store the bottle in the flammables cabinet.

**Day 3 – Data day**

1. Use a pipet (I suggest the Eppendorf Repeater M4) to dispense the liquid. You can take it straight from the bottle, so long as you avoid the silica.

a. I tried 1-2μL for hexane, decane, and hexadecane.
b. I tried 10-200μL for toluene.
c. If the drop doesn't come off the tip, carefully touch the drop to the solution without touching the tip to the solution.

2. Give the surface a minute to form, then take data as normal.
    a. With large amounts of toluene, I would see a dramatic increase in signal after a certain point, so I would try to capture both regions in the measurement.

**For graphene**

This experiment is done with Son Nguyen's help. Contact him to schedule the experiment, ngchison@gmail.com. This experiment also takes up a lot of space and uses a whole bottle of NoChromix. It is easier if you are the only one working in the hood that week. You will likely have to clean as much as possible every night to make sure everything is clean. Instead of stepping through when to do things, I will just tell you what additional glassware needs to be cleaned by when (solution making glassware is not included). Try to do as much of it the night before, if possible. Make sure that there are plenty of covers as well. You can cover the bottoms of all the dishes in a little NoChromix to achieve this. Solutions need to be 200mL.

**Day 2**

- 4 100x50mm Petri dishes
- graphene scoops (they are custom pieces with flow inlets and outlets)
- 4 1.5" teflon stir bars

Son will come in the morning to start the graphene etching. He will come back in the afternoon to transfer the graphene pieces to water several times to clean it. He may use the stir bars at this point. After his last transfer, he will leave the graphene covered in the hood.

**Day 3 – Data day**

- 1 100x50mm Petri dish per solution
- 1 teflon stir bar per solution
- 1 125x65mm Petri dish
- 1 5" glass disc
- all 4" glass discs
- graphene scoops
- the pieces of Teflon to put in the sample holder

Son will come in the morning to transfer the graphene one last time into the 125x65mm dish. Solutions are used from lowest concentration to highest. Put a stir bar in one of the 100x50mL Petri dishes and pour the first solution in it. Son will transfer the pieces of graphene from the water to the solution. **Keep track of these transfers.** You will have to calculate the new concentration afterward. There are two scoops, marked 'H' and 'L.' Their volumes are 9.4mL and 6.2mL respectively. 'H' works better for taking measurements, so we usually use this one. NaSCN solutions do not like mixing, so they need to be stirred **gently.** Stir until the visible ripples are gone. Now you can take data with these samples. Son will scoop them out one at a time and place them in the sample holder on top of the clean pieces of Teflon (to keep the scoops clean and to help see the graphene). Adjust the tip/tilt stage with a mirror beforehand

and avoid moving it again. Use the micrometers to adjust the alignment. Son can also nudge the graphene into place. Once it's measured, he will put it back in the solution. When all of the pieces are measured, he will transfer them to the next solution. Then, as a sanity check, I would pipet out samples and measure them without the graphene. Sometimes these samples are contaminated, so take the results with a grain of salt. Repeat this until all measurements are taken.

**Tips**

- A 2L beaker will hold 3 100x50mm Petri dishes and 3 4" glass discs. Place the beaker in another Petri dish to catch spill over though.
- Put smaller items in the Petri dishes being cleaned to save room.
- The scoops are constantly immersed in sample solutions, so only let them touch other clean surfaces.
- Keep track of the samples with the same piece of graphene.

**Notes**

- **PLAN OUT YOUR WEEK BEFORE YOU START!!!!**
- **PPE for working with acid and base:** splash goggles, labcoat made of majority Dacron (a PET fabric made by DuPont), double nitrile gloves.
- Use the large funnel when pouring NoChromix back into the bottle. Pour slowly, because the acid can gurgle back up the spout and spray a bit.
- Use lots of water when rinsing to dilute the acid as you go and to make sure all the NoChromix gets off. Be careful because this makes the glassware slippery.
- Piranha is quicker, but it changes results. Piranha can only be used in place of NoChromix for baking salts.
- A NoChromix bottle can be reused for a week. Piranha cannot be reused.
- NoChromix and piranha can get HOT. Times to be careful are when making piranha and when coming into contact with water (rinsing, pouring after condensation).
- Sulfuric acid is highly hydroscopic. Keep this in mind when cleaning. Cover large surface areas when possible. Try not to reuse NoChromix that has accumulated a lot of water.
- To saturate KOH in ethanol, use a stir bar. It takes 30-60 minutes. I usually try three scoops of KOH first and add more until it stops dissolving. It doesn't need to be analytic, so don't stress over amounts.
    - This can be reused, but I never used it often enough to do that.
    - It will turn a cider color over time. This is normal.
- Oven settings: 4 ~ 120C, 10 ~ 200C
- The only brand of NaSCN that can be baked without breaking down is J. T. Baker NaSCN 98% ACS reagent. The manufacturer is Avantor, so it is probably best to buy from them if you need more.
- Try to bake for 24 hours. If not possible, overnight will suffice.
- Try to soak in NoChromix for 24 hours. If not possible, overnight will suffice.
- All petri dishes except for sample dishes can be purchased in the stock room.

- The sample dishes have to be specifically Pyrex brand to fit in the holder. Other brands don't fit.
- The metal spatulas were hand sanded from rods in the student shop. Hopefully, the three I have will last, but you can make more if you need to.
- **Never** pour from a flask or use the top surface. The remaining impurities accumulate there. Always use a disposable pipet to draw from the bulk.
- Benchmark: 1M NaSCN should have normalized signal ~20.
- Use the amber flasks to prevent photoproducts from forming prematurely. If absolutely necessary, you can cover a clear flask in foil.
- I use NaSCN instead of KSCN like Dale because I've found crystals that turn green upon heating in the KSCN. KSCN also melts at < 200C. Dale did not bake KSCN, just filtered. Baking is necessary though.[1] SFG of these salts showed that filtering is not necessary.
- **NaSCN disposal:** I put extra solid in the blue bin and try to wipe everything down as best as possible. The residue on the glassware can be washed down the sink. The waste solution should go in a waste bottle. **Do not put anything but NaSCN and KSCN in the bottle.** For concentrated solutions, I rinse it with water and put that in the bottle, as well. Final rinses can go down the drain.
- Use Millipore water only for cleaning glassware to be used in the experiment, for solutions, and for water samples.
- When handling and storing the glassware, think about what surfaces will be touching samples. Only put those surfaces in contact with other clean surfaces. For example, in the graphene experiment, the sample dishes are submerged in solution frequently, so the dishes can only be placed on clean surfaces, including in the sample holder.
- Things to know about the pipets:
    - The Eppendorf Repeater M4 is a positive displacement pipet. This means the volume dispensed is the same, no matter the vapor pressure of the liquid being dispensed. It's also designed so that, if you need a different volume range, you buy new tips, not a new pipet. This is the one I've used most recently.
    - The Eppendorf Biomaster is a positive displacement pipet for 1-20μL. Unfortunately, the tips do not come sterile, so they have to be autoclaved. Check the autoclave on 8$^{th}$ floor Latimer for up to date information on how to use it. I do not suggest using this pipet.
    - The BrandTech Transferpette S is an air cushion pipet for 0.1-2.5μL. It has filter tips that can help prevent contamination to the pipet from volatile samples and ultra low retention tips for samples that don't like to come out. It is an air cushion, so the volume dispensed is dependent on vapor pressure.

**Items and part numbers**

- Sample dishes: 08747A from Fisher Scientific
- NoChromix: 328693-10PAK from Sigma Aldrich
- Vented caps for NoChromix: Z683205-3EA from Sigma Aldrich
- Spatula material: "Alloy 20 Stainless Steel Rod1/4" Diameter, ****1/2 ft length****" 9091K27 from McMaster Carr

- Sterile Pipets: 1367827F from Fisher Scientific
- Weigh boats: 13735743 from Fisher Scientific
- Amber flask 100mL: 980-28507-Z39 from Spectrum Chemical and Laboratory Companies
- Amber flask 200mL: 980-28510-Z39 from Spectrum Chemical and Laboratory Companies
- 4” glass lid: 8477K37 from McMaster Carr
- 5” glass lid: 8477K69 from McMaster Carr
- amber bottles: stock room, 1 or 2 oz will do, get wide mouth bottles with the green lids
- Casseroles: porcelain 70mm diameter, S325501B from Fisher Scientific
- Repeater tips: https://online-shop.eppendorf.us/US-en/Laboratory-Consumables-44512/Tips-44513/Eppendorf-Combitips-advanced-PF-18250.html, get whatever size is needed, make sure to get Biopur quality
- Silica, KOH, ethanol: stock room
- Hydrocarbons: Alfa Aesar

(1) Hua, W.; Verreault, D.; Adams, E. M.; Huang, Z.; Allen, H. C. Impact of Salt Purity on Interfacial Water Organization Revealed by Conventional and Heterodyne-Detected Vibrational Sum Frequency Generation Spectroscopy. *J. Phys. Chem. C* **2013**, *117*, 19577–19585.

# Appendix 3 – Data Analysis Procedure

Document has been converted from a Jupyter Notebook to a LaTeXdocument.

```
In [1]: %matplotlib inline
```

```
In [2]: import numpy
        import pandas
        import matplotlib.pyplot as plt
        import lmfit

        import fits
        import models
        import datatools
```

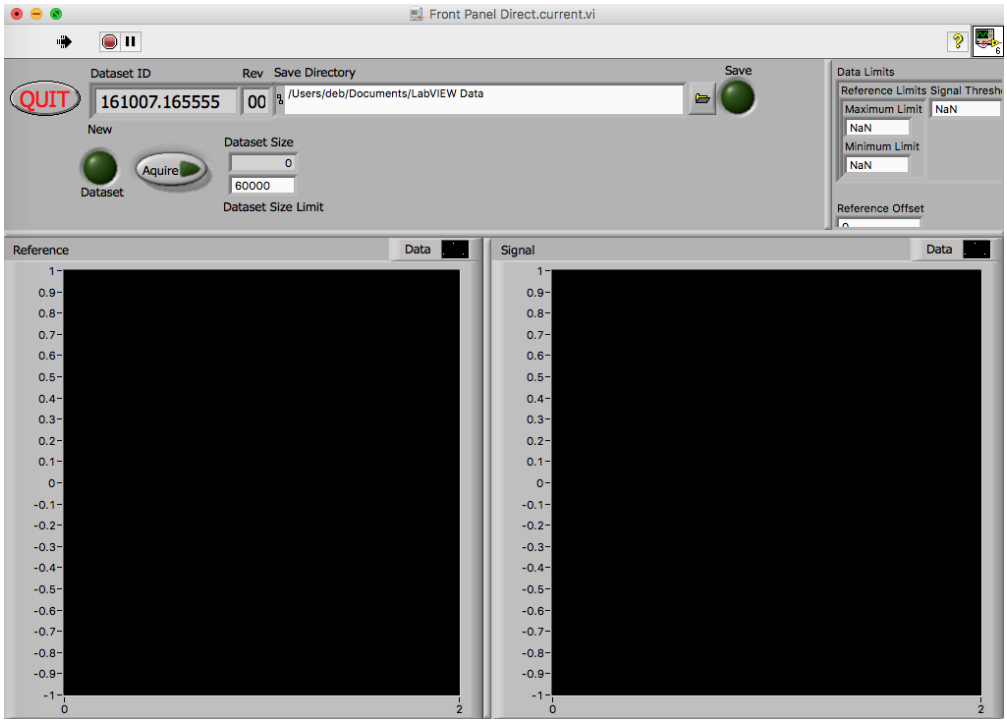
## 1 Data collection

Solutions are made in 100mL and 200mL flasks, then measured in ~21mL aliquots. This gives 4 aliquots for a 100mL flask and 8-9 aliquots for a 200mL flask. The last aliquot taken should be met with skepticism. Empirically, samples taken from the bottom of the flask tend to have more impurities. Each aliquot should have a water sample measured before and after it. A typical routine for a 100mL flask of 1M NaSCN is:

- water
- 1M NaSCN
- water
- 1M NaSCN
- water
- 1M NaSCN
- water
- 1M NaSCN
- water

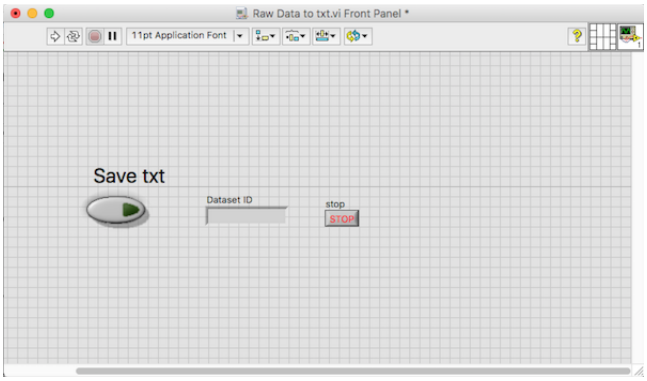
The stage has a height adjustment to account for the difference in surface heights of the samples. This is the largest source of error, so retake any samples that seem to have a poor height alignment, especially water. Later, the 1M NaSCN samples will be normalized by the average of the water samples before and after.

The input (reference) power, $I_\omega$, is measured by a photodiode and the signal power, $I_{2\omega}$, is measured with a PMT. The laser pulses are femtoseconds in width and have a repetition rate of 1kHz (1 pulse/ms), so a gated integrator is used to help reduce noise from dark counts. The output from the gated integrator is read by a DAQ card with LabView software. A trigger signal from the laser is used to correlate the pulses to give (reference, signal) data pairs. A neutral density filter wheel is used to modulate $I_\omega$ at a rate of 1Hz.




The LabView program for collecting data is called `Front Panel Direct.current.vi`. The above image shows a picture of the front panel. The program is shown running and it is meant to run continuously during all data collection. When the program is running, an inset on the top right appears. These are the shared global variables that will be discussed later. When the program is stopped, this panel is not shown. The upper left panel is where all the user control buttons are. The "Quit" button stops the program, as does pressing the red stop sign in the LabView toolbar. "Dataset ID" and "Rev" are the ID and revision number for the data in the current buffer stream. The "Save Directory" is the default directory for saving files. This can be changed from the program or during the save prompt. The "Save" button saves the current buffer to a .ddf file, which can only be read by LabView. The "New Dataset" button clears the buffer. "Dataset Size Limit" is how long you want the program to run for. One point is one pulse, which is one ms. "Dataset Size" is the actual size of the data. The "Reference" graph is the input intensity collected from the photodiode and the "Signal" graph is the signal intensity collected from the PMT.

At this point, the LabView files are converted to text and analyzed in Matlab. There are two LabView programs that can convert the data, `Channel Intensity Histograms.rawdata.vi` and `Raw Data to txt.vi`. `Raw Data to txt.vi` is




shown above. This program is also meant to run continuously, but is not shown running. Click the run arrow in the LabView toolbar to run. Click the "Save txt" button to choose the file to convert to txt and save it. The ID of the last dataset converted is in "Dataset ID." Use the "Stop" button or the stop sign button in the LabView toolbar to stop the program.

# 2 Matlab analysis of samples

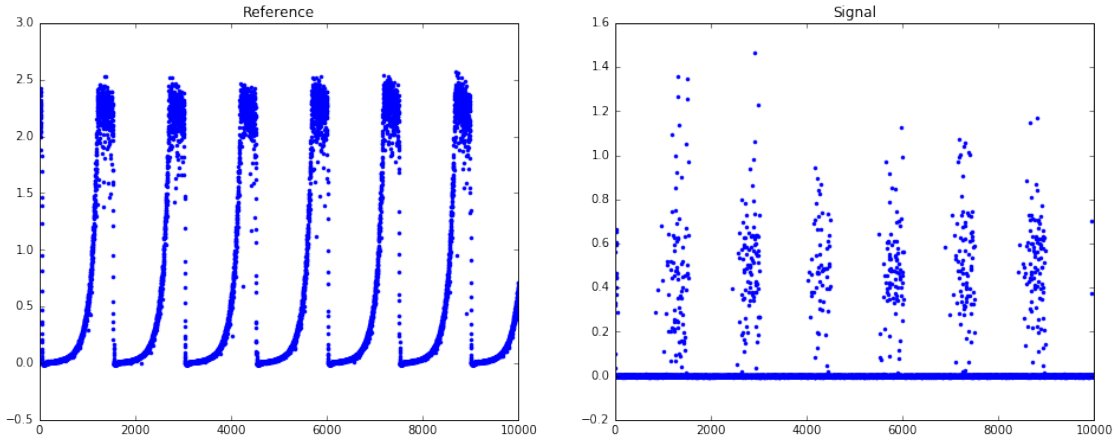
## 2.1 The raw time series data

The data generated is a time series that is exported to a .txt file. The first column is the reference and the second column is the signal. A row corresponds to 1 ms.

```
In [3]: with open('160428.145918.00.rawdata.txt') as file:
            water1 = numpy.loadtxt(file, delimiter='\t')
        print(water1[0:5])
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
        ax1.plot(water1[0:10000,0], '.')
        ax1.set_title('Reference')
        ax2.plot(water1[0:10000,1], '.')
        ax2.set_title('Signal')
```

```
[[  2.15811500e+00   9.76060000e-02]
 [  2.29381500e+00   5.61134000e-01]
 [  2.11731100e+00   1.64900000e-03]
 [  2.04961900e+00  -2.81500000e-03]
 [  2.20682800e+00   2.60500000e-03]]
```

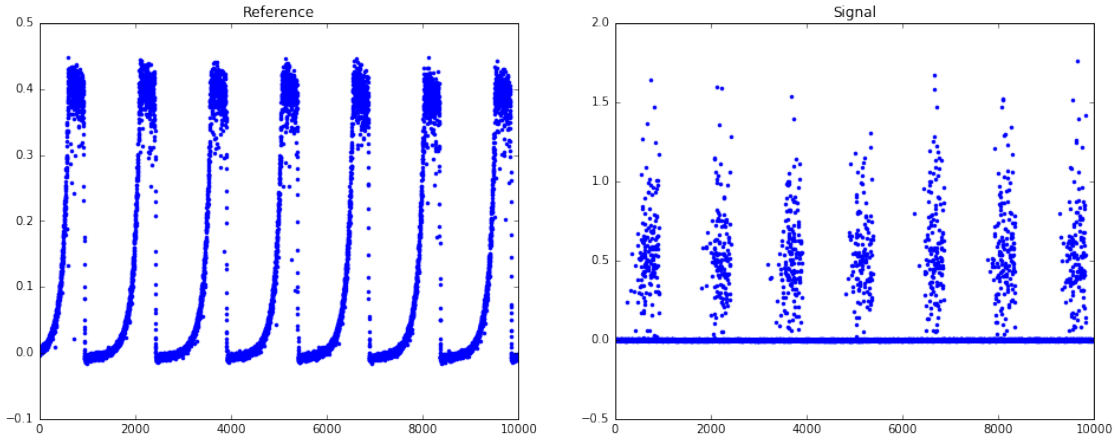
```
Out[3]: <matplotlib.text.Text at 0x117e8c198>
```




The above code shows an example of what a few rows of data look like, as well as the time series plots for the first 10000 ms. This sample is water, which is non-resonant, i.e. low signal. Here is what a thiocyanate sample (resonant, i.e. high signal) typically looks like:

```
In [4]: with open('160428.150342.00.rawdata.txt') as file:
            scn = numpy.loadtxt(file, delimiter='\t')
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
        ax1.plot(scn[0:10000,0], '.')
        ax1.set_title('Reference')
        ax2.plot(scn[0:10000,1], '.')
        ax2.set_title('Signal')
```

```
Out[4]: <matplotlib.text.Text at 0x1182b96d8>
```




Notice that there are more points above the baseline and that the magnitude is larger. The power here has been cut so that there is a maximum of one photon per pulse. Signal centered around ~0.5 corresponds to 1 pulse or less.

## 2.2 Choose the parameters based on the histogram

A matlab script called `BNanalysis_script.m` contains all the command line scripts that are needed to run the analysis in Matlab. I will talk about the individual lines of code here.

A parameter called "Signal Threshold" is used to discriminate between photons and null counts in the signal series. There are also two parameters called "Maximum Limit" and "Minimum Limit" that are used to discard the noisier points in the reference series. Plotting the histograms of these data allows you to choose an initial value for these three parameters. To plot in Matlab, run these lines once:

```
ref = figure;
sig = figure;
BNfit = figure;
limSearch = figure;
minV = 0;
maxV = 3;
thresh = .2;
nBins = 100;
binOpt = 0; %linear in R^2
sigOpt = 1; %ln(tau)
limType = 1; %start with signal threshold
stepSize = 0.01;
yMax = 100;
```

This creates workspace variables that are shared among the data sets. In particular, the first four variables are the figure handles that will be passed into the plotting functions. The windows generated should not be closed. Next, for each individual file, run the following lines to read the data into the Matlab workspace:

```
date = '######';
time = '######';
revision = '##';
readFile = strcat(date,'.',time,'.',revision,'.rawdata.txt');
data=dlmread(readFile);
dataCopy = data;
fprintf('\nPoints: %d\n\n', size(data,1));
```

The file name needs to be of the form "date.time.revision.rawdata.txt".

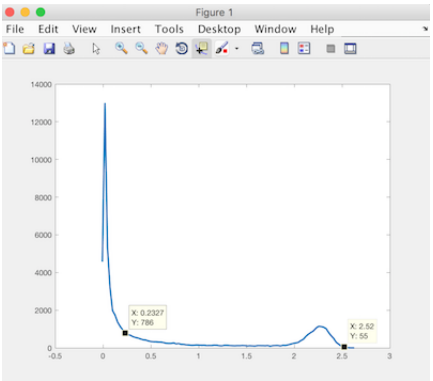
The histogram plotting function is called `intensityHist()`. The command line call is:

```
intensityHist(data, nBins, ref, sig, yMax);
```

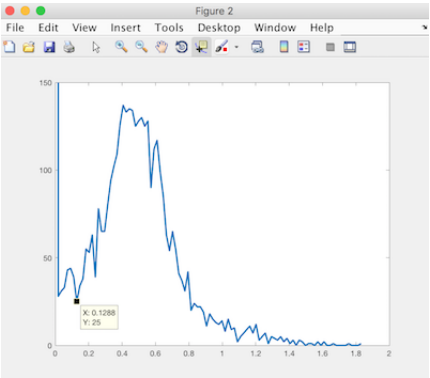
This plots the histogram of the data with the number of bins equaling `nBins`. The reference data is on the figure called `ref` and the signal data is on the figure called `sig`. `yMax` is the maximum for the y-axis on the signal histogram. Both `nBins` and `yMax` can

be adjusted. Here are example histograms with `nBins=100` and `yMax=150`:

**Reference**




**Signal**




The markers indicate good initial guesses for the parameters. They can be placed by clicking on the curve. Use Shift-Click for more than one marker. On the reference histogram, the "Minimum Limit" is the left marker and it is chosen to be at the midpoint of the curve. The "Maximum Limit" is the right marker and it is chosen to be in the right tail of the high intensity peak. On the signal histogram, the "Signal Threshold" is chosen to be where the signal peak begins to level out on the left. The large dark count peak, not shown in its entirety, also has a tail in that region that prevents the signal peak from returning to zero.

The corresponging X values for the markers can be extracted by right clicking on the marker and selecting "Export Cursor Data to Workspace." I choose to call all of the cursor variables `cursor_info`. For the reference histogram, this creates a 1x2 structure with three fields each. The `Position` field is a vector with [X,Y] coordinates. The following lines:

```
maxV = cursor_info(1).Position(1);
minV = cursor_info(2).Position(1);
```

copy the respective X positions to the `maxV` and `minV` variables. For the signal histogram, this creates a 1x1 structure with three fields. The following line:

```
thresh = cursor_info.Position(1);
```

copies the X position to the `thresh` variable.

## 2.3 Refine parameter choices

With the three parameters chosen, we can continue with the analysis. The next step is to bin the data with `binNorm()`:

```
[b, BN] = binNorm(data, minV, maxV, thresh, nBins, binOpt,...
    sigOpt, BNfit, true);
```

This function bins the signal data according to the reference value. The bins can either be linear in reference values or square reference values. This is encoded with the `binOpt` variable. Set `binOpt` to 1 for linear in reference values and anything else for linear in squared reference values. There are also two algorithms to calculate the signal. This is encoded in `sigOpt`. Set `sigOpt` to 1 to use Poissonian statistics (`nullCalc.m`) and anything else to use the signal value as is (`sigCalc.m`). In Poissonian statistics, the average photon count per bin (which is proportional to the second harmonic signal) is:

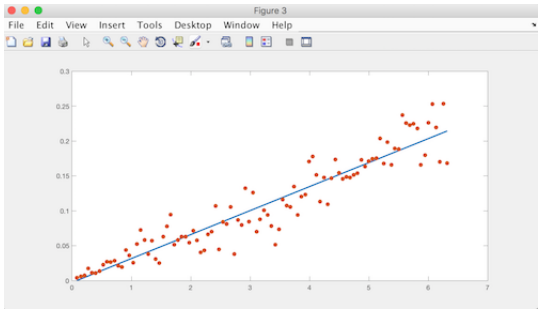
$$I_{2\omega} \propto \langle k \rangle = -ln\left[\frac{N_{k=0}^{pulse}}{N^{pulse}}\right]$$

where $\langle k \rangle$ is the average number of photons per pulse, $N_{k=0}^{pulse}$ is the number of null counts (no photon), and $N^{pulse}$ is the total number of pulses. `nullCalc.m` keeps track of the null counts in each bin and uses this to calculate $\langle k \rangle$.

Once the data is binned and the signal is calculated, the data is fit with a bisquare linear regression. This is a robust algorithm that is not influenced by outliers as much as regular linear regression. The slope is used as the measurement.

The remaining two inputs are `BNfit`, the figure handle that plots the data and the fit line, and `print`, a boolean for printing output to the screen. In this case, it is set to `true`.

I use `binOpts = 0` and `sigOpts = 1` for my data. A typical graph looks like this:



I decrease my power so that there is a maximum of one photon generated, but if this is not done, or if photoproducts are generated, you will see a graph that is not linear. In that case, adjust the parameters so that only the linear region is captured. This can usually be accomplished by adjusting "Maximum Limit".

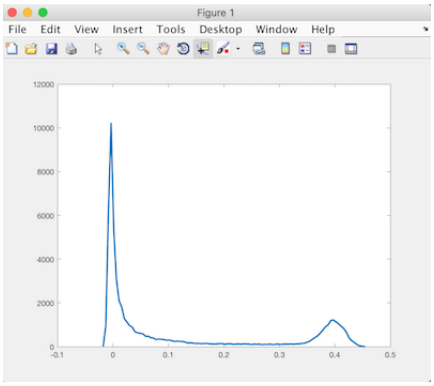
The outputs of `binNorm` are `b` and `BN`. `b` contains the slope and intercept values for the line. `BN` is a three column matrix with columns reference squared, signal, and fit.

The `limitSearch()` function is used to fine tune the parameters with the command line call:

```
limitSearch(data, minV, maxV, thresh, nBins, binOpt, sigOpt,...
    limSearch, limType, stepSize);
```

It calls `binNorm()` for five points on either side of the given parameter so that 11 values can be compared at once. The input `limType` encodes the parameter to search: 1 for "Signal Threshold," 2 for "Minimum Limit," and 3 for "Maximum Limit." The input `limSearch` is the figure handle to plot to. The input `stepSize` is the step size used when choosing the values to search. For "Signal Threshold," the amplitude of the signal, as seen on the histogram, should be similar for all samples when using the photon counting scheme, so `stepSize=0.01` is a good initial step size. For "Minimum Limit" and "Maximum Limit," the step size depends on the reference power. The water sample shown above was collected with the maximum available power at the collection wavelength. This corresponds to an amplitude of ~3 for the reference data. Appropriate step sizes in this case are 0.01 for "Minimum Limit" and 0.1 for "Maximum Limit." The graph below shows the reference histogram for a 2.5M NaSCN sample:

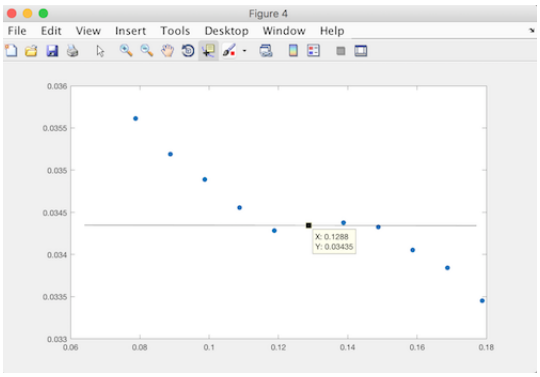
**Reference**



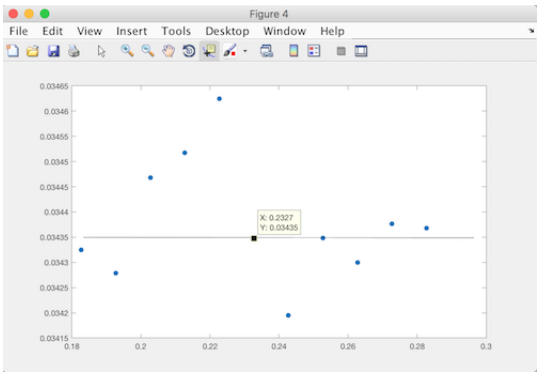

Notice that the reference amplitude is ~10 times smaller, so the step sizes should be 10 times smaller as well: 0.001 for "Minimum Limit" and 0.01 for "Maximum Limit." The "Maximum Limit" step size should always be 10 times larger than the "Minimum Limit" step size.

Here are the searches for the water sample:

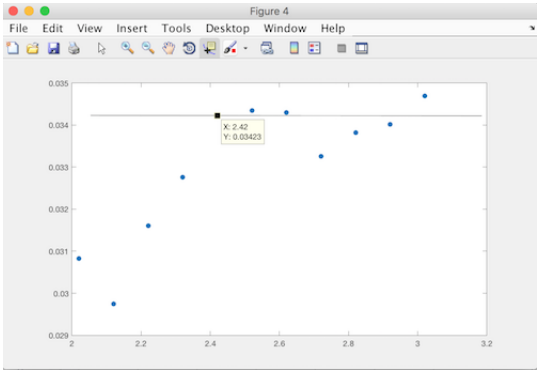
**Signal Threshold**



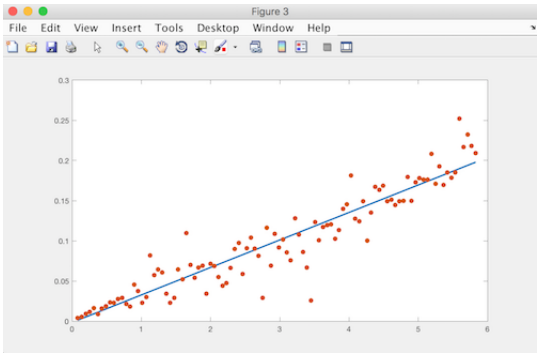

**Minimum Limit**




**Maximum Limit**

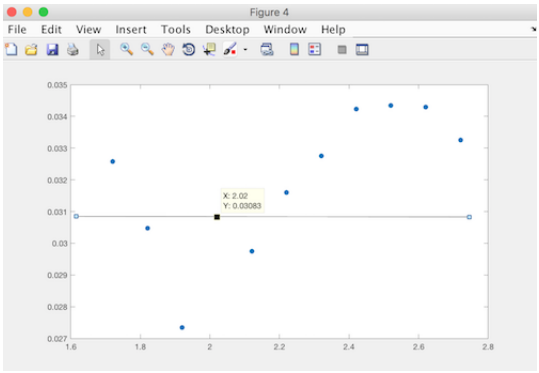



You are looking for a section in the graph where the values level out and then choosing a value that is roughly the average of the levelling out. The new values are selected on the graphs. Notice that "Maximum Limit" is the only one that will change in this case. If it is hard to find a good value, reduce the step size by half and try again. In general, it is good to go in order: "Signal Threshold," "Minimum Limit," "Maximum Limit." Once the new values are chosen and the corresponding variables updated, run `binNorm()` again:

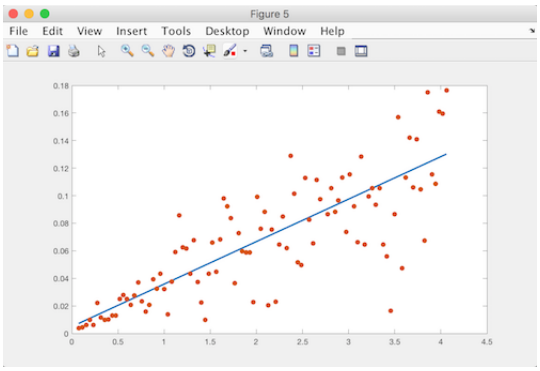



Here you can see that the high power end is starting to deviate from the fit line. This suggests that the highest power laser shots might be producing more than one photon. If this happens, choose a smaller value for `maxV` (I tried 2.2203 in this case) and run `limitSearch()` for the "Maximum Limit" again:

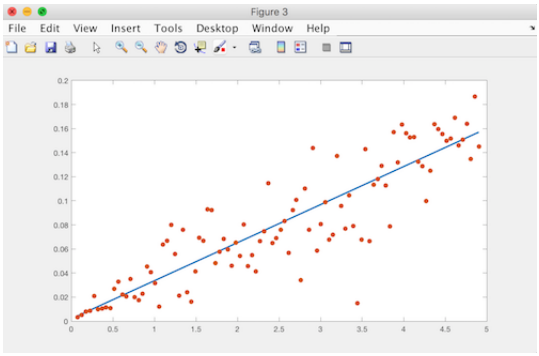
**Maximum Limit**




It looks like there are two level regions now. We already know the higher one is no good, so try a value for the lower region: 2.0203. Run `binNorm()` again:




Now the low power end is deviating. Try 2.2203:




This graph looks good; neither end is deviating from the fit. On the `limitSearch()` graph, it doesn't look like 2.2203 is an average for either of the level regions, but it does seem to be a pretty good average for the whole region. If these two programs don't ever converge to a reasonable value, give greater weight to the values that produce good graphs from `binNorm()`.

Once the final parameter values are chosen, run `binNorm()` one last time to get the slope measurement for the sample. The data can be saved as a .mat file with the following code:

```
writeMat = strcat(date,'.',time,'.',revision,'.BN.mat');
%can change revision number in command line
save(writeMat,'minV','maxV','thresh','nBins','b','BN','data',...
    'date','time','revision','dataCopy');
fprintf('.mat file saved\n');
```

It can also be saved as a text file with the following code:

```
writeFile = strcat(date,'.',time,'.',revision,'.BN.txt');
% can change revision number in command line
file = fopen(writeFile, 'w');
fprintf(file, strcat(writeFile,'\n'));
fprintf(file, 'Minimum Limit\tMaximum Limit\tSignal Threshold\t...
    # Bins\n%.6f\t%.6f\t%.6f\t%d\n',minV,maxV,thresh,nBins);
fprintf(file, 'Slope\tSlope Error\tIntercept\tIntercept Error...
    \n%.6f\t%.6f\t%.6f\t%.06f\n',b(2),b(4),b(1),b(3));
fprintf(file, 'R^2\tSignal\tFit\n');
fclose(file);
dlmwrite(writeFile,BN,'-append','delimiter','\t','newline','unix');
fprintf('.txt file saved\n');
```

## 2.4 Calculating final data points from the samples

The previous discussion was about how to analyze each sample. This section is how to analyze the data set of all your samples. You will have at least 6 samples per data point,

where a data point is one concentration for the system of interest, i.e. 1M NaSCN is a point and graphene on 1M NaSCN is another point.

At this point, I like to switch to Excel, but any sort of program that can handle tabular data will do. Excel is convenient because I can record slope values in the spreadsheet as I generate them. First, I look at the waters as a whole and get rid of one or two that seem to clearly be outliers. Also consider how the waters change over time when looking for outliers. Remove values cautiously, though. While we do know that poor height alignment can cause deviations in signal, it is still better practice to keep as many values as possible. Occasionally, solution samples can also be discarded.

The remaining samples need to be normalized to water by dividing by the average of the water samples before and after. For example, for the following samples:

- water1
- water2 (discarded)
- 1M NaSCN1
- water3

the normalized signal for 1M NaSCN1 is 1M NaSCN / average(water1, water3).

Once you have the normalized values, the data point is the average of all the normalized values and the error is the standard deviation. Since the error is deviation, it is not as dependent on sample size, so it is okay if the sample sizes differ. They most likely will because 4 aliquots in a flask doesn't mean you'll get 4 good normalized values.

# Appendix 4 – Text of the Matlab Functions for Measurement Processing

## createBins.m

```
function [edges] = createBins(minV, maxV, nBins)

%Creates bin edges based on the minimum and maximum reference limits and
%the number of bins.
%
% Inputs:
%     min: minimum reference value
%     max: maximum reference value
%     nBins: number of bins to create
%
% Output:
%     edges: a vector of bin edges, siz(nBins+1)

%Determine bin width
width = abs((maxV-minV)/nBins);

%Initialize edges
edges = 0:nBins;

%calculate edges
edges=edges*width+minV;

end
```

## lims.m

```
function [newData] = lims(data, minV, maxV)

%This function keeps data that have reference intensities between the min
%(exclusive) and max (exclusive) values.
%
% Inputs:
%     data: the data to be analyzed
%     min: minimum reference value
%     max: maximum reference value
%
```

```
% Output:
%     newData: analyzed data

newData = data;

cond = newData(:,1) >= maxV;
newData(cond,:) = [];
cond = newData(:,1) <= minV;
newData(cond,:) = [];

end
```

## sigCalc.m

```
function [binnedData] = sigCalc(data, edges, thresh)

%Calculates bin data using <R^2> and <S> for each bin.
%
% Inputs:
%     data: the data ready to be binned
%     edges: the end points for the bins (minimum inclusive)
%     thresh: signal threshold (exclusive)
%
% Output:
%     binnedData: Averaged ref and sig for each bin. Bins with fewer than 2
%         points are discarded

%Remove data below threshold
newData = data;

cond=newData(:,2) <= thresh;
newData(cond,:) = [];

%bin the data
[N, ~, bin] = histcounts(newData(:,1), edges);
N=N';

%calculate averages
R2 = accumarray(bin,newData(:,1),[],@mean);
S = accumarray(bin,newData(:,2),[],@mean);
binnedData = [R2.^2 S];

%keep only bins with more than one point
cond = N < 2;
binnedData(cond,:) = [];
```

```
end
```

# nullCalc.m

```
function [dataArray] = nullCalc(data, edges, nBins, thresh)

%Calculates bin data using <R^2> and <ln(tau)> for each bin. Uses unsorted
%raw data.
%
% Inputs:
%     data: the data to be binned, unsorted
%     edges: the end points for the bins (minimum inclusive)
%     nBins: the number of bins to use
%     thresh: signal threshold (exclusive)
%
% Output:
%     dataArray: Columns are total shots, <R^2>, null shots, <tau>,
%       <ln(tau)>. Bins are discarded with less than 2 null shots or with
%       <ln(tau)> == 0.

%find which bin each ref/sig pair belongs in
%all data is included, so some pairs might not have bins
[~,~,bin] = histcounts(data(:,1),edges);

m = size(data,1);

%data trackers are empty initially
dataArray = zeros(nBins,5)/0; %holds updating stats
internalArray = zeros(nBins,1)/0; %counts photons
%debugArray = zeros(m,6);

delta = 0;

for i = 1:m
    if bin(i)
        if isnan(dataArray(bin(i))) && isnan(internalArray(bin(i)))
            internalArray(bin(i)) = 0;
            dataArray(bin(i),:) = 0;
        else
            %update total shots and <R^2>
            dataArray(bin(i),1) = dataArray(bin(i),1) + 1;
            delta = data(i,1)^2 - dataArray(bin(i),2);
            dataArray(bin(i),2) = dataArray(bin(i),2)...
                + delta/dataArray(bin(i),1);
```

```
            %check for photon
            if data(i,2) <= thresh
                %data point
                internalArray(bin(i)) = internalArray(bin(i)) +1;
                y = log(internalArray(bin(i)));

                %update tau stats
                dataArray(bin(i),3) = dataArray(bin(i),3) + 1;
                delta = internalArray(bin(i)) - dataArray(bin(i),4);
                dataArray(bin(i),4) = dataArray(bin(i),4)...
                    + delta/dataArray(bin(i),3);
                delta = y - dataArray(bin(i),5);
                dataArray(bin(i),5) = dataArray(bin(i),5)...
                    + delta/dataArray(bin(i),3);

                internalArray(bin(i)) = 0; %reset photon counter
            else
                internalArray(bin(i)) = internalArray(bin(i)) +1;
            end
        end
    end

        %debugArray(i,:) = [dataArray(bin(i),:) internalArray(bin(i))];

end

cond = dataArray(:,3) < 2 | dataArray(:,5) == 0;
dataArray(cond,:) = [];

end
```

# binNorm.m

```
function [b,BN] = binNorm(data, minV, maxV, thresh, nBins, binOpt,...
    sigOpt, figHandle, print)

%Takes reference/signal data, bins it according to reference values, and
%fits the bins to a bisquare linear fit.
%
% Inputs:
%     data: the data to be analyzed
%     min: minimum reference value
%     max: maximum reference value
```

```
%     thresh: threshold value for photon event
%     nBins: number of bins to use
%     binOpt: determines whether bins should be linear in R or R^2. A
%      value of 1 is used for linear in R. All other values indicate R^2.
%     sigOpt: determines whether integrated signal or ln(tau) is used in
%      fit. A value of 1 is used for ln(tau). All other values indicate
%      integrated signal.
%     figHandle: the handle of the figure to plot to. Pass 0 or False to
%      suppress figure.
%     print: print is a boolean. True prints the slope, false does not
%      print the slope.
%
% Output:
%     b: a vector with slope and intercept values
%     BN: a matrix with columns R^2, ln(tau), yFit

%make sure data is two columns
[m,n] = size(data);

if m == 2 || n == 2 %makes sure data is two vectors

    if n > m
        data=data'; %makes sure data is two columns
    end

    %Determine if bins should be linear in R or R^2 and create edges
    %Final edges are in reference values
    if isempty(binOpt)
        binOpt = 0;
    end

    if binOpt ~= 1 %linear in R^2
        min2 = minV^2;
        max2 = maxV^2;
        edges = createBins(min2, max2, nBins);
        edges = sqrt(edges);
    else %linear in R
        edges = createBins(minV, maxV, nBins);
    end

    %Determine if integrated signal or null counts are used in fit.
    if isempty(sigOpt)
        sigOpt = 0;
    end
```

```
    if sigOpt ~= 1 %integrated signal
        Ldata = lims(data, minV, maxV); %Throw out data outside the
                                        %reference limits.
        Ldata = sortrows(Ldata);
        dataArray = sigCalc(Ldata, edges, thresh);
        X = dataArray(:,1);
        Y = dataArray(:,2);
    else %null counts
        dataArray = nullCalc(data, edges, nBins, thresh);
        X = dataArray(:,2);
        Y = dataArray(:,5);
    end

    [b, stats] = robustfit(X,Y);

    %calculate fit
    yFit = b(2)*X + b(1);

    if ~(figHandle == 0)
        figure(figHandle)
        plot(X,yFit,'-',X,Y,'.','LineWidth',2,'MarkerSize',22)
        datacursormode off
    end

    %make output
    b = [b stats.se];
    BN = [X Y yFit];

    %print out slope
    if print
        fprintf('Slope: %.6f\nDoF: %d\n',b(2),stats.dfe);
    end

%%all calculations before this point
else
    fprintf('Data must have one dimension equal to 2.')
end

end
```

# intensityHist.m

```
function intensityHist(data, nBins, figHandle1, figHandle2, yMax)

%Creates histograms of the reference and signal.
```

```
% Inputs:
%     data: the data to be histogramed
%     figHandle1: figure handle for the reference histogram
%     figHandle2: figure handle for the signal histogram

%reference
[N1, edges1] = histcounts(data(:,1), nBins);
width1 = edges1(2) - edges1(1);
centers1 = edges1 + width1/2;
centers1 = centers1(1:nBins);

figure(figHandle1)
plot(centers1, N1, 'LineWidth', 2)
datacursormode on

%signal
[N2, edges2] = histcounts(data(:,2), nBins);
width2 = edges2(2) - edges2(1);
centers2 = edges2 + width2/2;
centers2 = centers2(1:nBins);

figure(figHandle2)
plot(centers2, N2, 'LineWidth', 2)
ylim([0 yMax])
datacursormode on

end
```

# limitSearch.m

```
function limitSearch(data, minV, maxV, thresh, nBins, binOpt,...
 sigOpt, figHandle, limType, stepSize)

%Looks at a window of 11 values for a parameter and plots them.
%
% Inputs:
%     data: the data to be analyzed
%     min: minimum reference value
%     max: maximum reference value
%     thresh: threshold value for photon event
%     nBins: number of bins to use
%     binOpt: determines whether bins should be linear in R or R^2. A
%      value of 1 is used for linear in R. All other values indicate R^2.
%     sigOpt: determines whether integrated signal or ln(tau) is used in
%      fit. A value of 1 is used for ln(tau). All other values indicate
```

```
%      integrated signal.
%     figHandle: the handle of the figure to plot to. Pass 0 or False to
%       suppress figure.
%     limType: which parameter to search. 1 for thresh, 2 for minV, 3 for
%     maxV.
%     stepSize: width between values to search.

%start calculating axes
X = [-5:5];
X = X * stepSize;
bRange = zeros(11,1);

%calculate differently based on limType
switch limType
    case 1 %thresh
        X = X + thresh;
        cond = X < 0; %get rid of negative values
        X(cond) = [];
        for i = 1:11
            [b,~] = binNorm(data, minV, maxV, X(i), nBins,...
             binOpt, sigOpt, 0, false);
            bRange(i) = b(2);
        end
    case 2 %minV
        X = X + minV;
        cond = X < 0;
        X(cond) = [];
        for i = 1:11
            [b,~] = binNorm(data, X(i), maxV, thresh, nBins,...
             binOpt, sigOpt, 0, false);
            bRange(i) = b(2);
        end
    case 3 %maxV
        X = X + maxV;
        cond = X < 0;
        X(cond) = [];
        for i = 1:11
            [b,~] = binNorm(data, minV, X(i), thresh, nBins,...
             binOpt, sigOpt, 0, false);
            bRange(i) = b(2);
        end
    otherwise
        disp('not a valid limType')
end
```

```
figure(figHandle)
plot(X,bRange,'.','MarkerSize',22)
cursor = datacursormode(figHandle);
set(cursor, 'Enable', 'on', 'SnapToDataVertex', 'on');

end
```

## BNanalysis_script.m

```
%a script for analyzing datasets from the command line
%do at beginning
ref = figure;
sig = figure;
BNfit = figure;
limSearch = figure;
minV = 0;
maxV = 3;
thresh = .2;
nBins = 100;
binOpt = 0; %linear in R^2
sigOpt = 1; %ln(tau)
limType = 1; %start with signal threshold
stepSize = 0.01;
yMax = 100;

%create file name strings
date = '160428';
time = '145918';
revision = '00';

%import data
readFile = strcat(date,'.',time,'.',revision,'.rawdata.txt');
data=dlmread(readFile);
dataCopy = data;
fprintf('\nPoints: %d\n\n', size(data,1));

%histograms
intensityHist(data, nBins, ref, sig, yMax); %can change yMax and rerun

%bin data and fitminV
[b, BN] = binNorm(data, minV, maxV, thresh, nBins, binOpt,...
 sigOpt, BNfit, true);

%search limits
limitSearch(data, minV, maxV, thresh, nBins, binOpt, sigOpt,...
 limSearch, limType, stepSize);
```

```
%write cursor data to variables
thresh = cursor_info.Position(1);
maxV = cursor_info(1).Position(1);
minV = cursor_info(2).Position(1);

%write data to mat file
writeMat = strcat(date,'.',time,'.',revision,'.BN.mat');
%can change revision number in command line
save(writeMat,'minV','maxV','thresh','nBins','b','BN','data','date',...
    'time','revision','dataCopy');
fprintf('.mat file saved\n');

%write data to file
writeFile = strcat(date,'.',time,'.',revision,'.BN.txt');
% can change revision number in command line
file = fopen(writeFile, 'w');
fprintf(file, strcat(writeFile,'\n'));
fprintf(file, 'Minimum Limit\tMaximum Limit\tSignal Threshold\t#'...
 'Bins\n%.6f\t%.6f\t%.6f\t%d\n',minV,maxV,thresh,nBins);
fprintf(file, 'Slope\tSlope Error\tIntercept\tIntercept Error'...
 '\n%.6f\t%.6f\t%.6f\t%.06f\n',b(2),b(4),b(1),b(3));
fprintf(file, 'R^2\tSignal\tFit\n');
fclose(file);
dlmwrite(writeFile,BN,'-append','delimiter','\t','newline','unix');
fprintf('.txt file saved\n');
```

# Appendix 5 – Text of the Python Files for Fitting and Usage Examples

These Python modules were written to fit second harmonic generation datasets to a Langmuir equation using the Python module `lmfit`. There is also a module for calculating jackknife fits. The modules are:

- datatools.py
- fits.py
- models.py
- minimizefits.py
- jackknife.py

## 1 datatools.py

```
"""
Useful functions for handling the data used in the fitting
scripts.

Functions:
parse_csv_data(array)
create_flat_data(X, Y, Yerr)
plot_multi_data_sets(X, Y, Yerr)
extract_fit_values(fit)
extract_err_values(fit)
extract_corrs(fit)
"""
import numpy
import matplotlib.pyplot as plt

def parse_csv_data(array):
    """
    Takes data that has been imported from a csv file and parses
    it into X, Y, and Yerr arrays. It assumes the shape:
    X | Y(1) | Yerr(1) | ... |Y(num_sets) | Yerr(num_sets)

    Inputs:
        array: a numpy array of the data imported from a csv file
    Outputs:
```

```
        X: the 1D column of X values. It has the same units as in
           the csv file.
        Y: the Y values with data sets in columns, 1D or 2D
        Yerr: the Yerr values with data sets in columns. Same
           shape as Y.    """
    (rows, cols) = array.shape

    #determine number of data sets
    num_sets = (cols - 1) // 2

    #create X
    X = array[:,0]

    #create Y and Yerr
    Y = numpy.zeros_like(X)
    Yerr = numpy.zeros_like(X)
    for i in range(num_sets):
        Y = numpy.column_stack((Y, array[:,i*2 + 1]))
        Yerr = numpy.column_stack((Yerr, array[:,i*2 + 2]))
    Y = numpy.delete(Y, 0, 1)
    Yerr = numpy.delete(Yerr, 0, 1)

    return X, Y, Yerr

def create_flat_data(X, Y, Yerr):
    """
    Takes X, Y, and Yerr data and makes 1D flattened vectors.

    Inputs:
        X: the 1D column of X values.
        Y: Y values with datasets in columns, a 2D numpy array
        Yerr: the Yerr values with data sets in columns. Same
           shape as Y.
    Outputs:
        flat_X: a 1D vector with num_sets repeats of X. Same
            shape as Y and Yerr.
        flat_Y: a 1D vector with Y columns stacked on each other.
            Same shape as X and Yerr.
        flat_Yerr: a 1D vector with Yerr columns stacked on each
            other. Same shape as X and Y.
    """
    (rows, num_sets) = Y.shape

    #create flattened X
    flat_X = numpy.zeros_like(X)
```

```
    #create a column vector with num_sets repeats of the X vector
    for i in range(num_sets):
        flat_X = numpy.column_stack((flat_X, X))
    flat_X = numpy.delete(flat_X, 0, 1)
    flat_X = flat_X.flatten('F')


    #create flattened Y and Yerr
    flat_Y = Y.flatten('F')
    flat_Yerr = Yerr.flatten('F')

    return flat_X, flat_Y, flat_Yerr

def plot_multi_data_sets(X, Y, Yerr):
    """
    Takes X, Y, and Yerr data and creates a plot with errorbars.
    Using only the errorbar function results in the data set
    lines being connected.

    Inputs:
        X: the 1D column of X values
        Y: the Y values with data sets in columns, 1D or 2D
        Yerr: the Yerr values with data sets in columns. Same
           shape as Y.
    """
    #create flat data for errorbar
    (flat_X, flat_Y, flat_Yerr) = create_flat_data(X, Y, Yerr)

    #make plots
    plt.plot(X, Y) #plot can broadcast, errorbar can't
    plt.errorbar(flat_X, flat_Y, flat_Yerr, linestyle='None',
                 marker='o', color='0.5', ecolor='k',
                 elinewidth='3')

def extract_fit_values(fit):
    """
    Extract the parameter values of varied parameters into a list
    """
    fit_values = []
    for param in fit.params:
        if fit.params[param].vary:
            fit_values.append(fit.params[param].value)
    #fit_values = numpy.array(fit_values)

    return fit_values
```

```
def extract_err_values(fit):
    """
    Extract the parameter errors of varied parameters into a list
    """
    fit_errs = []
    for param in fit.params:
        if fit.params[param].vary:
            fit_errs.append(fit.params[param].stderr)
    #fit_values = numpy.array(fit_values)

    return fit_errs

 def extract_corrs(fit):
     """Extract the correlation matrix from fit parameters."""
     params = fit.params

     corrs_cols = []
     for param in params:
         try:
             for key, value in params[param].correl.items():
                 if abs(value) >= 0.1:
                     corrs_cols.append([param, key, value])
         except AttributeError:
             pass
     corrs_matrix = pandas.DataFrame(corrs_cols).pivot(index=0,
                                                       columns=1)

     return corrs_matrix
```

# 2 fits.py

```
"""
This module contains the Langmuir functions to be used by the
various fitting functions.

Functions:
lang_mol_frac(X, A, B, C, G, T=293, R=8.314)
lang_pot_mol_frac(X, A, B, C, D, E, G, T=293, R=8.314)
"""

import math
import numpy
```

```
def lang_mol_frac(X, A, B, C, G, T=293, R=8.314):
    """Calculates the simple Langmuir isotherm in mole fractions.
    Inputs:
        A - fit parameter
        B - fit parameter
        C - fit parameter
        G - free energy of adsorption in J/mol
        R - gas constant, fixed (default = 8.314 J/mol/K)
        T - temperature, fixed (default = 293K)
        X - concentration in mole fractions, type = numpy array
    Outputs:
        Yfit - the model Y values for X
    """
    Yfit = numpy.zeros_like(X)
    #creates an array of zeros the same shape as X

    Yfit = ((A + B *(X / (X + (1 - X) * math.exp(G/R/T)))) ** 2 +
            (C * (X / (X + (1 - X) * math.exp(G/R/T)))) ** 2)

    return Yfit

def lang_pot_mol_frac(X, A, B, C, D, E, G, T=293, R=8.314):
    """Calculates the Langmuir isotherm with surface potential
    included in mole fractions.
    Inputs:
        A - fit parameter
        B - fit parameter
        C - fit parameter
        D - fit parameter
        E - fit parameter
        G - free energy of adsorption in J/mol
        R - gas constant, fixed (default = 8.314 J/mol/K)
        T - temperature, fixed (default = 293K)
        X - concentration in mole fractions, type = numpy array
    Outputs:
        Yfit - the model Y values for X"""
    Yfit = numpy.zeros_like(X)
    #creates an array of zeros the same shape as X

    Yfit = ((A + B *(X / (X + (1 - X) * math.exp(G/R/T))) +
             D *(X / (X + (1 - X) * math.exp(G/R/T)))**2) ** 2 +
            (C * (X / (X + (1 - X) * math.exp(G/R/T))) +
             E *(X / (X + (1 - X) * math.exp(G/R/T)))**2) ** 2)
```

```
    return Yfit
```

# 3 models.py

```
"""
This module contains the corresponding Model object fits for the
fitting equations written in fits.py. These only work with single
data sets, i.e. one Y column.

Functions:
lang_mol_frac_fit(X, Y, Yerr, initial, print_out=True)
lang_pot_mol_frac_fit(X, Y, Yerr, initial, print_out=True)
"""
import lmfit
import matplotlib.pyplot as plt

import fits

#All models must start with args (X, Y, Yerr, initial, ...)
def lang_mol_frac_fit(X, Y, Yerr, initial, print_out=True):
    """Calculates a single fit using the simple Langmuir model.

    Inputs:
        X - concentration in mole fractions, a 1D numpy array
        Y - SHG data, a 1D numpy array the same size as X
        Yerr - error bars for Y, a 1D numpy array same size as Y
        initial - initial values for parameters, a 4 element list
        print_out - a flag for printing the output of the fit

    Outputs:
        fit_lmf - a ModelResult object from the lmfit module
    """
    #weights are inverse errors, multiplied by (y(exp)-y(model))
    W = 1 / Yerr

    #create Model object from fit function contained in fits.py
    lang_mol_frac_model = lmfit.Model(fits.lang_mol_frac)

    #initialize the parameters
    params_lmf = lang_mol_frac_model.make_params()
    params_lmf['A'].set(value=initial[0])
    params_lmf['B'].set(value=initial[1])
    params_lmf['C'].set(value=initial[2])
```

```
    params_lmf['G'].set(value=initial[3])
    params_lmf['R'].set(vary=False)
    params_lmf['T'].set(vary=False)

    #fit the Model
    fit_lmf = lang_mol_frac_model.fit(Y, params_lmf, W, X=X,
                                      scale_covar=False)

    #print out the results if desired
    if print_out:
        print(fit_lmf.fit_report())
        fit_lmf.plot_fit()
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2)

    return fit_lmf

def lang_pot_mol_frac_fit(X, Y, Yerr, initial, print_out=True):
    """Calculates a single fit using the Langmuir model with
        surface potential.

    Inputs:
        X - concentration in mole fractions, a 1D numpy array
        Y - SHG data, a 1D numpy array the same size as X
        Yerr - error bars for Y, a 1D numpy array same size as Y
        initial - initial values for parameters, a 6 element list
        print_out - a flag for printing the output of the fit

    Outputs:
        fit_lpmf - a ModelResult object from the lmfit module
    """
    #weights are inverse errors, multiplied by (y(exp)-y(model))
    W = 1 / Yerr

    #create Model object from fit function contained in fits.py
    lang_pot_mol_frac_model = lmfit.Model(fits.lang_pot_mol_frac)

    #initialize the parameters
    params_lpmf = lang_pot_mol_frac_model.make_params()
    params_lpmf['A'].set(value=initial[0])
    params_lpmf['B'].set(value=initial[1])
    params_lpmf['C'].set(value=initial[2])
    params_lpmf['D'].set(value=initial[3])
    params_lpmf['E'].set(value=initial[4])
    params_lpmf['G'].set(value=initial[5])
    params_lpmf['R'].set(vary=False)
```

```
    params_lpmf['T'].set(vary=False)

    #fit the Model
    fit_lpmf = lang_pot_mol_frac_model.fit(Y, params_lpmf, W,X=X,
                                           scale_covar=False)

    #print out the results
    if print_out:
        print(fit_lpmf.fit_report())
        fit_lpmf.plot_fit()
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2)

    return fit_lpmf
```

# 4 minimizefits.py

```
"""
A module for fitting the Langmuir functions in fits.py to
multiple datasets, i.e. multiple Y columns. It allows control
over shared parameters.

Functions:
make_full_params(T, R, Y, vars_list, *args)
lang_mol_frac_dataset(params, i, X, vars_list, func)
resid_multi_lang_mol_frac(params, X, Y, Yerr, num_sets,
                          vars_list, func)
report_and_plot(fit, X, Y, Yerr, num_sets, vars_list, func)
lang_mol_frac_multiset(X, Y, Yerr, func, *args, T=293, R=8.314,
                           print_out=True, handle_error=True)
"""
import numpy
import lmfit
import matplotlib.pyplot as plt
import inspect

import datatools

def make_full_params(T, R, Y, vars_list, *args):
    """
    Generate the full Parameter object for the dataset. For
    variables that aren't shared, it creates num_sets parameters
    with names e.g. 'A_1', 'A_2.' For shared parameters, only one
    parameter is created.
```

```
    Inputs:
        T: the temperature(s) for the experiment, list for
            multiple temps and numeric for same temp
        R: gas constant (8.314 kJ/mol/K)
        Y: SHG data, a 2D array
        vars_list: list of strings that represent the variables.
            Doesn't include temp.
        *args: the appropriate variables for the function,
            i.e. A, B, C, G

    Outputs:
        full_params: the full set of parameters
    """
    full_params = lmfit.Parameters()

    #handle R
    full_params.add('R', value=R, vary=False)

    #iterate over the variables that vary
    for var, var_value in zip(vars_list, args):
        #if the variable is a list, that means it's not shared
        #and multiple parameters need to be created
        if isinstance(var_value, list):
            for iy, y in enumerate(Y.T):
                full_params.add('{:s}_{:d}'.format(var, iy+1),
                                value=var_value[iy])
        #else the parameter is shared and one is created
        else:
            full_params.add(var, value = var_value)

    #handle T separately but similarly, only diff is 'vary'.
    if isinstance(T, list):
        for iy, y in enumerate(Y.T):
            full_params.add('T_{:d}'.format(iy+1), value=T[iy],
                            vary=False)
    else:
        full_params.add('T', value = T, vary=False)

    return full_params

def lang_mol_frac_dataset(params, i, X, vars_list, func):
    """
    Calculate the Langmuir fit for data set i using simple,
    hardwired naming convention.
```

```
    Inputs:
        params: a Parameters object that includes all of the
            parameters. Naming convention is '{model parameter
            name}_{i}'. R is always the same, so it is just 'R'.
        i: the dataset iterator
        X: concentration in mole fractions, a 1D numpy array
        vars_list: a list with the names of the variables in
            strings. Should include temperature.
        func: the name of the function to use

    Returns the fit values calculated from X and the parameters.
    """
    #create a list to store the valaues
    values = []

    #iterate over the variables
    for var in vars_list:
        try: #try as a list
            values.append(params['{:s}_{:d}'.format(var, i+1)]
                                    .value)
        except KeyError: #if not a list
            values.append(params[var].value)

    #handle R
    values.append(params['R'])

    #pass X and the values into the langmuir function
    Yfit = func(X, *values)

    return Yfit

def resid_multi_lang_mol_frac(params, X, Y, Yerr, num_sets,
                                vars_list, func):
    """
    Calculates the model residuals: (data - model) / uncertainty.
    This function is passed into lmfit.minimize() for least
    squares minimization.

    Inputs:
        params: a Parameters object that includes all of the
            parameters. Naming convention is '{model parameter
            name}_{i}'. R is always the same, so it is just 'R'.
        X: concentration in mole fractions, a 1D numpy array
        Y: the normalized SHG signal, a 2D numpy array with the
            same number of rows as X
```

```
        Yerr: the error in Y, a numpy array the same shape as Y
        num_sets: the total number of datasets
        vars_list: a list with the names of the variables in
            strings. Should include temperature.
        func: the name of the function to use

    Returns the residual array, flattened into 1D.
    """
    #calculate the full fit for the data sets
    Yfit = numpy.zeros_like(X) #add a row of zeros to start
    for i in range(num_sets):
        Ysubfit = lang_mol_frac_dataset(params, i, X, vars_list,
                                        func)
        Yfit = numpy.column_stack((Yfit, Ysubfit))
    Yfit = numpy.delete(Yfit, 0, 1)# take out the row of zeros

    #claculate the residual
    resid = (Y - Yfit) / Yerr

    return resid.flatten()

def report_and_plot(fit, X, Y, Yerr, num_sets, vars_list, func):
    """
    Creates a fit report and a plot of the fit with errorbars.

    Inputs:
        fit: the fit object from lmfit.minimizer()
        X: concentration in mole fractions, a 1D numpy array
        Y: the normalized SHG signal, a 2D numpy array with the
            same number of rows as X
        Yerr: the error in Y, a numpy array the same shape as Y
        num_sets: the total number of datasets
        vars_list: a list with the names of the variables in
            strings. Should include temperature.
        func: the name of the function to use
    """
    #create flattened data for errorbar
    flat_X, flat_Y, flat_Yerr = datatools.create_flat_data(X, Y,
                                                           Yerr)

    #create report
    lmfit.report_fit(fit.params)

    #create plot
    plt.figure()
```

```
    for i in range(num_sets):
        y_fit = lang_mol_frac_dataset(fit.params, i, X,
                                      vars_list, func)
        plt.plot(X, y_fit, '-')
    plt.errorbar(flat_X, flat_Y, flat_Yerr, linestyle='None',
                 marker='o', color='0.5', ecolor='k',
                 elinewidth='3')

def lang_mol_frac_multiset(X, Y, Yerr, func, *args,T=293,R=8.314,
                           print_out=True, handle_error=True):
    """
    Will fit multiple datasets with any Langmuir function from
    fits.py.

    Inputs:
        X: the 1D column of X values.
        Y: the Y values with data sets in columns, 2D numpy array
        Yerr: the Yerr values with data sets in columns. Same
            shape as Y.
        func: the name of the function to use
        *args: the appropriate variables for the function,
            i.e. A, B, C, G. For fit parameters, the arg must be
            a list the same length as the number of datasets or
            a scalar.
        T: temperature. Can be numeric or a list. Default is 293K
        R: the gas constant (default: 8.314 kJ/mol/K)
        print_out: a boolean flag for printing out the fit report
            and plot. True will print them out. Default is True.
        handle_error: a boolean flag for how to handle
            OverflowErrors in the fitting process. True will
            handle the errors in this function. Use false if
            another function calls this one, e.g. a jackknife
            function. Default is True.
    """
    #find the number of datasets
    (num_conc, num_sets) = Y.shape

    #find the parameters list and remove X, R, and T
    vars_list = inspect.signature(func)
    vars_list = list(vars_list.parameters.keys())
    vars_list.remove('X')
    vars_list.remove('R')
    vars_list.remove('T')

    #create the parameters
```

```
    full_params = make_full_params(T, R, Y, vars_list, *args)

    #add T back to vars_list
    vars_list.append('T')

    #create the Minimizer model
    model =lmfit.Minimizer(resid_multi_lang_mol_frac,full_params,
                           fcn_args=(X, Y, Yerr, num_sets,
                                     vars_list, func),
                           nan_policy='omit', scale_covar=False)

    #fit the data
    if handle_error: #handle errors in this function if true
        try:
            fit = model.minimize()

        except OverflowError:
        #prints out a message, but still raises error
            fit = []
            print("OverflowError. Use new inital values.")

        else:
            #print out the results
            if print_out:
                report_and_plot(fit,X,Y,Yerr,num_sets, vars_list,
                                func)

    else: #error handled in calling function
        fit = model.minimize()
        #print out the results
        if print_out:
            report_and_plot(fit, X, Y, Yerr, num_sets, vars_list,
                            func)

    return fit
```

# 5 jackknife.py

```
"""
This module contains functions used to calculate the jackknife
error for any of the Langmuir fits in models.py and
minimizefits.py. To create new functions to use in the jackknife
calculation, the order of parameters needs to match the order in
```

```
the jackknife call minus the model parameter.

Functions:
jackknife_one_set(model, X, Y, Yerr, *args)
subfit_redo_one_set(i, model, X, Y, Yerr,*args)
jackknife_multi_set(X, Y, Yerr, *args, **kwargs)
def subfit_redo_multi_set(k, X, Y, Yerr, *args, **kwargs):
jackknife_calcs(sub_fits, fit_values)
"""
import numpy

import datatools
n+nnminimizefits

def jackknife_one_set(model, X, Y, Yerr, *args):
    """A function for fitting subsets of data to be used in a
    jackknife calculation. This function only works for one set
    of Y, Yerr data and thus can only call models form the
    models.py file.

    Inputs:
        model: the function name of the desired Langmuir model
            from models.py
        X: concentration data in mole fractions, a 1D numpy array
        Y: SHG data, a 1D numpy array that is the same size as X
        Yerr: error bars for Y, a 1D numpy array that is the same
            size as Y
        *args: any additional arguments needed for model.

    Outputs:
        sub_fit: the parameters for each subset, a numpy array
        sub_errs: the fit errors for each subset, a numpy array
        fit_values: the parameter values for the full dataset, a
            numpy array
    """
    #number of data points
    num_data = numpy.size(X)

    #create the fit for the full dataset
    fit = model(X, Y, Yerr, *args)

    #get ordered list of parameter values
    fit_values = datatools.extract_fit_values(fit)

    #need an initialized array before I can append >:(
```

```
sub_fits = numpy.zeros_like(fit_values)
sub_errs = numpy.zeros_like(fit_values)

for i in range(num_data):
    #delete index at i
    sub_X = numpy.delete(X, i)
    sub_Y = numpy.delete(Y, i)
    sub_Yerr = numpy.delete(Yerr, i)

    #retry logic for OverflowError in fits
    MAX_RETRIES = 2
    new_args = list(args)

    #try the fit with inital values of one
    for retries in range(MAX_RETRIES):
        try:
            #same as args first time, modified second time
            sub_fit = model(sub_X, sub_Y, sub_Yerr,*new_args)

        #if there's an overflow, try again with initial
        #values set to fit_values
        except OverflowError:
            #create new initial values
            initial_new = list(fit_values)

            #copy args to new list variable for modification
            new_args[0] = initial_new

        #only does this if fit is valid and breaks loop
        #extracts fit values and errors
        else:
            column = datatools.extract_fit_values(sub_fit)
            col_err = datatools.extract_err_values(sub_fit)
            break

    #does this when no valid fit was generated
    #fill with nans and look at later
    else:
        column = numpy.zeros_like(fit_values)
        column.fill(numpy.nan)
        col_err = numpy.zeros_like(fit_values)
        col_err.fill(numpy.nan)
        print("Subset {:d} caused an overflow error.
                    ".format(i))
```

```
        #gather the sub fit info into one variable
        sub_fits = numpy.column_stack((sub_fits,column))
        sub_errs = numpy.column_stack((sub_errs,col_err))

    #delete the column of zeros from initialization
    sub_fits = numpy.delete(sub_fits, 0, axis=1)
    sub_errs = numpy.delete(sub_errs, 0, axis=1)

    return sub_fits, sub_errs, fit_values

def subfit_redo_one_set(i, model, X, Y, Yerr,*args):
    """
    A function to redo one of the sub fits in case any fits seem
    erroneous.Used with jackknife_one_set().

    Inputs:
        i: the number of the dataset to redo
            (the column index in the sub_fits variable)
        **the rest of the inputs are the same as for
            jackknife_one_set()**
        model: the function name of the desired Langmuir model
            from models.py
        X: concentration data in mole fractions, a 1D numpy array
        Y: SHG data, a 1D numpy array that is the same size as X
        Yerr: error bars for Y, a 1D numpy array that is the same
            size as Y
        *args: any additional arguments needed for model.

    Outputs:
        sub_fit_new: the new set of parameter values
        sub_err_new: the new set of parameter errors
    """
    #create the subsampled dataset
    sub_X = numpy.delete(X, i)
    sub_Y = numpy.delete(Y, i)
    sub_Yerr = numpy.delete(Yerr, i)

    #fit the dataset
    sub_fit = model(sub_X, sub_Y, sub_Yerr, *args)

    #get the sub values and sub errors
    sub_fit_new = datatools.extract_fit_values(sub_fit)
    sub_err_new = datatools.extract_err_values(sub_fit)

    return sub_fit_new, sub_err_new
```

```
def jackknife_multi_set(X, Y, Yerr, *args, **kwargs):
    """A function for performing a jackknife calculation with the
    lang_mol_frac_multiset() function from minimizefits.py.

    Inputs:
        X: concentration data in mole fractions, a 1D numpy array
        Y: SHG data, a 2D numpy array with the same number of
        rows as X
        Yerr: error bars for Y, a 2D numpy array that is the same
            size as Y
        *args: any additional positional arguments needed for
            model. Order must match the model.
        **kwargs: any additional keyword arguments needed for the
            model. Variable names must match the model.

    Outputs:
        sub_fit: the parameters for each subset, a numpy array
        sub_errs: the fit errors for each subset, a numpy array
        fit_values: the parameter values for the full dataset, a
            numpy array
    """
    #number of data points
    (m, n) = Y.shape

    #create the fit for the full dataset
    fit = minimizefits.lang_mol_frac_multiset(X, Y, Yerr, *args,
                                              **kwargs)

    #get ordered list of parameter values
    fit_values = datatools.extract_fit_values(fit)

    #need an initialized array before I can append >:(
    sub_fits = numpy.zeros_like(fit_values)
    sub_errs = numpy.zeros_like(fit_values)

    for i in range(m):
        for j in range(n):
            #change one value in Y to nan
            sub_Y = numpy.copy(Y)
            sub_Y[i, j] = numpy.nan

            #retry logic for OverflowError in fits
            MAX_RETRIES = 2
            new_args = list(args)
```

```
#try the fit with inital values of one
for retries in range(MAX_RETRIES):
    #print(retries)
    try:
        #passed args first time, modified second time
        sub_fit=minimizefits.lang_mol_frac_multiset(
                        X, sub_Y, Yerr, *new_args,
                        **kwargs)

    #if there's an overflow, try again with initial
    #values set to fit_values
    except OverflowError:
        #create new initial values from the full fit
        #values
        initial_new = list(fit_values)

        #replace the old initial values with new ones
        #remember that args also includes the func
        #variable and a variable number of parameters
        num_params = len(initial_new) // n

        #use the list to fill up all the variables
        #they can be different sizes
        #so element by element is the only way
        iterator = 0
        for p in range(num_params):
            for q in range(len(new_args[p+1])):
                new_args[p+1][q] = initial_new[
                                        iterator]
                iterator += 1

    #only does this if fit is valid and breaks loop
    #extracts fit values and errors
    else:
        column=datatools.extract_fit_values(sub_fit)
        col_err=datatools.extract_err_values(sub_fit)
        break

#does this when no valid fit was generated
#fill with nans and look at later
else:
    column = numpy.zeros_like(fit_values)
    column.fill(numpy.nan)
    col_err = numpy.zeros_like(fit_values)
```

```
                col_err.fill(numpy.nan)
                print("Subset {:d} caused an overflow error."
                      .format(i))

            #gather the sub fit info into one variable
            sub_fits = numpy.column_stack((sub_fits,column))
            sub_errs = numpy.column_stack((sub_errs,col_err))

    #delete the column of zeros from initialization
    sub_fits = numpy.delete(sub_fits, 0, axis=1)
    sub_errs = numpy.delete(sub_errs, 0, axis=1)

    return sub_fits, sub_errs, fit_values

def subfit_redo_multi_set(k, X, Y, Yerr, *args, **kwargs):
    """
    A function to redo one of the sub fits in case any fits seem
    erroneous.Used with jackknife_multi_set().

    Inputs:
        k: the number of the dataset to redo
            (the column index in the sub_fits variable)
        **the rest of the inputs are the same as for
                jackknife_multi_set()**
        X: concentration data in mole fractions, a 1D numpy array
        Y: SHG data, a 2D numpy array with the same number of
            rows as X
        Yerr: error bars for Y, a 2D numpy array that is the same
            size as Y
        *args: any additional positional arguments needed for
            model. Order must match the model.
        **kwargs: any additional keyword arguments needed for the
            model. Variable names must match the model.

    Outputs:
        sub_fit_new: the new set of parameter values
        sub_err_new: the new set of parameter errors
    """
    #get the data point that was removed
    m, n = Y.shape
    i = k // n
    j = k % n
    sub_Y = numpy.copy(Y)
    sub_Y[i, j] = numpy.nan
```

```
    #get the sub fit
    sub_fit = minimizefits.lang_mol_frac_multiset(X, sub_Y, Yerr,
                                                  *args, **kwargs)

    #get the sub values and sub errors
    sub_fit_new = datatools.extract_fit_values(sub_fit)
    sub_err_new = datatools.extract_err_values(sub_fit)

    return sub_fit_new, sub_err_new

def jackknife_calcs(sub_fits, fit_values):
    """
    Performs calulations based on the jackknife subfits.

    Inputs:
        sub_fits: the parameters for each subset, a numpy array
        fit_values: the fit values for the full data set, a list

    Outputs:
        estimate: the calculated jackknife parameters
        sdterr: the calculated jackknife errors
        est_corr: the bias corrected estimates
    """

    #shape of sub_fits
    num_params, num_data = sub_fits.shape

    #calculate jk estimate
    estimate = numpy.average(sub_fits, axis=1)

    #calculate residuals
    square_resids = (sub_fits.T - estimate).T ** 2
    sum_resids = numpy.sum(square_resids, axis=1)
    var = (num_data - 1) / num_data * sum_resids
    stderr = numpy.sqrt(var)

    #calculate the bias corrected estimates
    est_corr = num_data * numpy.array(fit_values)
               - (num_data - 1) * estimate

    return estimate, stderr, est_corr
```

# 6 Usage Examples

```
In [1]: %matplotlib inline
```

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        import datatools
        import fits
        import models
        import minimizefits
        import jackknife
```

## 6.1 Single dataset example

For data sets with only one Y, Yerr column pair, use the `models.py` module and `jackknife.jackknife_one_set()`.

Load the graphene data from Chapter 4.

```
In [3]: with open('graphene_data_sorted.csv') as file:
            load = pd.read_csv(file)
            data = load.values
        load
```

Out[3]:

| | Mol Frac | 193nm | 193err |
|---|---|---|---|
| 0 | 0.00152 | 0.82831 | 0.03773 |
| 1 | 0.00378 | 0.64777 | 0.14890 |
| 2 | 0.00413 | 0.88307 | 0.11317 |
| 3 | 0.00797 | 2.89190 | 0.43359 |
| 4 | 0.00830 | 3.45178 | 0.19691 |
| 5 | 0.01107 | 4.03015 | 1.03038 |
| 6 | 0.01192 | 4.13298 | 0.72800 |
| 7 | 0.01251 | 6.48660 | 2.49315 |
| 8 | 0.01496 | 8.87753 | 0.24343 |
| 9 | 0.01691 | 9.98051 | 0.64263 |
| 10 | 0.01902 | 12.63280 | 1.98260 |
| 11 | 0.02159 | 14.30370 | 1.13706 |
| 12 | 0.02350 | 12.30050 | 0.79586 |
| 13 | 0.02396 | 14.87840 | 2.68018 |
| 14 | 0.02591 | 20.79050 | 2.37478 |
| 15 | 0.03222 | 23.03210 | 3.35097 |
| 16 | 0.03271 | 26.37340 | 4.38724 |

```
        17   0.03784  30.00100  7.76124
        18   0.04201  42.68350  2.96554
```

```
In [4]: X = data[:,0]
        Y = data[:,1]
        Yerr = data[:,2]
```

Simple Langmuir model

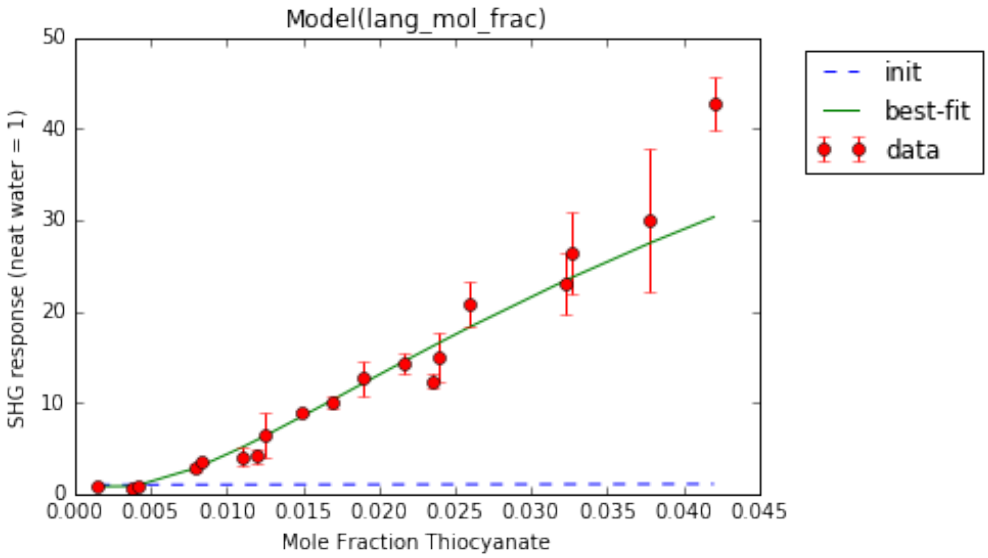
$$\frac{I_{2\omega}}{I_{\omega}^2} = \left(A + B' \frac{X_{SCN^-}}{(1 - X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2 + \left(C' \frac{X_{SCN^-}}{(1 - X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2$$

```
In [5]: initial = [1,1,1,1] #order is [A, B, C, G]
```

```
In [6]: fit = models.lang_mol_frac_fit(X, Y, Yerr, initial)
        plt.xlabel('Mole Fraction Thiocyanate')
        plt.ylabel('SHG response (neat water = 1)')
```

```
[[Model]]
    Model(lang_mol_frac)
[[Fit Statistics]]
    # function evals   = 151
    # data points      = 19
    # variables        = 4
    chi-square         = 58.391
    reduced chi-square = 3.893
    Akaike info crit   = 29.332
    Bayesian info crit = 33.109
[[Variables]]
    A:   1.21457115 +/- 0.073313 (6.04%) (init= 1)
    B:  -7.37693500 +/- 0.498521 (6.76%) (init= 1)
    C:   7.09623300 +/- 1.025682 (14.45%) (init= 1)
    G:  -8780.09329 +/- 382.2984 (4.35%) (init= 1)
    T:   293 (fixed)
    R:   8.314 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(C, G)                      =  0.975
    C(A, C)                      = -0.898
    C(A, G)                      = -0.826
    C(A, B)                      = -0.512
    C(B, C)                      =  0.244
```

```
Out[6]: <matplotlib.text.Text at 0x113ed9978>
```




Use `datatools.extract_fit_values()` and `datatools.extract_err_values()` to extract the parameter values and errors into variables.

```
In [7]: fit_values = datatools.extract_fit_values(fit)
        fit_errs = datatools.extract_err_values(fit)
        for item in zip(fit_values, fit_errs):
            print(item)
```

```
(1.2145711535307293, 0.073313399479016841)
(-7.3769350009735781, 0.49852126924926987)
(7.0962330039385488, 1.0256822436209414)
(-8780.0932975888154, 382.29848035741725)
```

Use `datatools.extract_corrs()` to extract the correlation matrix into a DataFrame. Unreported correlations are < 0.100.

```
In [8]: corrs_matrix = datatools.extract_corrs(fit)
        corrs_matrix
```

```
Out[8]:
          2
1         A         B         C         G
0
A       NaN -0.511753 -0.897563 -0.825834
B -0.511753       NaN  0.244406       NaN
C -0.897563  0.244406       NaN  0.974762
G -0.825834       NaN  0.974762       NaN
```

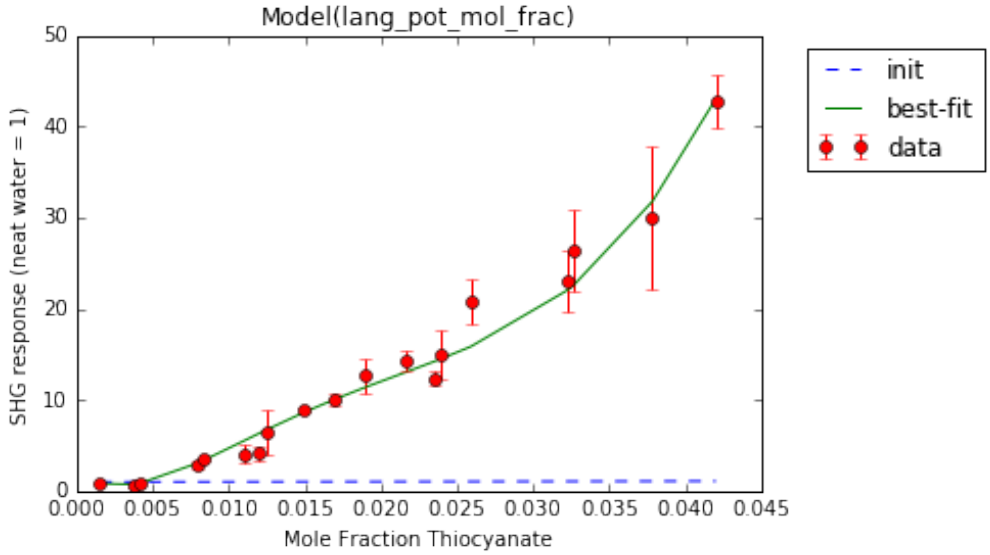
Langmuir model with surface potential term.

$$\frac{I_{2\omega}}{I_{\omega}^{2}} = \left(A + B'\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}} + D'\left(\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2\right)^2$$
$$+\left(C'\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}} + E'\left(\frac{X_{SCN^-}}{(1-X_{SCN^-})e^{\Delta G/RT} + X_{SCN^-}}\right)^2\right)^2$$

```
In [9]: initial = [1,1,1,1,1,1] #order is [A, B, C, D, E, G]

In [10]: fit = models.lang_pot_mol_frac_fit(X, Y, Yerr, initial)
         plt.xlabel('Mole Fraction Thiocyanate')
         plt.ylabel('SHG response (neat water = 1)')

[[Model]]
    Model(lang_pot_mol_frac)
[[Fit Statistics]]
    # function evals   = 308
    # data points      = 19
    # variables        = 6
    chi-square         = 25.566
    reduced chi-square = 1.967
    Akaike info crit   = 17.640
    Bayesian info crit = 23.307
[[Variables]]
    A:   1.36294785 +/- 0.107655 (7.90%) (init= 1)
    B:  -46.7350955 +/- 24.61348 (52.67%) (init= 1)
    C:   29.9149659 +/- 21.06090 (70.40%) (init= 1)
    D:   224.857829 +/- 245.3559 (109.12%) (init= 1)
    E:  -40.0893303 +/- 69.58432 (173.57%) (init= 1)
    G:  -5029.76623 +/- 1.72e+03 (34.12%) (init= 1)
    T:   293 (fixed)
    R:   8.314 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(C, E)                      = -0.998
    C(D, G)                      =  0.997
    C(C, G)                      =  0.997
    C(B, D)                      = -0.994
    C(C, D)                      =  0.992
    C(E, G)                      = -0.991
    C(D, E)                      = -0.986
    C(B, G)                      = -0.986
    C(B, C)                      = -0.974
    C(B, E)                      =  0.965
    C(A, E)                      =  0.866
    C(A, C)                      = -0.861
    C(A, G)                      = -0.835
```

```
    C(A, D)                          = -0.802
    C(A, B)                          =  0.738
```

```
Out[10]: <matplotlib.text.Text at 0x113ec7e10>
```




Use `datatools.extract_fit_values()` and `datatools.extract_err_values()` to extract the parameter values and errors into variables.

```
In [11]: fit_values = datatools.extract_fit_values(fit)
         fit_errs = datatools.extract_err_values(fit)
         for item in zip(fit_values, fit_errs):
             print(item)
```

```
(1.3629478542975455, 0.10765511100229698)
(-46.735095593092865, 24.613481768714191)
(29.914965988001157, 21.060906898845055)
(224.85782962537073, 245.35592777293945)
(-40.089330376751953, 69.584322494697787)
(-5029.7662389896768, 1716.1601978843578)
```

Use `jackknife.jackknife_one_set()` to perform the jackknife. It can be used with any model in the models.py file and should generalize to future models with model signature: `model(X, Y, Yerr,...)`.

Calculate the sub fits and sub errors.

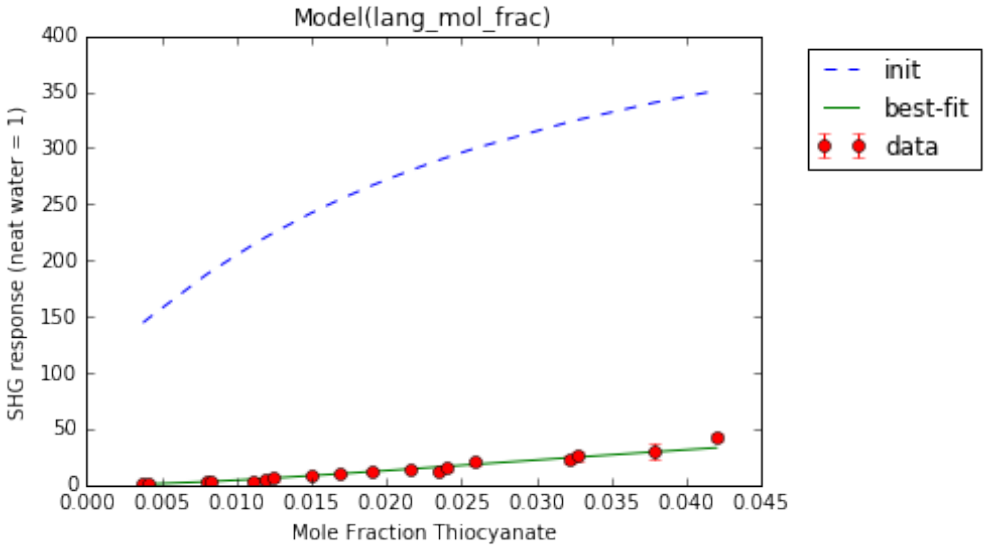
```
In [12]: sub_fits, sub_errs, fit_values =
        jackknife.jackknife_one_set( models.lang_mol_frac_fit,
                X, Y, Yerr, initial, False)
```

If one of the sub fits seems erroneous and needs to be recalculated, use the

`jackknife.subfit_redo_one_set()` function.

```
In [13]: i = 0
    initial_new = [10, 10, 10, -10000]
    sub_fit_new, sub_err_new = jackknife.subfit_redo_one_set(i,
                    models.lang_mol_frac_fit, X, Y, Yerr,
                    initial_new)
```

```
[[Model]]
    Model(lang_mol_frac)
[[Fit Statistics]]
    # function evals   = 122
    # data points      = 18
    # variables        = 4
    chi-square         = 49.634
    reduced chi-square = 3.545
    Akaike info crit   = 26.257
    Bayesian info crit = 29.819
[[Variables]]
    A:   0.00961593 +/- 13.03740 (135581.23%) (init= 10)
    B:   12.1936714 +/- 1.67e+04 (136894.27%) (init= 10)
    C:  -0.32063497 +/- 8.27e+05 (258080585.31%) (init= 10)
    G:  -7342.07935 +/- 539.7051 (7.35%) (init=-10000)
    R:   8.314 (fixed)
    T:   293 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(B, C)                      =  1.000
    C(A, C)                      = -1.000
    C(A, B)                      = -1.000
    C(B, G)                      =  0.400
    C(C, G)                      =  0.400
    C(A, G)                      = -0.394
```




Then these values can replace the column in `sub_fits` and `sub_errs`: `sub_fits[:,i] = sub_fit_new` and `sub_errs[:,i] = sub_err_new`

Calculate the parameter estimates, errors, and bias-corrected estimates.

```
In [14]: estimate, stderr, est_corr =
                jackknife.jackknife_calcs(sub_fits, fit_values)
```

```
In [15]: for item in zip(estimate, est_corr, stderr):
             print(item)
```

```
(1.1463899142272462, 2.4418334609934256, 1.1934790178429788)
(-7.5060089282163789, -5.0536043106031627, 2.891663267670463)
(7.3354984802708829, 2.7894544299565496, 4.9598441155718245)
(-8642.528644796721, -11256.257047846506, 2385.949068037296)
```

## 6.2 Multiple datasets

For data with multiple Y, Yerr column pairs (such as multiple temperatures or wavelengths), use `minimizefits.py` and `jackknife.jackknife_multi_set()`.

### 6.2.1 Datasets with no shared parameters

Load temperature data from Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.

```
In [16]: with open('dale_origin_data.csv') as file:
             load = pd.read_csv(file)
             data = load.values
         load
```

```
Out[16]:    Mol Frac    274K  274Kerr    283K  283Kerr    293K
          0  0.060460  86.400  1.69000  84.900  0.77000  80.500
          1  0.049540  74.000  2.32000  68.400  0.14100  65.900
          2  0.038990  59.500  1.95000  55.100  0.87300  51.500
          3  0.028250  41.000  0.37300  37.300  0.79600  33.500
          4  0.018900  23.800  0.27800  20.700  0.20600  18.800
          5  0.009250   8.960  0.29800   7.440  0.04290   6.310
          6  0.001830   1.420  0.00953   1.310  0.01070   1.220
          7  0.000184   0.983  0.01060   0.989  0.00278   0.979
          8  0.000018   0.991  0.01210   1.000  0.00272   0.990


             293Kerr    303K  303Kerr    313K  313Kerr
          0  0.36500  74.500  2.52000  70.200  2.23000
          1  1.57000  60.300  1.76000  51.900  1.06000
          2  0.91800  45.300  1.42000  40.500  0.23700
          3  0.81700  28.400  0.20400  26.600  0.18500
          4  0.10900  15.700  0.15500  13.200  0.29400
          5  0.04100   5.290  0.12200   4.540  0.04280
          6  0.00441   1.130  0.03440   1.110  0.00650
          7  0.00749   0.981  0.00595   0.995  0.00704
          8  0.01470   0.983  0.01020   0.993  0.01710
```

Use `datatools.parse_csv_data()` to easily take the above table and convert to X, Y, and Yerr variables.

```
In [17]: X, Y, Yerr = datatools.parse_csv_data(data)
```

Each parameter in the fitting function needs to be initialized with a list of length equal to the number of datasets. Here, it is five datasets.

```
In [18]: A = [1, 1, 1, 1, 1]
         B = [1, 1, 1, 1, 1]
         C = [1, 1, 1, 1, 1]
         G = [-5000, -5000, -5000, -5000, -5000]
         T = [274, 283, 293, 303, 313]
         func = fits.lang_mol_frac
```

The `minimizefits.lang_mol_frac_multiset()` function can be used with any fitting function in the `fits.py` module and should generalize. The `models.py` module uses the `fits.py` module internally. Here, it is called in the function call.

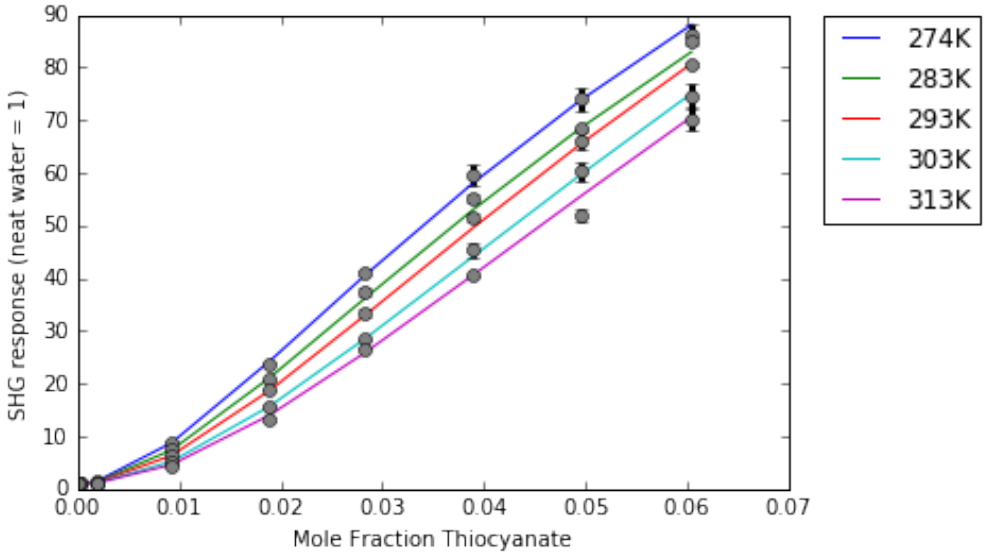
Simple Langmuir model.

```
In [19]: fit = minimizefits.lang_mol_frac_multiset(X, Y, Yerr,
                                                   func,A, B, C, G, T=T)
         plt.xlabel('Mole Fraction Thiocyanate')
         plt.ylabel('SHG response (neat water = 1)')
```

```
        plt.legend(labels=['274K','283K','293K','303K','313K'],
              bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
[[Variables]]
    R:     8.314 (fixed)
    A_1:   0.99110394 +/- 0.004373 (0.44%) (init= 1)
    A_2:   0.99724530 +/- 0.001110 (0.11%) (init= 1)
    A_3:   0.99232594 +/- 0.003701 (0.37%) (init= 1)
    A_4:   0.99199681 +/- 0.003190 (0.32%) (init= 1)
    A_5:   1.00214494 +/- 0.003619 (0.36%) (init= 1)
    B_1:   0.44590506 +/- 0.277481 (62.23%) (init= 1)
    B_2:  -0.33391959 +/- 0.204472 (61.23%) (init= 1)
    B_3:  -0.61838691 +/- 0.178988 (28.94%) (init= 1)
    B_4:  -1.10230528 +/- 0.751388 (68.17%) (init= 1)
    B_5:  -1.82979777 +/- 0.241977 (13.22%) (init= 1)
    C_1:   15.5474989 +/- 0.325040 (2.09%) (init= 1)
    C_2:   16.0060707 +/- 0.098077 (0.61%) (init= 1)
    C_3:   17.0942038 +/- 0.124659 (0.73%) (init= 1)
    C_4:   18.1406873 +/- 0.694663 (3.83%) (init= 1)
    C_5:   18.4920515 +/- 0.406092 (2.20%) (init= 1)
    G_1:  -7157.13515 +/- 87.18813 (1.22%) (init=-5000)
    G_2:  -7087.76391 +/- 32.62569 (0.46%) (init=-5000)
    G_3:  -6918.98859 +/- 31.62257 (0.46%) (init=-5000)
    G_4:  -6678.56301 +/- 147.4619 (2.21%) (init=-5000)
    G_5:  -6662.87511 +/- 84.54643 (1.27%) (init=-5000)
    T_1:   274 (fixed)
    T_2:   283 (fixed)
    T_3:   293 (fixed)
    T_4:   303 (fixed)
    T_5:   313 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(C_5, G_5)                  =  0.994
    C(C_4, G_4)                  =  0.990
    C(C_1, G_1)                  =  0.978
    C(C_2, G_2)                  =  0.973
    C(C_3, G_3)                  =  0.958
    C(A_3, B_3)                  = -0.838
    C(B_2, G_2)                  =  0.770
    C(A_5, B_5)                  = -0.755
    C(B_1, G_1)                  =  0.738
    C(B_4, G_4)                  =  0.664
    C(B_2, C_2)                  =  0.640
    C(B_3, G_3)                  =  0.629
    C(B_1, C_1)                  =  0.626
    C(A_4, B_4)                  = -0.593
```

```
    C(A_1, B_1)                    = -0.580
    C(B_5, G_5)                    =  0.578
    C(B_4, C_4)                    =  0.565
    C(B_5, C_5)                    =  0.525
    C(A_2, B_2)                    = -0.499
    C(B_3, C_3)                    =  0.457
    C(A_3, G_3)                    = -0.391
    C(A_4, G_4)                    = -0.391
    C(A_5, G_5)                    = -0.378
    C(A_5, C_5)                    = -0.344
    C(A_2, G_2)                    = -0.343
    C(A_4, C_4)                    = -0.334
    C(A_2, C_2)                    = -0.275
    C(A_3, C_3)                    = -0.257
    C(A_1, G_1)                    = -0.163
Out[19]: <matplotlib.legend.Legend at 0x113fc0e80>
```




Langmuir model with surface potential term.

```
In [20]: D = [1, 1, 1, 1, 1]
         E = [1, 1, 1, 1, 1]
         func = fits.lang_pot_mol_frac

In [21]: fit = minimizefits.lang_mol_frac_multiset(X, Y, Yerr,
                                  func, A, B, C, D, E, G, T=T)

OverflowError. Use new inital values.
```

In the event that the fitting function encounters values that are too large or too small, it

throws an `OverflowError`. The function will print out a response prompting you to choose new initial values.

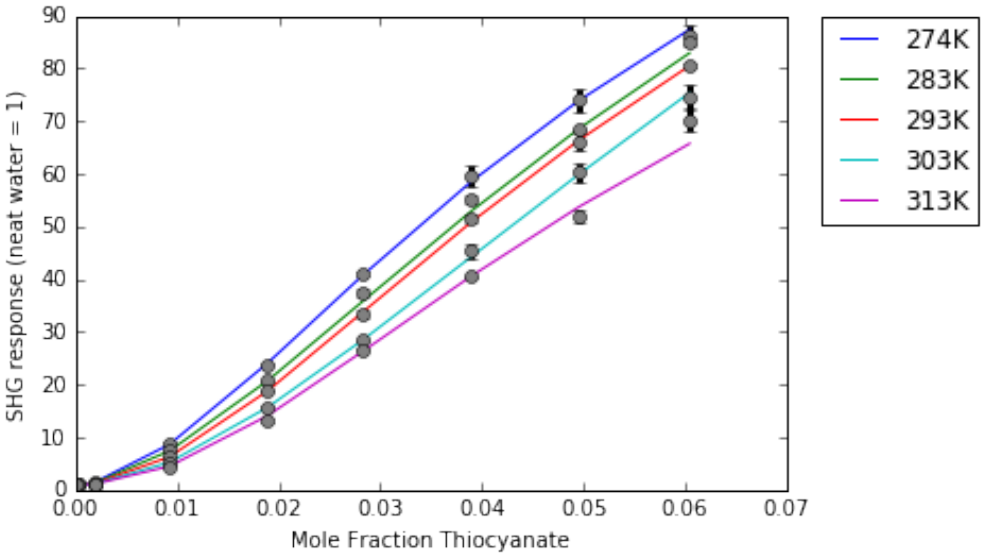
```
In [22]: G = [-10000, -10000, -10000, -10000, -10000]

In [23]: fit = minimizefits.lang_mol_frac_multiset(X, Y, Yerr,
                                 func, A, B, C, D, E, G, T=T)
         plt.xlabel('Mole Fraction Thiocyanate')
         plt.ylabel('SHG response (neat water = 1)')
         plt.legend(labels=['274K','283K','293K','303K','313K'],
             bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
[[Variables]]
    R:     8.314 (fixed)
    A_1:   0.99853557 +/- 0.006778 (0.68%) (init= 1)
    A_2:   1.00067677 +/- 0.001523 (0.15%) (init= 1)
    A_3:   0.99708340 +/- 0.007449 (0.75%) (init= 1)
    A_4:   0.99323442 +/- 0.004251 (0.43%) (init= 1)
    A_5:   1.00581269 +/- 0.008677 (0.86%) (init= 1)
    B_1:  -1.20806694 +/- 0.841545 (69.66%) (init= 1)
    B_2:  -1.07698128 +/- 0.236949 (22.00%) (init= 1)
    B_3:  -1.15005924 +/- 0.826099 (71.83%) (init= 1)
    B_4:  -0.95544960 +/- 0.660009 (69.08%) (init= 1)
    B_5:  -1.26891100 +/- 0.904327 (71.27%) (init= 1)
    C_1:   6.81165050 +/- 1.015459 (14.91%) (init= 1)
    C_2:   7.45746045 +/- 0.608821 (8.16%) (init= 1)
    C_3:   5.78640843 +/- 0.874010 (15.10%) (init= 1)
    C_4:   9.14604673 +/- 7.882020 (86.18%) (init= 1)
    C_5:   8.65747205 +/- 0.868773 (10.03%) (init= 1)
    D_1:   14.5643006 +/- 0.870708 (5.98%) (init= 1)
    D_2:   14.9212753 +/- 0.486822 (3.26%) (init= 1)
    D_3:   15.0031433 +/- 0.401931 (2.68%) (init= 1)
    D_4:   13.3725304 +/- 7.906370 (59.12%) (init= 1)
    D_5:  -13.3537176 +/- 1.248679 (9.35%) (init= 1)
    E_1:  -6.14728816 +/- 2.644143 (43.01%) (init= 1)
    E_2:  -3.87271355 +/- 1.529059 (39.48%) (init= 1)
    E_3:  -6.09691623 +/- 4.466606 (73.26%) (init= 1)
    E_4:   2.13162554 +/- 6.105663 (286.43%) (init= 1)
    E_5:  -8.29690871 +/- 5.781653 (69.68%) (init= 1)
    G_1:  -9323.50224 +/- 320.8468 (3.44%) (init=-10000)
    G_2:  -8957.65148 +/- 229.8494 (2.57%) (init=-10000)
    G_3:  -9579.75562 +/- 392.9832 (4.10%) (init=-10000)
    G_4:  -8227.54492 +/- 1.92e+03 (23.28%) (init=-10000)
    G_5:  -10303.5221 +/- 452.3032 (4.39%) (init=-10000)
    T_1:   274 (fixed)
    T_2:   283 (fixed)
```

```
    T_3:   293 (fixed)
    T_4:   303 (fixed)
    T_5:   313 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(C_4, G_4)                  =  0.997
    C(B_5, E_5)                  =  0.996
    C(D_4, E_4)                  = -0.995
    C(D_5, G_5)                  = -0.994
    C(E_3, G_3)                  =  0.992
    C(B_3, E_3)                  =  0.984
    C(B_5, C_5)                  = -0.976
    C(B_3, C_3)                  = -0.973
    C(C_5, E_5)                  = -0.970
    C(B_3, G_3)                  =  0.969
    C(D_5, E_5)                  = -0.967
    C(B_5, D_5)                  = -0.960
    C(E_5, G_5)                  =  0.956
    C(C_3, D_3)                  =  0.949
    C(C_3, E_3)                  = -0.941
    C(B_5, G_5)                  =  0.940
    C(C_1, D_1)                  =  0.931
    C(A_5, B_5)                  = -0.927
    C(A_5, E_5)                  = -0.909
    C(E_1, G_1)                  =  0.901
    C(C_3, G_3)                  = -0.898
    C(A_5, D_5)                  =  0.897
    C(D_4, G_4)                  = -0.895
    C(A_3, B_3)                  = -0.894
    C(B_1, D_1)                  = -0.893
    C(C_5, D_5)                  =  0.882
    C(A_5, C_5)                  =  0.881
    C(B_3, D_3)                  = -0.880
    C(B_2, D_2)                  = -0.873
    C(E_4, G_4)                  =  0.870
    C(A_5, G_5)                  = -0.866
    C(C_4, D_4)                  = -0.865
    C(C_5, G_5)                  = -0.857
    C(A_3, C_3)                  =  0.841
    C(A_3, G_3)                  = -0.840
    C(A_3, E_3)                  = -0.839
    C(C_4, E_4)                  =  0.837
    C(C_2, G_2)                  =  0.830
    C(A_3, D_3)                  =  0.814
    C(D_2, E_2)                  = -0.808
    C(D_3, E_3)                  = -0.805
```

```
C(A_1, B_1)                    = -0.800
C(B_1, E_1)                    =  0.799
C(B_2, E_2)                    =  0.793
C(E_2, G_2)                    =  0.763
C(A_2, B_2)                    = -0.748
C(B_1, C_1)                    = -0.747
C(D_3, G_3)                    = -0.740
C(A_1, D_1)                    =  0.675
C(A_1, E_1)                    = -0.630
C(A_2, E_2)                    = -0.615
C(A_4, D_4)                    =  0.609
C(A_4, E_4)                    = -0.597
C(B_1, G_1)                    =  0.593
C(A_2, D_2)                    =  0.577
C(A_4, B_4)                    = -0.577
C(D_1, E_1)                    = -0.571
C(A_1, G_1)                    = -0.522
C(A_2, G_2)                    = -0.517
C(A_4, G_4)                    = -0.512
C(A_1, C_1)                    =  0.510
C(B_2, G_2)                    =  0.503
C(A_4, C_4)                    = -0.471
C(C_2, E_2)                    =  0.299
C(C_2, D_2)                    =  0.294
C(C_1, E_1)                    = -0.276
C(D_2, G_2)                    = -0.275
C(B_4, E_4)                    =  0.244
C(B_4, D_4)                    = -0.232
C(D_1, G_1)                    = -0.220
C(A_2, C_2)                    = -0.169
C(B_4, C_4)                    = -0.124
```

```
Out[23]: <matplotlib.legend.Legend at 0x11765f630>
```




Use `jackknife.jackknife_multi_set()` to perform the jackknife. It can be used with `minimizefits.lang_mol_frac_multiset()` and should generalize well to other functions.

Using the simple Langmuir only.

```
In [24]: A = [1, 1, 1, 1, 1]
         B = [1, 1, 1, 1, 1]
         C = [1, 1, 1, 1, 1]
         G = [-5000, -5000, -5000, -5000, -5000]
         T = [274, 283, 293, 303, 313]
         func = fits.lang_mol_frac
```

```
In [25]: sub_fits, sub_errs, fit_values =
                jackknife.jackknife_multi_set(X, Y, Yerr, func,
                                              A, B, C, G, T=T,
                                              print_out=False,
                                              handle_error=False)
```

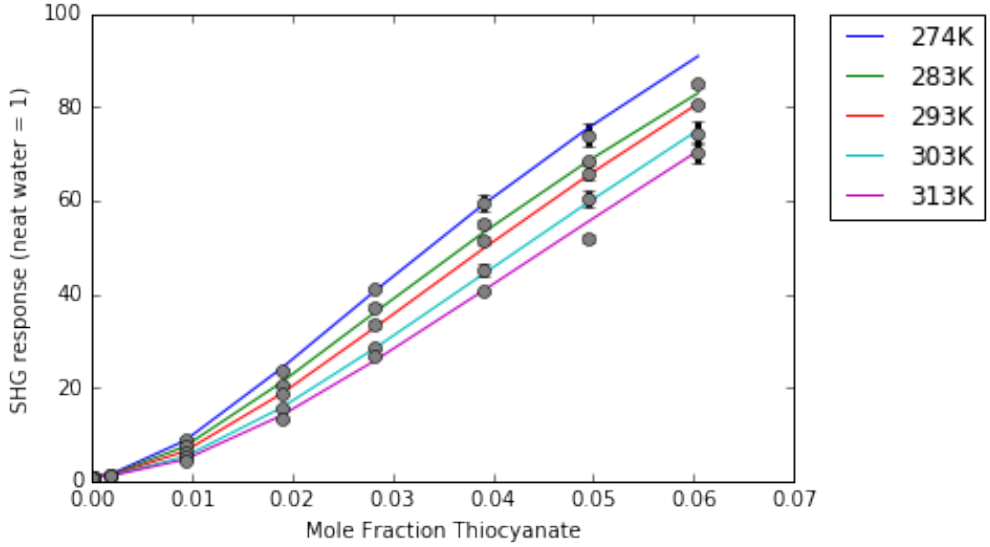
If one of the sub fits seems erroneous and needs to be recalculated, use the `jackknife.subfit_redo_multi_set()` function.

```
In [26]: k = 0
         G = [-1000, -1000, -1000, -1000, -1000]
         sub_fit_new, sub_err_new =
                              jackknife.subfit_redo_multi_set(k,
                              X, Y, Yerr, func, A, B, C, G, T=T)
```

```
[[Variables]]
```

```
    R:     8.314 (fixed)
    A_1:   0.99039539 +/- 0.004391 (0.44%) (init= 1)
    A_2:   0.99724529 +/- 0.001110 (0.11%) (init= 1)
    A_3:   0.99232594 +/- 0.003701 (0.37%) (init= 1)
    A_4:   0.99199680 +/- 0.003190 (0.32%) (init= 1)
    A_5:   1.00214480 +/- 0.003619 (0.36%) (init= 1)
    B_1:   0.75937503 +/- 0.358065 (47.15%) (init= 1)
    B_2:  -0.33391767 +/- 0.204472 (61.23%) (init= 1)
    B_3:  -0.61838687 +/- 0.178989 (28.94%) (init= 1)
    B_4:  -1.10230462 +/- 0.751395 (68.17%) (init= 1)
    B_5:  -1.82978357 +/- 0.241957 (13.22%) (init= 1)
    C_1:   16.2110636 +/- 0.550859 (3.40%) (init= 1)
    C_2:   16.0060713 +/- 0.098076 (0.61%) (init= 1)
    C_3:   17.0942038 +/- 0.124659 (0.73%) (init= 1)
    C_4:   18.1406882 +/- 0.694668 (3.83%) (init= 1)
    C_5:   18.4920925 +/- 0.406049 (2.20%) (init= 1)
    G_1:  -6997.49529 +/- 132.8867 (1.90%) (init=-1000)
    G_2:  -7087.76368 +/- 32.62568 (0.46%) (init=-1000)
    G_3:  -6918.98858 +/- 31.62258 (0.46%) (init=-1000)
    G_4:  -6678.56283 +/- 147.4624 (2.21%) (init=-1000)
    G_5:  -6662.86651 +/- 84.54204 (1.27%) (init=-1000)
    T_1:   274 (fixed)
    T_2:   283 (fixed)
    T_3:   293 (fixed)
    T_4:   303 (fixed)
    T_5:   313 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(C_5, G_5)                  =  0.994
    C(C_1, G_1)                  =  0.990
    C(C_4, G_4)                  =  0.990
    C(C_2, G_2)                  =  0.973
    C(C_3, G_3)                  =  0.958
    C(A_3, B_3)                  = -0.838
    C(B_1, G_1)                  =  0.825
    C(B_2, G_2)                  =  0.770
    C(B_1, C_1)                  =  0.760
    C(A_5, B_5)                  = -0.755
    C(B_4, G_4)                  =  0.664
    C(B_2, C_2)                  =  0.640
    C(B_3, G_3)                  =  0.629
    C(A_4, B_4)                  = -0.593
    C(B_5, G_5)                  =  0.578
    C(B_4, C_4)                  =  0.565
    C(A_1, B_1)                  = -0.533
    C(B_5, C_5)                  =  0.525
```

```
C(A_2, B_2)                    = -0.499
C(B_3, C_3)                    =  0.457
C(A_3, G_3)                    = -0.391
C(A_4, G_4)                    = -0.391
C(A_5, G_5)                    = -0.378
C(A_5, C_5)                    = -0.344
C(A_2, G_2)                    = -0.343
C(A_4, C_4)                    = -0.334
C(A_2, C_2)                    = -0.275
C(A_3, C_3)                    = -0.257
C(A_1, G_1)                    = -0.179
C(A_1, C_1)                    = -0.124
```




Then these values can replace the column in sub_fits and sub_errs: `sub_fits[:,i] = sub_fit_new` and `sub_errs[:,i] = sub_err_new`

Calculate the parameter estimates, errors, and bias-corrected estimates.

```
In [27]: estimate, stderr, est_corr =
                jackknife.jackknife_calcs(sub_fits, fit_values)

In [28]: for item in zip(estimate, est_corr, stderr):
             print(item)

(0.99118964191265069, 0.98733305985793862, 0.0071729487069151377)
(0.99724103836915956, 0.99743291956565372, 0.0058012653183970161)
(0.99245687588878695, 0.98656515710414538, 0.0082387802532657654)
(0.99197683815223925, 0.99287559357624389, 0.0013753104729033246)
(1.0023425590749866, 0.99344990649638021, 0.0089119961369202548)
(0.41215515136230552, 1.9309010878008479, 1.6411323766697861)
(-0.34291966543201269, 0.06208372374248583, 1.0241216007495333)
```

```
(-0.65155268756858342, 0.84090694512060438, 1.5512807112079243)
(-1.0951965694856927, -1.4150888102256687, 0.66240381708882079)
(-1.8900027749438146, 0.81922229223006582, 2.7700269935741382)
(15.547452736558595, 15.549533164787363, 0.83469836305666745)
(16.014787382788487, 15.622539699291792, 0.5228393219584504)
(17.103398539689405, 16.689636957651828, 1.1040236887461212)
(18.146413982989518, 17.888715467502948, 0.38534599003217712)
(18.424261846961393, 21.474798672598695, 2.9796277258646704)
(-7158.7286711928436, -7087.0204339167685, 246.04447171362531)
(-7086.2766674294926, -7153.2028370163753, 144.28670029372367)
(-6918.9258280410013, -6921.7502785508404, 241.79934845266177)
(-6677.3060520997515, -6733.8693803983042, 88.375173939878735)
(-6678.6353065573212, -5969.4265907790395, 673.55369686352446)
```

### 6.2.2 Datasets with one or more shared parameter

Load data from Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. *J Phys Chem B* **2005**, *109* (21), 10915–10921.

```
In [29]: with open('poul_origin_data_nascn.csv') as file:
             load = pd.read_csv(file)
             data = load.values
         load
```

Out[29]:

| | Molarity | 200nm | 200err | 212nm | 212err |
|---|---|---|---|---|---|
| 0 | 4.00000 | 217.95047 | 21.89505 | 91.25119 | 9.22512 |
| 1 | 3.00000 | 194.16100 | 19.51610 | 69.57555 | 7.05755 |
| 2 | 2.00000 | 135.90941 | 13.69094 | 45.42832 | 4.64283 |
| 3 | 1.00000 | 65.58297 | 6.65830 | 23.85600 | 2.48560 |
| 4 | 0.50000 | 25.38578 | 2.63858 | 9.99346 | 1.09935 |
| 5 | 0.25000 | 9.97127 | 1.09713 | 4.76236 | 0.57624 |
| 6 | 0.12500 | 4.36041 | 0.53604 | 2.45415 | 0.34542 |
| 7 | 0.06250 | 2.31932 | 0.33193 | 1.64823 | 0.26482 |
| 8 | 0.03125 | 1.52001 | 0.25200 | 1.18353 | 0.21835 |
| 9 | 0.01563 | 1.33945 | 0.23395 | 1.03818 | 0.20382 |
| 10 | 0.00782 | NaN | NaN | 1.01939 | 0.20194 |

| | 219nm | 219err | 227nm | 227err | 241nm | 241err |
|---|---|---|---|---|---|---|
| 0 | 58.35191 | 5.93519 | 11.99235 | 1.29924 | 4.67430 | 0.56743 |
| 1 | 44.27587 | 4.52759 | 9.34094 | 1.03409 | 3.31323 | 0.43132 |
| 2 | 28.90151 | 2.99015 | 5.87993 | 0.68799 | 2.21728 | 0.32173 |
| 3 | 12.45554 | 1.34555 | 3.33201 | 0.43320 | 1.45477 | 0.24548 |
| 4 | 5.76458 | 0.67646 | 2.02902 | 0.30290 | 1.03564 | 0.20356 |
| 5 | 2.59660 | 0.35966 | 1.27060 | 0.22706 | 0.86803 | 0.18680 |
| 6 | 1.75263 | 0.27526 | 1.05313 | 0.20531 | 0.86250 | 0.18625 |

```
7  1.20572  0.22057   1.01610  0.20161  0.85649  0.18565
8  1.00112  0.20011   0.96178  0.19618  0.90994  0.19099
9  0.88356  0.18836   1.00476  0.20048  0.96695  0.19669
10 0.88551  0.18855   1.14910  0.21491  0.87871  0.18787
```

```
In [30]: X, Y, Yerr = datatools.parse_csv_data(data)

         #X in molarity. Need mol frac
         X = X * 18.02 / 0.9982067 / 1000
         X = np.array([x / (1 + x) for x in X])
```

Simple Langmuir model.

G and T are both shared here, so they are scalars instead of lists.

```
In [31]: A = [1, 1, 1, 1, 1]
         B = [1, 1, 1, 1, 1]
         C = [1, 1, 1, 1, 1]
         G = -5000
         T = 293
         func = fits.lang_mol_frac
```

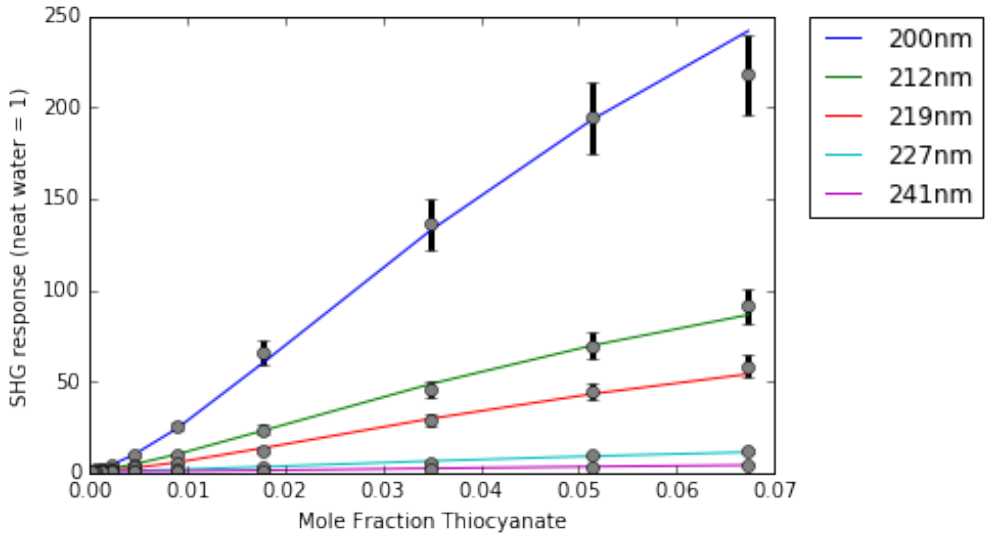
```
In [32]: fit = minimizefits.lang_mol_frac_multiset(X, Y, Yerr,
                                func, A, B, C, G, T=T)
         plt.xlabel('Mole Fraction Thiocyanate')
         plt.ylabel('SHG response (neat water = 1)')
         plt.legend(labels=[
             '200nm','212nm','219nm','227nm','241nm'],
             bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
[[Variables]]
    R:     8.314 (fixed)
    A_1:   0.97074452 +/- 0.065885 (6.79%) (init= 1)
    A_2:   0.95057720 +/- 0.071437 (7.52%) (init= 1)
    A_3:   0.94540591 +/- 0.063009 (6.66%) (init= 1)
    A_4:   1.00783250 +/- 0.054194 (5.38%) (init= 1)
    A_5:   0.96178162 +/- 0.050229 (5.22%) (init= 1)
    B_1:   23.6084216 +/- 1.838475 (7.79%) (init= 1)
    B_2:   12.5996696 +/- 4.737637 (37.60%) (init= 1)
    B_3:   3.97263220 +/- 2.785257 (70.11%) (init= 1)
    B_4:   0.26218028 +/- 1.126889 (429.81%) (init= 1)
    B_5:  -0.85368560 +/- 0.717996 (84.11%) (init= 1)
    C_1:   0.00620139 +/- 1.57e+04 (252494721.46%) (init= 1)
    C_2:   5.17341000 +/- 11.99342 (231.83%) (init= 1)
    C_3:   10.5578898 +/- 1.409116 (13.35%) (init= 1)
    C_4:   5.11124640 +/- 0.528884 (10.35%) (init= 1)
    C_5:   3.25897094 +/- 0.290538 (8.92%) (init= 1)
    G:    -7568.72125 +/- 285.6773 (3.77%) (init=-5000)
```

```
    T:     293 (fixed)
[[Correlations]] (unreported correlations are <  0.100)
    C(B_2, C_2)                  = -0.982
    C(B_1, G)                    =  0.861
    C(B_3, C_3)                  = -0.837
    C(B_4, C_4)                  = -0.765
    C(A_2, C_2)                  =  0.726
    C(A_2, B_2)                  = -0.706
    C(B_5, C_5)                  = -0.688
    C(A_3, B_3)                  = -0.671
    C(A_4, B_4)                  = -0.656
    C(B_2, G)                    =  0.633
    C(A_5, B_5)                  = -0.630
    C(A_3, C_3)                  =  0.624
    C(B_3, G)                    =  0.571
    C(B_1, B_2)                  =  0.545
    C(A_1, G)                    =  0.498
    C(C_2, G)                    = -0.496
    C(B_1, B_3)                  =  0.491
    C(A_4, C_4)                  =  0.486
    C(C_5, G)                    =  0.470
    C(B_1, C_1)                  =  0.469
    C(B_1, C_2)                  = -0.427
    C(B_1, C_5)                  =  0.404
    C(B_2, B_3)                  =  0.361
    C(A_5, C_5)                  =  0.340
    C(B_4, G)                    =  0.336
    C(A_1, B_2)                  =  0.315
    C(A_1, B_1)                  =  0.302
    C(B_2, C_5)                  =  0.297
    C(B_1, B_4)                  =  0.290
    C(A_1, B_3)                  =  0.284
    C(B_3, C_2)                  = -0.283
    C(B_3, C_5)                  =  0.268
    C(C_4, G)                    =  0.249
    C(A_1, C_2)                  = -0.247
    C(A_1, C_5)                  =  0.234
    C(C_2, C_5)                  = -0.233
    C(B_1, C_4)                  =  0.215
    C(B_2, B_4)                  =  0.213
    C(B_3, B_4)                  =  0.192
    C(A_3, G)                    = -0.189
    C(A_2, G)                    = -0.181
    C(A_1, B_4)                  =  0.167
    C(B_4, C_2)                  = -0.167
```

```
C(A_3, B_1)                    = -0.163
C(B_4, C_5)                    =  0.158
C(B_2, C_4)                    =  0.158
C(A_2, B_1)                    = -0.156
C(B_3, C_4)                    =  0.142
C(B_5, G)                      =  0.132
C(A_1, C_4)                    =  0.124
C(C_2, C_4)                    = -0.124
C(A_3, B_2)                    = -0.120
C(C_4, C_5)                    =  0.117
C(B_1, B_5)                    =  0.114
C(C_1, G)                      =  0.113
C(A_4, G)                      = -0.112
C(A_2, B_3)                    = -0.103
C(A_1, C_1)                    =  0.100
```

```
Out[32]: <matplotlib.legend.Legend at 0x117cbf048>
```




# 7 Notes on future use

The `lmfit` module is very powerful and has many more options available for customization. For example, the fit objects also have a method for plotting a residual plot and the parameters can have bounds. The minimizefits.py module in particular has several functions that are called by minimizefits.lang_mol_frac_multiset() which can be used individually, if necessary.

# Appendix 6 – Error Analysis Full Results

## A6.1 Model comparisons

"frac_lmf" is the fraction of correlations above 0.1 that were 0.9 or greater for Equation (2.18), the simple Langmuir model. "frac_lpmf" is the fraction of correlations above 0.1 that were 0.9 or greater for Equation (5.13), the Langmuir model with surface potential.

### *A6.1.1 NaSCN graphene/water*

| | Value | Result |
|---|---|---|
| **F-test** | 0.004662381 | evidence for LSP |
| **AIC** | 11.69174609 | no support for SL over LSP |
| **BIC** | 9.802868131 | little support for SL over LSP |
| **AICc** | 7.548888946 | some support for SL over LSP |
| **frac_lmf** | 0.2 | support for SL |
| **frac_lpmf** | 0.6666 | |

### *A6.1.2 KSCN temperature dependence*[1]

| | Value | Result |
|---|---|---|
| **F-test** | 0.176363705 | no evidence for LSP |
| **AIC** | 13.80597374 | no support for SL over LSP |
| **BIC** | -4.260651161 | some support for LSP over SL |
| **AICc** | -84.05116912 | no support for LSP over SL |
| **frac_lmf** | 0.172413793 | support for SL |
| **frac_lpmf** | 0.236111111 | |

### *A6.1.3 NaSCN dodecanol/water*[2]

| | Value | Result |
|---|---|---|
| **F-test** | 0.026450721 | evidence for LSP |
| **AIC** | 6.686053608 | some support for SL over LSP |
| **BIC** | -5.880013765 | some support for LSP over SL |
| **AICc** | -1.475313264 | substantial support for LSP over SL |
| **frac_lmf** | 0.212121212 | support for LSP |
| **frac_lpmf** | 0.132352941 | |

*A6.1.4 NaSCN < 4M dodecanol/water*[2]

| | Value | Result |
|---|---|---|
| **F-test** | 0.039592985 | evidence for LSP |
| **AIC** | 6.418925522 | some support for SL over LSP |
| **BIC** | -5.05321251 | some support for LSP over SL |
| **AICc** | -4.424897322 | some support for LSP over SL |
| **frac_lmf** | 0.030303030 | support for SL |
| **frac_lpmf** | 0.182795698 | |

*A6.1.5 KSCN dodecanol/water*[2]

| | Value | Result |
|---|---|---|
| **F-test** | 0.470415594 | no evidence for LSP |
| **AIC** | -3.178140601 | some support for LSP over SL |
| **BIC** | -9.72848524 | little support for LSP over SL |
| **AICc** | -9.598653421 | little support for LSP over SL |
| **frac_lmf** | 0.076923076 | support for LSP |
| **frac_lpmf** | 0.05 | |

*A6.1.6 KSCN < 4M dodecanol/water*[2]

| | Value | Result |
|---|---|---|
| **F-test** | 0.60133765 | no evidence for LSP |
| **AIC** | -3.874757192 | some support for LSP over SL |
| **BIC** | -9.479546719 | little support for LSP over SL |
| **AICc** | -13.45051477 | no support for LSP over SL |
| **frac_lmf** | 0.058823529 | support for SL |
| **frac_lpmf** | 0.255813953 | |

*A6.1.7 NaSCN air/water*[3]

| | Value | Result |
|---|---|---|
| **F-test** | 5.09E-06 | evidence for LSP |
| **AIC** | 53.78832625 | no support for SL over LSP |
| **BIC** | 33.89848579 | no support for SL over LSP |
| **AICc** | 16.49102896 | no support for SL over LSP |
| **frac_lmf** | 0.024691358 | support for SL |
| **frac_lpmf** | 0.051401869 | |

## A6.2 Jackknife calculations

All $\Delta G_{ads}$ values are presented in units of J/mol.

### *A6.2.1 NaSCN graphene/water*

*A6.2.1.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A** | 1.214571154 | 1.146389914 | 2.441833461 | 0.073313399 | 1.193479018 |
| **B** | -7.376935001 | -7.506008928 | -5.053604311 | 0.498521269 | 2.891663268 |
| **C** | 7.096233004 | 7.33549848 | 2.78945443 | 1.025682244 | 4.959844116 |
| **G** | -8780.093298 | -8642.528645 | -11256.25705 | 382.2984804 | 2385.949068 |

*A6.2.1.2 Langmuir with surface potential*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A** | 1.362947854 | 1.357337044 | 1.463942437 | 0.107655111 | 0.204498606 |
| **B** | -46.73509559 | -48.27231634 | -19.06512213 | 24.61348177 | 48.00917313 |
| **C** | 29.91496599 | 31.2385991 | 6.089569913 | 21.0609069 | 37.40254043 |
| **D** | 224.8578296 | 254.9998424 | -317.6984001 | 245.3559278 | 514.6220773 |
| **E** | -40.08933038 | -48.84691788 | 117.5472446 | 69.58432249 | 136.5415953 |
| **G** | -5029.766239 | -5022.919115 | -5153.014475 | 1716.160198 | 3155.499825 |

### *A6.2.2 KSCN temperature dependence*[1]

*A6.2.2.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 0.991103937 | 0.991189643 | 0.987332874 | 0.004373219 | 0.007172971 |
| **A2** | 0.997245303 | 0.99724104 | 0.997432867 | 0.001110788 | 0.005801247 |
| **A3** | 0.992325949 | 0.992456876 | 0.986565166 | 0.003701974 | 0.008238778 |
| **A4** | 0.991996812 | 0.99197684 | 0.992875594 | 0.003190564 | 0.001375308 |
| **A5** | 1.002144929 | 1.002342544 | 0.993449895 | 0.00361973 | 0.008911921 |
| **B1** | 0.445905365 | 0.412154689 | 1.930935086 | 0.277482029 | 1.641150788 |
| **B2** | -0.333919697 | -0.342920139 | 0.06209977 | 0.204472442 | 1.024120046 |
| **B3** | -0.618386957 | -0.651552718 | 0.840906507 | 0.178988922 | 1.551280589 |
| **B4** | -1.102305944 | -1.095197245 | -1.415088683 | 0.751383178 | 0.662402429 |
| **B5** | -1.829796216 | -1.890000901 | 0.819209898 | 0.241975228 | 2.77000319 |
| **C1** | 15.5474992 | 15.54745259 | 15.54954992 | 0.325040045 | 0.834700293 |
| **C2** | 16.00607074 | 16.01478724 | 15.62254455 | 0.09807705 | 0.522839325 |
| **C3** | 17.09420382 | 17.10339851 | 16.68963757 | 0.124659856 | 1.104022831 |
| **C4** | 18.14068664 | 18.14641305 | 17.88872473 | 0.69465924 | 0.385345925 |
| **C5** | 18.49205601 | 18.42426634 | 21.47480161 | 0.406087885 | 2.979608225 |

| | | | | | |
|---|---|---|---|---|---|
| **G1** | -7157.135082 | -7158.72872 | -7087.015025 | 87.18815028 | 246.0455197 |
| **G2** | -7087.763928 | -7086.276725 | -7153.200872 | 32.62569516 | 144.2866315 |
| **G3** | -6918.988599 | -6918.925835 | -6921.750199 | 31.62257332 | 241.7992144 |
| **G4** | -6678.563171 | -6677.306254 | -6733.867514 | 147.4615566 | 88.37512431 |
| **G5** | -6662.874178 | -6678.634361 | -5969.42613 | 84.54595042 | 673.5489972 |

*A6.2.2.2 Langmuir with surface potential*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 0.998535576 | 1.002175418 | 0.838382533 | 0.006778039 | 0.152984055 |
| **A2** | 1.000676913 | 1.00049536 | 1.00866521 | 0.00152388 | 0.008164566 |
| **A3** | 0.997083366 | 0.997086734 | 0.996935175 | 0.00744941 | 0.006195142 |
| **A4** | 0.993234343 | 0.993217584 | 0.99397174 | 0.004252014 | 0.002742015 |
| **A5** | 1.005810636 | 1.010152197 | 0.81478193 | 0.008678229 | 0.20194191 |
| **B1** | -1.208066711 | -1.598352424 | 15.96450465 | 0.841546612 | 12.45056594 |
| **B2** | -1.077001465 | -1.056406446 | -1.983182292 | 0.236937006 | 1.570269629 |
| **B3** | -1.150054525 | -1.154320993 | -0.962329962 | 0.826096372 | 0.761600916 |
| **B4** | -0.955451306 | -0.956482826 | -0.910064449 | 0.65996588 | 0.578022107 |
| **B5** | -1.268678152 | -1.71901089 | 18.54596233 | 0.904273388 | 15.45051824 |
| **C1** | 6.811650349 | 6.994005407 | -1.211972188 | 1.015460105 | 4.942018616 |
| **C2** | 7.457372542 | 8.840732213 | -53.41045295 | 0.60849404 | 38.29829851 |
| **C3** | 5.786403837 | 5.791839742 | 5.547224015 | 0.874027286 | 0.927003259 |
| **C4** | 9.146364405 | 9.160699476 | 8.515621245 | 7.879093875 | 5.91213358 |
| **C5** | 4.598955105 | 4.900519695 | -8.66988687 | 1.866279108 | 7.321948916 |
| **D1** | 14.56430043 | 14.81235726 | 3.649799862 | 0.870708819 | 11.60583684 |
| **D2** | 14.92129717 | 15.81126684 | -24.23736852 | 0.486541319 | 28.24941571 |
| **D3** | 15.00314158 | 14.98273466 | 15.90104627 | 0.401893507 | 0.779351899 |
| **D4** | 13.37224617 | 13.43501024 | 10.61062728 | 7.904271637 | 3.406022201 |
| **D5** | 13.38949894 | 13.1369955 | 24.49965057 | 0.629108546 | 24.54358151 |
| **E1** | -6.147287532 | -6.435831271 | 6.548636994 | 2.644145649 | 12.56461498 |
| **E2** | -3.872912656 | -5.537866869 | 69.38507271 | 1.528709695 | 52.7022984 |
| **E3** | -6.096890023 | -6.202948993 | -1.430295347 | 4.466212215 | 4.261975994 |
| **E4** | 2.131838747 | 2.002500795 | 7.82270863 | 6.106798569 | 3.619658378 |
| **E5** | -8.238068821 | -8.243303892 | -8.007725706 | 4.274949081 | 21.95146127 |
| **G1** | -9323.502174 | -9385.292498 | -6604.727905 | 320.8470737 | 1817.628843 |
| **G2** | -8957.691156 | -8755.323712 | -17861.8587 | 229.7565321 | 5360.203418 |
| **G3** | -9579.753296 | -9589.609133 | -9146.096464 | 392.9626736 | 358.5915531 |
| **G4** | -8227.467556 | -8231.338989 | -8057.124483 | 1915.020529 | 1278.077884 |
| **G5** | -10303.39851 | -10407.74978 | -5711.94275 | 452.0785478 | 3072.067494 |

### *A6.2.3 NaSCN dodecanol/water²*

*A6.2.3.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 0.996302006 | 0.996104324 | 1.008558282 | 0.010225997 | 0.015156582 |
| **A2** | 0.992608471 | 0.992355704 | 1.008279992 | 0.008941815 | 0.022295755 |
| **A3** | 1.000348241 | 1.000366061 | 0.999243375 | 0.009191646 | 0.00229599 |
| **B1** | 5.223242397 | 5.272256818 | 2.184348281 | 3.571414128 | 3.414289482 |
| **B2** | 1.660875305 | 1.680132968 | 0.466900215 | 1.152986874 | 1.367096567 |
| **B3** | 0.241964721 | 0.244998386 | 0.053877502 | 0.420247025 | 0.353687578 |
| **C1** | 3.602915059 | 3.510973457 | 9.30329437 | 2.123812606 | 3.928874094 |
| **C2** | 2.61953624 | 2.601685988 | 3.726251872 | 0.650239956 | 1.39499542 |
| **C3** | 3.14426194 | 3.157990624 | 2.293083561 | 1.067162996 | 1.002202865 |
| **G** | -3228.283628 | -3216.671912 | -3948.210001 | 1391.115792 | 1301.862555 |

*A6.2.3.2 Langmuir with surface potential*

| | Fit values | Fit error |
|---|---|---|
| **A1** | 1.00018656 | 0.01099976 |
| **A2** | 0.998271852 | 0.009781721 |
| **A3** | 0.998808308 | 0.009857809 |
| **B1** | 0.274347763 | 1.156999012 |
| **B2** | -0.021781893 | 0.546680571 |
| **B3** | 0.237229796 | 0.612217428 |
| **C1** | 6.320812724 | 1.417292381 |
| **C2** | 1.535866173 | 2.678397305 |
| **C3** | 1.113737007 | 2460.684677 |
| **D1** | -7.333805526 | 2.79500974 |
| **D2** | 1.62049533 | 1.150025653 |
| **D3** | -1.207242098 | 2760.291764 |
| **E1** | -4.601321814 | 2.771225203 |
| **E2** | -1.708948206 | 5.214647476 |
| **E3** | 1.381538952 | 2439.108713 |
| **G** | -6650.924197 | 1525.31116 |

### *A6.2.4 NaSCN < 4M dodecanol/water²*

*A6.2.4.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 0.997367722 | 0.99703312 | 1.014097808 | 0.010551718 | 0.018807276 |
| **A2** | 0.997708328 | 0.997518106 | 1.007219431 | 0.009344384 | 0.010896222 |

| | | | | | |
|---|---|---|---|---|---|
| **A3** | 1.000735109 | 1.00079448 | 0.997766551 | 0.009490078 | 0.004825598 |
| **B1** | 1.209741573 | 1.24499028 | -0.552693766 | 1.086194412 | 1.438955553 |
| **B2** | 0.115926207 | 0.125130065 | -0.344266701 | 0.305161405 | 0.343567166 |
| **B3** | -0.029115088 | -0.026816038 | -0.14406758 | 0.193875041 | 0.168766501 |
| **C1** | 3.075431946 | 1.262743232 | 93.70986762 | 0.353132942 | 19.81512555 |
| **C2** | 2.173821686 | 2.185541466 | 1.587832649 | 0.51633606 | 0.591162374 |
| **C3** | 1.508436055 | 1.516681548 | 1.096161396 | 0.412293782 | 0.689987609 |
| **G** | -6203.22923 | -6172.838259 | -7722.777755 | 1272.316194 | 1462.732136 |

*A6.2.4.2 Langmuir with surface potential*

| | **Fit values** | **JK estimate** | **JK bias-corr values** | **Fit error** | **JK error** |
|---|---|---|---|---|---|
| **A1** | 0.998796087 | 0.998568319 | 1.010184497 | 0.010836002 | 0.012740154 |
| **A2** | 0.998195777 | 0.997948786 | 1.010545299 | 0.009831949 | 0.014243295 |
| **A3** | 1.000149141 | 1.000190476 | 0.998082434 | 0.009977329 | 0.0023314 |
| **B1** | 1.504956102 | 1.627691403 | -4.631808967 | 4.047610054 | 4.435857166 |
| **B2** | -0.002257792 | 0.008391023 | -0.534698516 | 1.173114223 | 0.705063887 |
| **B3** | -0.483625565 | -0.50404197 | 0.537194647 | 1.146553738 | 0.826047254 |
| **C1** | 5.59592842 | 5.551124787 | 7.836110063 | 12.55034181 | 16.97360922 |
| **C2** | 3.45075592 | 3.690136559 | -8.518276053 | 9.289731128 | 8.132178895 |
| **C3** | -4.867401423 | -4.06645096 | -44.91492461 | 6.522547761 | 22.94638917 |
| **D1** | 3.270968377 | 3.259675109 | 3.835631764 | 11.94690923 | 10.14161612 |
| **D2** | 3.775589566 | 4.092796973 | -12.0847808 | 12.37678072 | 19.79570981 |
| **D3** | 3.276361803 | 3.694731445 | -17.64212029 | 8.039577676 | 14.71922068 |
| **E1** | -8.804018028 | -9.143591513 | 8.174656201 | 46.80348229 | 64.48048746 |
| **E2** | -7.805085711 | -9.126901511 | 58.2857043 | 37.80956789 | 59.68705644 |
| **E3** | 16.57908666 | 14.88529092 | 101.268874 | 48.02210558 | 108.0998872 |
| **G** | -5001.479851 | -4913.936119 | -9378.666431 | 5740.88291 | 3373.134222 |

## *A6.2.5 KSCN dodecanol/water[2]*

*A6.2.5.1 Simple Langmuir*

| | **Fit values** | **JK estimate** | **JK bias-corr values** | **Fit error** | **JK error** |
|---|---|---|---|---|---|
| **A1** | 1.000317057 | 1.000365864 | 0.99851118 | 0.015153506 | 0.00171595 |
| **A2** | 0.995509034 | 0.995239768 | 1.005471882 | 0.013118715 | 0.014023486 |
| **B1** | 5.315025728 | 5.293072753 | 6.127285774 | 1.023097963 | 0.665607684 |
| **B2** | 0.495982463 | 0.497613504 | 0.435633963 | 0.383374415 | 0.245375277 |
| **C1** | -0.000497718 | -0.090326245 | 3.323157757 | 1403.461544 | 2.41019354 |
| **C2** | 2.238866657 | 2.229526999 | 2.584434017 | 0.703515484 | 0.374316925 |
| **G** | -3471.759102 | -3479.881189 | -3171.241871 | 624.6316365 | 283.9235848 |

*A6.2.5.2 Langmuir with surface potential*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 1.00072271 | 1.00120603 | 0.98283986 | 0.016856155 | 0.018706811 |
| **A2** | 0.998977191 | 0.998616675 | 1.012316269 | 0.014540534 | 0.014463619 |
| **B1** | 0.736382771 | 0.729993746 | 0.972776708 | 2.060142074 | 0.773705416 |
| **B2** | -0.310494786 | -0.303833638 | -0.556957263 | 0.674849281 | 0.324621635 |
| **C1** | -2.675021482 | -2.700935887 | -1.716188461 | 2.914324626 | 1.125277317 |
| **C2** | 2.004988907 | 1.9957573 | 2.346558393 | 1.897054618 | 0.775287709 |
| **D1** | 2.191574234 | 2.217569721 | 1.229741204 | 2.205397479 | 0.89983391 |
| **D2** | 1.174684873 | 1.168105808 | 1.418110254 | 1.211931589 | 0.520121834 |
| **E1** | 5.728231406 | 5.756405382 | 4.685794282 | 2.217300707 | 1.263945903 |
| **E2** | -3.317688195 | -3.319161167 | -3.263188239 | 2.691473472 | 1.024313236 |
| **G** | -7263.343302 | -7252.683823 | -7657.744044 | 2505.458805 | 986.0474665 |

## *A6.2.6 KSCN < 4M dodecanol/water*[2]

*A6.2.6.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 1.000168917 | 1.000214829 | 0.998837459 | 0.016352093 | 0.001942521 |
| **A2** | 0.996429014 | 0.995973132 | 1.009649579 | 0.013685834 | 0.016584579 |
| **B1** | 2.119674617 | 2.147746168 | 1.305599645 | 3.539552109 | 1.59683946 |
| **B2** | 0.120400108 | 0.128992979 | -0.128793164 | 0.437954394 | 0.299866308 |
| **C1** | 2.679138683 | 2.652903302 | 3.439964731 | 1.385174539 | 0.736552334 |
| **C2** | 1.529106153 | 1.523586834 | 1.689166408 | 1.005638866 | 0.480464921 |
| **G** | -5538.504273 | -5525.598583 | -5912.769287 | 3070.723178 | 1380.680681 |

*A6.2.6.2 Langmuir with surface potential*

| | Fit values | Fit error |
|---|---|---|
| **A1** | 1.000369194 | 0.016842927 |
| **A2** | 0.999124132 | 0.014714954 |
| **B1** | 1.004217665 | 5.069224524 |
| **B2** | -0.369649484 | 1.043775403 |
| **C1** | 1.764473497 | 7.438753539 |
| **C2** | 2.3045937 | 6.599871647 |
| **D1** | 1.882844989 | 3.567920287 |
| **D2** | 1.35311609 | 5.073054757 |
| **E1** | -3.752616086 | 9.82326334 |
| **E2** | -3.973266347 | 17.33342232 |
| **G** | -7059.323138 | 9987.47835 |

### *A6.2.7 NaSCN air/water*[3]

*A6.2.7.1 Simple Langmuir*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 0.970772452 | 0.970632784 | 0.978314541 | 0.121111498 | 0.034367807 |
| **A2** | 0.950520857 | 0.950712933 | 0.940148737 | 0.072426198 | 0.017848292 |
| **A3** | 0.945393447 | 0.945383205 | 0.945946514 | 0.063955113 | 0.022294147 |
| **A4** | 1.007827441 | 1.007814548 | 1.008523635 | 0.054478111 | 0.027177245 |
| **A5** | 0.961778966 | 0.961757652 | 0.962929925 | 0.050373275 | 0.01639255 |
| **B1** | 23.60966013 | 23.61507564 | 23.3172223 | 12.01756023 | 1.844332271 |
| **B2** | 12.60526234 | 12.59413305 | 13.20624427 | 5.483862105 | 2.31312249 |
| **B3** | 3.974072938 | 3.978861191 | 3.715507274 | 3.145830439 | 1.662932531 |
| **B4** | 0.262493346 | 0.262478212 | 0.263310607 | 1.179639532 | 0.747781347 |
| **B5** | -0.853627381 | -0.853159148 | -0.878911974 | 0.723331268 | 0.429417194 |
| **C1** | -0.161297048 | -0.018519892 | -7.871263481 | 12954.99967 | 0.35665245 |
| **C2** | -5.159671919 | -5.102327074 | -8.256293583 | 13.2304248 | 6.760786087 |
| **C3** | 10.55773884 | 10.55266137 | 10.8319222 | 1.413413149 | 0.779596484 |
| **C4** | -5.111360028 | -5.110257654 | -5.170888229 | 0.542708237 | 0.361085149 |
| **C5** | 3.259093019 | 3.258144413 | 3.310317742 | 0.316510294 | 0.280393441 |
| **G** | -7568.482538 | -7568.416273 | -7572.060821 | 388.129045 | 252.4508327 |

*A6.2.7.2 Langmuir with surface potential*

| | Fit values | JK estimate | JK bias-corr values | Fit error | JK error |
|---|---|---|---|---|---|
| **A1** | 1.041274446 | 1.040506038 | 1.082768494 | 0.141111815 | 0.063314794 |
| **A2** | 0.962506493 | 0.962420442 | 0.96715324 | 0.087724343 | 0.0230882 |
| **A3** | 0.908211224 | 0.908426304 | 0.896596906 | 0.079322031 | 0.020373132 |
| **A4** | 1.02029114 | 1.021316468 | 0.964923377 | 0.070717865 | 0.051577242 |
| **A5** | 0.964725672 | 0.96516256 | 0.941133723 | 0.065612463 | 0.028156011 |
| **B1** | 17.7771435 | 17.87304839 | 12.59827926 | 28.78063646 | 25.73485389 |
| **B2** | 12.82679498 | 12.77673488 | 15.53004037 | 22.59097656 | 15.43971628 |
| **B3** | 8.926942182 | 8.84621959 | 13.28596215 | 13.81247067 | 10.00043978 |
| **B4** | -1.236584648 | -1.286747948 | 1.472233577 | 3.680028938 | 2.761836195 |
| **B5** | -1.792750884 | -1.796612157 | -1.584242132 | 2.787494067 | 0.572295801 |
| **C1** | 27.85763695 | 27.42783168 | 51.06712153 | 23896.81513 | 21.5308448 |
| **C2** | 15.56878665 | 15.42524465 | 23.32005506 | 16.99204365 | 12.63003522 |
| **C3** | 5.47646047 | 5.569193264 | 0.46888959 | 39.25365418 | 10.15111695 |
| **C4** | -8.209387335 | -8.224382216 | -7.399663798 | 7.584738333 | 5.474360518 |
| **C5** | 7.137571679 | 7.128482519 | 7.628386298 | 4.689916727 | 3.346336907 |
| **D1** | -7.587947619 | -7.763724711 | 1.904015379 | 639058.206 | 27.44518754 |

| | | | | | |
|---|---|---|---|---|---|
| **D2** | -26.00374919 | -26.03258337 | -24.44670361 | 40.91747102 | 37.36293424 |
| **D3** | -0.999911067 | -1.701479234 | 36.88476998 | 89.18584247 | 18.12667047 |
| **D4** | 9.479335154 | 9.545811403 | 5.889617723 | 9.215247596 | 7.516118386 |
| **D5** | -5.699091735 | -5.692590641 | -6.050150817 | 7.93953941 | 5.467549637 |
| **E1** | -11.60140625 | -11.35878083 | -24.70317866 | 418077.3732 | 32.03394082 |
| **E2** | 1.821442891 | 1.833337621 | 1.179127477 | 27.51308137 | 11.4211583 |
| **E3** | 5.895061584 | 5.650378576 | 19.10794402 | 49.21908206 | 11.93130361 |
| **E4** | 12.15538263 | 12.26897149 | 6.021584065 | 17.3625606 | 15.07409088 |
| **E5** | -9.514823088 | -9.539140268 | -8.201695383 | 10.31078539 | 6.956263433 |
| **G** | -7125.048166 | -7165.496062 | -4940.861767 | 3190.79582 | 2758.936896 |

## A6.3 References


(1) Otten, D. E.; Shaffer, P. R.; Geissler, P. L.; Saykally, R. J. Elucidating the Mechanism of Selective Ion Adsorption to the Liquid Water Surface. *Proc. Natl. Acad. Sci.* **2012**, *109* (3), 701–705.

(2) Onorato, R. M.; Otten, D. E.; Saykally, R. J. Adsorption of Thiocyanate Ions to the Dodecanol/Water Interface Characterized by UV Second Harmonic Generation. *Proc. Natl. Acad. Sci.* **2009**, *106* (36), 15176–15180.

(3) Petersen, P. B.; Saykally, R. J.; Mucha, M.; Jungwirth, P. Enhanced Concentration of Polarizable Anions at the Liquid Water Surface: SHG Spectroscopy and MD Simulations of Sodium Thiocyanide. *J Phys Chem B* **2005**, *109* (21), 10915–10921.